Empirical Validation of the Architecture Quality Assurance Framework (AQAF): A Technical Report

Andreas Johnsen[1], Kristina Lundqvist[1], Kaj Hänninen[1], Paul Pettersson[1], and Martin Torelm[2]

[1] School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
{andreas.johnsen,kristina.lundqvist,kaj.hanninen,paul.pettersson}@mdh.se
[2] Bombardier Transportation Sweden AB
Propulsion & Converter Control Standardization
Västerås, Sweden
martin.torelm@se.transport.bombardier.com

Abstract. Architecture engineering is essential to achieve dependability of critical embedded systems and affects large parts of the system life cycle. Architectural faults may consequently cause substantial costs and devastating harm. Verification in architecture engineering, known as architecture-based verification, should therefore be holistically and systematically managed in the development of critical embedded systems, from requirements analysis and architecture design to implementation and maintenance. The Architecture Quality Assurance Framework (AQAF) for critical embedded systems modeled in the Architecture Analysis and Design Language (AADL) has been developed to address this issue. By means of a standardized representation of system architectures in AADL, formal methods can be applied to perform rigorous verification of complex systems. The framework provides a holistic set of verification techniques with a common formalism and semantic domain, architecture flow graphs and timed automata, enabling completely formal and automated verification processes covering a broad area of architecture engineering. More precisely, the framework includes model checking, model-based testing, and selective regression verification techniques for the detection of architecture design and implementation faults, as well as faults introduced at design updates. In this technical report, we present an empirical validation of AQAF where it is applied to a safety-critical train control system. The objective of the case study is to assess fault-finding effectiveness and resource efficiency. The method of fault injection is used to ensure coverage of fault types and to produce an adequate data set from which resource consumption statistically can be assessed. Results suggest an effective fault-finding capacity and an efficient use.

Table of Contents

1 Introduction

Computer systems are an integral part of the human society and increasingly used in environments where our safety is dependent on them. For example, computers are embedded in anti-lock braking (ABS) and electronic stability control (ESC/ESP) systems of cars, in air traffic control (ATC) systems of control towers, in fly-by-wire and navigation systems of aircraft, and in control systems of trains, nuclear power plants, and medical equipment. This type of embedded computer system, where the safety of its environment is dependent on it, is known as a *safety-critical* system. The main requirement in the development of safety-critical systems is that they must not cause hazardous failures that are more frequent and more severe than acceptable [1]. Generally, this is achieved through a combination of hazard analysis, fault forecasting, verification and validation (V&V), and mechanisms of fault tolerance. However, the utilization and complexity of safety-critical systems is increasing beyond what current safety-achieving engineering is able to manage [2, p. 4]. A problem is that failures emerge in the component interactions when the complexity increases, whereas current engineering has given most attention to failures of components rather than their interactions [2, p. 8]. Component interactions is a central concern of architecture design, which together with requirement analysis typically account for the majority of faults introduced in the development process [3]. In addition, the main objective of designing the architecture is to build necessary non-functional properties, such as safety, reliability, and availability, into the system. Non-functional properties are achieved by allocating the functionality to certain software, hardware, and data structures, such as reliability and availability through redundancy, safety trough hazard avoidance and fail-safe mechanisms, and deterministic behavior through time-driven and synchronous computation systems. An architectural design fault may therefore not only cause a failure of the safety-critical functionality, but also to built-in safety mechanisms that are supposed to maintain safety in presence of erroneous system states.

The effects of a faulty architecture design also have a significant impact on the cost and performance of the development process. In [4], a survey quantitatively presents the return on investment of system engineering based on an analysis of 161 projects. Results show that 20% of the defects account for 80% of the rework costs, and that these 20% of defects primarily came from an inadequate architecture definition and risk resolution. In [5], a survey of projects executed by defence contractors quantifies the relationship between system engineering best practices and the performance of the projects in terms of cost, schedule, and scope goals. Results show that there is a strong positive relationship between architecture engineering capabilities and the performance of the projects. For example, only 11% of the projects with lower architecture engineering capabilities exhibited a good performance compared to 46% of the projects with higher capabilities. Consequently, improved architecture engineering capabilities provide the ability to build safer systems at lower costs.

Architecture engineering generally includes the processes of designing the architecture, documenting the architecture, analyzing the architecture, realizing/implementing the architecture, and maintaining the architecture. The need for advancement in architecture engineering has led to the development of architecture description languages (ADLs). The Architecture Analysis and Design Language (AADL) [6] is an ADL that has been developed for modeling architectures of critical embedded systems. The use of AADL generates standardized, computer-readable, and semi-formal models of the system architectures. These properties contribute to the assurance of quality by facilitating understandability, communication, and analysis [7]. In addition, they provide the necessary prerequisites for developing computerized verification techniques that are effective and efficient enough in detecting architectural faults that emerge in the development of complex safety-critical embedded systems.

State of the art architecture-based verification in the domain of AADL integrates formal methods with the standard. The incentive for using formal methods is threefold: (i) fault avoidance based on mathematics of software is an essential method for developing dependable systems [8]; (ii) there is a need for verification evidences based on mathematics in the certification of safety-critical systems [9]; and (iii) formal methods enable automation of verification processes through powerful computer tools, thereby reducing the cost of labor and the risk of human error [10]. Since architecture engineering is conducted in various phases of the system lifecycle [11, 12], a variety of architecture-based verification techniques is necessary to assure architecture quality. Given that the system requirements are correct, architectural faults may be introduced during the architecture design phase, the architecture realization phase, and any time subsequent changes or updates are made to the design or implementation.

In this technical report, we present the Architecture Quality Assurance Framework (AQAF); a possible solution to the complexity of architecture-based verification of critical embedded systems modeled in AADL, and a validation thereof. The framework has been developed on the hypothesis of combining necessary formal verification techniques through a common formal underpinning such that they can be effectively and efficiently used in an integrated manner. AQAF includes a model checking technique to address architecture design faults, a model-based testing technique to address architecture implementation faults, and a selective regression verification technique based on change impact analysis through slicing to address faults introduced in response to design changes. Through a common formal underpinning, we enable the method of performing model-based testing based on the results of model checking [10]. The model-based testing oracle, thereby, is inherently consistent with the model-checked architecture behavior. Furthermore, a common formal underpinning provides explicit trace links between the verification runs, the coverage of the AADL model, and the coverage of the architecture implementation. Regression verification, thereby, can efficiently be executed by only selecting verification runs of the model and implementation that can be traced from the impact analysis.

To validate these propositions, we present an empirical evaluation of AQAF by means of an industrial case study where the framework is applied to a safety-critical train control system developed by Bombardier Transportation AB. The main research objectives are to assess its fault-finding effectiveness and to quantify efficiency in terms of resource consumption. The case study design is composed of two stages. The first stage includes an application of AQAF on the original model of the control system. In the second stage, we use the technique of fault injection and repeat the application of the framework on mutated versions of the system to collect data for statistical evaluation of effectiveness and efficiency.

2 Background information

2.1 The Architecture Analysis and Design Language

AADL was initially released and published as a Society of Automotive Engineers (SAE) Standard AS5506 [13] in 2004, and a second version (AADLv2) [6] was published in 2009. It is a textual and graphical language used to model, specify, and analyze software- and hardware-architectures of real-time embedded systems. AADL is based on a component-connector paradigm that hierarchically describes components, component interfaces, and the interactions (connections) among components. Hence, the language captures functional properties of the system, such as input and output through component interfaces, as well as structural properties through configurations of components, subcomponents, and connectors. Furthermore, means to describe quality attributes, characteristics, and constraints, such as timing and reliability, of application software and execution platform components are also provided through explicit property associations. Changes to the runtime architecture can be described by modes and transitions of modes, and behavior of components can be described by state transitions systems defined in the Behavioral Annex (BA) [14]. AADL defines component abstractions dividable into three groups:

- **Application software components**
  - **Process component**: represents a protected address space containing at least one thread.
  - **Thread component**: represents a schedulable and concurrent unit of sequentially executed source code.
  - **Thread group component**: represents a single reference to a group of threads that have common characteristics (and does not represent a unit of execution).
  - **Subprogram component**: represents a callable piece of sequentially executed source code that operates on data or provides functions to the component that calls it.
  - **Data component**: represents a data type to type port and subprogram parameter interfaces, and static data shareable among components.
- **Execution platform components**
  - **Processor component**: represents hardware with associated software that schedules and executes threads.
  - **Virtual processor component**: represents a logical resource that is able to schedule and execute threads.
  - **Memory component**: represents a storage for executable code and data.
  - **Bus component**: represents a component that can exchange control and data between processors, memories, and devices.
  - **Virtual bus component**: represents a logical bus abstraction.
  - **Device component**: represents a dedicated entity within the system, or an entity in or interfacing with the external environment, such as GPS systems, counters, timers, sensors, and actuators.

- **General composite components**
  - **System component**: represents a composition of software, hardware, and/or system components, where the software components can be allocated to the hardware components.

A component is modeled by a *component type* declaration and a *component implementation* declaration. A component type is declared with an unique identifier and specifies the external interfaces (known as "features" in AADL) of the component, externally visible properties, and explicit data and control flows between the external interfaces. Consequently, a component type provides a black-box view of the component. Interfaces are declared in a *features* sub clause and represent interaction points for the exchange of data and control to other components. There are three types of features: ports, component accesses, and parameters. Ports represent interaction points for directional exchange of data, events, or both. A port can either be declared as a *data* port, an *event* port, or an *event data* port. A data port communicates typed state data without queuing, such as sensor data streams, where the connection between data ports can be declared as immediate (transmitted upon completion of a thread) or delayed (transmitted upon the deadline). An event port communicates events with queueing, such as dispatch triggers of threads, triggers for mode switches, and alarms. An event data port communicates messages, i.e., data associated with events, with queuing. Parameters exclusively represent interaction points of subprograms for the transmission of call (in parameter) and return (out parameter) data values. Component access declarations support modeling of static data shareable among components and modeling of hardware components communicating through buses. Access declarations are named and can be declared with a *provides* or *requires* statement. A provides statement denotes that a component provides access to a data or bus component internal to it. A requires statement denotes that a component requires access to a data or bus component external to it.

Property declarations may be included in a *properties* sub clause of a component type. A property constraints the expression it is associated with, and in this case, as a sub clause of a component type, constraints the component type. Examples of other expressions that can be associated with property declarations are: component implementations, subcomponents, features, connections, flows, modes, mode transitions, and subprogram calls. A property declaration consists of a name, a type, and a value. The name corresponds to the identifier of the property. The property type specifies a set of values that is accepted for a given property, and each property must be assigned a value or a list of values. There exist built-in predeclared properties in the language, but creation of new properties is supported.

A component implementation declaration represents the internal component structure in terms of subcomponents and their connections, component-internal properties, and modes and the transitions between them. The component implementation therefore can be viewed as a white box in contrast to its component type. A component implementation must be coupled with a component type.

The component implementation *subcomponents* sub clause represents a component's internal components. These internal components can themselves have subcomponents resulting in a hierarchy that eventually describes a whole system.

The connections between interfaces of subcomponents are declared within a *connections* sub clause of a component implementation. There are three types of connections: *port connections*, *component access connections* and *parameter connections*. Port connections represent directional transfer of data and control between ports. A component access connection represents the path from the component providing access to the component requiring access. Parameter connections represent flows of data into and out of subprograms. Subprogram components are accessed through *call* statements. Calls are declared in the component implementation *calls* sub clause or in behavioral models. Finally, component implementations may be modeled with mode state machines to specify the set of components, connections, and properties that are active in a specific mode, and with behavioral models (automata) by means of the Behavioral Annex to refine the execution behavior of the component.

Definition 1 Formally, an AADL model is a tuple:

$$AADLMDL = \langle PROC, COMP, THR, DATA, SUB, C, CALL\rangle$$

$PROC = \{processor_1, processor_2, \ldots, processor_n\}$ denotes the set of processors in the architecture. A $processor = \langle B\_THR, Scheduling\_Protocol\rangle$ has a set of threads $B\_THR \subseteq THR$ bound to it and a scheduling protocol property. $COMP = \{comp_1, comp_2, \ldots, comp_n\}$ denotes the set of software components in the architecture, where THR denotes the set of *thread components*, $DATA$ denotes the set of *data components*, and SUB denotes the set of *subprogram components*. Let thr, $data$, and sub range over THR, $DATA$, and SUB respectively. A thread $thr = \langle DATA\_S, SUB\_S, DP, EP, EDP, DA, SA, MSM, BM,$ $SCH\_PROP\rangle$ has a set of *data subcomponents* $DATA\_S \subseteq DATA$; a set of *subprogram subcomponents* $SUB\_S \subseteq SUB$; a set of data ports $DP = \{dp(data) \mid dp(data)$ is an in/out/in out *data port* of data type $data \in DATA$ and of the form *port* (see Table 1)$\}$; a set of event ports $EP = \{ep \mid ep$ is an in/out/in out *event port* and of the form *port*$\}$; a set of event data ports $EDP = \{edp(data) \mid edp(data)$ is an in/out/in out *event data port* of data type $data \in DATA$ and of the form *port*$\}$; a set of data accesses $DA = \{da(data) \mid da(data)$ is a *data access* to shared data $data \in DATA$ and of the form *component_access*$\}$; a set of subprogram accesses $SA = \{sa(sub) \mid sa(sub)$ is a *subprogram access* to subprogram $sub \in SUB$ and of the form *component_access*$\}$; a *Mode State Machine* MSM; a *Behavioral Model* BM; and a set of scheduling properties $SCH\_PROP = \{Dispatch\_Protocol, Period, Compute\_Execution\_Time, Compute\_Deadline,$ $Priority\}$ of the form *Property*.

A subprogram $sub = \langle DATA\_S, SP, EP, EDP, DA, SA, MSM, BM\rangle$ has a set of *data subcomponents* $DATA\_S \subseteq DATA$; a set of *subprogram parameters* $SP = \{sp(data) \mid sp$ is an in/out/in out *parameter* of data type $data \in DATA$ and of the form *parameter* $\}$; a set of event ports $EP = \{ep \mid ep$ is an out *event port* of data type $d \in DATA\}$; a set of event data ports $EDP = \{edp(data) \mid$

$edp(data)$ is an out *event data port* of data type $data \in DATA\}$; a set of data accesses $DA = \{da(data) \mid da(data)$ is a *data access* to shared data $data \in DATA\}$; a set of subprogram accesses $SA = \{sa(sub) \mid sa(sub)$ is a *subprogram access* to subprogram $sub \in SUB\}$; a *Mode State Machine* MSM; and a *Behavioral Model* BM.

Let DP_U, EP_U, EDP_U, SP_U, DA_U and SA_U denote the union of all sets of component data ports, event ports, event data ports, parameters, data accesses, and subprogam accesses respectively. C denotes the set of connections in the architecture, $C = \{c(source, destination) \mid c$ is a *port connection* from $source \in DP\_U \cup EP\_U \cup EDP\_U$ to $destination \in DP\_U \cup EP\_U \cup EDP\_U$ of the form *port_connection*; or a *data access connection* (access to shared data) from $source \in DATA$ to $destination \in DA\_U$ of the form *data_access_connection*; or a *subprogram access connection* from $source \in SUB$ to $destination \in SA\_U$ of the form *subp_access_connection*; or a *parameter connection* from $source \in SP\_U \cup DP\_U \cup EDP\_U$ to $destination \in SP\_U \cup DP\_U \cup EDP\_U$ and $\langle source, destination\rangle \notin DP\_U \times DP\_U \cup DP\_U \times EDP\_U \cup EDP_U \times DP\_U$ of the form *parameter_connection*$\}$.

$CALL$ denotes the set of subprogram calls in the architecture, $CALL = \{call(source) \mid call$ is a *subprogram call* of the form *subprogram_call* and $source \in SUB\}$.

A Behavioral Model $comp_i.BM = \langle S, s_o, CPL, FIL, VAR, TR\rangle$ has a set of *states* S of the form *state*; an *initial state* $s_0 \in S$; a set of *complete states* $CPL \subseteq S$; a set of *final states* $FIL \subseteq S$; a set of typed variables VAR of the form *variable*; and a set of *state transitions* $TR \subseteq S \times PRI \times G \times ACT \times S$ of the form *state_transition*. A state $s \notin CPL \cup FIL \cup s_0$ is called an *execution state*. We shall use the denotation $s \xrightarrow{pri,g,act} s'$ iff $\langle s, pri, g, act, s'\rangle \in TR$. $pri \in \mathbb{N}$ is the *priority* of the transition. g is a (possibly empty) set of *guards*, which are predicates (also known as *execute conditions*) over local variables, component ($comp_i$) in ports, component in parameters, subcomponent ($comp_i.sub\_s_j$) out ports, subcomponent out parameters, data subcomponents, or accesses to shared data components iff $s \notin CPL \cup FIL$; or predicates (also known as dispatch conditions) over (dispatch triggered by) event ports or event data ports (including receipt of a call) iff $s \in CPL$. act is a (possibly empty) set of *actions* which are sequences (elements of a sequence are separated by ";" and executes in that order) and sets (separated by "&" and executes nondeterministically) of: subprogram calls with arguments of the form $sub!(list)$ where $sub \in SUB$ and $list \in ARG \times ARG^*$ where ARG is the union of local variables, component ($comp_i$) in ports and parameters, subcomponent ($comp_i.sub\_s_j$) out ports and parameters, data subcomponents, and accesses to shared data components; of *assignments* of the form $target := expr$ where $target \in VAR \cup comp_i.DATA \cup comp_i.DA \cup comp_i.DP \cup comp_i.EP \cup comp_i.EDP$ where $expr$ is an arithmetic expression over local variables, component in ports and parameters, subcomponent out ports and parameters, data subcomponents, and accesses to shared data components; and of timed actions of the form **com-**

putation(min **..** max**)** which represent the use of the bounded CPU in terms of a duration between $min \in \mathbb{N}$ and $max \in \mathbb{N}$ time units.

A Mode State Machine $comp_i.MSM = \langle M, m_o, MTR\rangle$ has a set of operational states (runtime configurations) called *modes* M of the form *mode*; an *initial mode* $m_0 \in S$; and a set of *mode transitions* $MTR \subseteq M \times TRI \times M$ of the form *mode_transition*. We shall use the denotation $m \xrightarrow{tri} m'$ iff $\langle s, tri, s'\rangle \in MTR$. TRI is a set of *triggers* which is the union of component ($comp_i$) in event and event data ports, and subcomponent ($comp_i.sub\_s_j$) out event and event data ports.

Table 1. AADL grammar in Backus-Naur Form (BNF)

port_connection	::=	*identifier* **:** (**data port** | **event port** | **event data port**) *source_port_reference* (**−>** | **−>>**) *destination_port_reference*
data_access_connection	::=	*identifier* **: data access** *data_component_reference* (**−>** | **<−>**) *access_require_reference*
subp_access_connection	::=	*identifier* **: subprogram access** *subprogram_component_reference* **<−>** *access_require_reference*
parameter_connection	::=	*identifier* **: parameter** *source_parameter_reference* **−>** *destination_parameter_reference*
subprogram_call	::=	*identifier* **: subprogram** *subprogram_reference*
port	::=	*identifier* **:** (**in** | **out** | **inout**) (**data port** | **event data port** | **event**) *data_component_reference*
component_access	::=	*identifier* **: requires** (**data access** | **subprogram access**) *component_reference*
parameter	::=	*identifier* **:** (**in** | **out** | **in out**) **parameter** [*data_component_reference*]
state	::=	*state_identifier* **:** [**initial**][**complete**][**final**] **state**
variable	::=	*variable_declarator* **:** *data_component_reference*
state_transition	::=	[*identifier* [*priority*] **:**] *source_state_identifier* **−[**[*guard*]**]−>** *destination_state_identifier* [*action*]
mode	::=	*identifier* **:** [**initial**] **mode**
mode_transition	::=	[*identifier* **:**] *source_mode_identifier* **−[** *trigger* **]−>** *destination_mode_identifier*
property	::=	*identifier* **=>** *value*

2.2 Uppaal timed automata

In this study, we use the Uppaal model-checker [15] to model-check AADL models and to generate test cases. We have chosen Uppaal due to its maturity and its ability to verify timing constraints. In Uppaal, a system is represented by a network of timed finite state automata and checked by formulae in Timed Computational Tree Logic (TCTL) [16]. A timed finite state automaton consists of *locations* (nodes), *edges* (arcs) connecting locations, and *labels* (alphabet) associated with these elements. Uppaal extends the automata theory with clock variables to model time, where all clocks progress synchronously through real numbers, and with discrete variables that can be read, assigned, or used in arithmetic operations and propositional formulae. Furthermore, Uppaal allows coding of functions (in UCode, a subset of C) callable in labels of transitions.

Definition 2 Formally, a network of timed automata $NTA = \langle \overline{TA}, Var_G, Ch \rangle$ has a vector of n timed automata $\overline{TA} = \langle TA_0, TA_1, \ldots, TA_{n-1} \rangle$, a set of shared (global) variables Var_G, and a set of synchronization channels Ch. A timed automaton $TA = \langle L, \ell_0, X, Var, I, E \rangle$ has a set of locations L, an initial location $\ell_0 \in L$, a set of real-valued variables X called *clocks*, a set of (bounded) integer-typed variables Var, a function assigning invariants to locations $I : L \to G$, and a set of edges $E \subseteq L \times G \times Act \times U \times L$. G is a set of *guards*, which are predicates over variables and clock constraints of the form $x \, expr_1 \, c$, where $x \in X \cup Var \cup Var_G$, $c \in \mathbb{N}$, and $expr_1 \in \{<, \leq, =, \geq, >\}$. $Act = I \cup O \cup \{\tau\}$ is a set of input (denoted a?) and output (denoted a!) synchronization actions and the non-synchronization τ. U is a set of *updates* which are sequences of variable-assignments of the form $v := expr_2$ and/or clock resets of the form $x := 0$, where $v \in Var \cup Var_G$, $x \in X$, and $expr_2$ is an arithmetic expression over integers. We shall use the denotation $\ell \xrightarrow{g,a,u} \ell'$ iff $\langle \ell, g, a, u, \ell' \rangle \in E$. In addition, locations may be labelled as urgent or committed. In an urgent location, time is not allowed to progress whereas in a committed location, time is not allowed to progress and the next transition must involve one of its outgoing edges.

The semantics of a network of timed automata is defined in terms of a timed transition system over system states. A *system state* is a triple $\langle \overline{\ell}, \overline{\phi}, \overline{\sigma} \rangle$ where $\overline{\ell}$ is a location vector over all automata such that $\overline{\ell^0}, \overline{\ell^1}, \ldots, \overline{\ell^{n-1}}$ denotes the current location of $\underline{TA_0, TA_1, \ldots, TA_{n-1}}$, $\overline{\phi}$ is a clock valuation vector over all automata such that $\overline{\phi^0}, \overline{\phi^1}, \ldots, \overline{\phi^{n-1}} \in \mathbb{R}_+^X$ and satisfies the invariants of the locations $(\overline{\phi} \models I(\overline{\ell}))$, and $\overline{\sigma}$ is a variable valuation vector that maps variables to values and $\overline{\sigma} \models I(\overline{\ell})$. The *initial system state* is a state $\langle \overline{\ell_0}, \overline{\phi_o}, \overline{\sigma_o} \rangle$ where $\overline{\ell_0}$ is the initial location vector, $\overline{\phi_o}$ maps each clock to zero, and $\overline{\sigma_o}$ maps each variable to its default value. Progress is made through *delay transitions* or *discrete transitions*. A *delay transition* is of the form $\langle \overline{\ell}, \overline{\phi}, \overline{\sigma} \rangle \xrightarrow{d} \langle \overline{\ell}, \overline{\phi} \oplus d, \overline{\sigma} \rangle$ where $\overline{\phi} \oplus d$ is the result of synchronously adding the delay d to each clock valuation in $\overline{\phi}$. Let $\overline{\ell}[\ell'_i/\ell_i]$ denote that the ith vector element ℓ_i is replaced by ℓ'_i. A *discrete transition* is of the form $\langle \overline{\ell}, \overline{\phi}, \overline{\sigma} \rangle \xrightarrow{a} \langle \overline{\ell}[\ell'_i/\ell_i, \ell'_j/\ell_j, \ell'_k/\ell_k, \ldots], \overline{\phi'}, \overline{\sigma'} \rangle$ such that there are edges $\ell_{i/j/k\ldots} \xrightarrow{g_{i/j/k\ldots}, a_{i/j/k\ldots}, u_{i/j/k\ldots}} \ell'_{i/j/k\ldots}$ where $\overline{\phi}$ and $\overline{\sigma}$ satisfies $g_i \wedge g_j \wedge g_k \ldots$, the result of updating $\overline{\phi}$ and $\overline{\sigma}$ according to $u_i, u_j, u_k, \ldots$ is $\overline{\phi'}$ and $\overline{\sigma'}$, and the edges are synchronous over complementary actions ($a?$ complements $a!$). A trace is a sequence of states such that there exist a delay or discrete transition from each state in the sequence leading to its successor state.

A model is verified by TCTL queries in the form of path formulae and state formulae, where the UPPAAL model-checker searches the state space of the model to check if it satisfies the formulae. If the model satisfies the expected behavior, the information gathered during the search may be used to generate test cases. State formulae are expressions that describe properties of individual states while path formulae are expressions that describe properties over paths of states. A state formula is a predicate such as $x == 4$ or $x \leq 10$, where these formulae are valid in a state whenever x equals four or x is less or equal to ten. State formulae can be evaluated to valid or invalid for a state without analyzing the behavior of the model to or from the particular state. Path formulae are classified into reachability, safety, and liveness property formulae: a reachability property formula checks whether a predicate can be satisfied by a reachable state along some path; a safety property formula checks whether a predicate invariantly is satisfied in each state of a path, or all paths; and a liveness property formula checks whether a predicate eventually is satisfied by a reachable state in all paths.

Reachability properties are verified using temporal operators E (pronounced "for some path" or "exists one path") and $<>$ (pronounced "eventually"). For example, in order to verify if a state formula p is reachable, we simple check it by the formula $E <> p$ (pronounced "for some path eventually p holds"). Safety properties are verified using temporal operators A (pronounced "for all paths"), E, and $[]$ (pronounced "always" or "globally"). Formula $A[]p$ (pronounced "for all paths globally p holds") is used if the property should hold in all states for all paths whereas formula $E[]p$ (pronounced "for some path globally p holds") is used if the property should hold for all states in at least one path. Liveness properties are verified by using temporal operators A, $<>$, and $\rightarrow$ (pronounced"leads to"). Formula $A <> p$ (pronounced "for all paths eventually p holds") checks whether p eventually holds in all paths whereas $p \rightarrow q$ (pronounced "whenever p holds eventually q holds") checks whether q eventually holds whenever p holds.

3 Framework overview

The primary focus of evaluation at the architectural level is the integration of components, including the structure and the resulting emergent behavior and non-functional properties [17]. General verification objectives are to ensure consistency, completeness, and correctness of component interfaces and the control and data interactions among them [17, 11]. To obtain these objectives, the quality assurance framework, illustrated in Fig. 1, is based on architectural control and data flow verification criteria (c). This is of industrial importance as some contemporary safety standards (e.g., ISO 26262 [18]) request control- and data-flow analysis of software architecture designs. The verification criteria essentially demand all architectural control and data flows of the AADL model to be exercised in the verification of the model and the implementation. The modeled flows are extracted from the AADL model through the generation of an architecture flow graph (b). The application of the verification criteria on the flow graph results in a set of verification sequences (d) that must be run on the model, and later on the implementation when available. Each verification sequence corresponds to a path of control and data flows through the architecture design and the requirements (functional and non-functional) that must be achieved when executing the path.

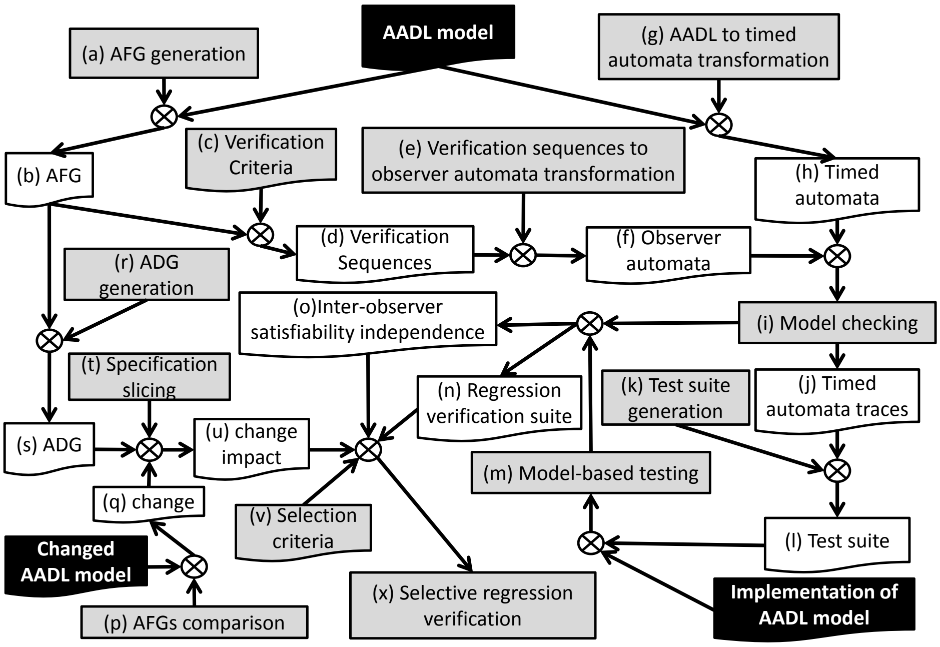

AADL model
(a) AFG generation
(g) AADL to timed automata transformation
(b) AFG
(c) Verification Criteria
(e) Verification sequences to observer automata transformation
(h) Timed automata
(d) Verification Sequences
(f) Observer automata
(r) ADG generation
(o)Inter-observer satisfiability independence
(i) Model checking
(t) Specification slicing
(n) Regression verification suite
(k) Test suite generation
(j) Timed automata traces
(s) ADG
(u) change impact
(m) Model-based testing
(q) change
(v) Selection criteria
(l) Test suite
Changed AADL model
(x) Selective regression verification
Implementation of AADL model
(p) AFGs comparison

Fig. 1. Flowchart of the verification framework. A black shape denotes a necessary framework input. A gray shape denotes a formally defined process or rule set (c and v). A white shape denotes an artifact produced by the framework.

Based on the control and data flow verification criteria, model checking (i) and model-based testing (m) techniques are used to automatically and formally execute the verification sequences. The semantic domain of AADL, however, is not based on a mathematical language and cannot be directly explored by a model-checker. The framework therefore includes a formal semantics of AADL in timed automata (h) such that the verification processes can be executed by the Uppaal model checker. Verification sequences are executed on the model by transforming them to observer automata (f) [19] – a flexible method for specifying coverage criteria for model checking and model-based test case generation. Satisfied observers indicate a complete, consistent, and correct model. In that case, the satisfied observers produce a set of traces (k) (one for each observer) which can be transformed into a test suite (l) that tests the conformance of the implementation with respect to the model. The implementation conforms to the model if each path may be executed in the implementation while the expected functional and non-functional properties are met.

Nevertheless, the design is typically subjected to modifications. The artifacts must therefore undergo regression verification to verify that no new faults have been introduced in response to a design change. In addition, architectural variants may be designed to develop a product line or to analyze trade-offs, or an architecture may be incrementally designed through iterations of added design decisions. All these scenarios, where different instances of the architecture design or implementation are created, are common and challenged with inefficient regression verification if equivalent parts among the artifacts that are not affected by the changes or variations are unnecessarily re-verified. To efficiently perform regression verification, the framework includes an approach to selective regression verification (x). The approach is to trace the impact a change may have on the residual architecture, and thereby only selecting verification sequences that are affected by the change. The first step of the approach is to locate the change by comparing the flow graph of the original model with the flow graph of the changed model (p). The possibly affected parts of the architecture with respect to the located change are then traced through static slicing (t). Slicing is conducted by means of an architecture dependence graph (s) generated from the architecture flow graph of the changed model. The slice may be further reduced based on inter-observer coverage data (from which satisfiability independences between observes can be deduced) of the preceding verification cycle, which constraint dependencies in the architecture with additional dynamic information. The regression verification process is then efficiently executed by only selecting verification sequences that cover parts of the sliced AADL model.

4 Case study design

The case study presented in this report yields an application of AQAF to the safety-critical train control system presented in Section 5. The objective is to validate the framework in terms of effectiveness and efficiency. An application of the complete framework involves the following steps:

1. Generate the architecture flow graph (AFG) of the AADL model.
2. Generate verification sequences by applying the verification criteria to the AFG.
3. Transform the AADL model to a network of timed automata and each verification sequence to an observer automaton.
4. Verify the satisfiability of each observer using the Uppaal model-checker.
   - The AADL model is complete, consistent, and correct if all observers are satisfiable. The resulting traces may be used to test the conformance of an implementation with respect to the AADL model, i.e., go to step 5.
   - The AADL model is faulty if not all observers are satisfiable. The model should be updated where steps 8-11 subsequently may be used to perform selective regression verification of the updated model.
5. Generate a test suite from the produced timed automata traces – one test case for each trace.
6. Execute the test suite on the implementation.
   - The implementations conforms to the AADL model if each test case passes.
7. Modify the AADL model.
8. Generate the AFG and architecture dependence graph (ADG) of the modified model.
9. Compare the AFG of the previously verified AADL model with the AFG of the modified model to identify the modification.
10. Slice the ADG of the modified model with respect to the identified modification. The slice determines which parts of the modified model that may be impacted by the modification.
11. Perform selective regression verification of the modified model where only verification sequences that cover parts in the slice are selected.

The faultiness of the AADL model must be controlled in the study to be able to validate efficiency and effectiveness. Firstly, an AADL model and an implementation thereof may have different types of faults which the framework is expected to find. Secondly, the presence of multiple faults may result in a complex combined error behavior that makes it difficult to determine if the framework produces any false positives or false negatives, i.e. if the framework falsely declares presence or absence of faults. Finally, under the assumption that no false positives or negatives are produced by the model checking technique, a conformance test suite generation demands a fault-free AADL model (satisfied observers). In order to apply the complete framework and simultaneously ensure validity of the results and coverage of fault types, the framework must be systematically applied to controlled versions of the model.

4.1 A systematic application of AQAF

Our approach to a systematic application involves two stages. The first stage is to perform steps 1-6 based on the AADL model presented in Section 5, which we assume to be free from architectural faults. If the steps are valid and the implementation truly conforms to the model, the result of model checking and model-based testing must necessarily be satisfied observers and passed test cases. Since the model certainly conforms to itself, it is treated as the implementation in step 6. Under the assumption that the premise is true, i.e. that the AADL model truly is free from faults, these steps inductively support the absence of false negatives and false positives if the results meet the expectations.

The second stage of the approach is to use the technique of fault injection to create mutated versions of the AADL model and extend the steps performed in stage one such that the application covers the complete framework and ranges over the possible fault types. By means of the steps taken in the first stage, each fault injection correspond to a modification. If steps 1-4 and 8-11 are valid, the result of regression verification must necessarily be at least one unsatisfied observer – since there now definitely exist a fault. Nevertheless, the selective approach must be contrasted with a re-run all approach to conclude the selection effectiveness and assess its efficiency. If steps 1-4 and 8-11 are valid, the result must necessarily be satisfied observers for all non-selected verification sequences (since the impact analysis is expected to select all verification sequences that possibly are affected by the modification). The required overhead expense of conducting the selection in addition to the resource consumption of running the selection must either not exceed the cost of a re-run all approach to be efficient.

Furthermore, each mutated version may be treated as an implementation to validate the fault-finding effectiveness of the test suite generated in the first stage. If steps 1-5 are valid, the result must necessarily be at least one test case which did not pass for each tested mutation. Each result that meets the expectations increase the validity of its fault-finding capabilities, which in turn strengthen the truthfulness of the undertaken premise and soundness of the framework as a whole. The repeated framework applications also provide a basis from which efficiency can be statistically quantified.

The outline of this technical report follows the structure of this framework application.

4.2 Research measures

Effectiveness of model checking and model-based testing is measured in terms of the ratio of found faults to the number of injected faults. Effectiveness of selective regression verification is measured in terms of the ratio of unsatisfied non-selected observers to the number of selected and non-selected unsatisfied observers. Note that a lower ratio denotes a higher effectiveness in the latter case. A ratio of zero means that the technique did not exclude any verification sequence that reveals a fault in the modified architecture. Efficiency is measured in terms of time and memory consumption. The resource consuming activities of model checking are observers generation (including AFG generation and verification sequences extraction), AADL to timed automata transformation, and satisfiability checking. For model-based testing, the resource consuming activities are identical except that the test data is extracted by searching the resultant traces. In either case, the bottleneck is satisfiability checking of observers due to the state space explosion problem, which is the resource consuming activity of interest in this case study. With respect to efficiency of the selective verification technique, the resource consumption of slicing is included to make sure that the overhead the selective regression verification technique brings does not exceed the savings. All measurements in this study are performed in Windows 7 64-bit edition running an Intel Core i7-3667U 2.0 GHz CPU with 8 GB RAM.

4.3 Fault injections

In this study, we consider the following fault types:

1. Absent, unachievable, and (logically) incorrect control expressions (guards)
2. Absent and incorrect data assignments, events, and calls (actions)
3. Absent and incorrect connections
4. Absent, incorrect, and incompatible non-functional properties
5. Absent, incorrect, and incompatible protocols of critical sections and shared resources
6. Absent, incorrect, and incompatible scheduling properties
7. Deadlock, livelock, starvation, and priority inversion of processes and threads

Note that a fault may yield a combination of these fault types.

5 The Line Trip Relay Interface and Supervision system

Line trip relay interface and supervision (LTRIS) is a safety-critical train control sub-system embedded in a system of systems developed by Bombardier Transportation AB. In this section, we present an AADL model of LTRIS that has been created for the purpose of conducting the case study. The model has been created by informal means and properties of the operating system and hardware platform have been abstracted to delimit the workload of the case study. Consequently, analysis of the AADL model cannot be deduced to the source system without taking these abstractions into consideration. Moreover, LTRIS interacts with numerous components in the system of systems it is embedded within. A sound architecture representation of LTRIS should include the architecture of the surrounding system. However, inclusion of the complete architecture is out of scope of the study plan. Instead, we abstract the functionality of the software in the environment into a single process component, denoted *LineTripEnvironment* in the model, and assume it executes with LTRIS on a common processing platform, as shown in Table 2. Any required input by the LTRIS software, denoted *LineTripSoftware.Impl*, is assumed to be produced by *LineTripEnvironment* although the specifics of the connections are not modelled. Each of these is referred to as "some connection" in the model. Non-functional properties of LTRIS have been adjusted in the model according to the abstraction of the environment.

LineTripSoftware.Impl, presented in Table 3, is essentially composed of two connected periodic threads: *Controller* and *Tester*. The functionality of *Controller*, presented in Table 4, is to control a critical relay, monitor its status, and output feedback data. The feedback data is information on the status of the relay and the status relative to the expected one. *Controller* controls the relay according to data on in ports and shared variables assigned by components in the environment (abstracted to *LineTripEnvironment*). In this manner, *Controller* acts as an interface to the relay.

The behavior of *Controller* includes two consecutive subprogram calls, first a call to subprogram *LtrInt*, presented in Table 5, followed by a call to subprogram dcu2_line_trip, presented in Table 6. The possible opening and closing requests received at in ports of *Controller*, together with state information of *LineTripEnvironment*, are given as arguments to *LtrInt* when called. *LtrInt* then performs a sequence of operations on the input, as specified by its behavioral model, to determine whether on opening or closing output (return) signal shall be produced. The logic has been composed such that an opening request shall be prioritized over a closing request.

The output signal produced by *LtrInt* is used as argument in the second call to *dcu2_line_trip*, which is responsible of controlling the relay and produces feedback through out parameters, as specified by its behavioral model. The feedback is made available for the environment through out ports of *Controller* (which are connected to the out parameters of *dcu2_line_trip*).

The input domain of *Controller* is partly set by *Tester*, presented in Table 7, through *connection1* and *connection2*, as specified in *LineTripSoftware.Impl* (see Table 3). The functionality of *Tester* is to execute a test sequence, *LtrTsSq* presented in Table 8, verifying a correct functioning of the relay. The behavior is specified in a behavioral model, from which subprogram *LtrTsSq* is invoked. *LtrTsSq* may transmit opening and closing requests to *Controller* through *connection1* and *connection2* respectively.

As shown by the behavioral model of subprogram *LtrTsSq.Impl* in Table 8, the test sequence exercises the relay in the possible ways it can be exercised: starting from an unidentified state of the relay, the test sequence signals an opening request (*B_OpLtr := true*), followed by a closing request (*B_CdLtr := true*), which finally is followed by an opening request. The final request is delayed 512 ms in order to make sure that there is enough time for the power supply to close the relay before it finally is opened. Each request is validated by boolean conditions, which evaluation is dependent on the data assigned to in parameters *B_LtrFl* and *S_LtrCd*, before a subsequent request is sent. The data assigned to in parameters *B_LtrFl* and *S_LtrCd* correspond to feedback produced by *Controller*. The feedback data assigned to *S_LtrCd* represent the state of the relay and is transmitted to *Tester* through *connection3* (see Table 3). Data assigned to *B_LtrFl*, on the other hand, represent the functioning of the relay and is sent from *LineTripEnvironment*. The test process is reset to its default state any time a malfunction signal is transmitted to *B_LtrFl* during the execution of the test sequence. Whenever the test sequence successfully has been executed, a signal is assigned to out parameter *A_LtrTs* and made available to *LineTripEnvironment* through out port *DHSSMG_S_LtrTsRdy* of *Tester*.

Table 2: LTRIS system component

```
system LineTripSystem
end LineTripSystem;

system implementation LineTripSystem.Impl
subcomponents
LTSoftware: process LineTripSoftware.Impl;
LTEnvironment: process LineTripEnvironment;
Hardware: processor ProcessorPlatform.Impl;
properties
Actual_Processor_Binding =>(reference (hardware)) applies to LTSoftware;
Actual_Processor_Binding =>(reference (hardware)) applies to LTEnvironment;
end LineTripSystem.Impl;
```

Table 3: LTRIS software component

```
process LineTripSoftware
features
PRASMZ_B_RqPrSd: in data port Base_Types::Boolean;
APSIMZ_B_OpLtr: in data port Base_Types::Boolean;
SSSCMZ_NX_MnSqSt: in data port Base_Types::Integer;
PCTHMZ_A_PctMo: in data port Base_Types::Boolean;
```

```
PLTTMG_B_OpLtr: in data port Base_Types::Boolean;
APSIMZ_B_EnCdLnTrpSlt: in data port Base_Types::Boolean;
PARAGP_L_CnfHpp: in data port Base_Types::Boolean;
DHSSMG_B_LtrHwOpFl: in data port Base_Types::Boolean;
DCUIMG_C_LtrTs: in data port Base_Types::Boolean;
DHSSMG_B_LtrFl: in data port Base_Types::Boolean;
DIGIMG_S_LtrOp: in data port Base_Types::Boolean;
DCUIMG_S_DcuNtRdy: in data port Base_Types::Boolean;
PARAGP_L_LtrSvEn: in data port Base_Types::Boolean;
DIGOMG_B_CdLtr: out data port Base_Types::Boolean;
DHWOMG_B_FpgaLtrOn: out data port Base_Types::Boolean;
DHWOMG_B_DcuLtrFl: out data port Base_Types::Boolean;
DHSSMG_NX_LtrSaSq: out data port Base_Types::Integer;
DHSSMG_S_LtrTsRdy: out data port Base_Types::Boolean;
DHSSMG_S_LtrOpVd: out data port Base_Types::Boolean;

GPIO_OUT: requires data access GPIO_OUT;
LTRIP_EN_N: requires data access LTRIP_EN_N;
MCU_LT_ON: requires data access MCU_LT_ON;
FPGA_LTRCR: requires data access FPGA_LTRCR;
FPGA2_LT_ON: requires data access FPGA2_LT_ON;
LT_RELAY_FB: requires data access LT_RELAY_FB;
end LineTripSoftware;

process implementation LineTripSoftware.Impl
subcomponents
relayController: thread Controller;
relayTester: thread Tester;
connections
some_connection1: port PRASMZ_B_RqPrSd ->relayController.PRASMZ_B_RqPrSd;
some_connection2: port APSIMZ_B_OpLtr ->relayController.APSIMZ_B_OpLtr;
some_connection3: port SSSCMZ_NX_MnSqSt ->relayController.SSSCMZ_NX_MnSqSt;
some_connection4: port PCTHMZ_A_PctMo ->relayController.PCTHMZ_A_PctMo;
some_connection5: port PLTTMG_B_OpLtr ->relayController.PLTTMG_B_OpLtr;
some_connection6: port APSIMZ_B_EnCdLnTrpSlt ->
relayController.APSIMZ_B_EnCdLnTrpSlt;
some_connection7: port PARAGP_L_CnfHpp ->relayController.PARAGP_L_CnfHpp;
some_connection8: port DHSSMG_B_LtrHwOpFl ->
relayController.DHSSMG_B_LtrHwOpFl;
some_connection18: port relayController.DIGOMG_B_CdLtr ->DIGOMG_B_CdLtr;
some_connection20: port relayController.DHWOMG_B_FpgaLtrOn ->
DHWOMG_B_FpgaLtrOn;
some_connection21: port relayController.DHWOMG_B_DcuLtrFl ->
DHWOMG_B_DcuLtrFl;
some_connection9: port DCUIMG_C_LtrTs ->relayTester.DCUIMG_C_LtrTs;
some_connection10: port DHSSMG_B_LtrFl ->relayTester.DHSSMG_B_LtrFl;
some_connection12: port DIGIMG_S_LtrOp ->relayTester.DIGIMG_S_LtrOp;
some_connection13: port DCUIMG_S_DcuNtRdy ->
relayTester.DCUIMG_S_DcuNtRdy;
some_connection14: port PARAGP_L_LtrSvEn ->relayTester.PARAGP_L_LtrSvEn;
```

```
some_connection15: port relayTester.DHSSMG_NX_LtrSaSq ->
DHSSMG_NX_LtrSaSq;
some_connection16: port relayTester.DHSSMG_S_LtrTsRdy ->
DHSSMG_S_LtrTsRdy;
some_connection17: port relayTester.DHSSMG_S_LtrOpVd ->
DHSSMG_S_LtrOpVd;
connection1:port relayTester.DHSSMG_B_OpLtr ->
relayController.DHSSMG_B_OpLtr {Timing =>Immediate; Latency =>0ms .. 1ms;};
connection2: port relayTester.DHSSMG_B_CdLtr ->
relayController.DHSSMG_B_CdLtr {Timing =>Immediate; Latency =>0ms .. 1ms;};
connection3: port relayController.DHWOMG_S_LtrCd ->
relayTester.DHWOMG_S_LtrCd {Timing =>Immediate; Latency =>0ms .. 3ms;};

data access GPIO_OUT ->relayController.GPIO_OUT;
data access LTRIP_EN_N ->relayController.LTRIP_EN_N;
data access MCU_LT_ON ->relayController.MCU_LT_ON;
data access FPGA_LTRCR ->relayController.FPGA_LTRCR;
data access FPGA2_LT_ON ->relayController.FPGA2_LT_ON;
data access LT_RELAY_FB ->relayController.LT_RELAY_FB;
end LineTripSoftware.Impl;
```

Table 4: Thread Controller

```
thread Controller
features
PRASMZ_B_RqPrSd: in data port Base_Types::Boolean;
APSIMZ_B_OpLtr: in data port Base_Types::Boolean;
SSSCMZ_NX_MnSqSt: in data port Base_Types::Integer;
PCTHMZ_A_PctMo: in data port Base_Types::Boolean;
PLTTMG_B_OpLtr: in data port Base_Types::Boolean;
DHSSMG_B_OpLtr: in data port Base_Types::Boolean;
DHSSMG_B_CdLtr: in data port Base_Types::Boolean;
APSIMZ_B_EnCdLnTrpSlt: in data port Base_Types::Boolean;
PARAGP_L_CnfHpp: in data port Base_Types::Boolean;
DHSSMG_B_LtrHwOpFl: in data port Base_Types::Boolean;
DIGOMG_B_CdLtr: out data port Base_Types::Boolean;
DHWOMG_S_LtrCd: out data port Base_Types::Boolean;
DHWOMG_B_FpgaLtrOn: out data port Base_Types::Boolean;
DHWOMG_B_DcuLtrFl: out data port Base_Types::Boolean;

GPIO_OUT: requires data access GPIO_OUT {Access_Right =>read_write;};
LTRIP_EN_N: requires data access LTRIP_EN_N {Access_Right =>read_only;};
MCU_LT_ON: requires data access MCU_LT_ON {Access_Right =>read_only;};
FPGA_LTRCR: requires data access FPGA_LTRCR
{Access_Right =>write_only;};
FPGA2_LT_ON: requires data access FPGA2_LT_ON
{Access_Right =>read_only;};
LT_RELAY_FB: requires data access LT_RELAY_FB
```

```
{Access_Right =>read_only;};
properties
Dispatch_Protocol =>Periodic;
Period =>4 ms;
Priority =>1;
Compute_Execution_Time =>1 ms .. 2 ms;
Compute_Deadline =>2 ms;
end Controller;

thread implementation Controller.Impl
calls call_list: {LtrInt: subprogram LtrInt;
dcu2_line_trip: subprogram dcu2_line_trip;};
connections
B_RqPrSd_in: parameter PRASMZ_B_RqPrSd ->LtrInt.B_RqPrSd;
B_OpLtr_AppSpec_in: parameter APSIMZ_B_OpLtr ->LtrInt.B_OpLtr_AppSpec;
NX_SqSt_in: parameter SSSCMZ_NX_MnSqSt ->LtrInt.NX_SqSt;
A_PctMo_in: parameter PCTHMZ_A_PctMo ->LtrInt.A_PctMo;
B_LtrTsOpLtr_in: parameter PLTTMG_B_OpLtr ->LtrInt.B_LtrTsOpLtr;
B_OpLtr_LtrTs_in: parameter DHSSMG_B_OpLtr ->LtrInt.B_OpLtr_LtrTs;
B_CdLtr_LtrTs_in: parameter DHSSMG_B_CdLtr ->LtrInt.B_CdLtr_LtrTs;
B_EnCdLnTrpSlt_in: parameter APSIMZ_B_EnCdLnTrpSlt ->
LtrInt.B_EnCdLnTrpSlt;
L_CnfHpp_in: parameter PARAGP_L_CnfHpp ->LtrInt.L_CnfHpp;
B_ClLtr_out: parameter LtrInt.B_ClLtr ->DIGOMG_B_CdLtr;
enable_in: parameter DHSSMG_B_LtrHwOpFl ->dcu2_line_trip.enable;
act_in: parameter DIGOMG_B_CdLtr ->dcu2_line_trip.act;
fb_out: parameter dcu2_line_trip.fb ->DHWOMG_S_LtrCd;
fpga2_on_out: parameter dcu2_line_trip.fpga2_on ->DHWOMG_B_FpgaLtrOn;
fb_ne_out: parameter dcu2_line_trip.fb_ne ->DHWOMG_B_DcuLtrFl;

annex behavior_specification
{**
variables
temp0 : Baste_Types::Boolean;
states
state0 : initial complete final state;
state1 : state;
transitions
state0 -[ on dispatch ]->state1 { LtrInt(PRASMZ_B_RqPrSd,APSIMZ_B_OpLtr,
SSSCMZ_NX_MnSqSt,PCTHMZ_A_PctMo,PLTTMG_B_OpLtr,DHSSMG_B_OpLtr
DHSSMG_B_CdLtr,APSIMZ_B_EnCdLnTrpSlt,PARAGP_L_CnfHpp,
DIGOMG_B_CdLtr)};
state1 -[ ]->state0 { dcu2_line_trip(not DHSSMG_B_LtrHwOpFl,
DIGOMG_B_CdLtr,DHWOMG_S_LtrCd,DHWOMG_B_FpgaLtrOn,DHWOMG_B_DcuLtrFl)};
**};
end Controller.Impl;
```

Table 5: Subprogram LtrInt

```
subprogram LtrInt
features
B_RqPrSd: in parameter Base_Types::Boolean;
B_OpLtr_AppSpec: in parameter Base_Types::Boolean;
NX_SqSt: in parameter Base_Types::Integer;
A_PctMo: in parameter Base_Types::Boolean;
B_LtrTsOpLtr: in parameter Base_Types::Boolean;
B_OpLtr_LtrTs: in parameter Base_Types::Boolean;
B_CdLtr_LtrTs: in parameter Base_Types::Boolean;
B_EnCdLnTrpSlt: in parameter Base_Types::Boolean;
L_CnfHpp: in parameter Base_Types::Boolean;
B_ClLtr: out parameter Base_Types::Boolean;
end LtrInt;

subprogram implementation LtrInt.Impl
annex behavior_specification
{**
variables
temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8,temp9,
temp10,temp11,temp12,temp13,temp14,temp15,temp16,
temp17: Base_Types::Boolean;
states
ENTRY : initial state;
state1,state2,state3,state4,state5,state6,state7,state8,
state9,state10,state11,state12,state13,state14,state15,
state16,state17 : state;
EXIT : final state;
transitions
ENTRY -[]->state1 {temp1 := NX_SqSt >= 3};
state1 -[]->state2 {WITHIN_I(true,NX_SqSt,27,4,temp2)};
state2 -[]->state3 {temp3 := NX_SqSt >= 38};
state3 -[]->state4 {temp4 := NX_SqSt = 30};
state4 -[]->state5 {temp5 := NX_SqSt = 31};
state5 -[]->state6 {temp6 := B_RqPrSd and temp1};
state6 -[]->state7 {temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec};
state7 -[]->state8 {temp8 := temp2 or temp3};
state8 -[]->state9 {F_TRIG(temp4,temp9)};
state9 -[]->state10 {F_TRIG(B_OpLtr_AppSpec,temp10)};
state10 -[]->state11 {R_TRIG(temp8,temp11)};
state11 -[]->state12 {temp12 := temp9 or temp5};
state12 -[]->state13 {temp13 := L_CnfHpp or B_EnCdLnTrpSlt};
state13 -[]->state14 {temp14 := not(A_PctMo and B_LtrTsOpLtr)};
state14 -[]->state15 {temp15 := temp12 and temp13};
state15 -[]->state16 {temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15};
state15 -[]->state17 {RS(temp7,temp16,temp17)};
state17 -[]->EXIT {B_ClLtr := temp14 and temp17};
**};
```

```
end LtrInt.Impl;
```

Table 6: Subprogram dcu2_line_trip

```
subprogram dcu2_line_trip
features
enable: in parameter Base_Types::Boolean;
act: in parameter Base_Types::Boolean;
fb: out parameter Base_Types::Boolean;
fpga2_on: out parameter Base_Types::Boolean;
fb_ne: out parameter Base_Types::Boolean;
GPIO_OUT: requires data access Base_Types::Boolean
{Access_Right =>read_write;};
LTRIP_EN_N: requires data access Base_Types::Boolean
{Access_Right =>read_only;};
MCU_LT_ON: requires data access Base_Types::Boolean
{Access_Right =>read_only;};
FPGA_LTRCR: requires data access Base_Types::Boolean
{Access_Right =>write_only;};
FPGA2_LT_ON: requires data access Base_Types::Boolean
{Access_Right =>read_only;};
LT_RELAY_FB: requires data access Base_Types::Boolean
{Access_Right =>read_only;};
end dcu2_line_trip;

subprogram implementation dcu2_line_trip.Impl
annex behavior_specification
{**
variables
temp : Base_Types::Boolean;
btemp : Base_Types::Boolean;
states
ENTRY : initial state;
state0,state1,state2,state3,state4, state5, state6 : state;
EXIT : final state;
transitions
ENTRY -[]->state0 {temp := false; btemp := false};
state0 [2] -[enable]->state1 {GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N};
state0 [1] -[]->state1 {GPIO_OUT:= GPIO_OUT or LTRIP_EN_N};
state1 [2] -[act]->state2 {temp := temp or MCU_LT_ON};
state1 [1] -[]->state2 {};
state2 -[]->state3 {FPGA_LTRCR := temp; fpga2_on := temp and FPGA2_LT_ON;
fb := temp and LT_RELAY_FB ;};
state3 [2] -[enable and act and fpga2_on]->state4 {};
state3 [1] -[]->state5 {};
state4 [2]-[not fb]->state6 {btemp := true};
state4 [1]-[]->state6 {};
state5 [2]-[fb]->state6 {btemp:= true};
state5 [1]-[]->state6 {};
state6 -[]->EXIT {fb_ne := btemp};
**};
end dcu2_line_trip.Impl;
```

Table 7: Thread Tester

```
thread Tester
features
DCUIMG_C_LtrTs: in data port Base_Types::Boolean;
DHSSMG_B_LtrFl: in data port Base_Types::Boolean;
DHWOMG_S_LtrCd: in data port Base_Types::Boolean;
DIGIMG_S_LtrOp: in data port Base_Types::Boolean;
DCUIMG_S_DcuNtRdy: in data port Base_Types::Boolean;
PARAGP_L_LtrSvEn: in data port Base_Types::Boolean;
DHSSMG_B_OpLtr: out data port Base_Types::Boolean;
DHSSMG_B_CdLtr: out data port Base_Types::Boolean;
DHSSMG_NX_LtrSaSq: out data port Base_Types::Integer;
DHSSMG_S_LtrTsRdy: out data port Base_Types::Boolean;
DHSSMG_S_LtrOpVd: out data port Base_Types::Boolean;
properties
Dispatch_Protocol =>Periodic;
Period =>64 ms;
Priority =>2;
Compute_Execution_Time =>1 ms .. 10 ms;
Compute_Deadline =>64 ms;
end Tester;

thread implementation Tester.Impl
calls call_list: LtrTsSq: subprogram LtrTsSq;;
connections
C_LtrTs_in: parameter DCUIMG_C_LtrTs ->LtrTsSq.C_LtrTs;
B_LtrFl_in: parameter DHSSMG_B_LtrFl ->LtrTsSq.B_LtrFl;
S_LtrCd_in: parameter DHWOMG_S_LtrCd ->LtrTsSq.S_LtrCd;
S_LtrOp_in: parameter DIGIMG_S_LtrOp ->LtrTsSq.S_LtrOp;
S_DCUNtRdy_in: parameter DCUIMG_S_DcuNtRdy ->LtrTsSq.S_DCUNtRdy;
L_EnLtrSv_in: parameter PARAGP_L_LtrSvEn ->LtrTsSq.L_EnLtrSv;
B_OpLtr_out: parameter LtrTsSq.B_OpLtr ->DHSSMG_B_OpLtr;
B_CdLtr_out: parameter LtrTsSq.B_CdLtr ->DHSSMG_B_CdLtr;
NX_LtrSaSq_out: parameter LtrTsSq.NX_LtrSaSq ->DHSSMG_NX_LtrSaSq;

annex behavior_specification
{**
variables
temp1,temp2,temp3 : Baste_Types::Boolean;
states
state0 : initial complete final state;
state1,state2,state3 : state;
transitions
state0 -[ on dispatch ]->state1 {LtrTsSq(DCUIMG_C_LtrTs,DHSSMG_B_LtrFl,
DHWOMG_S_LtrCd,DIGIMG_S_LtrOp,DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,
DHSSMG_B_OpLtr,DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)};
state1 -[ ]->state2 {temp3 := temp1 and temp2};
state2 -[ ]->state3 {SR(temp1,false,DHSSMG_S_LtrTsRdy)};
state3 -[ ]->stateo {SR(temp3,false,DHSSMG_S_LtrOpVd)};
**};
end Tester.Impl;
```

Table 8: Subprogram LtrTsSq

```
subprogram LtrTsSq
features
C_LtrTs: in parameter Base_Types::Boolean;
B_LtrFl: in parameter Base_Types::Boolean;
S_LtrCd: in parameter Base_Types::Boolean;
S_LtrOp: in parameter Base_Types::Boolean;
S_DCUNtRdy: in parameter Base_Types::Boolean;
L_EnLtrSv: in parameter Base_Types::Boolean;
B_OpLtr: out parameter Base_Types::Boolean;
B_CdLtr: out parameter Base_Types::Boolean;
NX_LtrSaSq: out parameter Base_Types::Integer;
A_LtrTs: out parameter Base_Types::Boolean;
A_LtrOpVd: out parameter Base_Types::Boolean;
state_LtrTsSq : requires data access Base_Types::String
{Access_Right =>read_write;};
Dy : requires data access Base_types::Boolean {Access_Right =>read_write;};
end LtrTsSq;

subprogram implementation LtrTsSq.Impl
annex behavior_specification
{**
variables
states
CheckState : initial state;
Start,OpenLtr1,CloseLtr,OpenLtr2,Ready : state;
Return : final state;
transitions
CheckState [4]-[state_LtrTsSq = Start]->Start {NX_LtrSaSq = 0};
CheckState [3]-[state_LtrTsSq = OpenLtr1]->OpenLtr1 {NX_LtrSaSq = 1};
CheckState [2]-[state_LtrTsSq = CloseLtr]->CloseLtr {NX_LtrSaSq = 2};
CheckState [1]-[state_LtrTsSq = OpenLtr2]->OpenLtr2 {NX_LtrSaSq = 3};
CheckState [0]-[state_LtrTsSq = Ready]->Ready {NX_LtrSaSq = 4};
Start [1]-[(not L_EnLtrSv) and C_LtrTs]->Return {state_LtrTsSq := Ready};
Start [0]-[L_EnLtrSv and C_LtrTs]->Return {B_OpLtr := true; state_LtrTsSq :=
OpenLtr1};
OpenLtr1 [1]-[B_LtrFl]->Return {state_LtrTsSq := Start};
OpenLtr1 [0]-[not S_LtrCd]->Return {B_CdLtr := true; Delay(512ms); Dy := true;
state_LtrTsSq := CloseLtr};
CloseLtr [1]-[B_LtrFl]->Return {state_LtrTsSq := Start};
CloseLtr [0]-[S_LtrCd and Dy]->Return {B_OpLtr := true; state_LtrTsSq :=
OpenLtr2};
OpenLtr2 [1]-[B_LtrFl]->Return {state_LtrTsSq := Start};
OpenLtr2 [0]-[not S_LtrCd]->Return {A_LtrTs := true, state_LtrTsSq := Ready};
Ready -[S_DCUNtRdy]->Return {state_LtrTsSq := Start};
**};
end LtrTsSq.Impl;
```

6 Architecture flow graphs

AADL models essentially express control and data flows through the architecture that define the architectural behavior. *Control flows* refer to the orders in which software components and their instructions are executed. *Data flows* refer to the orders in which data variables (including interfaces and connections) are assigned by a component and subsequently used, possibly by a different component. In AADL, the flows are dependent on how components transfer control and data through their interfaces – in conjunction with scheduling properties and communication protocols of shared resources. The possible interactions among components are represented by: *port connections*, *data access connections*, *subprogram calls*, and *parameter connections.* A *port connection* represents a transfer of unqueued data, queued control, or queued control with associated data (messages), depending on the type of interconnected interfaces (data port, event port, or event data port). A *subprogram call* represents a transfer of control whereas a *parameter connection* and a *data access connection* represent a transfer of data. In addition, components may have behavioral models (BMs) describing their logical executions. A BM therefore both yields internal flows of a component (between input and output interfaces) and refines flows to other components as it operates on inter-component connections. All these constructs together determines the control and data flows of an AADL model. In order to verify consistency, completeness, and correctness of the system architecture (according to the criteria presented in Section 7), the control and data flows must be analyzed. A common approach to control and data flow analysis is to extract the flows into a directed graph. In this section, we define a type of directed graph that has been developed for the representation of AADL flows, referred to as architecture flow graphs (AFGs).

Definition 3 An architecture flow graph $AFG(M) = \langle V, A \rangle$ of an AADL model M is a directed graph of a set of vertices $V = \{v \mid v \in EXPR \cup \langle \text{``}ENTRY\text{''}, comp_i \rangle \cup \langle \text{``}REENTRY\text{''}, comp_i \rangle \cup \langle \text{``}EXIT\text{''}, comp_i \rangle\}$ representing AADL expressions and scheduling states, and a set of directed arcs $A \subseteq V \times V$ describing how control and data flow through the vertices. The possible AADL expressions are determined by the abstract syntax in Definition 1. v of an arc $\langle v, v' \rangle \in A$ is the tail and v' is the head, denoting that the flow is directed from v to v', i.e., an execution of v is followed by an execution of v' if it is a control flow or a write of a variable in v is followed by a read of the variable in v' if it is a data flow. The denotation $\langle v, v' \rangle$ is used interchangeably with $v \rightarrow v'$. A vertex $v = \langle expr \rangle$ and an arc $\langle v, v' \rangle$ may be attributed with a set of AADL properties: $\langle v, \{prop_1, prop_2, \ldots, prop_n\} \rangle$ and $\langle \langle v, v' \rangle, \{prop_1, prop_2, \ldots, prop_n\} \rangle$. Furthermore, an arc has one of the following labels to distinguish different types of control and data flows:

- $\langle v, v' \rangle_c$: represents a component-internal control flow. A vertex v is called a **direct predecessor** of v' and v' a **direct successor** of v iff $\langle v, v' \rangle_c \in A$. Let $outdegree(v)$ be a function mapping the number of direct successors of v and $indegree(v)$ the number of direct predecessors. A vertex can have zero, one, or two direct successors. A vertex v with two direct successors represents a so called **control expression** constituting a Boolean condition. The two outgoing arcs of v are attributed with $\langle v, v_x\rangle_{cT}$ for $true$ and $\langle v, v_y\rangle_{cF}$ for $false$ and correspond to the control flow in response to the condition evaluation.

- $\langle v, v'\rangle_{c-inter}$: represents an interaction-based control flow due to the activation of a communication protocol. The execution of v' coincides with the execution of v according to the protocol.
- $\langle v, v'\rangle_{call}$: represents an inter-component control flow due to a raised event or a call.
- $\langle v, v'\rangle_{d}$: represents a component-internal data flow.
- $\langle v, v'\rangle_{d-in}$: represents an inter-component data flow due to a data passing by value or by reference protocol. The arc indicates data flowing from an output to an input interface. If used together with a function call, the arc indicates the data flowing from an argument to the corresponding subprogram input parameter.
- $\langle v, v'\rangle_{d-out}$: represents an inter-component data flow due to a data passing by value protocol activated to return from a call. The arc indicates data flowing from an output parameter of a subprogram to the variable assigned by the call.

The $\langle$"$ENTRY$"$, comp\rangle$ vertex represents the point of the component $comp$ through which control enters and $outdegree(\langle$"$ENTRY$"$, comp\rangle) = 1$.
A $\langle$"$REENTRY$"$, comp\rangle$ vertex represents a point of the component $comp$ through which control suspends, and reenters when the component has been reactivated/dispatched after the suspension and $outdegree(\langle$"$REENTRY$"$, comp\rangle) = 1$. A component may have any number of reenter vertices. The $\langle$"$EXIT$"$, comp\rangle$ vertex represents the point of the component $comp$ through which control exits and $outdegree(\langle$"$EXIT$"$, comp\rangle) = 0$. A **control path** $P = v_1 \rightarrow_c v_2 \rightarrow_c \cdots \rightarrow_c v_n$ of a flow graph is a sequence of n vertices such that $n \geq 2$ and for $i = 1, 2, \ldots, n-1$, $\langle v_i, v_{i+1}\rangle_c \in A$. A control path is called a **basic block** if $v_1 \neq ENTRY \cup REENTRY$; $outdegree(v_1) = 1$; for $n > 2$ and for $i = 2, 3 \ldots, n-1$, $indegree(v_i) = 1$, $outdegree(v_i) = 1$, and $indegree(v_n) = 1$; and $outdegree(v_n) \geq 2$.

6.1 Architecture flow graph generation

An AADL model is transformed into an AFG through three operations. The result of applying these operations to LTRI is shown in Fig. 7. The first operation is to generate an individual control flow graph (CFG) for each AADL component representing a, possibly concurrent, unit of sequential execution, i.e., for each thread and subprogam component. This is achieved by analyzing each thread and subprogram component in isolation to find all possible control flows of type $\langle v, v'\rangle_c$. The second operation is to compute all component-internal data flows through def-use pair analysis and annotate them to the CFGs, resulting in graphs referred to as program flow graphs (PFGs). Again, the analysis is performed on each component in isolation. The third and final step is to integrate the set of PFGs according to the modeled component connections, resulting in the AFG of the AADL model.

AFG generation: step 1 The individual control flow of each component is entirely determined by the BM of the component. A BM essentially consists of state transitions. A state transition $s \xrightarrow{pri,g,act} s'$, from a state s to a successor state s', has a priority $pri \in \mathbb{N}$, a (possibly empty) set of predicate guards g, and a (possibly empty) sequence of actions act. Each state transition corresponds to a fixed execution order of operations: the guard of the transition is first computed and, if evaluated to the Boolean value $true$, the sequence of actions is executed. Thus, BM guards and actions are the executable operations and yield the vertices of the CFG. Consequently, each transition $s \xrightarrow{pri,g,act} s'$, where $act = action_1; action_2; ...; action_n$ is a sequence of n actions, maps to a CFG construct of one vertex $v_1 = g$ representing the guard of the state transition; a basic block of n vertices $v_2 = action_1, v_3 = action_2, \ldots, v_{n+1} = action_n$ representing the actions of the state transition; and n arcs $\langle v_1, v_2\rangle_{cT}$, $\langle v_2, v_3\rangle_c$, $\ldots$, $\langle v_n, v_{n+1}\rangle_c$ representing the control flow through the executable operations. Note that the arc from the guard to the first action is attributed with a "T". Let $stateTrToV : TR \to \mathcal{P}(V)$ be a function mapping a state transition to a set of vertices and $stateTrToA : TR \to \mathcal{P}(A)$ to a set of arcs in this fashion, i.e., such that $stateTrToV(s \xrightarrow{pri,g,act} s') = \{v_1, v_2, v_3, \ldots, v_{n+1}\}$ and $stateTrToA(s \xrightarrow{pri,g,act} s') = \{\langle v_1, v_2\rangle_T, \langle v_2, v_3\rangle, \ldots, \langle v_n, v_{n+1}\rangle\}$ where $v_1 = g, v_2 = action_1, v_3 = action_2, \ldots, v_{n+1} = action_n$.

The fixed execution order of operations is repeated throughout the BM (until a final state is reached), as shown in Fig. 2. If g_1 is evaluated to the Boolean value $true$, act_1 is executed, resulting in the arrival of a new state s_i whereupon the transition going out from s_i with the highest priority is executed according to the fixed order. On the other hand, if g_1 is evaluated to $false$, another state transition going out from s_1 with the (next) highest priority is executed in the fixed order (in this case, the transition with priority pri_2 is next in line). Let $guardVertex : TR \to V$ be a function mapping a state transition $s \xrightarrow{pri,g,act} s'$ to the vertex v_x representing the guard of the state transition. Let $lastActionVertex : TR \to V$ be a function mapping a state transition $s \xrightarrow{pri,g,act} s'$ to the vertex v_x representing the last action of the state transition. Let $guardVertexPrio : S \times \mathbb{N} \to V \cup \{false\}$ be a function mapping a state s to the vertex v_x representing the guard of the state transition going out from v and with the highest priority, or with the highest priority but less than n if a natural number is given as argument, or $false$ if there exist no such vertex. In the case of an evaluation of a guard to $true$, the control flow from the last action of the transition to the guard of the second transition with the highest priority is simply represented by an arc $\langle lastActionVertex(s \xrightarrow{pri,g,act} s'), guardVertexPrio(s')\rangle$. In case of an evaluation to $false$, the control flow to the guard with the (next) highest priority is represented by an ($false$-)arc $\langle guardVertex(s \xrightarrow{pri,g,act} s'), guardVertexPrio(s, pri)\rangle_F$.

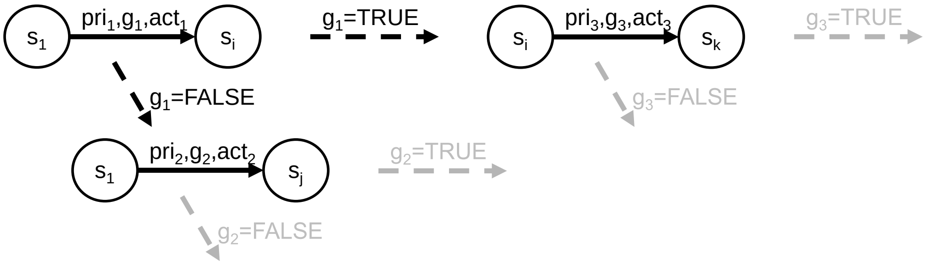


Fig. 2. Illustration of behavioral model semantics. Assume s_1 is the initial state and $pri_1 > pri_2$.

It should be mentioned that actions may be of **if**, **while** and **for** constructs. Such an action comprises multiple vertices where control can leave the construct (action) from several vertices rather than a single one. In such constructs are predicates and nested actions also represented through distinguished vertices. Assume that v_x represents a control predicate expression of a loop or conditional, and v_y represents an action expression immediately nested within the loop or condition. If v_x is the predicate of a conditional expression the arc $\langle v_x, v_y\rangle$ is labeled with "T" or "F" according to weather v_y exists in the **then** branch or **else** branch. If v_x is the predicate of a while- or for-loop, the arc $\langle v_x, v_y\rangle$ is labeled with "T". In case a state transition consists of an action sequence where the last action consists of an **if** construct, each (nested) action ending the control flow of the construct, including the current state transition, must be connected to the subsequent transition guard vertex, $REENTRY$ vertex, or $EXIT$ vertex by an arc.

The order in which transitions are executed is determined by the possible orders states can be visited through state transitions (also known as the possible *paths* in the BM) and by the priorities of the state transitions, as shown in Fig. 3. However, a BM has different types of states that must be taken into consideration when building the CFG. A BM of a subprogram component has: one initial state representing the starting point of a call; zero or more intermediate execution states representing the logical execution between start and completion of a call; and one final state representing the completion of a call. Thus, the initial state of a subprogram maps to an $ENTRY$ vertex whereas the final state maps to an $EXIT$ vertex. A BM of a thread component, on the other hand, has: one initial state representing the state of the thread before it is initialized (halted); zero or more intermediate execution states representing the initialization steps (such as checking correctness of initial values of input and output ports) of the thread between the initial state and one, first, complete state (any path from the initial state will reach the same complete state before any other complete state); one or more complete states representing that the thread has suspended itself and is awaiting dispatch/reactivation (the first complete state reached from an initial state does also represent completion of initialization the first time it is reached); zero or more intermediate execution states representing logical execution between dispatches, that is, from and back to a complete state or between complete states; and one final state representing completion of finalization. Moreover, a state may be of a combination of these types. For example, a complete state may also be a final state.

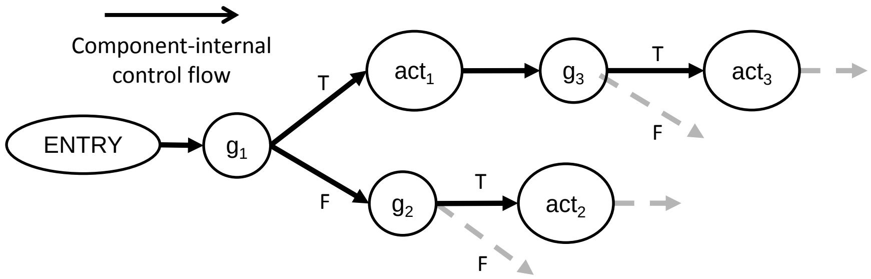


Fig. 3. The control flow graph of the behavioral model example in Figure 2.

Execution of a subprogram component is triggered by incoming calls where the transition out from the initial state with the highest priority (if several) and with valid execute conditions is executed. A thread component, on the other hand, must first be initialized by an initialize action triggered when the process containing the thread is completely loaded into its virtual address space before it can be executed. An initialize action triggers the transition out from the initial state eventually leading to one, first, complete state. A state transition to a complete state means that the thread is calling an "await dispatch" run-time service, whereupon the thread is suspended after the action of the state transition has been executed. A dispatch of the thread component is triggered according to the dispatch conditions of the transitions out from the current complete state in conjunction with the specified scheduling properties. Dispatches of a periodic thread are solely triggered by a clock according to the specified time interval (period). In this case, the dispatch conditions (guards of transitions out from complete states) are left empty or labeled "on dispatch". Dispatches of aperiodic, sporadic, timed, and hybrid threads are, in essence, triggered by the arrival of an event or a remote subprogram call arriving to a provides subprogram access feature of the thread. By default, any arrival of event or subprogram call triggers a dispatch where dispatch conditions restrict the number of triggers if modeled.

In either case, an input-compute-output model of execution is triggered. Input on in ports is frozen at the time of dispatch, where input from each port connection is read and assigned to a corresponding port variable which value (by default) is not affected by new arrivals for the remainder of the current dispatch. Output on out ports is transmitted through the connections at the time of completion, deadline or at specific output times according to an $Output\_Time$ property. For simplicity, we assume that the output is transmitted at completion. A state transition to a final state means that the thread completes and is calling a “finalize” run-time service, whereupon the thread terminates after the action of the state transition has been executed. Consequently, a BM of a thread component, in contrast to a BM of a subprogram component, expresses state transitions which are not relevant to the logical execution (such as initialization transitions).

Algorithm 1 Algorithm for generating a control flow graph

```
Input: $comp_i.BM = \langle S, s_o, CPL, FIL, VAR, TR \rangle$ and $TR_{rel} \subseteq TR$
Output: $CFG(comp_i) = \langle V, A \rangle$
$V \leftarrow \emptyset \cup \{\langle "ENTRY", comp_i \rangle, \langle "EXIT", comp_i \rangle\}$
$A \leftarrow \emptyset$
for all $s \xrightarrow{pri,g,act} s' \in TR_{rel}$ do    ▷ generate vertices and arcs for each relevant transition
    $V \leftarrow V \cup stateTrToV(s \xrightarrow{pri,g,act} s')$
    $A \leftarrow A \cup stateTrToA(s \xrightarrow{pri,g,act} s')$
end for
$A \leftarrow A \cup \{\langle\langle "ENTRY", comp_i \rangle, guardVertexPrio(firstState(comp_i.BM))\rangle$    ▷ Generate the arc representing control flow from the $ENTRY$ vertex to the guard vertex with highest priority
for all $cpl_j \in CPL$ do    ▷ generate possible $REENTRY$ vertices
    if $\exists s \xrightarrow{pri,g,act} s' \in TR_{rel}[s' = cpl_j]$ then
        $V \leftarrow V \cup \{"REENTRY_j"\}$
        $A \leftarrow A \cup \{\langle "REENTRY_j", guardVertexPrio(s') \rangle\}$    ▷ Any control flow to $"REENTRY_j"$ will successively flow to $guardVertexPrio(s')$
    end if
end for
for all $s \xrightarrow{pri,g,act} s' \in TR_{rel}$ do    ▷ Generate arcs to connect each transition representation to the subsequent guard, complete state (reentry) or final state representation
    if $s' \in CPL)$ then
        $A \leftarrow A \cup \{\langle lastActionVertex(s \xrightarrow{pri,g,act} s'), CPLStateVertex(s') \rangle\}$
    else if $s' \in FIL$ then
        $A \leftarrow A \cup \{\langle lastActionVertex(s \xrightarrow{pri,g,act} s'), \langle "EXIT", comp_i \rangle \rangle\}$
    else$A \leftarrow A \cup \{\langle lastActionVertex(s \xrightarrow{pri,g,act} s'), guardVertexPrio(s') \rangle\}$
    end if
    if $guardVertexPrio(s, pri)$ then    ▷ generate a false arc if a subsequent guard exists
        $A \leftarrow A \cup \{\langle guardVertex(s \xrightarrow{pri,g,act} s'), guardVertexPrio(s, pri) \rangle_F\}$
    end if
end for
return $\langle V, A \rangle$
```

The relevant set of state transitions includes each transition that exists on every path from every complete state in the BM. Each of these are either from a complete state to a complete state, execution state, or a final state; or from an execution state to a complete state, an execution state, or a final state. Thus, the first complete state reached from an initial state maps to an $ENTRY$ vertex and any subsequently reachable complete states, including the one first reached from an initial state, maps to a $REENTRY$ vertex. The final state maps to an $EXIT$ vertex. Let $firstState : \mathcal{P}(S) \times S \times \mathcal{P}(S) \times \mathcal{P}(S) \times \mathcal{P}(VAR) \times \mathcal{P}(TR) \rightarrow CPL$ be a function mapping a BM to the initial state if the BM is of a subprogram component, or the first complete state reachable from the initial state if the BM is of a thread component. Let $CPLStateVertex : CPL \rightarrow V$ be a function mapping a complete state to its corresponding $REENTRY$ vertex. Note that the first complete state is mapped to both an $ENTRY$ and an $REENTRY$ vertex if there exist a transition back to the state. However, there exist only one distinguished $ENTRY$ vertex, so there is no need to define a function to retrieve it. Let $TR_{rel} \subseteq TR$ be the relevant set of state transitions of a BM. If the BM is of a subprogram component, then $TR_{rel} = TR$. The transformation from a $comp_i.BM$ to the corresponding $CFG(comp_i.BM)$ is then be calculated according to Algorithm 1. As examples, the result of applying the algorithm on subprogram *LtrTsSq.Impl* (8) is presented in Figure 5.

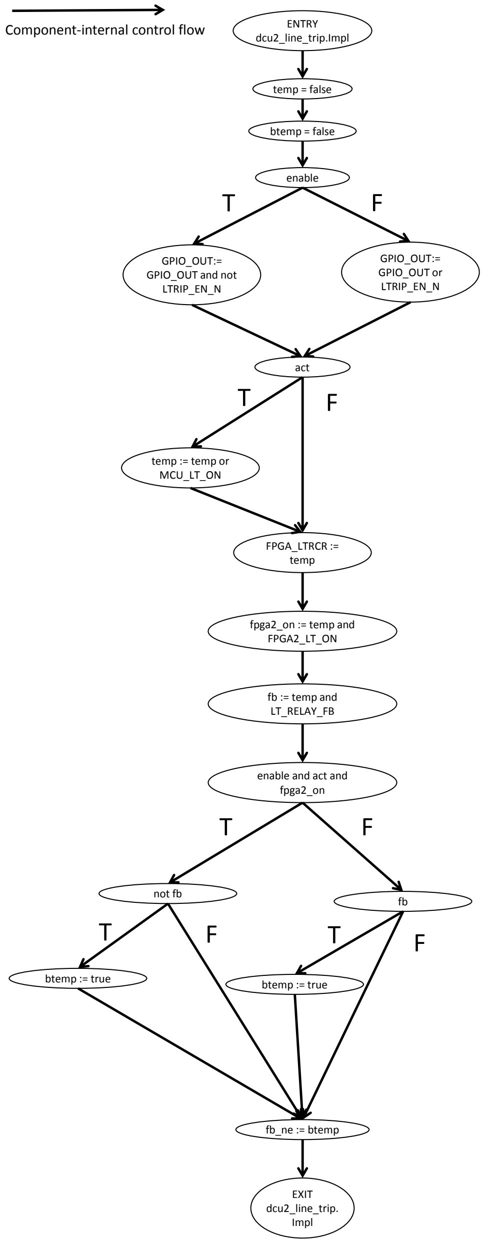

Component-internal control flow
ENTRY dcu2_line_trip.Impl
temp = false
btemp = false
enable
T
F
GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N
GPIO_OUT:= GPIO_OUT or LTRIP_EN_N
act
T
F
temp := temp or MCU_LT_ON
FPGA_LTRCR := temp
fpga2_on := temp and FPGA2_LT_ON
fb := temp and LT_RELAY_FB
enable and act and fpga2_on
T
F
not fb
fb
T
F
T
F
btemp := true
btemp := true
fb_ne := btemp
EXIT dcu2_line_trip. Impl

Fig. 4. The control flow graph of *dcu2_line_trip.Impl* (Table 6).

AFG generation: step 2 The second operation is to compute the component-internal data flows for each component and annotate them to the CFGs to produce the PFGs. Such flows are computed by performing definition-use pairs analysis of each CFG. Assume that V_{def} is the set of vertices that defines/assigns variable var_i, and V_{use} is the set of vertices that uses/reads var_i. A component-internal data flow is defined as:

Definition 4 For each pair of vertices $\langle v_x, v_y \rangle \in V_{def} \times V_{use}$ such that there exists a control path $P = v_1 \rightarrow_c v_2 \rightarrow_c \cdots \rightarrow_c v_n$ from v_x to v_y (where $v_1 = v_x$ and $v_n = v_y$) and any other vertex v_z in P does not define/assign var_i, i.e., $v_z \neq V_{def}$ for $z = 2, 3, \ldots, n-1$, there exist a component-internal data flow $\langle v_x, v_y \rangle_d$.

If the principle is applied to all variables for each CFG, all the possible component-internal data flows are generated. As an example, the PFG of
dcu2_line_trip.Impl is presented in Figure 5.

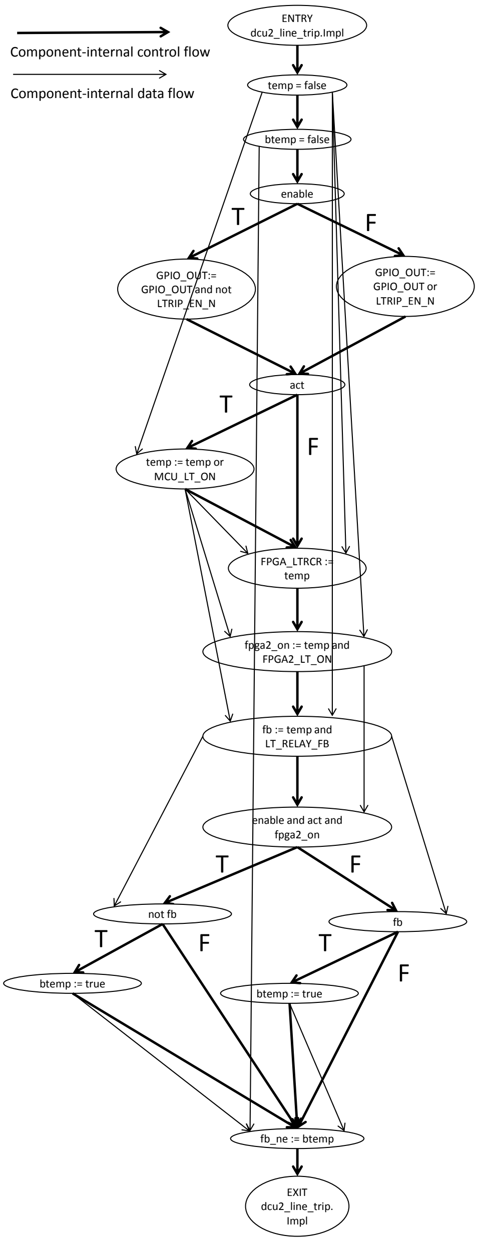


Fig. 5. The program flow graph of *dcu2_line_trip.Impl*.

AFG generation: step 3 The third and final operation is to integrate the CFGs according to the component interactions to produce the AFG. Components of AADL models may both transfer data and control through interfaces, where ports and parameters are accessible as variables. Control may be transferred to threads through event ports and event data ports that are included in dispatch conditions, and to subprograms through subprogram calls. In either case, control is transferred to the entry point (including reentry points of threads) of the target component. Data may be transferred through data ports, event data ports, subprogram parameters, and shared data components.

Following the default input-compute-output semantics of AADL, each thread dispatch and subprogram invocation includes assignments to input ports and parameters if the component has such connections. In addition, each thread-execution completion and subprogram return includes transmission of output data on output ports and parameters if such connections exist. Consequently, input assignments coincide with entry vertices of subprograms, and with dispatch condition vertices (extensions of entry and reentry vertices to explicitly represent dispatch conditions) of entry and reentry vertices of threads. Output assignments, on the other hand, coincide with exit vertices of subprograms whereupon control is returned to the caller. In threads, output assignments coincide with exit and reentry vertices of threads as both represent a completion of the current dispatch when entered.

These inter-component flows are explicitly represented through four distinguished types of vertices (similarly to system dependence graphs defined by Horwitz et al. [20]): (1) *actual-in* vertices on the form $connection = out\_interface$ representing assignments that copy the values of output interfaces to connections; (2) *formal-in* vertices on the form $in\_interface = connection$ representing assignments that copy the values of connections to input interfaces; (3) *formal-out* vertices on the form $connection = out\_parameter$ representing assignments that copy return values of a callee's output parameters to connections; and (4) *actual-out* vertices on the form $in\_interface = connection$ representing assignments that copy return values of connections to variables assigned by the call.

By using these vertices, inter-component flows are added according to Rule 1-5. Each vertex of the PFG that operates on an interface must subsequently be connected to the corresponding distinguished vertex, as illustrated in Figure 6, to finalize the AFG.

Rule 1 For each data port (dp) connection $c_x(dp_y, dp_z)$, where dp_y is the source and dp_z is the sink, generate an actual-in vertex $c_x = dp_y$ and a formal-in vertex $dp_z = c_x$ connected through a data-in arc $\langle c_x = dp_y, dp_z = c_x \rangle_{d-in}$. To conform to the transmission of output semantics, an interaction-based control flow arc shall be created from each reentry vertex and the exit vertex of the sending thread to the actual-in vertex. Similarly, dispatch conditions of entry and reentry vertices of the receiving thread must have an interaction-based control flow to the formal-in vertex.

Rule 2 For each event port (ep) connection $c_x(ep_y, ep_z)$, generate an actual-in vertex $c_x = ep_y$ and a formal-in vertex $ep_z = c_x$ connected through a call/event

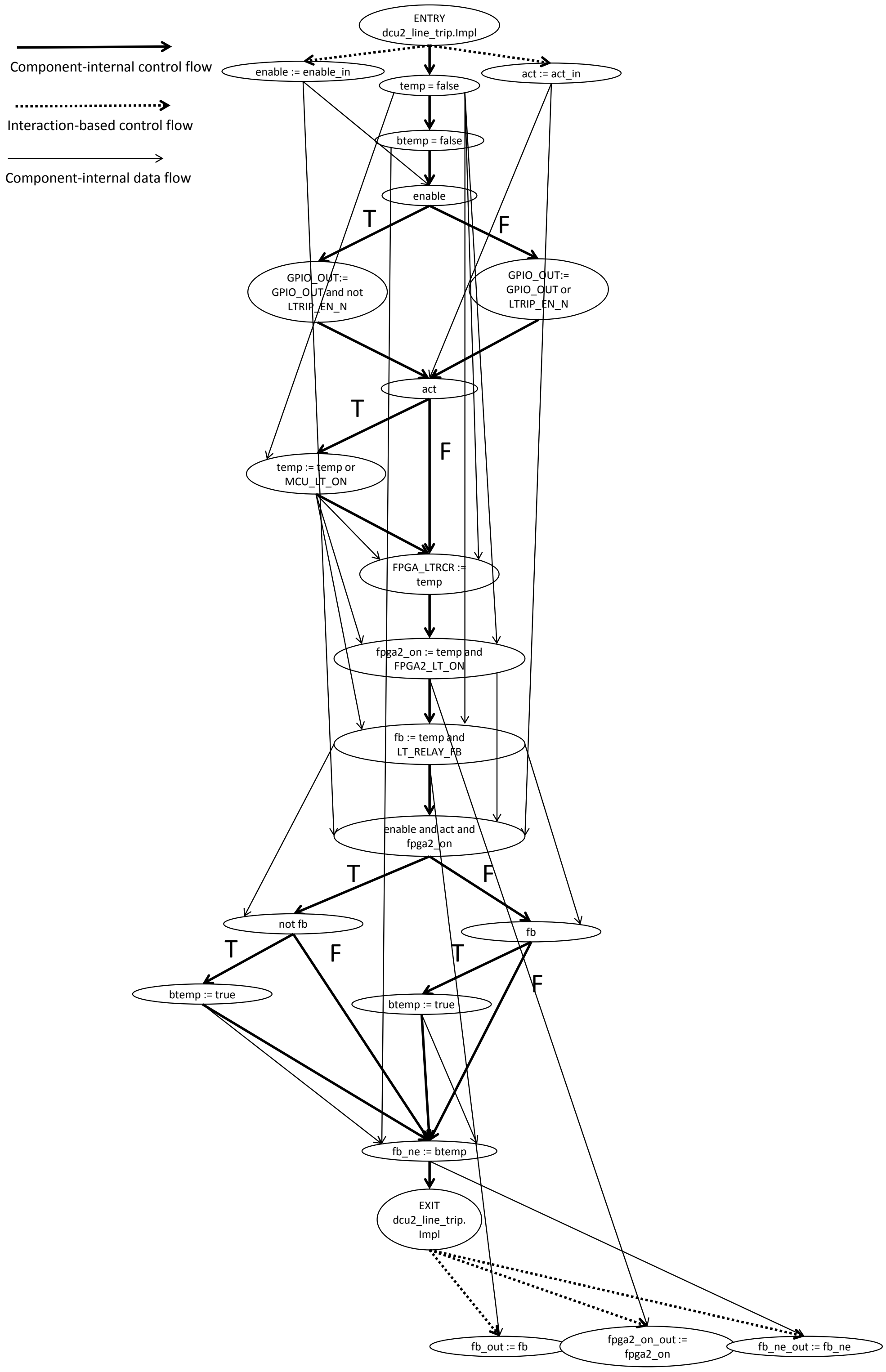


Fig. 6. The program flow graph of *dcu2_line_trip.Impl* with formal-in and formal-out vertices.

and a data-in arc. c_x and ep_y are assumed to be of Boolean type and ep_z is a Boolean-typed array implementing a FIFO queue. The index range is defined by a *Queue_Size* property which by default is one. It is assumed that ep_y is set to $false$ immediately after an execution of the actual-in vertex. From each reentry vertex and the exit vertex of the sending thread, an interaction-based control flow arc shall be connected to the actual-in vertex. Finally, each dispatch condition of each entry and reentry vertex of the receiving thread have an interaction-based control flow arc to the formal-in vertex.

Rule 3 For each event data port (edp) connection $c_x(edp_y, edp_z)$, generate a construct similar to event port connections, however, where the variables and array are assumed to be of a complex data type composed of a boolean type associated with some data type.

Rule 4 For each subprogram (*sub*) call $call(sub_x)$, generate a call arc $\langle call(sub_x), \langle \text{``}ENTRY\text{''}, sub_x\rangle\rangle$. If any parameter connections to the callee are associated with the call, they are represented similarly to data port connections, however, where interaction-based control flows to actual-in vertices are flowing from the call vertex rather than reentry and exit vertices. If any (return) parameter connections to the caller is associated with the call, it is represented by a formal-out and an actual-out vertex connected through a data-out arc. The exit vertex of the sending subprogram has an interaction-based control flow arc to the formal-out vertex whereas the call vertex has such an arc to the actual-out vertex.

Rule 5 *Data access connections* to a common data component $data_x$ may represent transfers of data (by reference) if there exist both write-right and read-right access connections. In case this condition holds, and to represent the possible combinations of data flows with respect to concurrency, the data flow between the component $comp_y$ with write-right access and the component $comp_z$ with read-right access is represented through an actual-in vertex $comp_y - comp_z = data_x$, representing the write-right connection, and an inverting formal-in vertex $data_x = comp_y - comp_z$, representing the read-right connection, connected through a data-in arc. Given that a thread or a subprogram gets the data source upon dispatch and releases it upon a completion, each reentry vertex and the exit vertex of the sending thread, or the exit vertex of the sending subprogram, have interaction-based control arcs to the actual-in vertex. On the other hand, the dispatch conditions of entry and reentry vertices of the receiving thread, or the entry vertex of the receiving subprogram, have interaction-based control arcs to the formal-in vertex.

6.2 Results

The result of applying these steps to LTRIS is presented in Figure 7 and a detailed representation of each component part is presented in Figures 8–13.

Fig. 7. The architecture flow graph of LTRIS

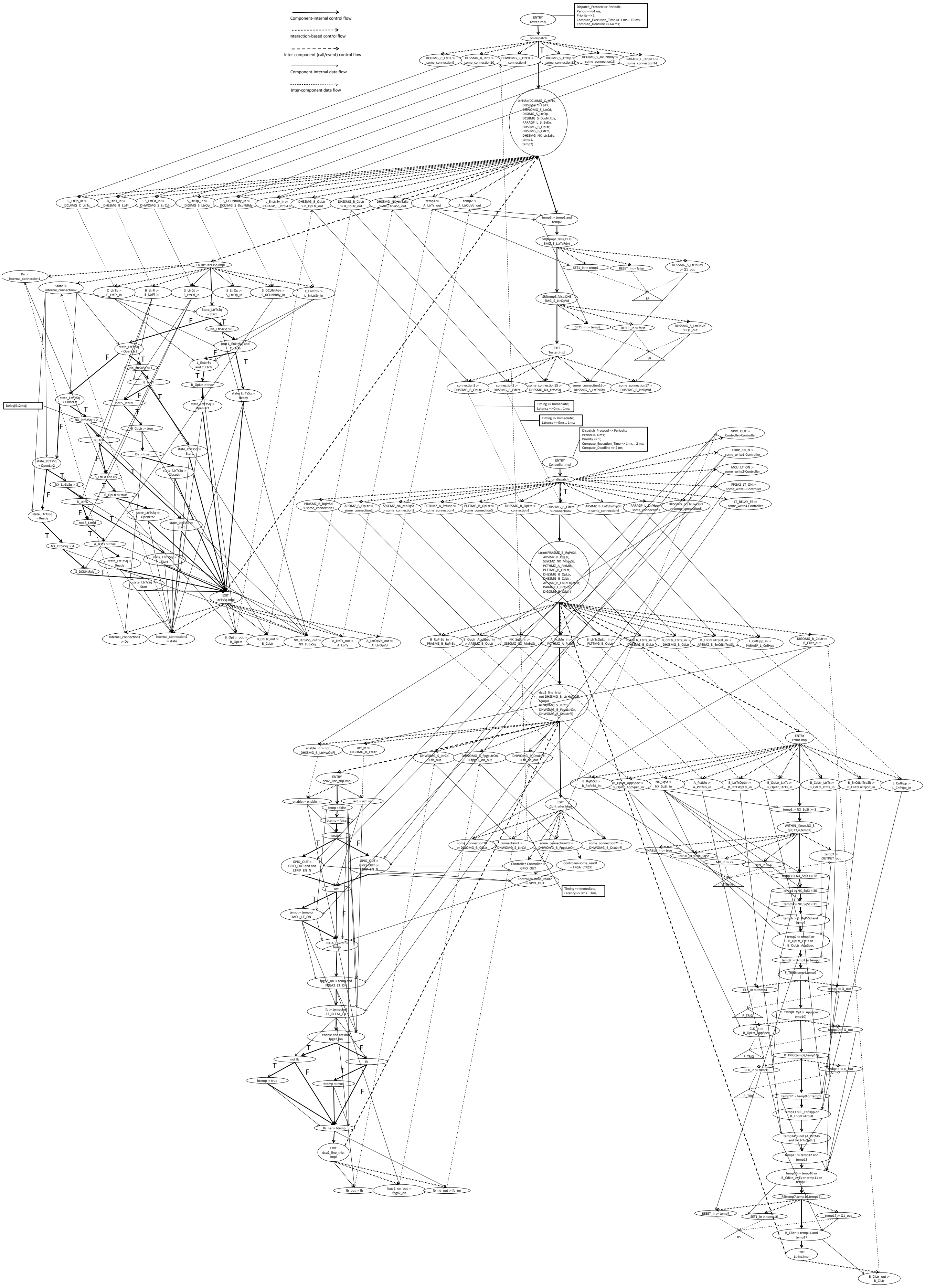

Component-internal control flow
Interaction-based control flow
Inter-component (call/event) control flow
Component-internal data flow
Inter-component data flow
ENTRY Tester.Impl
Dispatch_Protocol => Periodic;
Period => 64 ms;
Priority => 2;
Compute_Execution_Time => 1 ms .. 10 ms;
Compute_Deadline => 64 ms;
on dispatch
T
DCUIMG_C_LtrTs := some_connection9
DHSSMG_B_LtrFl := some_connection10
DHWOMG_S_LtrCd := connection3
DIGIMG_S_LtrOp := some_connection12
DCUIMG_S_DcuNtRdy := some_connection13
PARAGP_L_LtrSvEn := some_connection14
LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl, DHWOMG_S_LtrCd, DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy, PARAGP_L_LtrSvEn, DHSSMG_B_OpLtr, DHSSMG_B_CdLtr, DHSSMG_NX_LtrSaSq, temp1, temp2)
temp3 := temp1 and temp2
SET1_in := temp1
RESET_in := false
DHSSMG_S_LtrTsRdy := Q1_out
SR
SET1_in := temp3
DHSSMG_S_LtrOpVd := Q1_out
EXIT Tester.Impl
ENTRY LtrTsSq.Impl
Delay(512ms);
EXIT LtrTsSq.Impl
Timing => Immediate;
Latency => 0ms .. 1ms;
Timing => Immediate;
Latency => 0ms .. 1ms;
ENTRY Controller.Impl
Dispatch_Protocol => Periodic;
Period => 4 ms;
Priority => 1;
Compute_Execution_Time => 1 ms .. 2 ms;
Compute_Deadline => 2 ms;
on dispatch
GPIO_OUT := Controller-Controller
LTRIP_EN_N := some_write1-Controller
MCU_LT_ON := some_write2-Controller
FPGA2_LT_ON := some_write3-Controller
LT_RELAY_FB := some_write4-Controller
EXIT Controller.Impl
ENTRY dcu2_line_trip.Impl
EXIT dcu2_line_trip.Impl
fb_out := fb
fpga2_on_out := fpga2_on
fb_ne_out := fb_ne
Controller-Controller := GPIO_OUT
Controller-some_read1 := FPGA_LTRCR
Controller-some_read2 := GPIO_OUT
Timing => Immediate;
Latency => 0ms .. 3ms;
ENTRY Ltrint.Impl
temp1 := NX_SqSt != 3
F_TRIG
B_TRIG
RS
EXIT Ltrint.Impl
B_OlLtr_out := B_OlLtr
T
F

Fig. 8. The AFG of LTRIS – part 1.

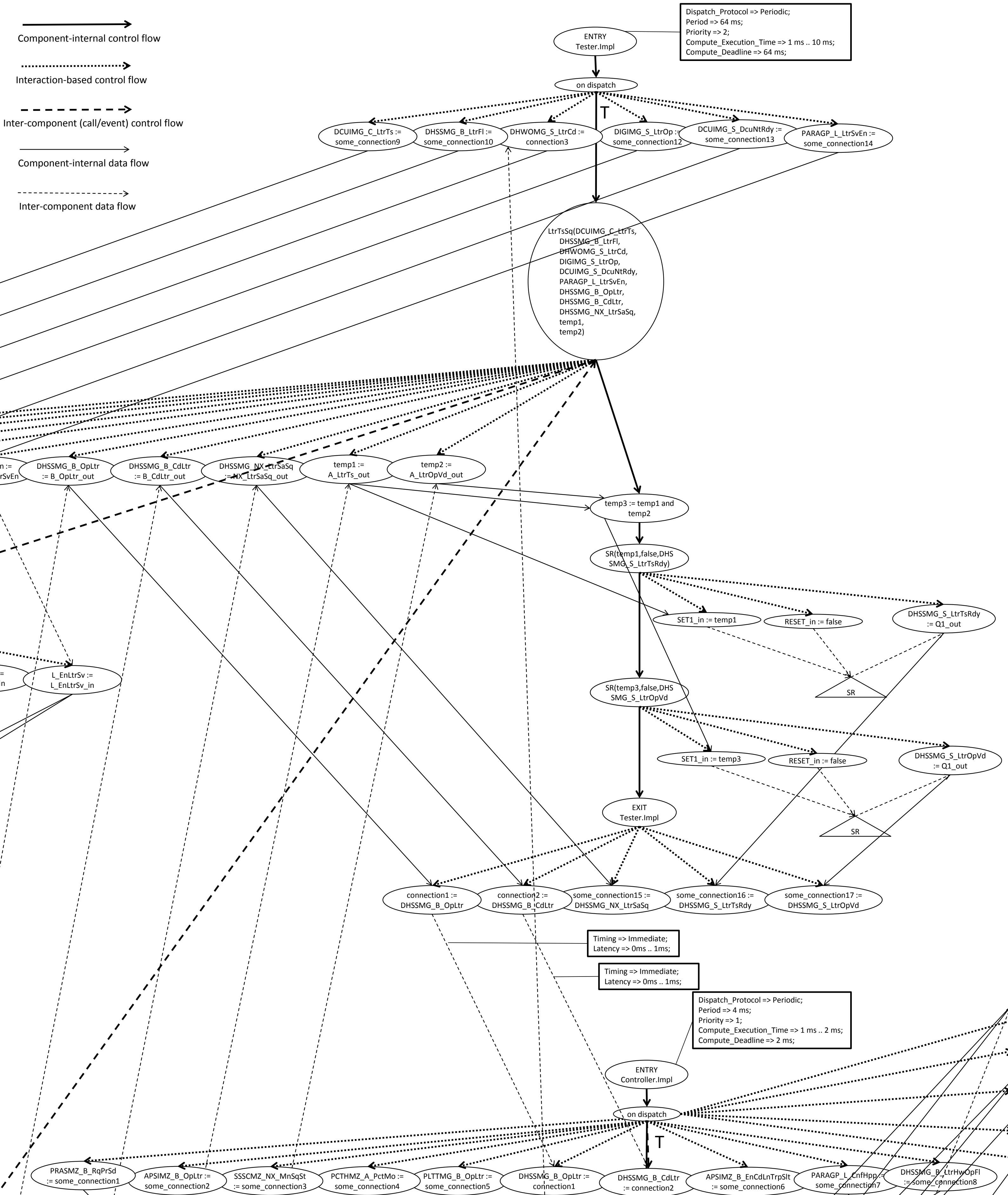

Component-internal control flow
Interaction-based control flow
Inter-component (call/event) control flow
Component-internal data flow
Inter-component data flow
ENTRY Tester.Impl
Dispatch_Protocol => Periodic;
Period => 64 ms;
Priority => 2;
Compute_Execution_Time => 1 ms .. 10 ms;
Compute_Deadline => 64 ms;
on dispatch
T
DCUIMG_C_LtrTs := some_connection9
DHSSMG_B_LtrFl := some_connection10
DHWOMG_S_LtrCd := connection3
DIGIMG_S_LtrOp := some_connection12
DCUIMG_S_DcuNtRdy := some_connection13
PARAGP_L_LtrSvEn := some_connection14
LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl, DHWOMG_S_LtrCd, DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy, PARAGP_L_LtrSvEn, DHSSMG_B_OpLtr, DHSSMG_B_CdLtr, DHSSMG_NX_LtrSaSq, temp1, temp2)
n := rSvEn
DHSSMG_B_OpLtr := B_OpLtr_out
DHSSMG_B_CdLtr := B_CdLtr_out
DHSSMG_NX_LtrSaSq := NX_LtrSaSq_out
temp1 := A_LtrTs_out
temp2 := A_LtrOpVd_out
temp3 := temp1 and temp2
SR(temp1,false,DHSSMG_S_LtrTsRdy)
SET1_in := temp1
RESET_in := false
DHSSMG_S_LtrTsRdy := Q1_out
SR
= n
L_EnLtrSv := L_EnLtrSv_in
SR(temp3,false,DHSSMG_S_LtrOpVd)
SET1_in := temp3
RESET_in := false
DHSSMG_S_LtrOpVd := Q1_out
SR
EXIT Tester.Impl
connection1 := DHSSMG_B_OpLtr
connection2 := DHSSMG_B_CdLtr
some_connection15 := DHSSMG_NX_LtrSaSq
some_connection16 := DHSSMG_S_LtrTsRdy
some_connection17 := DHSSMG_S_LtrOpVd
Timing => Immediate;
Latency => 0ms .. 1ms;
Timing => Immediate;
Latency => 0ms .. 1ms;
Dispatch_Protocol => Periodic;
Period => 4 ms;
Priority => 1;
Compute_Execution_Time => 1 ms .. 2 ms;
Compute_Deadline => 2 ms;
ENTRY Controller.Impl
on dispatch
T
PRASMZ_B_RqPrSd := some_connection1
APSIMZ_B_OpLtr := some_connection2
SSSCMZ_NX_MnSqSt := some_connection3
PCTHMZ_A_PctMo := some_connection4
PLTTMG_B_OpLtr := some_connection5
DHSSMG_B_OpLtr := connection1
DHSSMG_B_CdLtr := connection2
APSIMZ_B_EnCdLnTrpSlt := some_connection6
PARAGP_L_CnfHpp := some_connection7
DHSSMG_B_LtrHwOpFl := some_connection8

Fig. 9. The AFG of LTRIS – part 2.

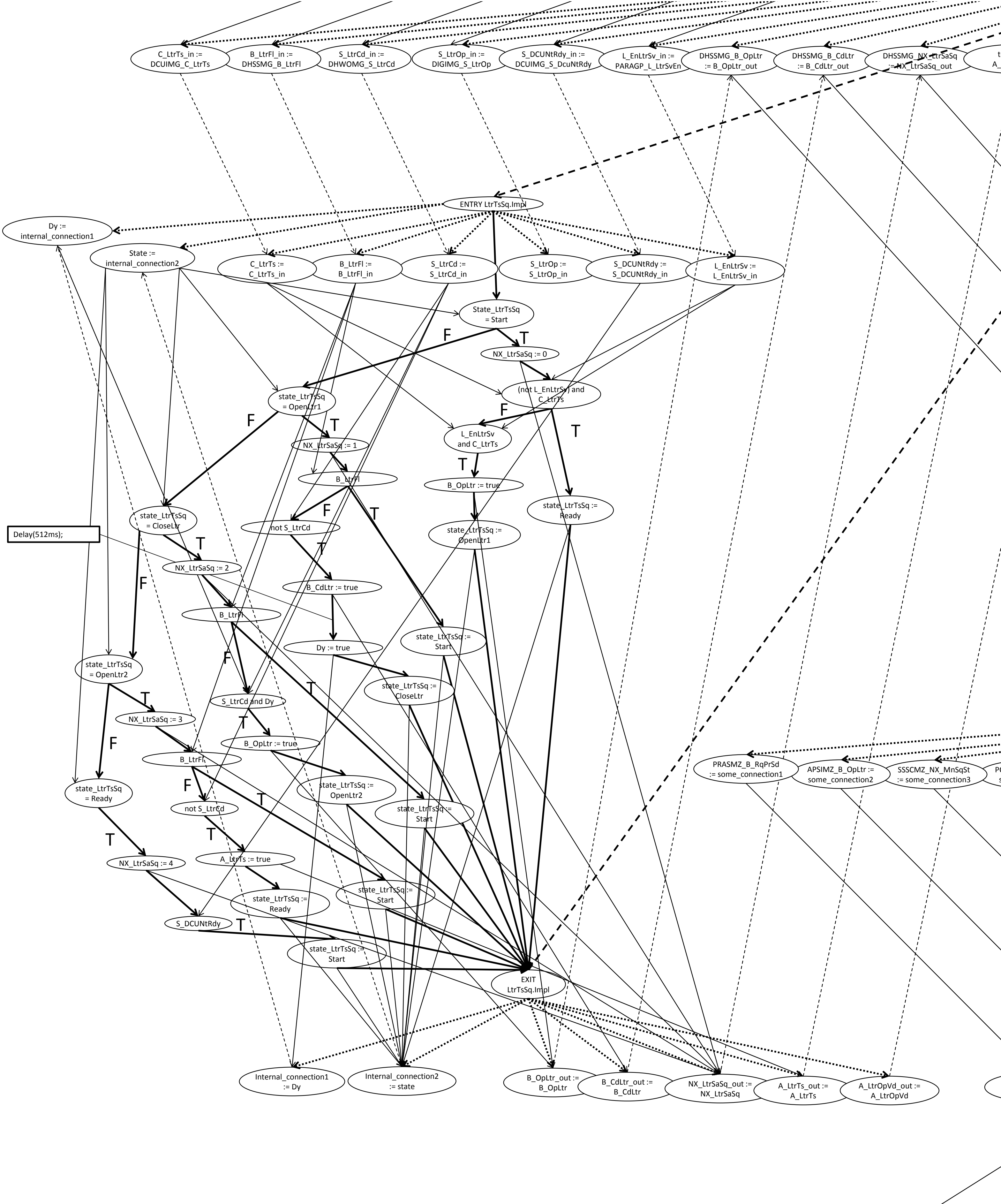


Fig. 10. The AFG of LTRIS – part 3.

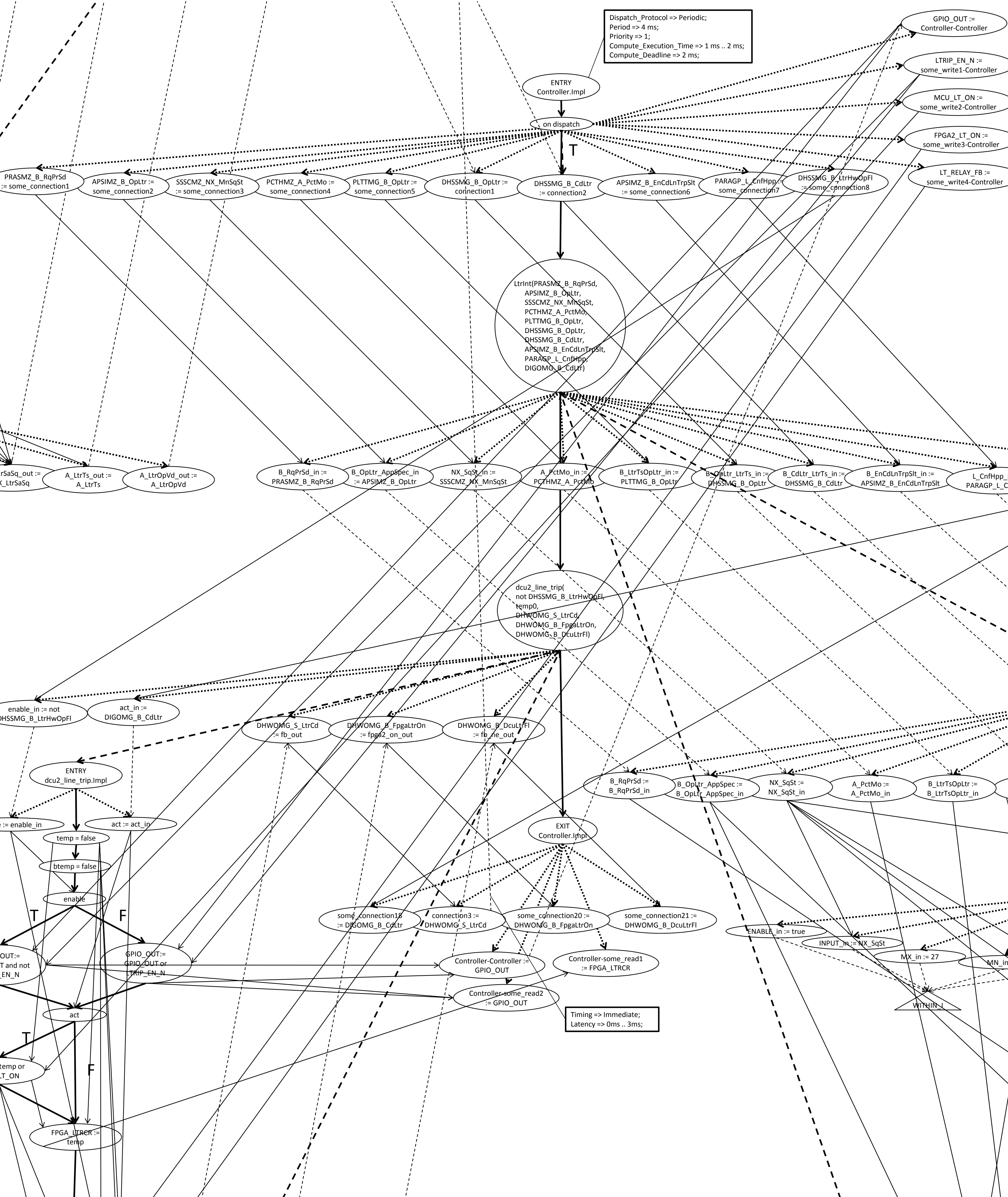


Fig. 11. The AFG of LTRIS – part 4.

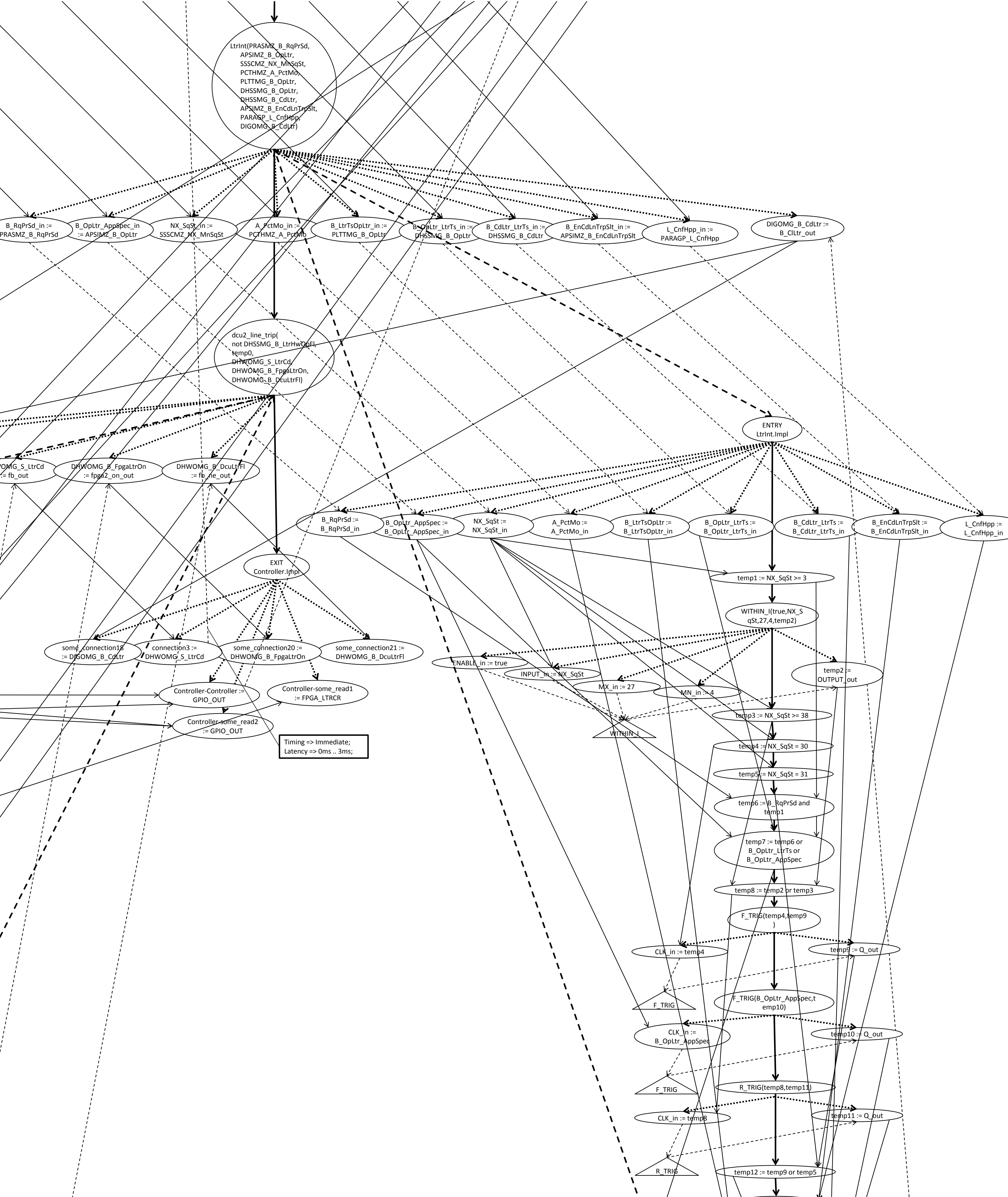


Fig. 12. The AFG of LTRIS – part 5.

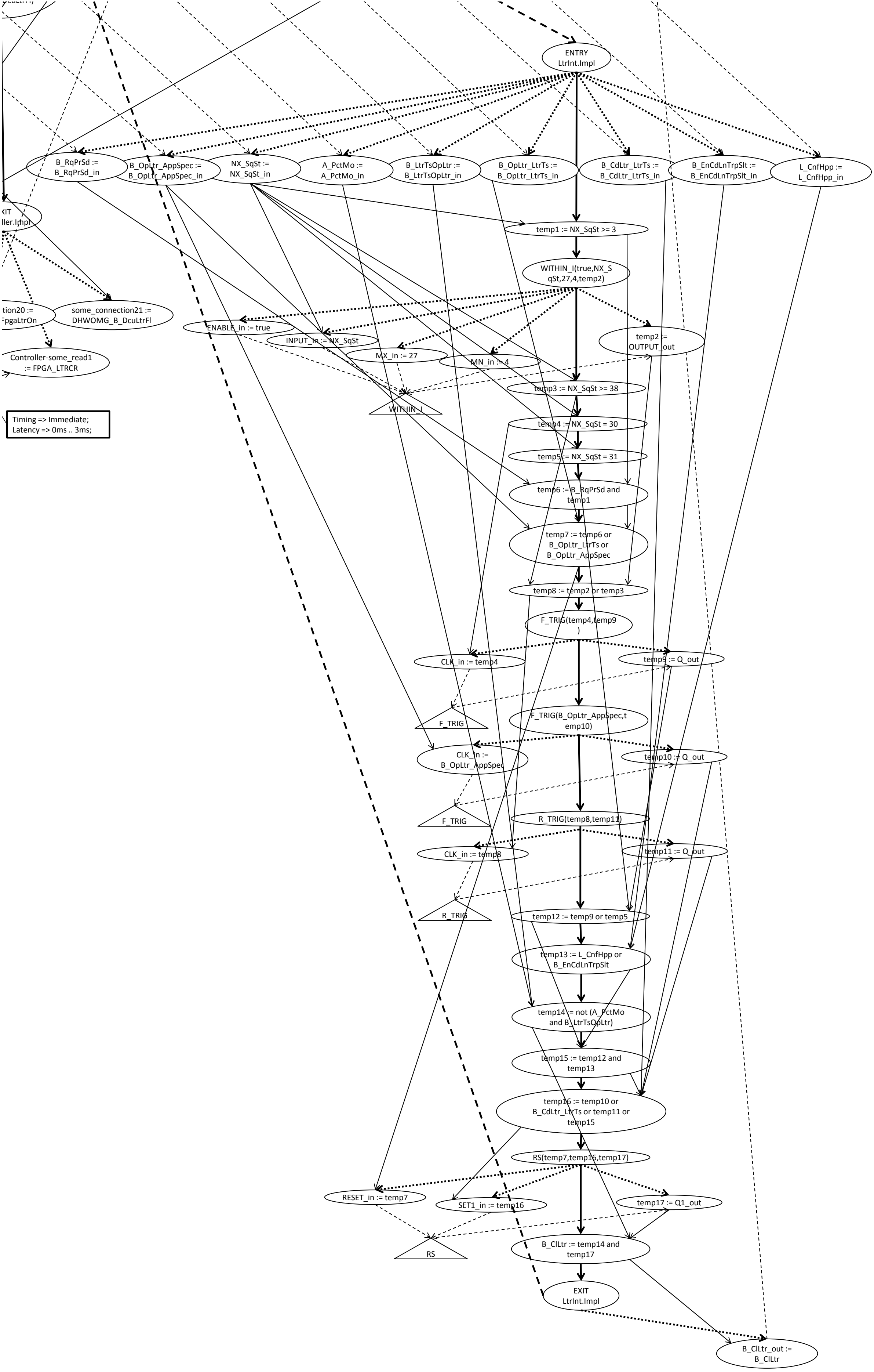
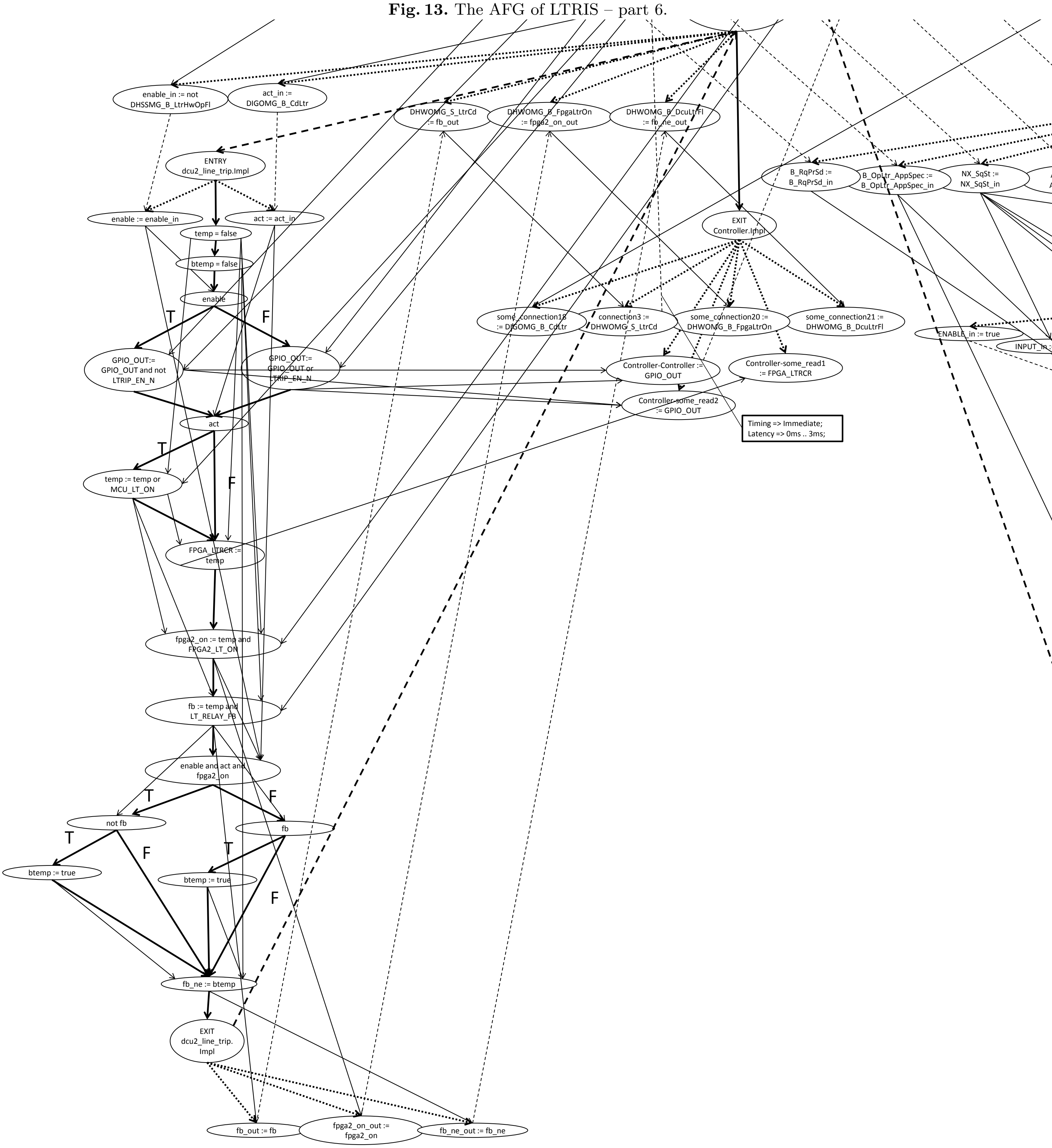


Fig. 13. The AFG of LTRIS – part 6.


enable_in := not DHSSMG_B_LtrHwOpFl
act_in := DIGOMG_B_CdLtr
DHWOMG_S_LtrCd := fb_out
DHWOMG_B_FpgaLtrOn := fpga2_on_out
DHWOMG_B_DcuLtrFl := fb_ne_out
ENTRY dcu2_line_trip.Impl
B_RqPrSd := B_RqPrSd_in
B_OpLtr_AppSpec := B_OpLtr_AppSpec_in
NX_SqSt := NX_SqSt_in
enable := enable_in
act := act_in
EXIT Controller.Impl
temp = false
btemp = false
enable
T
F
some_connection18 := DIGOMG_B_CdLtr
connection3 := DHWOMG_S_LtrCd
some_connection20 := DHWOMG_B_FpgaLtrOn
some_connection21 := DHWOMG_B_DcuLtrFl
ENABLE_in := true
INPUT_in :
GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N
GPIO_OUT:= GPIO_OUT or TRIP_EN_N
Controller-Controller := GPIO_OUT
Controller-some_read1 := FPGA_LTRCR
Controller-some_read2 := GPIO_OUT
Timing => Immediate; Latency => 0ms .. 3ms;
act
T
F
temp := temp or MCU_LT_ON
FPGA_LTRCR := temp
fpga2_on := temp and FPGA2_LT_ON
fb := temp and LT_RELAY_FB
enable and act and fpga2_on
T
F
not fb
fb
T
F
T
F
btemp := true
btemp := true
fb_ne := btemp
EXIT dcu2_line_trip. Impl
fb_out := fb
fpga2_on_out := fpga2_on
fb_ne_out := fb_ne

7 Verification criteria and sequences

In order to verify consistency, completeness, and correctness, the architecture flows must be analyzed with respect to requirements and constraints associated with the model, and in conjunction with the semantic rules of AADL. Each control and data flow is composed of a sequence of elements. A flow is constrained if any member in the sequence is associated with a property. A model is consistent if each control and data flow can be fully executed while not contradicting any constraints imposed by properties. In other words, the model must be able to be executed in compliance with the semantic rules such that each flow can be exercised, from the first element to the last according to the order of the sequence, while each (active) property and requirement is valid in each state of the execution. Correctness can only be determined if requirements are associated with the model or if property declarations are considered as requirements. Consistency implies correctness in the latter case. In the former case, the model is correct if no flow exceeds any requirement declarations while they are executed. The model is complete if all flows can be activated by the specified input classes and a flow will be activated for every class of input.

These objectives can be defined in terms of control- and data-flow reachability. Control-flow reachability is the property where each architectural element in an execution order can reach the subsequent element to be executed without conflicting any constraints or requirements. Data-flow reachability is the property where each data element can reach its target component, where the data is used, from its source component, where the data is defined, without conflicting any constraints or requirements. Thus, reachability of each flow imply architecture consistency, correctness, and, if all possible input classes have been covered, completeness. Note that reachability analysis consider properties such as the minimum and maximum latencies of connections, or the period, execution time, and deadline of threads. It therefore implies analysis of aspects such as timing and schedulability. Consequently, flow reachability cannot be achieved if timing constraints are not met or the system is not schedulable.

An AFG contains different structural path types including different types of control and data flows. An AFG path in conjunction with the (possibly empty) set of path constraints and requirements is referred to as a *verification sequence*. Three types of paths exist: (1) *component-internal paths* including component-internal flows between interfaces of a component; (2) *direct component to component paths* including inter-component flows between interfaces of two components; and (3) *indirect component to component paths* including flows between interfaces of two components through one or several intermediate components.

Recall that $COMP = \{comp_1, comp_2, \dots, comp_n\}$ denotes the set of software components in the architecture. Let $I = \{comp_x.i \mid comp_x.i \in DP\_U \cup EP\_U \cup EDP\_U \cup SP\_U \cup DA\_U \cup SA\_U\}$ denote the set of component interfaces (data ports, event ports, event data ports, parameters, data accesses, and subprogram accesses) in the architecture. The set of interfaces may interact through connection declarations $C = \{c(s, d) \mid \text{source } s \in I \text{ and destination } d \in I\}$ and through BMs. A BM refines the interaction of a connection if it operates on either the source or destination interface. Let $BMIR = \{bmir(i) \mid bmir(i)$ is a control path of a behavioral model that operates on $i \in I\}$ denote the set of refined interface behaviors. A BM connects an input interface to an output interface if there is a control path through the BM that operates on the output interface in response to an operation on the input interface. Let $BMC = \{bmc(s, d) \mid bmc(s, d)$ is a control path of a behavioral model that operates on $s \in I$ in response to an operation on $d \in I\}$ denote the component interface connections connected through a BM. The possible types of AFG paths are defined by the following relations between component interfaces.

Definition 5 $CIR \subseteq I \times I$ is the set of component internal relations such that $\langle comp_1.i_1, comp_1.i_2\rangle \in CIR$ iff $c(comp_1.i_1, comp_1.i_2) \in C$ or $bmc(comp_1.i_1, comp_1.i_2) \in BMC$.

Definition 6 $DCCR \subseteq I \times I$ is the set of direct component to component relations such that $\langle comp_1.i_1, comp_2.i_2\rangle \in DCCR$ iff $c(comp_1.i_1, comp_2.i_2) \in C$ or $c(comp_1.i_1, comp_2.i_2) \in C$ and $bmir(comp_1.i_1) \in BMIR$ or $c(comp_1.i_1, comp_2.i_2) \in C$ and $bmir(comp_2.i_2) \in BMIR$.

Definition 7 $ICCR \subseteq I \times I \times I^*$ is the set of indirect component to component relations and is defined recursively to include any number of components. The base case is: $\langle comp_1.i_1, comp_3.i_4, t\rangle \in ICCR$ iff $\langle comp_1.i_1, comp_2.i_2\rangle \in DCCR$ and $\langle comp_2.i_2, comp_2.i_3\rangle \in CIR$ and $\langle comp_2.i_3, comp_3.i_4\rangle \in DCCR$ and $t = \langle\langle comp_1.i_1, comp_2.i_2\rangle, \langle comp_2.i_2, comp_2.i_3\rangle, \langle comp_2.i_3, comp_3.i_4\rangle\rangle$. The inductive clause is: $\langle comp_1.i_1, comp_x.i_y, t\rangle \in ICCR$ iff $\langle comp_1.i_1, comp_2.i_2\rangle \in DCCR$ and $\langle comp_2.i_2, comp_2.i_3\rangle \in CIR$ and $\langle comp_2.i_3, comp_x.i_y, t'\rangle \in ICCR$ and $t = \langle\langle comp_1.i_1, comp_2.i_2\rangle, \langle comp_2.i_2, comp_2.i_3\rangle, \langle t'\rangle\rangle$

These types of relations are illustrated in Figure 14. There is a component internal relation between $comp_2.i_1$ and $comp_2.i_2$ as they are connected through bmc_1: a control path of $comp_2$ which is guarded by $comp_2.i_1$ and updates $comp_2.i_2$. There are direct component to component relations both between $comp_1.i_1$ and $comp_2.i_1$ and between $comp_2.i_2$ and $comp_3.i_1$, where $c1$ connects $comp_1.i_1$ to $comp_2.i_1$ and $c2$ connects $comp_2.i_2$ to $comp_3.i_1$. Note that there exist refined versions of these relations. Finally, there is an indirect component to component relation, where $c1$, bmc_1, and $c2$ indirectly connect $comp_1.i_1$ to $comp_3.i_1$.

If any of these relations exist between two interfaces, there exist a corresponding – internal, direct, or indirect – path. These also correspond to the possible coverage criteria that can be applied, where each type provides an increasing amount of coverage at more cost and time [21].

- **Component Internal Coverage**: requires coverage of all *Component Internal Transfer Paths.*
- **Direct Component to Component Coverage**: requires coverage of all *Direct Component to Component Paths.*

- **Indirect Component to Component Coverage**: requires coverage of all *Indirect Component to Component Paths.*

Note that each successive type subsumes the preceding types, e.g., indirect paths subsume both direct and internal paths. In the simplest case, only component-internal flows may be covered. Inter-component flows may be added to cover the complete AFG. However, covering the complete graph does not mean that all possible combinations of flows have been covered. Covering all indirect paths ensures this, which also is a criterion for ensuring completeness, correctness, and consistency.

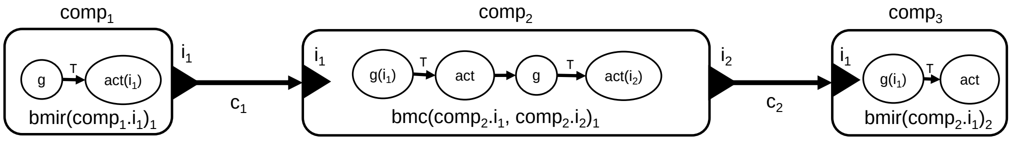


Fig. 14. Illustration of relations between three interconnected components.

7.1 Results

In Appendix A, the verification sequences that are extracted when applying the coverage criteria are presented. The AFG contains 34 internal paths, 6 direct paths (three of which are calls with associated parameter data flows), and 17 indirect paths.

8 Formal semantics in Uppaal timed automata

In order to execute the verification sequences through formal verification, the AADL model semantics must be formalized and implemented. This is achieved through a transformation to Uppaal timed automata. The transformation rules are defined by means of functions, where a transformation is initiated through function $aadlToNta : AADLMDL \rightarrow NTA$ that maps an AADL model to a network of timed automata. Let *processor* denote the processor component a set of threads THR is bound to, $Port_i = InPort_i \cup OutPort_i$ denote the set of in and out ports of a thread thr_i, and $connect : Port \rightarrow C$ be a function which assigns connections to ports. We further use $id(element)$ to denote the *identifier* of an element, $val(element)$ to denote its *value*, and $defVal(element)$ to denote its *default value*. The function, defined in Rule 6, maps a processor component to a scheduler automaton through function $prToTa$ defined in Rule 7, each (bounded) thread component to a thread automaton through function $thrToTa$ defined in Rule 8, and each port connection and shared data component to a global variable through functions $cToVarg$ and $dataCompToVarg$ defined in Rule 9 and Rule 10.

Rule 6 $aadlToNta : AADLMDL \rightarrow NTA$ where $aadlToNta(AADLMDL) = \langle \overline{TA}, Var_G, \emptyset, Ch \rangle$ such that $\overline{TA}[0] = prToTa(processor)$ and for $0 \leq i < |T|$, $\overline{TA}[i+1] = thrToTa(t_i)$, $Var_G = \{cToVarg(c) \mid c \in C\} \cup \{dataCompToVarg(data\_s) \mid data\_s \in DATA\}$ and Ch is as described in Section 8.2.

Rule 7 $prToTa : \mathcal{P}(processor) \rightarrow TA$ where $prToTa(processor) = \langle L, \ell_o, X, Var, I, E \rangle$ such that L, ℓ_o, X, Var, I and E are as described in Section 8.2.

Rule 8 $thrToTa : THR \rightarrow TA$ where $thrToTa(thr_i) = \langle L, \ell_o, X, Var, I, E \rangle$ such that $L = \{awaiting\_dispatch, ready, running, awaiting\_resource\}$; $\ell_0 = awaiting\_dispatch$; $X = \{cl\}$; $Var = \{Period, C\_E\_T, C\_D, Priority\} \cup PortIn \cup PortOut$ where $val(Period) = val(Period_i), val(C\_E) = val(Compute\_Execution\_Time_i), Val(C\_D) = val(Compute\_Deadline_i)$, $val(Priority) = val(Priority_i)$, $PortIn = \{portToVar(inport) \mid inport \in In\_Port_i\}$ and $PortOut = \{portToVar(outport) \mid outport \in Out\_Port_i\}$; $I(awaiting\_dispatch) = \{cl <= Period\}$; and $E = \{awaiting\_dispatch \xrightarrow{cl>=Period, dispatched[i]!, u_1} ready, ready \xrightarrow{run[i]?} running, running \xrightarrow{preempt[i]?} ready, running \xrightarrow{complete[i]?, u_2} awaiting\_dispatch, running \xrightarrow{blocked[i]!} awaiting\_resource, awaiting\_resource \xrightarrow{unblocked[i]!} ready\}$ where $u_1 = \langle sch\_info[i][0] := C\_E\_T, sch\_info[i][1] := C\_D, sch\_info[i][2] := Priority, portin_0 = id(connect(in\_port_0)), portin_1 = id(connect(in\_port_1)), \ldots, portin_n = id(connect(in\_port_n)), cl := 0 \rangle$ and $u_2 = \langle id(connect(out\_port_0)) = port\_out_0, id(connect(out\_port_1)) = port\_out_1, \ldots, id(connect(out\_port_m)) = port\_out_m, out\_port_1 = defVal(port\_out_1), out\_port_2 = defVal(port\_out_2), \ldots, out\_port_m = defVal(port\_out_m)) \rangle$.

Rule 9 $cToVarg : C \rightarrow Var_G \times Var_G$ where $cToVarg(c) = \langle v_{G1}, v_{G2} \rangle$ such that $id(c) = id(v_{G1})$ and v_{G2} is a semaphore for access of v_{G1}.

Rule 10 $dataCompToVarg : DATA \rightarrow Var_G$ where $dataCompToVarg(data\_s) = v_G$ such that $id(data\_s) = id(v_G)$.

Rule 11 $portToVar : Port \rightarrow Var$ where $portToVar(port) = v$ such that $Id(port) = Id(v)$.

The mapped scheduling automaton, which details is described in Section 8.2, controls the transition of thread states, from dispatches to completions, and of preemptions and context switches. A thread automaton, in its most basic form, consists of four locations: *awaiting_dispatch*, *ready*, *running*, and (if the thread operates on shared resources) *awaiting_resource*. Each thread is initially in the *awaiting_dispatch* location. An edge $awaiting\_dispatch \xrightarrow{cl>=Period,a,u_1} ready$ is then taken to the *ready* location depending on the dispatch protocol. For periodic threads, the time of dispatch is entirely dependent on the clock in relation to the period of the thread. At the time of dispatch, data on port connections are assigned to input ports of the thread. Ports are mapped to local variables of the corresponding thread automaton through function *portToVar* defined in Rule 11. These assignments $(u_1 = \langle \ldots, portin_1 = c_x, portin_2 = c_y, \ldots, portin_n = c_z \rangle)$ correspond to actual-in vertices of Rule 1–3. Threads in the *ready* location are assigned to be executed by the processor component they are bound to according to a scheduling policy property. Assuming a scheduler with fixed priority preemptive scheduling policy, the thread with the highest priority is selected to run on the processor and thus transits to the *running* location. No more than one thread (per processing unit) is allowed to be in a *running* location simultaneously. A thread that operates on a shared resource access it through a $Get\_Resource()$ service call. If the semaphore of the resource already is locked, the thread transits to the *awaiting_resource* location. Shared resources are released through a *Release_Resource* service call.

A running thread is preempted and transits back to the ready location if another thread with higher priority enters the *ready* location. A thread in the *running* location that completes its execution transits to the *awaiting_dispatch* location through $running \xrightarrow{g,a,u_2} awaiting\_dispatch$. Output is simultaneously assigned to connections $(u_2 = \langle c_x = portout_1, c_y = portout_2, \ldots, c_z = portout_n, \ldots \rangle)$. These correspond to formal-in vertices of Rule 1–3. Note that out ports also (as defined in Rule 8) are set to default values after they have been assigned to connections such that a subsequent dispatch conforms to the initialization settings.

If the thread is specified with a BM, the *running* location is replaced with the BM automaton under the assumption that its execution starts from $firstState(thr_i.BM)$. Transitions of the BM automaton that are not constrained by computation time declarations are assumed to be instantaneous. A transition specified with computation time is mapped to a timed automata transition with an intermediate location wherein the corresponding time must progress before the target state is reached. The location in which time may progress must be integrated with the preemption mechanism if the scheduling policy is preemptive.

BMs of subprogram components may be inserted where they are called as long as the subprogram is local to the calling thread. Similarly to port connections, local state variables are used as temporary in and out variables for actual-in, formal-in, formal-out, and actual-out assignments associated with the call. If the AADL model includes remote subprogram calls, the scheduler and threads servicing remote calls must include the concurrency mechanism defined by remote subprogram call properties (properties for both synchronous and semi-synchronous remote calls are supported by AADL).

8.1 Results

According to the transformation rules, Tester (Table 7) is transformed into the automaton presented in Figure 15, Controller (Table 4) into the automaton in Figure 16, and the environment (Table 2) into the automaton in Figure 17.

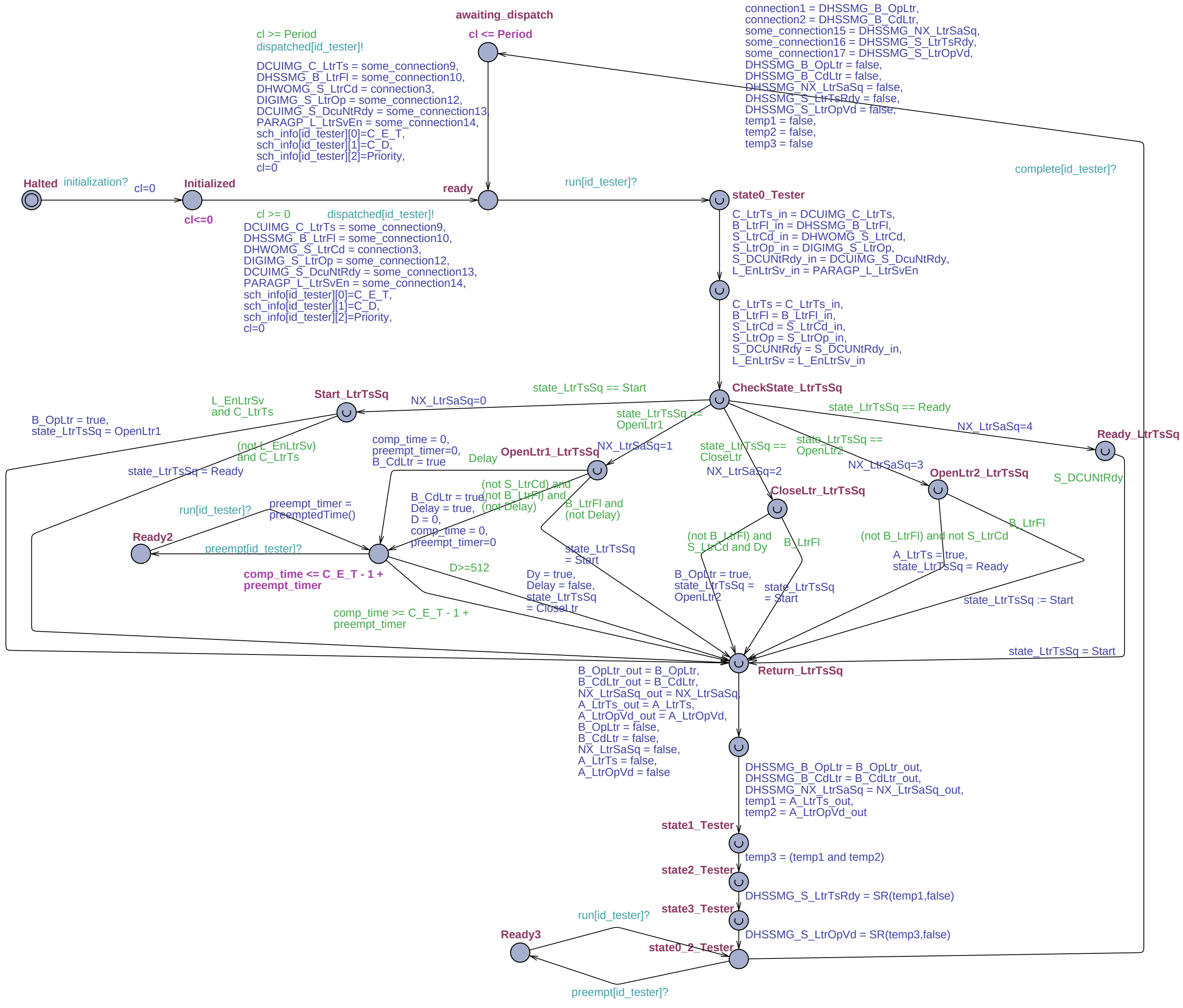


Fig. 15. Timed automata model of Tester.Impl.

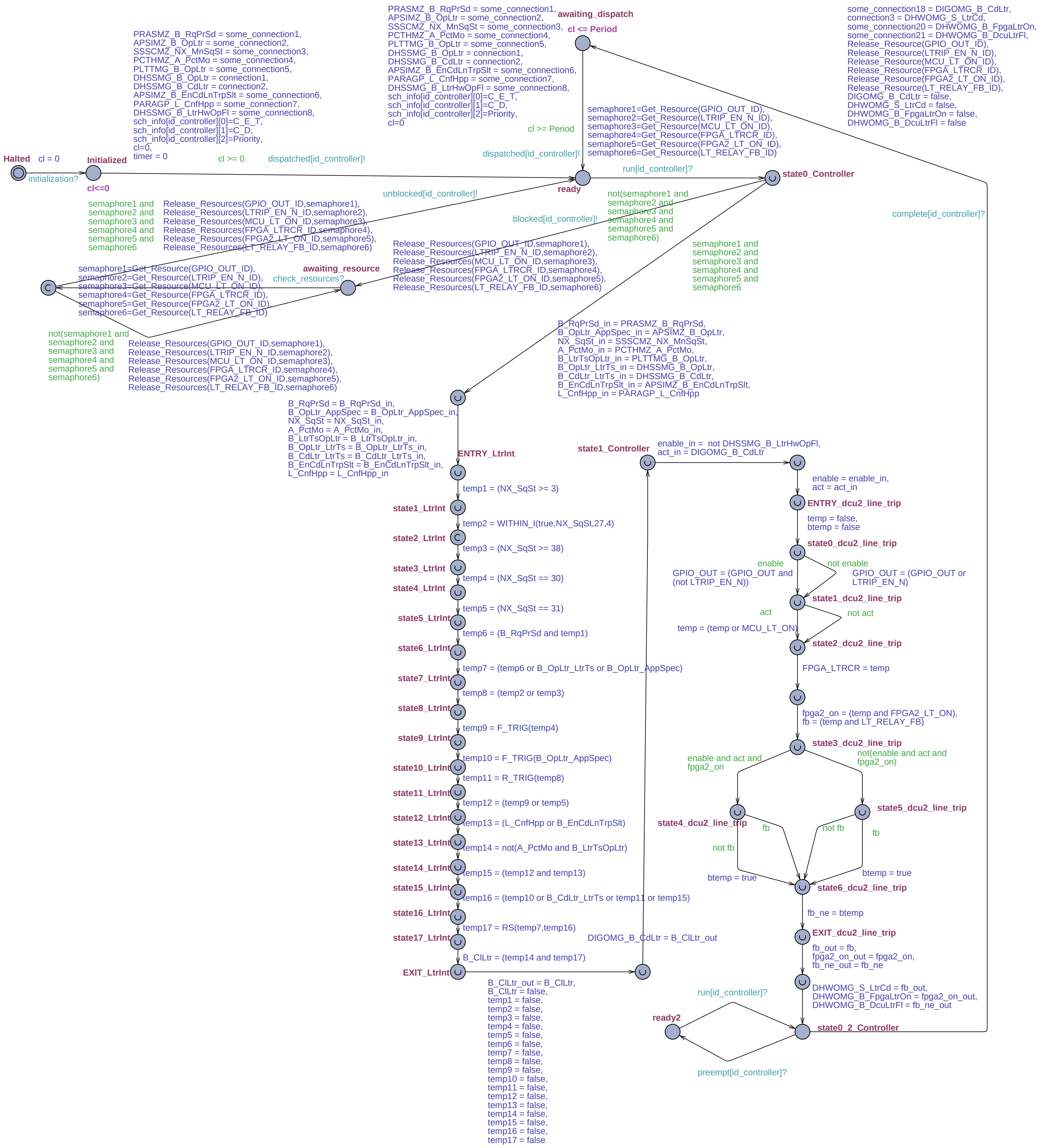

PRASMZ_B_RqPrSd = some_connection1,
APSIMZ_B_OpLtr = some_connection2,
SSSCMZ_NX_MnSqSt = some_connection3,
PCTHMZ_A_PctMo = some_connection4,
PLTTMG_B_OpLtr = some_connection5,
DHSSMG_B_OpLtr = connection1,
DHSSMG_B_CdLtr = connection2,
APSIMZ_B_EnCdLnTrpSlt = some_connection6,
PARAGP_L_CnfHpp = some_connection7,
DHSSMG_B_LtrHwOpFl = some_connection8,
sch_info[id_controller][0]=C_E_T,
sch_info[id_controller][1]=C_D,
sch_info[id_controller][2]=Priority,
cl=0,
timer = 0
PRASMZ_B_RqPrSd = some_connection1,
APSIMZ_B_OpLtr = some_connection2,
SSSCMZ_NX_MnSqSt = some_connection3,
PCTHMZ_A_PctMo = some_connection4,
PLTTMG_B_OpLtr = some_connection5,
DHSSMG_B_OpLtr = connection1,
DHSSMG_B_CdLtr = connection2,
APSIMZ_B_EnCdLnTrpSlt = some_connection6,
PARAGP_L_CnfHpp = some_connection7,
DHSSMG_B_LtrHwOpFl = some_connection8,
sch_info[id_controller][0]=C_E_T,
sch_info[id_controller][1]=C_D,
sch_info[id_controller][2]=Priority,
cl=0
awaiting_dispatch
cl <= Period
some_connection18 = DIGOMG_B_CdLtr,
connection3 = DHWOMG_S_LtrCd,
some_connection20 = DHWOMG_B_FpgaLtrOn,
some_connection21 = DHWOMG_B_DcuLtrFl,
Release_Resource(GPIO_OUT_ID),
Release_Resource(LTRIP_EN_N_ID),
Release_Resource(MCU_LT_ON_ID),
Release_Resource(FPGA_LTRCR_ID),
Release_Resource(FPGA2_LT_ON_ID),
Release_Resource(LT_RELAY_FB_ID),
DIGOMG_B_CdLtr = false,
DHWOMG_S_LtrCd = false,
DHWOMG_B_FpgaLtrOn = false,
DHWOMG_B_DcuLtrFl = false
semaphore1=Get_Resource(GPIO_OUT_ID),
semaphore2=Get_Resource(LTRIP_EN_N_ID),
semaphore3=Get_Resource(MCU_LT_ON_ID),
semaphore4=Get_Resource(FPGA_LTRCR_ID),
semaphore5=Get_Resource(FPGA2_LT_ON_ID),
semaphore6=Get_Resource(LT_RELAY_FB_ID)
cl >= Period
dispatched[id_controller]!
Halted
cl = 0
initialization?
Initialized
cl<=0
cl >= 0
dispatched[id_controller]!
run[id_controller]?
ready
state0_Controller
unblocked[id_controller]!
blocked[id_controller]!
complete[id_controller]?
semaphore1 and
semaphore2 and
semaphore3 and
semaphore4 and
semaphore5 and
semaphore6
Release_Resources(GPIO_OUT_ID,semaphore1),
Release_Resources(LTRIP_EN_N_ID,semaphore2),
Release_Resources(MCU_LT_ON_ID,semaphore3),
Release_Resources(FPGA_LTRCR_ID,semaphore4),
Release_Resources(FPGA2_LT_ON_ID,semaphore5),
Release_Resources(LT_RELAY_FB_ID,semaphore6)
not(semaphore1 and
semaphore2 and
semaphore3 and
semaphore4 and
semaphore5 and
semaphore6)
Release_Resources(GPIO_OUT_ID,semaphore1),
Release_Resources(LTRIP_EN_N_ID,semaphore2),
Release_Resources(MCU_LT_ON_ID,semaphore3),
Release_Resources(FPGA_LTRCR_ID,semaphore4),
Release_Resources(FPGA2_LT_ON_ID,semaphore5),
Release_Resources(LT_RELAY_FB_ID,semaphore6)
semaphore1 and
semaphore2 and
semaphore3 and
semaphore4 and
semaphore5 and
semaphore6
semaphore1=Get_Resource(GPIO_OUT_ID),
semaphore2=Get_Resource(LTRIP_EN_N_ID),
semaphore3=Get_Resource(MCU_LT_ON_ID),
semaphore4=Get_Resource(FPGA_LTRCR_ID),
semaphore5=Get_Resource(FPGA2_LT_ON_ID),
semaphore6=Get_Resource(LT_RELAY_FB_ID)
awaiting_resource
check_resources?
C
not(semaphore1 and
semaphore2 and
semaphore3 and
semaphore4 and
semaphore5 and
semaphore6)
Release_Resources(GPIO_OUT_ID,semaphore1),
Release_Resources(LTRIP_EN_N_ID,semaphore2),
Release_Resources(MCU_LT_ON_ID,semaphore3),
Release_Resources(FPGA_LTRCR_ID,semaphore4),
Release_Resources(FPGA2_LT_ON_ID,semaphore5),
Release_Resources(LT_RELAY_FB_ID,semaphore6)
B_RqPrSd_in = PRASMZ_B_RqPrSd,
B_OpLtr_AppSpec_in = APSIMZ_B_OpLtr,
NX_SqSt_in = SSSCMZ_NX_MnSqSt,
A_PctMo_in = PCTHMZ_A_PctMo,
B_LtrTsOpLtr_in = PLTTMG_B_OpLtr,
B_OpLtr_LtrTs_in = DHSSMG_B_OpLtr,
B_CdLtr_LtrTs_in = DHSSMG_B_CdLtr,
B_EnCdLnTrpSlt_in = APSIMZ_B_EnCdLnTrpSlt,
L_CnfHpp_in = PARAGP_L_CnfHpp
B_RqPrSd = B_RqPrSd_in,
B_OpLtr_AppSpec = B_OpLtr_AppSpec_in,
NX_SqSt = NX_SqSt_in,
A_PctMo = A_PctMo_in,
B_LtrTsOpLtr = B_LtrTsOpLtr_in,
B_OpLtr_LtrTs = B_OpLtr_LtrTs_in,
B_CdLtr_LtrTs = B_CdLtr_LtrTs_in,
B_EnCdLnTrpSlt = B_EnCdLnTrpSlt_in,
L_CnfHpp = L_CnfHpp_in
ENTRY_LtrInt
temp1 = (NX_SqSt >= 3)
state1_LtrInt
temp2 = WITHIN_I(true,NX_SqSt,27,4)
state2_LtrInt
C
temp3 = (NX_SqSt >= 38)
state3_LtrInt
temp4 = (NX_SqSt == 30)
state4_LtrInt
temp5 = (NX_SqSt == 31)
state5_LtrInt
temp6 = (B_RqPrSd and temp1)
state6_LtrInt
temp7 = (temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec)
state7_LtrInt
temp8 = (temp2 or temp3)
state8_LtrInt
temp9 = F_TRIG(temp4)
state9_LtrInt
temp10 = F_TRIG(B_OpLtr_AppSpec)
state10_LtrInt
temp11 = R_TRIG(temp8)
state11_LtrInt
temp12 = (temp9 or temp5)
state12_LtrInt
temp13 = (L_CnfHpp or B_EnCdLnTrpSlt)
state13_LtrInt
temp14 = not(A_PctMo and B_LtrTsOpLtr)
state14_LtrInt
temp15 = (temp12 and temp13)
state15_LtrInt
temp16 = (temp10 or B_CdLtr_LtrTs or temp11 or temp15)
state16_LtrInt
temp17 = RS(temp7,temp16)
state17_LtrInt
B_ClLtr = (temp14 and temp17)
EXIT_LtrInt
DIGOMG_B_CdLtr = B_ClLtr_out
B_ClLtr_out = B_ClLtr,
B_ClLtr = false,
temp1 = false,
temp2 = false,
temp3 = false,
temp4 = false,
temp5 = false,
temp6 = false,
temp7 = false,
temp8 = false,
temp9 = false,
temp10 = false,
temp11 = false,
temp12 = false,
temp13 = false,
temp14 = false,
temp15 = false,
temp16 = false,
temp17 = false
state1_Controller
enable_in = not DHSSMG_B_LtrHwOpFl,
act_in = DIGOMG_B_CdLtr
enable = enable_in,
act = act_in
ENTRY_dcu2_line_trip
temp = false,
btemp = false
state0_dcu2_line_trip
enable
GPIO_OUT = (GPIO_OUT and
(not LTRIP_EN_N))
not enable
GPIO_OUT = (GPIO_OUT or
LTRIP_EN_N)
state1_dcu2_line_trip
act
not act
temp = (temp or MCU_LT_ON)
state2_dcu2_line_trip
FPGA_LTRCR = temp
fpga2_on = (temp and FPGA2_LT_ON),
fb = (temp and LT_RELAY_FB)
state3_dcu2_line_trip
enable and act and
fpga2_on
not(enable and act and
fpga2_on)
state4_dcu2_line_trip
state5_dcu2_line_trip
fb
not fb
not fb
fb
btemp = true
btemp = true
state6_dcu2_line_trip
fb_ne = btemp
EXIT_dcu2_line_trip
fb_out = fb,
fpga2_on_out = fpga2_on,
fb_ne_out = fb_ne
DHWOMG_S_LtrCd = fb_out,
DHWOMG_B_FpgaLtrOn = fpga2_on_out,
DHWOMG_B_DcuLtrFl = fb_ne_out
run[id_controller]?
ready2
state0_2_Controller
preempt[id_controller]?

Fig. 16. Timed automata model of Controller.Impl.

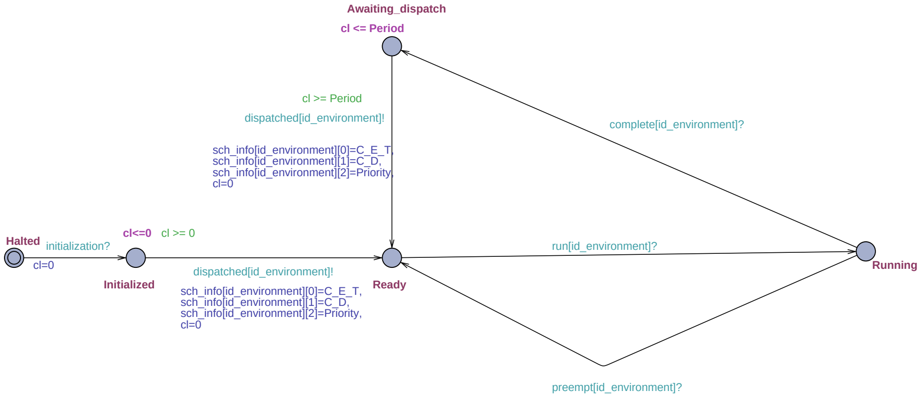

Awaiting_dispatch
cl <= Period
cl >= Period
dispatched[id_environment]!
sch_info[id_environment][0]=C_E_T,
sch_info[id_environment][1]=C_D,
sch_info[id_environment][2]=Priority,
cl=0
complete[id_environment]?
Halted
initialization?
cl=0
cl<=0
cl >= 0
Initialized
dispatched[id_environment]!
sch_info[id_environment][0]=C_E_T,
sch_info[id_environment][1]=C_D,
sch_info[id_environment][2]=Priority,
cl=0
Ready
run[id_environment]?
Running
preempt[id_environment]?

Fig. 17. Timed automata model of the abstracted LTRIS environment.

8.2 The scheduler automaton

A processor component is mapped to a scheduler automaton such as the template shown in Figure 19. The labels of the scheduling automaton are defined as follows:

- (int)ready_queue[x]: is a sorted queue of currently dispatched threads. The queue is sorted according to a given scheduling policy where the first element in the queue (x=0) is the (identifier of the) thread being processed and where the second element is the next thread to be processed, and so forth.
- (clock)sch_clocks[x][2]: is a list of clocks in sets of two, each set referenced by an identifier x of a currently dispatched thread. Each dispatched thread has two clocks, the first (sch_clocks[x][0] of thread with identifier x) is used to keep track of a thread's execution time, and the second (sch_clocks[x][1] of thread with identifier x) is used to keep track of a thread's deadline.
- (int)sch_info[x][3]: is a list of threads' scheduling properties (integers) in sets of three, each set referenced by an identifier x of a currently dispatched thread. Each dispatched thread has three scheduling properties, the first (sch_info[x][0] of thread with identifier x) is the execution time, the second (sch_info[x][1] of thread with identifier x) is the deadline, and the third (sch_info[x][2] of thread with identifier x) is the priority.
- (int)preempt_stack[x][2]: is a stack of sets of currently preempted threads (integer identifiers) and the amount of time each thread has been preempted. Given a stack of preempted threads, the first set of elements in the stack (preempt_stack[0][0] is the thread identifier and preempt_stack[0][1] is the amount of time) corresponds to the thread that first was preempted.
- (int)nr_preempted: number of currently preempted threads.
- (int)threads: number of currently dispatched threads.
- (int)check_preempt: holds the identity of a thread that is dispatched at the same time as another thread is running. It is used to check if the dispatched thread preempts the running thread.

- (chan)dispatched[(int)x],(chan)run[(int)x],(chan)complete[(int)x], (chan)preempt[(int)x],(chan)blocked[(int)x],(chan)unblocked[(int)x]: are channels used to synchronize every thread transition of every thread in the system. Synchronization with a particular thread is done through its identity. For example, run[2] is a synchronization channel with thread having identity "2".
- (void)schprotocol((int)x): is a function sorting threads in the ready_queue according to a given scheduling policy. The function is called each time a thread dispatches where the thread's identity is given as argument to the function. In this example, we assume fixed priority scheduling.
- (void)completion((int)x): is a function removing threads from the ready_queue. The function is called each time a thread completes its execution, where the thread's identity is given as argument to the function.
- (void)remove((int)x): is a function removing threads from the ready_queue. The function is called when a thread is blocked due to shared resources.
- (void)addTime(): is a function adding preempted time to the threads in the preempt_stack. The function is called when a preemption occurs, whereupon the execution time of the thread causing the preemption is added to the preemption time of every preempted thread.
- (void)removeTime(): is a function removing preempted time from the threads in the preempt_stack. The function is called when a block due to shared resources occurs, whereupon the execution time of the thread is removed from the preemption time of every preempted thread.
- (void)checkTime((int)x): is a function adding preempted time to the threads in (int)preempt_stack[x][2]. The function is called when a thread dispatch not causing any preemption occurs, to check if the dispatched thread is prior to any preempted threads in the ready_queue whereupon preemption time is added.

The automaton includes two clocks per thread, lists and functions with corresponding variables to handle thread execution and preemption. The reason for having two clocks per thread is that the UPPAAL language does only allow reset and comparison of clocks, i.e., clocks cannot be read or assigned. Because of these constraints, a preempted thread's time of completion cannot be obtained solely from its execution time. In order to model thread preemption, a method considering the execution time of the threads causing preemption is used to calculate preempted threads' time of completion.

The method is illustrated in Figure 18. A, B and C are denotations for threads where priority of $A <$ priority of $B <$ priority of C. C_A is the execution time of A and D_A is the deadline for A. c_A (sch_clocks[i][0]) and d_A (sch_clocks[i][1]) are clocks for A, which are used to measure the time of completion and the time of a missed deadline respectively. r_A is a variable used to summarize the time required to complete thread A and all – during the execution of A – dispatched threads with priorities higher than A. As shown in the illustration, the time of completion for thread A is when the comparison $c_A = r_A$ evaluates to true. In addition to this comparison, $d_A > D_A$ should not evaluate to true before or while A's completion. The comparison is used for schedulability analysis where an evaluation to true indicates a missed deadline. Note that we are illustrating the method explicitly for thread A though the methodology is applied to each thread. A formal proof of the methodology is presented in [22]. Furthermore, the behavior of the scheduler assumes immediate switching time of threads. If the processor the threads are bound to is specified with a *Thread_Swap_Execution_Time* property, the scheduler has to be modified with intermediate locations delaying the switching-time according to the specified property.

The scheduler is initially in the *Empty* location when the system has been initialized. When dispatch occurs, the scheduler transits to the *Schedule1* location whereby the corresponding thread is added to the *ready_queue* (via $schprotocol()$) and its deadline clock is reset (corresponds to $d_A = 0$ in Figure 18). The *Schedule1* location is a committed repetition of the *Empty* location, allowing several threads to be dispatched simultaneously through the edge labeled with channel *dispatched*. The other edge to *Schedule1* itself, labeled with channel *unblocked*, allows for unblocking of threads. However, the edge can only be fired in response to a completion of a thread whereupon the availability of resources are checked. Succeeding to all simultaneous dispatches, the scheduler synchronizes with the first thread in the *ready_queue* and transits to the *Running* location through one of two different edges depending on which action should be executed. If the number of preempted threads is zero, or if the number is more than zero and the latest preempted thread is not the first in the *ready_queue*, the execution time clock of the thread to be run is reset (corresponds to $c_A = 0$). If the number of preempted threads are more than zero and the latest preempted thread is the first thread in the *ready_queue*, the scheduler transits to the *Running* location without resetting its execution time clock since it already has been reset (corresponds to the start of execution of A after preemption by B and C). The

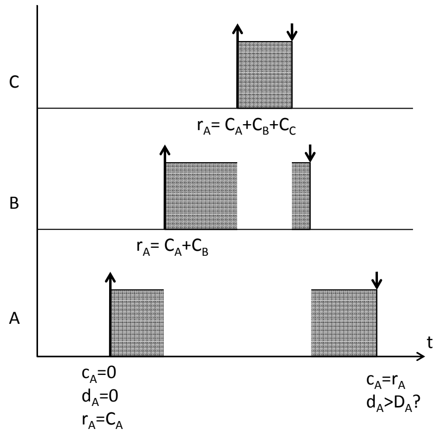


Fig. 18. Thread execution schema for threads A, B and C, where ↑ indicates dispatch and ↓ indicates completion.

scheduler remains in the *Running* location until the running thread gets blocked due to shared resources, until the running thread completes its execution, until another thread is dispatched, or until the running thread misses its deadline. A running thread that gets blocked due to shared resources synchronizes with the scheduler back to $Schedule1$, whereby the blocked thread is removed from the *ready_queue* and preemption time of possible preempted threads is adjusted with respect to the execution time of the blocked thread. If a running thread completes its execution (corresponds to $c_A = r_A$), the scheduler transits to the *LookUp* location through one of two different edges. Note that the running location is modeled with an invariant in order to force a fire of the completion edge at the time of completion. The two edges have guards for execution time where additional expressions are used to differentiate between a preempted thread and a thread which has not been preempted. If the thread has not been preempted, the thread is simply removed from the *ready_queue* (through the *completion*() function). A preempted thread, on the other hand, is not only removed from the *ready_queue*, but also from the *preempt_stack*. From the *LookUp* location, the scheduler transits to the *Empty* or *Schedule1* location depending on whether there are any dispatched threads left or not. If dispatched threads still exist, the scheduler synchronizes with possibly blocked threads to check the availability of shared resources in response to the thread completion. This allows for unblocking of threads when the scheduler enters $Schedule1$.

If a dispatch occurs when the scheduler is in the *Running* location, an edge is fired to the *Schedule2* location, whereupon the thread is added to the *ready_queue* and the corresponding execution time clock is reset. Three different edges are available from the *Schedule2* location depending on if the recently dispatched thread was scheduled as the first thread in the *ready_queue* or not. If scheduled as the first thread in the *ready_queue*, that is, if it preempts the previously running thread, the scheduler transits to the *Preemption* location through one of the two edges depending on whether the preempted thread already exist in the *preempt_stack* or not. Whereby the edges from the *Schedule2* location to the *Preemption* location, the preempted thread is added to the *preempt_stack* if it previously has not been preempted, and preempted time is added – to all preempted threads – through the *addTime()* function (corresponds to $r_A = C_A + C_B$ or $r_A = C_A + C_B + C_C$). On the other hand, if the recently dispatched thread does not cause a preemption, no further actions are taken other than adding preempted time – if the thread is scheduled prior to currently preempted threads – to preempted threads through the *checkTime()* function. From the *Preemption* location, the scheduler synchronizes with the first thread in the *ready_queue* for execution. Note that the *Preemption* location has an edge to itself to allow simultaneous dispatches of threads.

The edge from the *Running* location to the *MissedDeadline* location is modeled for schedulability analysis. The *Running* location is modeled with an invariant that forces a fire of the edge whenever the running thread misses its deadline (corresponds to $d_A > D_A$). Thus, any internal, direct, or indirect path that is not consistent with the scheduling properties cannot be reached.

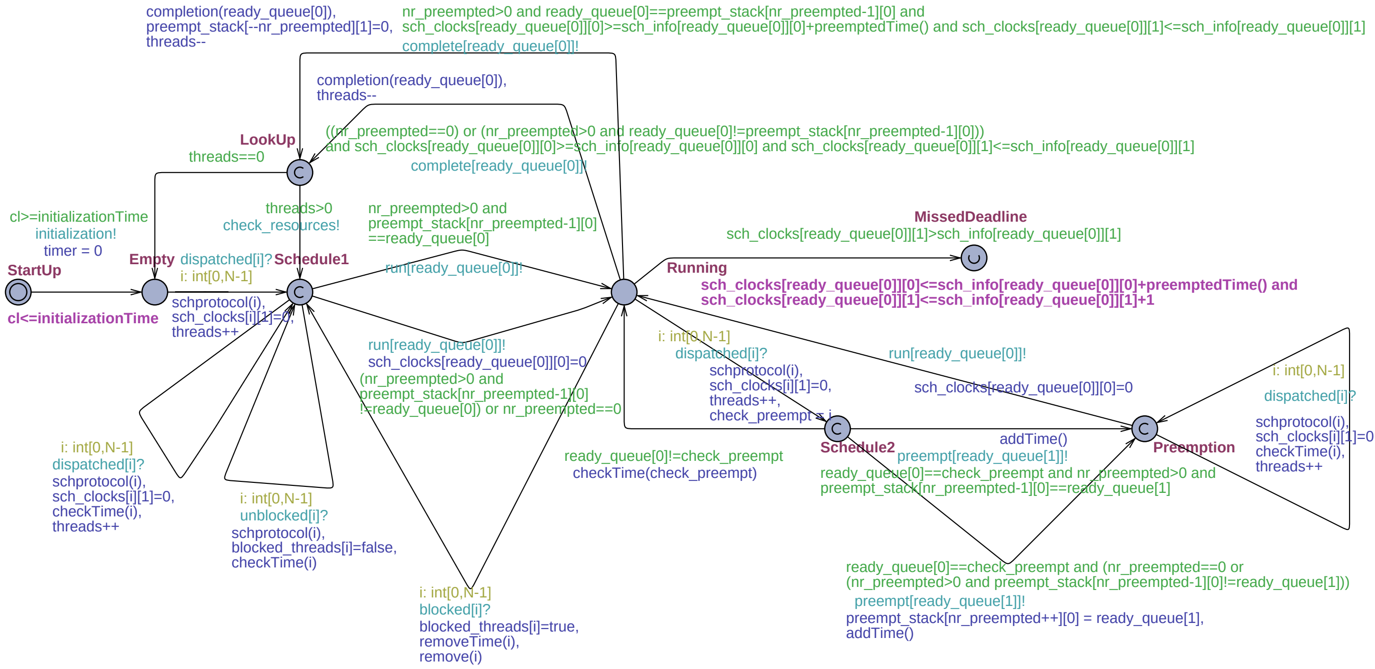

completion(ready_queue[0]),
preempt_stack[--nr_preempted][1]=0,
threads--
nr_preempted>0 and ready_queue[0]==preempt_stack[nr_preempted-1][0] and
sch_clocks[ready_queue[0]][0]>=sch_info[ready_queue[0]][0]+preemptedTime() and sch_clocks[ready_queue[0]][1]<=sch_info[ready_queue[0]][1]
complete[ready_queue[0]]!
completion(ready_queue[0]),
threads--
((nr_preempted==0) or (nr_preempted>0 and ready_queue[0]!=preempt_stack[nr_preempted-1][0]))
and sch_clocks[ready_queue[0]][0]>=sch_info[ready_queue[0]][0] and sch_clocks[ready_queue[0]][1]<=sch_info[ready_queue[0]][1]
complete[ready_queue[0]]!
LookUp
threads==0
threads>0
check_resources!
cl>=initializationTime
initialization!
timer = 0
StartUp
cl<=initializationTime
Empty
dispatched[i]?
i: int[0,N-1]
Schedule1
schprotocol(i),
sch_clocks[i][1]=0,
threads++
nr_preempted>0 and
preempt_stack[nr_preempted-1][0]
==ready_queue[0]
run[ready_queue[0]]!
MissedDeadline
sch_clocks[ready_queue[0]][1]>sch_info[ready_queue[0]][1]
Running
sch_clocks[ready_queue[0]][0]<=sch_info[ready_queue[0]][0]+preemptedTime() and
sch_clocks[ready_queue[0]][1]<=sch_info[ready_queue[0]][1]+1
run[ready_queue[0]]!
sch_clocks[ready_queue[0]][0]=0
(nr_preempted>0 and
preempt_stack[nr_preempted-1][0]
!=ready_queue[0]) or nr_preempted==0
i: int[0,N-1]
dispatched[i]?
schprotocol(i),
sch_clocks[i][1]=0,
threads++,
check_preempt = i
run[ready_queue[0]]!
sch_clocks[ready_queue[0]][0]=0
i: int[0,N-1]
dispatched[i]?
schprotocol(i),
sch_clocks[i][1]=0,
checkTime(i),
threads++
i: int[0,N-1]
dispatched[i]?
schprotocol(i),
sch_clocks[i][1]=0,
checkTime(i),
threads++
i: int[0,N-1]
unblocked[i]?
schprotocol(i),
blocked_threads[i]=false,
checkTime(i)
ready_queue[0]!=check_preempt
checkTime(check_preempt)
Schedule2
addTime()
preempt[ready_queue[1]]!
Preemption
ready_queue[0]==check_preempt and nr_preempted>0 and
preempt_stack[nr_preempted-1][0]==ready_queue[1]
ready_queue[0]==check_preempt and (nr_preempted==0 or
(nr_preempted>0 and preempt_stack[nr_preempted-1][0]!=ready_queue[1]))
preempt[ready_queue[1]]!
preempt_stack[nr_preempted++][0] = ready_queue[1],
addTime()
i: int[0,N-1]
blocked[i]?
blocked_threads[i]=true,
removeTime(i),
remove(i)

Fig. 19. The scheduler automaton template.

9 Observers generation and model checking

Once the UPPAAL model has been generated, the possible paths of the AFG can be verified against their constraints (properties), and possibly requirements, through flow-reachability analysis. Note that properties that have an effect on the dynamic semantics are transformed into the timed automata model, such as scheduling properties. They are therefore not explicitly included in verification sequences as their validity automatically is verified when the corresponding timed automata paths are exercised. Each verification sequence is executed through transformation to an observer automaton [19] and auxiliary variables and clocks (if it is constrained by timing properties). Observers have been developed to provide a flexible method for specifying coverage criteria for model checking and model-based test case generation. The execution is formulated as a reachability problem, which conforms to our verification criteria. An observer essentially monitors a trace of the timed automata model and reaches an acceptance state whenever the coverage criterion has been met. With respect to verification sequences, reaching an acceptance state denotes flow-reachability of the corresponding AFG path. Thus, reaching all acceptance states imply consistency, completeness (assuming all input classes have been used), and correctness of the AADL model. Validity is preserved as observers cannot interfere with the state space.

Formally, an observer $\langle O, o_0, o_{accept}, E_{obs}\rangle$ over a set of auxiliary clocks and variables has a set of observer locations O, an initial observer location $o_0 \in O$, an accepting location $o_{accept} \in O$, and a set of observer edges E_{obs} on the form $o \xrightarrow{g,a,u} o'$. A coverage criterion is created by dividing it into atomic timed automata items that must be covered and, for each item, generate an observer edge which predicate (g and a) is dependent on that item. An observer edge will thereby be fired when the item has been covered. If the criterion requires the items to be covered in a specific sequence, the edges are structured correspondingly. Moreover, locations may be labelled with invariants (including urgent and committed) and guards, actions, and clocks may be used to specify additional constraints in which items must be covered. With respect to a verification sequence, the coverage criterion is the corresponding timed automata path. Since each control flow arc in a verification sequence corresponds to an edge in the timed automata model, and each data flow arc to a sequence of two edges (one where the variable is defined and one where it is used), the corresponding edges or sequence of edges are the atomic items to be covered. Thus, an observer automaton is created for each verification sequence by creating an observer edge or a sequence of two edges for each arc in the path. In addition, each path constraint (property) and requirement is specified through location invariants and transition guards and actions.

Assuming no existence of data flows, a verification sequence of m vertices $\langle v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \cdots \rightarrow v_m, \{properties\}\rangle$ maps to an observer automaton of $m-1$ observer edges $\langle \{o_1, o_2, o_3, \ldots o_m\}, o_1, o_m, \{o_1 \xrightarrow{g,a,u} o_2, o_2 \xrightarrow{g,a,u} o_3, \ldots, o_{m-1} \xrightarrow{g,a,u} o_m\}\rangle$, where an execution of the timed automaton edge that corresponds to $v_1 \to v_2$ (see correspondence below) is observed by $o_1 \xrightarrow{g,a,u} o_2$, $v_2 \to v_3$ by $o_2 \xrightarrow{g,a,u} o_3$, etc. If the sequence contains any control flow due to a true or false evaluation of a control expression, it must be complemented with an observer edge that resets the observer to its initial location in case the complementing branch is fired instead. For example, a verification sequence $\langle v_1 \to_c v_2 \to_{cT} v_3 \to_{cF} v_4, \{properties\}\rangle$ maps to $\langle \{o_1, o_2, o_3, o_4\}, o_1, \{o_3\}, \{o_1 \xrightarrow{g,a,u} o_2, o_2 \xrightarrow{g,a,u} o_3, o_2 \xrightarrow{g,a,u} o_1, o_3 \xrightarrow{g,a,u} o_4, o_3 \xrightarrow{g,a,u} o_1\}\rangle$ where the timed automaton edge that corresponds to $v_2 \to_{cF} v_x$ is observed by $o_2 \xrightarrow{g,a,u} o_1$ and $v_3 \to_{cT} v_y$ by $o_3 \xrightarrow{g,a,u} o_1$.

Given that there exist an automaton that may stimulate the model with the possible system inputs, the verification sequence is executed by the reachability formula $E <> o_m$ (meaning "there exists one path where o_m eventually holds"). The verification sequence passes if the model satisfies the formula.

An observer edge primarily observers the coverage item through a broadcast synchronization channel, i.e., given that an execution of $\ell \xrightarrow{g,a_x,u} \ell'$ corresponds to $v_1 \to v_2$, $\ell \xrightarrow{g,a_x!,u} \ell'$ synchronizes with $o_1 \xrightarrow{g,a_x?,u} o_2$ through channel a_x. By following the transformation rules from AADL to timed automata, each control flow arc corresponds to the execution of exactly one edge (coverage item).

- An arc $\langle\langle \text{“}ENTRY\text{”}/\text{“}REENTRY\text{”}, thread_i\rangle, v'\rangle_c$ corresponds to an execution of the edge $awaiting\_dispatch \xrightarrow{cl>=Period,a,u} ready$ in the automaton of $thread_i$.
- An arc $\langle\langle \text{“}ENTRY\text{”}, sub_i\rangle, v'\rangle_c$ corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ where u represents formal-in assignments of sub_i.
- An arc $\langle g_x, act_y\rangle_{cT}$ corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ where g represents g_x.
- An arc $\langle act_x, act_y/g_y\rangle_c$ corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ where some assignment $u_z \in u$ represents act_x.
- An arc $\langle sub_i!(argument\_list), \langle \text{“}ENTRY\text{”}, sub_i\rangle\rangle_{call}$ corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ where u represents actual-in assignments of the call to sub_i.
- An arc $\langle\langle \text{“}EXIT\text{”}, sub_i\rangle, sub_i!(argument\_list)\rangle_{call}$ corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ where u represents formal-out assignments of sub_i.
- An arc $\langle sub_i!(argument\_list), v'\rangle_c$ corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ where u represents actual-out assignments of the call to sub_i.
- An arc $\langle g_x/act_y, \langle \text{“}REENTRY\text{”}/\text{“}EXIT\text{”}, thread_i\rangle\rangle_c$ corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} awaiting\_dispatch$, where g represents g_x/some assignment $u_z \in u$ represents act_y.
- An arc $\langle g_x/act_y, \langle \text{“}EXIT\text{”}, sub_i\rangle\rangle_c$ corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$, where g represents g_x/some assignment $u_z \in u$ represents act_y.

A verification sequence that contains a data-flow arc requires observer edges that observe at least two successive coverage items: the edge where the data is

defined followed by the edge where it is used. In addition, two auxiliary variables, v_{aux1} and v_{aux2}, are introduced together with the observer edges to ensure that the use edge actually uses the data instance defined by the definition edge. The data $Data_{def}$ that is defined at the definition edge, and the data $Data_{use}$ that is used at the use edge, are additionally stored in the auxiliary variables. Once the observer edges have observed a definition (through channel $Chan_{def}$) followed by a use (through channel $Chan_{use}$) of the data component, a guard g composed of predicate $v_{aux1} == v_{aux2}$ of an edge $o \xrightarrow{g,a,u} o'$ ensures data flow reachability before the accepting state o' is reached. Nevertheless, in case of inter-component data flows, threads may be modeled with under-sampled data communication (the receiving thread has a lower dispatch frequency than the sending thread) where a fraction of defined data instances are not supposed to reach the us edge. To prevent false negatives of data flow reachability, alternative definition-observer edges may synchronize with new definitions of the data component. Consequently, for sequences that contain an inter-component data flow arc $v_1 \rightarrow v_2$, i.e. $\langle v_1, v_2 \rangle_{d-in/d-out}$, the two coverage items (def and use) are observed by two sequential observer edges, one for possibly under-sampled communication, and one for assurance of data-flow reachability:
$\langle \{o_1, o_2, o_3, o_4\}, o_1, o_4, \{o_1 \xrightarrow{g,Chan_{def}?,\langle v_{aux1}:=Data_{def}\rangle} o_2,$
$o_2 \xrightarrow{g,Chan_{def}?,\langle v_{aux1}:=Data_{def}\rangle} o_2, o_2 \xrightarrow{g,Chan_{use}?,\langle v_{aux2}:=Data_{use}\rangle} o_3,$
$o_3 \xrightarrow{v_{aux1}==v_{aux2},\tau,u} o_4\}\rangle$
where $Chan_{def}?$ observes the definition edge that corresponds to v_1 and $Chan_{use}?$ observes the use edge that corresponds to v_2. The correspondence of inter-component data flow arcs are as follows:

- An arc $\langle v, v' \rangle_{d-in}$, where v is an actual-in vertex of $thread_a$ and v' a formal-in vertex of $thread_b$, corresponds to an execution of the edge $running \xrightarrow{g,a,u} awaiting\_dispatch$ of $thread_a$ and subsequently an execution of the edge $awaiting\_dispatch \xrightarrow{g,a,u} ready$ of $thread_b$.
- An arc $\langle v, v' \rangle_{d-in}$, where v is an actual-in vertex of $comp_a$ (either a thread or subprogram) and v' a formal-in vertex of sub_b, corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ of $comp_a$ where an element of u represents the actual-in assignment v, and subsequently an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ of sub_b where an element of u represents the formal-in assignment v'.
- An arc $\langle v, v' \rangle_{d-out}$, where v is an formal-out vertex of sub_a and v' a actual-out vertex of $comp_b$, corresponds to an execution of the edge $\ell \xrightarrow{g,a,u} \ell'$ of sub_a where an element of u represents formal-out assignment v, and subsequently an execution of the edge $\ell' \xrightarrow{g,a,u} \ell''$ of $comp_b$ where an element of u represents actual-out assignment v'.

For example, the direct component to component path $P = connection3 := DHWOMG\_S\_LtrCd \rightarrow DHWOMG\_S\_LtrCd := connection$ corresponds to a verification sequence $\langle P, \{\langle\langle connection3 := DHWOMG\_S\_LtrCd,$ $DHWOMG\_S\_LtrCd := connection3\rangle_{d-in}, Latency => 0ms..3ms\rangle\}\rangle$. Assuming that the time units in the Uppaal model are milliseconds, the verification sequence generates, according to above rules, an observer automaton $\langle\{o_1, o_2, o_{3(committed)}, o_4\}, o_1, o_4, \{o_1 \xrightarrow{g, Chan_{def}?, \langle v_{aux1}:=Data_{def}, cl=0\rangle} o_2,$ $o_2 \xrightarrow{g, Chan_{def}?, \langle v_{aux1}:=Data_{def}, cl=0\rangle} o_2, o_2 \xrightarrow{g, Chan_{use}?, \langle v_{aux2}:=Data_{use}\rangle} o_3,$ $o_3 \xrightarrow{cl>=0 \text{ } and \text{ } cl<=3 \text{ } and \text{ } v_{aux1}==v_{aux2}, \tau, \langle\rangle} o_4\}\rangle$ where $Chan_{def}$ synchronizes with $running \xrightarrow{g_{ctrl}, a_{ctrl}, u_{ctrl}} awaiting\_dispatch$ of the controller automaton; $Data_{def}$ is a copy of the value assigned to variable *connection*3 in u_{ctrl}; $Chan_{use}$ synchronizes with $awaiting\_dispatch \xrightarrow{g_{tst}, a_{tst}, u_{tst}} ready$ of the tester automaton; and $Data_{use}$ is a copy of the value assigned to $DHWOMG\_S\_LtrCd$ in u_{tst}. Note that a clock cl is used to verify the validity of the latency property; the data should be received at least after 0 millisecond and at most after 3.

For sequences that contain a component-internal data flow arc, the coverage item is decomposed to and observed as the underlying control-flow path.

9.1 Results

In order to exercise the architectural paths by every class of input, the input domain of LTRIS is divided through equivalence portioning where concrete values are determined through boundary value analysis. In the LTRIS AADL model, we assume that any required input is generated by the *LineTripEnvironment* process. The abstracted environment is transformed to the automaton presented in Figure 17, which should be able to stimulate LTRIS with the possible input classes at completions. In addition, the values of (input) data objects at the time of LTRIS initialization depends on its environment. The possible stimuli is added to the timed automata model by creating an automaton for each input object such that it may assign any of the possible concrete values to the object in response to LTRIS initialization and *LineTripEnvironment* completions. A template for a Boolean-typed data object is presented in Figure 20.

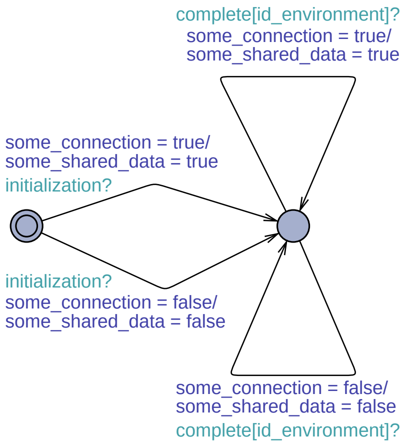


Fig. 20. Template for input generation of a boolean typed connection/shared variable.

The possible boundary values of each data object are presented in Table 9. Note that these values are generated based on informal assumptions about the input domain as the specifics of the environment are unknown.

Table 9. Value sets of each input data object.

Data object	**Initialization**	**Env. completion**
some_connection1-2	{0(false),1(true)}	{0(false),1(true)}
some_connection3	{0,3,4,27,30,31,38}	{0,3,4,27,30,31,38}
some_connection4-14	{0(false),1(true)}	{0(false),1(true)}
GPIO_OUT	{0(false)}	
LTRIP_EN_N	{1(true)}	
MCU_LT_ON	{1(true)}	
FPGA2_LT_ON	{0(false),1(true)}	{0(false),1(true)}
LT_RELAY_FB	{0(false),1(true)}	{0(false),1(true)}

The results of model checking are presented in Table 10 and descriptive statistics for the data set in Table 11. The results conform to the expectations. All observers were satisfied by the model. On average, satisfiability checking consumed 15 seconds and 154 MB per observer. The average trace size is 1.8 MB. In total, it took 855 seconds and 9327 MB to check satisfiability of the 57 observers. The aggregate set of traces yields 105 MB. The distribution of each unit is positively skewed where occurrences are clustered in the lower end of the scale. The inter-observer coverage for the traces that satisfied each observer in this run is listen in Table 12. The information may be used to reduce the resource consumption of regression verification, as described in Section 11.

Table 10. Model checking results

Observer	**Verdict**	**Time cons.(sec)**	**Memory cons. (MB)**	**Trace size (KB)**
InternalTester1	satisfied	1	36	119
InternalLtrTsSq1	satisfied	1	35	29
InternalLtrTsSq2	satisfied	1	36	29
InternalLtrTsSq3	satisfied	1	40	391
InternalLtrTsSq4	satisfied	17	98	3383
InternalLtrTsSq5	satisfied	28	300	3676
InternalLtrTsSq6	satisfied	28	300	3676
InternalLtrTsSq7	satisfied	43	438	4049
InternalLtrTsSq8	satisfied	43	438	4049
InternalLtrTsSq9	satisfied	1	40	391
InternalLtrTsSq10	satisfied	1	39	390
InternalLtrTsSq11	satisfied	1	40	390
InternalLtrTsSq12	satisfied	2	65	756
InternalLtrTsSq13	satisfied	26	283	3675
InternalLtrTsSq14	satisfied	42	417	4047
InternalLtrTsSq15	satisfied	42	417	4047
InternalLtrTsSq16	satisfied	59	649	4420
InternalLtrTsSq17	satisfied	59	649	4420
InternalLtrTsSq18	satisfied	2	78	756
InternalLtrTsSq19	satisfied	27	289	3676
InternalController1	satisfied	1	56	25
InternalLtrInt1	satisfied	1	57	18
Internaldcu2_line_trip1	satisfied	1	59	34
Internaldcu2_line_trip2	satisfied	1	39	34
Internaldcu2_line_trip3	satisfied	1	40	34
Internaldcu2_line_trip4	satisfied	1	40	34
Internaldcu2_line_trip5	satisfied	1	39	34
Internaldcu2_line_trip6	satisfied	1	41	34
Internaldcu2_line_trip7	satisfied	1	41	34
Internaldcu2_line_trip8	satisfied	1	41	34
Internaldcu2_line_trip9	satisfied	1	41	63
Internaldcu2_line_trip10	satisfied	1	48	365
Internaldcu2_line_trip11	satisfied	1	48	365
Internaldcu2_line_trip12	satisfied	1	42	63
Direct1	satisfied	1	38	27
Direct2	satisfied	1	39	7
Direct3	satisfied	1	39	19
Direct4	satisfied	1	40	125
Direct5	satisfied	1	40	125
Direct6	satisfied	1	42	379
Indirect1	satisfied	1	42	145
Indirect2	satisfied	1	42	145
Indirect3	satisfied	29	307	3917
Indirect4	satisfied	29	308	3917
Indirect5	satisfied	17	108	3531
Indirect6	satisfied	17	108	3531
Indirect7	satisfied	17	108	3531
Indirect8	satisfied	17	108	3531
Indirect9	satisfied	17	105	3531
Indirect10	satisfied	18	111	3531
Indirect11	satisfied	23	135	3521
Indirect12	satisfied	18	112	3521
Indirect13	satisfied	30	334	3909
Indirect14	satisfied	30	335	3909
Indirect15	satisfied	30	334	3909
Indirect16	satisfied	66	642	4295
Indirect17	satisfied	50	531	4292

Table 11. Descriptive statistics of Table 10

	Time cons. (sec)	**Memory cons. (MB)**	**Trace size (KB)**
Mean	15	164	1840
Median	1	59	391
SD	18	178	1853
Minimum	1	35	7
Maximum	66	649	4420
Sum	855	9327	104888
N	57	57	57

Table 12. Inter-observer coverage

	InternalTester1	InternalLtrTsSq1	InternalLtrTsSq2	InternalLtrTsSq3	InternalLtrTsSq4	InternalLtrTsSq5	InternalLtrTsSq6	InternalLtrTsSq7	InternalLtrTsSq8	InternalLtrTsSq9	InternalLtrTsSq10	InternalLtrTsSq11	InternalLtrTsSq12	InternalLtrTsSq13	InternalLtrTsSq14	InternalLtrTsSq15	InternalLtrTsSq16	InternalLtrTsSq17	InternalLtrTsSq18	InternalLtrTsSq19	InternalController1	InternalLtrInt1	Internaldcu2_line_trip1	Internaldcu2_line_trip2	Internaldcu2_line_trip3	Internaldcu2_line_trip4	Internaldcu2_line_trip5	Internaldcu2_line_trip6	Internaldcu2_line_trip7	Internaldcu2_line_trip8	Internaldcu2_line_trip9	Internaldcu2_line_trip10	Internaldcu2_line_trip11	Internaldcu2_line_trip12	Direct1	Direct2	Direct3	Direct4	Direct5	Direct6	Indirect1	Indirect2	Indirect3	Indirect4	Indirect5	Indirect6	Indirect7	Indirect8	Indirect9	Indirect10	Indirect11	Indirect12	Indirect13	Indirect14	Indirect15	Indirect16	Indirect17
InternalTester1	x		x																		x	x									x				x	x	x	x																				
InternalLtrTsSq1		x																			x	x									x					x	x	x																				
InternalLtrTsSq2			x																		x	x									x					x	x	x																				
InternalLtrTsSq3	x		x	x								x									x	x									x				x	x	x	x	x	x	x		x															
InternalLtrTsSq4	x		x		x							x									x	x								x	x				x	x	x	x	x	x	x		x															
InternalLtrTsSq5	x		x		x	x						x		x							x	x								x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq6	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq7	x		x		x		x	x				x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq8	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq9	x	x								x	x										x	x									x				x	x	x	x	x	x	x																	
InternalLtrTsSq10	x	x									x										x	x									x				x	x	x	x	x	x	x																	
InternalLtrTsSq11	x		x									x									x	x									x				x	x	x	x	x	x	x		x															
InternalLtrTsSq12	x		x	x								x	x								x	x									x				x	x	x	x	x	x	x		x															
InternalLtrTsSq13	x		x		x							x		x							x	x								x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq14	x		x		x	x						x		x	x						x	x								x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq15	x		x		x		x					x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq16	x		x		x		x	x				x		x		x	x			x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		
InternalLtrTsSq17	x		x		x		x		x			x		x		x		x		x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		x
InternalLtrTsSq18	x	x								x	x								x		x	x									x				x	x	x	x	x	x	x																	
InternalLtrTsSq19	x		x		x							x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalController1																					x	x									x						x	x																				
InternalLtrInt1																						x															x																					
Internaldcu2_line_trip1																						x	x														x	x																				
Internaldcu2_line_trip2																						x		x													x	x																				
Internaldcu2_line_trip3																						x			x												x	x																				
Internaldcu2_line_trip4																						x				x											x	x																				
Internaldcu2_line_trip5																						x					x										x	x																				
Internaldcu2_line_trip6																						x						x									x	x																				
Internaldcu2_line_trip7																						x							x								x	x																				
Internaldcu2_line_trip8																						x								x							x	x																				
Internaldcu2_line_trip9			x																		x	x					x				x				x	x	x																					
Internaldcu2_line_trip10	x		x	x								x									x	x					x			x	x	x		x	x	x	x	x	x	x	x																
Internaldcu2_line_trip11	x	x								x	x										x	x					x			x	x		x	x	x	x	x	x	x	x																	
Internaldcu2_line_trip12			x																		x	x									x				x	x	x	x																				
Direct1																					x	x									x					x	x	x																				
Direct2																																				|x|																				
Direct3																						x															x	x																				
Direct4	x		x																		x	x									x				x	x	x	x	x																			
Direct5	x		x																		x	x									x				x	x	x	x		x																		
Direct6	x	x																			x	x									x				x	x	x	x	x	x	x																	
Indirect1	x		x																		x	x					x					x				x	x	x	x	x		x																
Indirect2	x		x																		x	x									x				x	x	x	x	x	x			x															
Indirect3	x		x		x		x					x		x						x	x	x			x	x	x					x				x	x	x	x	x	x	x		x					x			x			x			
Indirect4	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect5	x		x		x							x									x	x	x				x					x				x	x	x	x	x	x	x				x						x						
Indirect6	x		x		x							x									x	x		x			x					x				x	x	x	x	x	x	x					x					x						
Indirect7	x		x		x							x									x	x			x		x					x				x	x	x	x	x	x	x						x				x						
Indirect8	x		x		x							x									x	x				x	x					x				x	x	x	x	x	x	x							x			x						
Indirect9	x		x		x							x									x	x						x		x				x	x	x	x	x	x	x		x							x			x					
Indirect10 68	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x								x		x					
Indirect11	x		x		x							x									x	x				x	x					x				x	x	x	x	x	x	x										x						
Indirect12	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x										x					
Indirect13	x		x		x		x					x		x						x	x	x		x		x	x					x				x	x	x	x	x	x	x							x			x		x				
Indirect14	x		x		x		x					x		x						x	x	x			x	x	x					x				x	x	x	x	x	x	x							x			x			x			
Indirect15	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x			x		
Indirect16	x		x		x		x		x			x		x		x				x	x	x			x	x	x					x				x	x	x	x	x	x	x		x	x	x				x			x				x	x	
Indirect17	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		x

10 Model-based testing

To verify an implementation, its conformance to the – complete, consistent, and correct – model must be tested. A satisfied observer generates a *trace* $\langle\overline{\ell_o}, \overline{\phi_o}, \overline{\sigma_o}\rangle \xrightarrow{a_1/d_1} \langle\overline{\ell_1}, \overline{\phi_1}, \overline{\sigma_1}\rangle \xrightarrow{a_2/d_2} \cdots \xrightarrow{a_n/d_n} \langle\overline{\ell_n}, \overline{\phi_n}, \overline{\sigma_n}\rangle$ that contains information about the initial state of the system and its environment before the path is executed, the input or the sequence of inputs needed to stimulate an execution of the system according to the expected path, and the expected output or sequence of outputs. In addition, the trace holds information on expected non-functional properties, including timing of input and output.

Thus, depending on which automata are accredited as an environment $\overline{E}(p_i) = \langle TA_1, TA_2, TA_3, \dots\rangle$ for a specific verification sequence with path p_i, an observer trace over its $E(p_i)$ yields a test case. Let $MV(p_i)$ and $MAct(p_i)$ denote the sets of variables and actions (on the form $a!$) in environment $E(p_i)$ that are monitored by system under test (SUT). Let $CV(p_i)$ and $CAct(p_i)$ denote the sets of variables and actions (on the form $a?$) in $E(p_i)$ that are controlled by SUT. For end-to-end paths, sensor variables and actions are monitored while actuator variables and actions are controlled.

Assuming that the SUT at time $t = 0$ is set according to system state $\langle\overline{\ell_o}, \overline{\phi_o}, \overline{\sigma_o}\rangle$, depending on the used test harness, the tester, test script, or test model is responsible of following the sequence such that:

1. for each encountered *delay transition*$\langle\overline{\ell}, \overline{\phi}, \overline{\sigma}\rangle \xrightarrow{d} \langle\overline{\ell}, \overline{\phi} \oplus d, \overline{\sigma}\rangle$, wait until $t = t + d$.
2. for each encountered *discrete transition* $\langle\overline{\ell}, \overline{\phi}, \overline{\sigma}\rangle \xrightarrow{a} \langle\overline{\ell}[\ell'_i/\ell_i, \ell'_j/\ell_j, \ell'_k/\ell_k, \dots],$ $\overline{\phi'}, \overline{\sigma'}\rangle$ where $\ell_{i/j/k\dots} \xrightarrow{g_{i/j/k\dots}, a_{i/j/k\dots}, u_{i/j/k\dots}} \ell'_{i/j/k\dots}$ are edges of the environment $E(p_i)$ and $a_{i/j/k\dots}$ is a member of $MAct(p_i)$ and/or any assignment $u^x_{i/j/k\dots}$ is on the form $v := expr$ such that $v \in MV(p_i)$, stimulate SUT at time t with actions $a_{i/j/k\dots}$ and data updates $u^x_{i/j/k\dots}$.
3. for each encountered *discrete transition* where $\ell_{i/j/k\dots} \xrightarrow{g_{i/j/k\dots}, a_{i/j/k\dots}, u_{i/j/k\dots}} \ell'_{i/j/k\dots}$ are not edges of the environment $E(p_i)$ and $a_{i/j/k\dots}$ is a member of $CAct(p_i)$ and/or any assignment $u^x_{i/j/k\dots}$ is on the form $v := expr$ such that $v \in CV(p_i)$, assure at time t that SUT responds with actions $a_{i/j/k\dots}$ and data updates $u^x_{i/j/k\dots}$.

The collective set of generated tests creates a test suite that tests the conformance of the implementation with respect to the architecture model.

10.1 Results

The sets of controlled and monitored variables for each path are listed in Table 13. All connections from the environment to in ports of LTRIS, and shared variables which are read by LTRIS, constitute the set of monitored variables for all paths. The set of controlled variables for each internal path corresponds to each connection that is connected to an output interface of the component covered by the path. For direct paths, the controlled variables are the involved input interfaces of the destination component. Finally, each connection from an out port of LTRIS to the environment, and shared variables which are written by LTRIS, constitute the set of controlled variables for all indirect paths.

Table 13: Controlled and monitored variables for each path type

Path type p	$MV(p)$	$CV(p)$
InternalTester	some_connection1 some_connection2 some_connection3 some_connection4 some_connection5 some_connection6 some_connection7 some_connection8 LTRIP_EN_N MCU_LT_ON FPGA2_LT_ON LT_RELAY_FB some_connection9 some_connection10 some_connection12 some_connection13 some_connection14	connection1 connection2 some_connection15 some_connection16 some_connection17
InternalLtrTsSq	— ” —	Tester.B_OpLtr_out Tester.B_CdLtr_out Tester.NX_LtrSaSq_out Tester.A_LtrTs_out Tester.A_LtrOpVd_out
InternalController	— ” —	connection3 some_connection18 some_connection20 some_connection21 GPIO_OUT FPGA_LTRCR
InternalLtrInt	— ” —	Controller.B_ClLtr_out
Internaldcu2_line_trip	— ” —	GPIO_OUT FPGA_LTRCR Controller.fb_out Controller.fpga2_on_out Controller.fb_ne_out
Direct1	— ” —	Tester.C_LtrTs_in Tester.B_LtrFl_in Tester.S_LtrCd_in Tester.S_LtrOp_in Tester.S_DCUNtRdy_in

		Tester.L_EnLtrSv_in
Direct2	— ” —	Controller.B_RqPrSd_in Controller.B_OpLtr_AppSpec_in Controller.NX_SqSt_in Controller.A_PctMo_in Controller.B_LtrTsOpLtr_in Controller.B_OpLtr_LtrTs_in Controller.B_CdLtr_LtrTs_in Controller.B_EnCdLnTrpSlt_in Controller.L_CnfHpp_in
Direct3	— ” —	Controller.enable_in Controller.act_in
Direct4	— ” —	Controller.DHSSMG_B_OpLtr
Direct5	— ” —	Controller.DHSSMG_B_CdLtr
Direct6	— ” —	Tester.DHWOMG_S_LtrCd
Indirect	— ” —	connection3 GPIO_OUT FPGA_LTRCR some_connection15 some_connection16 some_connection17 some_connection18 some_connection20 some_connection21

According to this configuration, the results of the test case generation algorithm are presented in Table 28–46 in Appendix B. In the tables, a normal font denotes a test input, a cursive font denotes a delay, and a bold font denotes an expected output. For the purpose of the case study, these are scripted in timed automata. As an example, we consider the test case generated from the trace that satisfied the verification sequence (observer) verifying indirect path No. eight:

some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
GPIO_OUT = 0
LTRIP_EN_N = 1
MCU_LT_ON = 1

FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=60
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 0
FPGA2_LT_ON = 0
LT_RELAY_FB = 0
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=536
some_connection15 = 1
some_connection16 = 0
some_connection17 = 0
d=2
connection3 = 0
some_connection18 = 1
some_connection20 = 0
some_connection21 = 0
GPIO_OUT = 0
FPGA_LTRCR = 1

The test case is scripted in timed automata as shown in Fig. 21. The environment component in the LTRIS model is disconnected from the system when running the test scripts. Similarly to observers, if the test script may reach the accepted location, the test case passes, i.e., the system reacted as expected with respect to the input. The result of running all tests on the original LTRIS model is presented in Table 14. The result conforms to the expectation – all test cases passed.

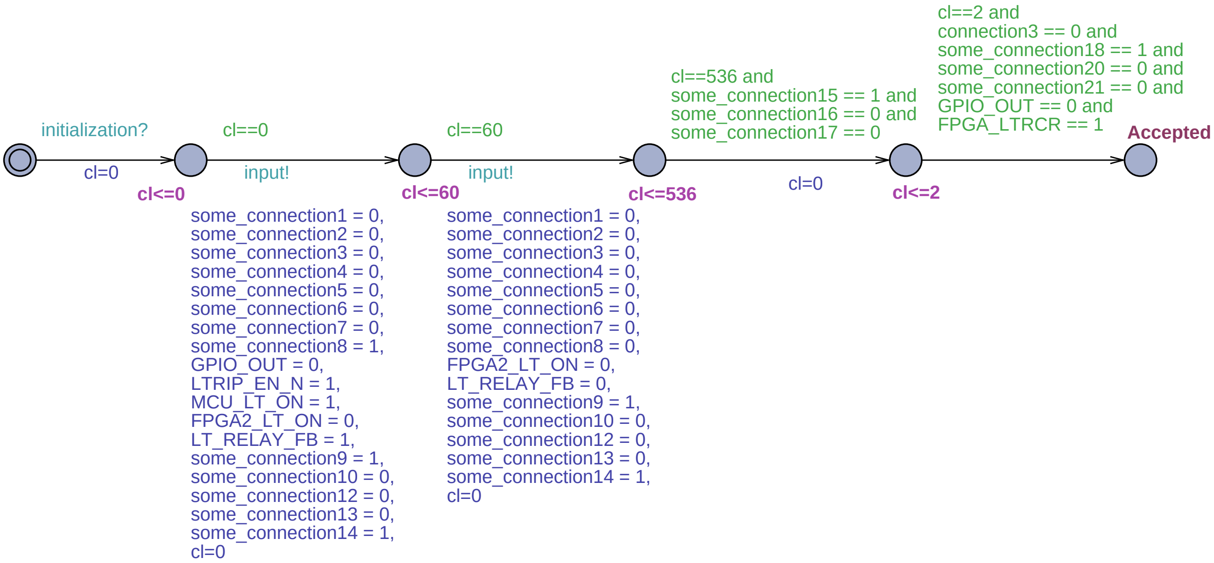


Fig. 21. TestcaseIndirect8 in timed automata.

Table 14. Model-based testing results

Test case	**Verdict**
TestcaseInternalTester1	passed
TestcaseInternalLtrTsSq1	passed
TestcaseInternalLtrTsSq2	passed
TestcaseInternalLtrTsSq3	passed
TestcaseInternalLtrTsSq4	passed
TestcaseInternalLtrTsSq5	passed
TestcaseInternalLtrTsSq6	passed
TestcaseInternalLtrTsSq7	passed
TestcaseInternalLtrTsSq8	passed
TestcaseInternalLtrTsSq9	passed
TestcaseInternalLtrTsSq10	passed
TestcaseInternalLtrTsSq11	passed
TestcaseInternalLtrTsSq12	passed
TestcaseInternalLtrTsSq13	passed
TestcaseInternalLtrTsSq14	passed
TestcaseInternalLtrTsSq15	passed
TestcaseInternalLtrTsSq16	passed
TestcaseInternalLtrTsSq17	passed
TestcaseInternalLtrTsSq18	passed
TestcaseInternalLtrTsSq19	passed
TestcaseInternalController1	passed
TestcaseInternalLtrInt1	passed
TestcaseInternaldcu2_line_trip1	passed
TestcaseInternaldcu2_line_trip2	passed
TestcaseInternaldcu2_line_trip3	passed
TestcaseInternaldcu2_line_trip4	passed
TestcaseInternaldcu2_line_trip5	passed
TestcaseInternaldcu2_line_trip6	passed
TestcaseInternaldcu2_line_trip7	passed
TestcaseInternaldcu2_line_trip8	passed
TestcaseInternaldcu2_line_trip9	passed
TestcaseInternaldcu2_line_trip10	passed
TestcaseInternaldcu2_line_trip11	passed
TestcaseInternaldcu2_line_trip12	passed
TestcaseDirect1	passed
TestcaseDirect2	passed
TestcaseDirect3	passed
TestcaseDirect4	passed
TestcaseDirect5	passed
TestcaseDirect6	passed
TestcaseIndirect1	passed
TestcaseIndirect2	passed
TestcaseIndirect3	passed
TestcaseIndirect4	passed
TestcaseIndirect5	passed
TestcaseIndirect6	passed
TestcaseIndirect7	passed
TestcaseIndirect8	passed
TestcaseIndirect9	passed
TestcaseIndirect10	passed
TestcaseIndirect11	passed
TestcaseIndirect12	passed
TestcaseIndirect13	passed
TestcaseIndirect14	passed
TestcaseIndirect15	passed
TestcaseIndirect16	passed
TestcaseIndirect17	passed

11 Selective regression verification

Given a model M, and possibly an implementation of the model $IMPL$, for which a verification suite VS of verification sequences has been generated and executed on M as described in Section 9, and on $IMPL$ as described in Section 10, it is likely that M eventually is modified into another version M', which later may be modified into another version M'', and so forth. A conventional approach to regression verification would be to ensure that a modification has not introduced faults in the model M' and has not violated the conformance with the implementation $IMPL$ by (1) re-executing all "old" but still valid verification sequences $VS'_{old} \subseteq VS$ on M' and $IMPL$, and, if the modification includes an added functionality, behavior, or property, (2) generate a new verification suite VS'_{new} that covers the added part(s) and execute it. A modification corresponds to the set of expressions (vertices) and flows (arcs) that are different among the models, i.e., expressions that exist in one version but not in the other. Re-execution of all valid verification sequences is inefficient if the modification does not affect the complete architecture. Moreover, determining which ones that still are valid and new sequences that are necessary to cover new parts is difficult. The problem is that the effect of a modification on the remaining architecture is complex to manually trace. In order to perform regression verification efficiently, the framework includes a technique that selectively re-executes only those verification sequences that are affected by the modification and generates new verification sequences that only cover added parts.

The technique uses the concept of *specification slicing* [23] through architecture dependence graphs (ADGs), to exactly identify the parts of a modified AADL model that directly or indirectly are affected by the modification and must be covered by verification sequences in the regression verification process. The concept of slicing is to remove statements that do not have an effect on and are not affected by the value of a variable at some statement. ADGs provide these dependencies such that causality can be precisely traced. The approach is to apply this idea to variables of the changed or added part such that other parts of the model which behavior now might behave incorrectly are identified for regression verification.

The first step is to determine what expressions, or flows to an expression, that have been removed or changed or added. This is simply done by comparing AFG' of M' with AFG of M to determine the set of removed vertices and arcs $AFG \backslash AFG'$ and the set of added or changed vertices and arcs $AFG' \backslash AFG$. VS'_{old} is thereby easily computed: any $vs \in VS$ that covers a vertex or arc in $(AFG' \backslash AFG) \cup (AFG \backslash AFG')$ is no longer valid. Invalid verification sequences are discarded in the regression verification process if the corresponding architectural paths are removed by the modification. If the paths still exist, the verification sequences are updated according to the modification to become valid. Nevertheless, valid verification sequences that do not cover the parts that are affected by the modification are unnecessary to re-execute on M'. Affected vertices V'_{aff} are determined through forward-slicing of the ADG'. The ADG' is generated from the AFG' according to Section 11.1. The regression verification suite

VS'_{old} is subsequently efficiently executed by only selecting verification sequences that cover vertices in V'_{aff}.

The set of affected vertices in relation to old verification sequences may be further trimmed by means of the inter-observer coverage data (Table 12) from the preceding verification cycle. Under the assumption that all observers were satisfied, satisfiability (reachability) independence between observers can be deduced from the data, which in turn may provide an even more precise slice with respect to the regression verification process. Meanwhile a marked intersection of two observers imply that the trace that satisfied the observer on the vertical axis also satisfied the observer on the horizontal axis, the absence of a coverage mark imply that the vertical axis observer is satisfiable independently from the satisfaction of the horizontal axis observer. Note that the contrary does not (necessarily) hold, i.e. that a marked intersection imply that the satisfiability of the former is dependent on the latter, since the data only represents one out of possibly several traces. Provided that the data set is generated from full path coverage, a previously satisfied observer which satisfiability is independent to each observer that covers the modification will also be satisfiable in the regression verification process. Thus, the vertices in the corresponding observed path may be reliably removed from the slice.

Finally, changed and added vertices and arcs must be covered with new verification sequences (unless updates of old verification sequences maintain full coverage). VS'_{new} is generated by applying the verification criteria to the changed and added set $AFG' \backslash AFG$, from which the possible new paths and corresponding set of verification sequences are extracted. If yet another version M'' is developed, the regression verification process is repeated upon the verification history $VS' = VS'_{old} \cup VS'_{new}$, instead of VS.

11.1 Generation of architecture dependence graphs

Let $EXPR$ be the set of possible expressions described by the abstract syntax. The slicing algorithm we define builds on the general definition of program slicing, originally defined by Weiser [24]:

Definition 8 A backward slice of an AADL model with respect to slicing criterion $CRI = \langle expr, var \rangle$, where $expr \in EXPR$ and var is a variable or data component defined or used at $expr$, consists of all control flow and data flow determining expressions of the model that the value of var at $expr$ possibly depend on. A forward AADL slice with respect to slicing criterion $CRI = \langle expr, var \rangle$ consists of all control flow and data flow determining expressions of the model that possibly are dependent on the value of var at $expr$.

Consequently, an architecture flow graph provides the necessary information basis for conducting slicing of AADL models. There exist two types of dependencies: control dependence and data dependence.

Definition 9 An AADL expression $expr_1 \in EXPR$ is control dependent on an AADL expression $expr_2 \in EXPR$ if $expr_2$ possibly decides whether $expr_1$ will be executed or not. $expr_1$ is data dependent on $expr_2$ if $expr_2$ defines a data variable possibly used by an execution of $expr_1$.

Data dependencies are therefore synonymous to data flows in an AFG. Control dependencies, on the other hand, are determined by performing post-domination analysis of the component-internal control flows of the AFG. Assume that v_x, v_y, and v_z are non-actual in/out and non-formal in/out vertices and contained within the same component. A vertex v_x is post-dominated by a vertex v_y if every path $P = v_1 \rightarrow_c v_2 \rightarrow_c \cdots \rightarrow_c v_n$ from v_x to the EXIT vertex (i.e. $v_1 = v_x$ and v_n is the exit vertex) includes v_y. Control dependency is then defined as:

Definition 10 A vertex v_y is control dependent on a vertex v_x iff **1)** v_x is an $ENTRY$ vertex and v_y is not nested within any loop or conditional vertex, or **2)** there exists a path $P = v_1 \rightarrow_c v_2 \rightarrow_c \cdots \rightarrow_c v_n$ from v_x to v_y such that any vertex v_z in P is post-dominated by v_y, and v_x is not post-dominated by v_y (v_x must be a control expression).

An algorithm to generate control dependencies based on this definition can be found in [25]. With respect to component interactions, interaction-based control flows (on the form $v \rightarrow_{c-inter} v'$) and calls (on the form $v \rightarrow_{call} v'$) are themselves control dependencies since the source vertex initiates the execution of the target vertex. The union of data dependencies and control dependencies form the architecture dependence graph, which formally is a directed graph $ADG = \langle V, A \rangle$ where the set of vertices is equal to the AFG and the set of arcs represent control and data dependencies. An arc $v \rightarrow v'$ of an ADG denotes that v' is control or data dependent on v. The ADG of LTRIS with data dependencies excluded is partly presented in Figure 22 and 23.

By means of an ADG, a forward slice $fSlice(Cri)$ with respect to a slicing criterion $Cri = \langle v, var \rangle$, where v is a vertex and $var \in v$ is a variable or data component defined or used at v, consists of all vertices that are forward-reachable (through arcs) from v. The set of affected vertices $V'_{aff} \subseteq V'$ is thereby determined by, for each $v_x \in AFG \backslash AFG' \cup AFG \backslash AFG'$, and for each defined or used variable $var_y \in v_x$, compute $fSlice(\langle v_x, var_y \rangle)$ of ADG'.

Fig. 22. Control dependencies of *dcu2_line_trip.Impl*.

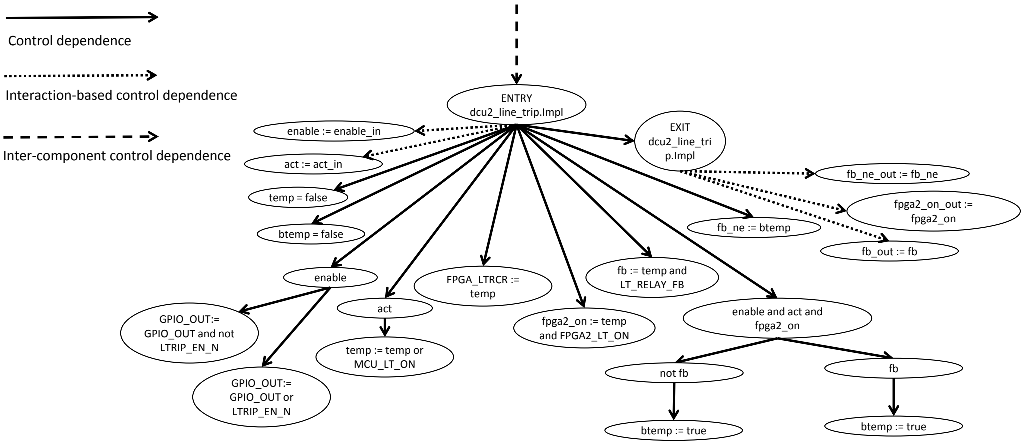
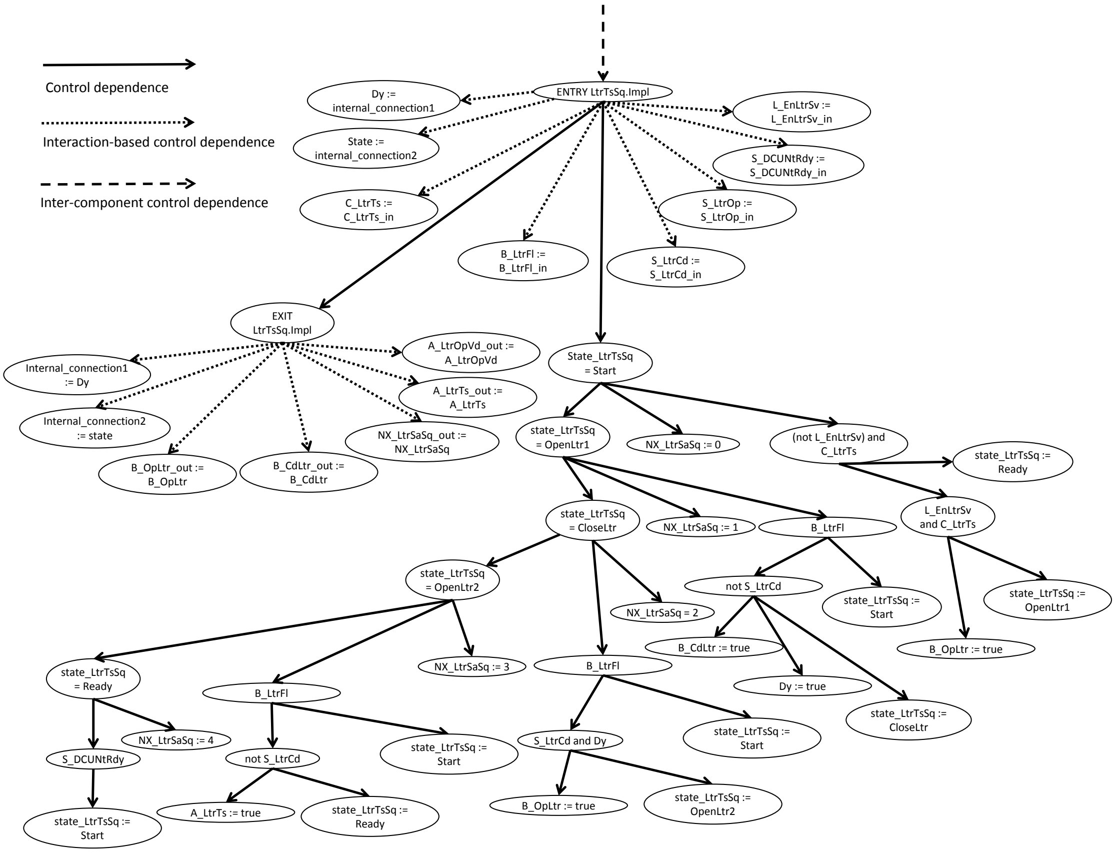

Control dependence
Interaction-based control dependence
Inter-component control dependence
ENTRY dcu2_line_trip.Impl
EXIT dcu2_line_trip.Impl
enable := enable_in
act := act_in
temp = false
btemp = false
enable
act
FPGA_LTRCR := temp
fb := temp and LT_RELAY_FB
fb_ne := btemp
fpga2_on := temp and FPGA2_LT_ON
enable and act and fpga2_on
fb_ne_out := fb_ne
fpga2_on_out := fpga2_on
fb_out := fb
GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N
GPIO_OUT:= GPIO_OUT or LTRIP_EN_N
temp := temp or MCU_LT_ON
not fb
fb
btemp := true
btemp := true

Fig. 23. Control dependencies of *LtrTsSq.Impl*.



12 Case study stage two: validation through fault injections

In this section, we present the results of applying the framework on mutated versions of the LTRIS AADL model, each of which contains an injected fault. By means of the original LTRIS model and the model checking and model-based testing activities performed in the first stage, each fault injection corresponds to a modification upon the selective verification technique can be applied. The expectation is that the result of regression verification is at least one unsatisfied selected observer per modification. According to the study design, the selective approach is contrasted with a re-run all approach to assess the selection effectiveness and efficiency. The expectation is that all non-selected verification sequences are satisfied when executed against the mutated version since all verification sequences that possibly reveals a fault in the modified model should be selected. In other words, the number of unsatisfied observers in the selective regression verification suite should be equal to a re-run all regression verification suite (number of unsatisfied selected observers is equal to number of unsatisfied selected and non-selected observers). Furthermore, the total resource consumption of selective regression verification, including the required overhead expense of conducting the selection, is expected to not exceed the resource consumption of a re-run all approach to be efficient. Finally, each mutated version is treated as an implementation to validate the effectiveness of the test suite generated in the first stage of the case study. The expectation is at least one failed test case for each tested mutation.

12.1 Injected faults

In Table 15, the chosen fault injections that cover all considered fault types are listed. A total of seven mutated versions of the original LTRIS model are created by seven different fault injections. The first fault injection is a negation of a transition guard in the test sequence subprogram component. The negation changes the predicate condition such that the transition is fired if the relay is closed rather than opened. In order to reach this guard, a transition which action sends an opening order to the relay-controller thread must previously have been executed. However, the controller thread prioritizes an opening order over a closing order. Thus, the negation renders the guard unachievable. The second fault injection is a changed value assignment to the variable that determines the test sequence. The changed, incorrect, value simply makes the test to jump one step back in the sequence rather than forward, where finalization of the test sequence cannot be performed. The third fault injection is the removal of connection3. The connection is crucial for the test sequence as it transmits feedback data of the relay. The fourth fault injection is the removal of a parameter connection to the subprogram that controls the relay. Without the connection, larger parts of the controlling behavior cannot be reached or behaves incorrectly. The fifth fault injection is a change of the latency constraint of connection2, where the latency window is changed to a minimum of 1 ms and a maximum of 2 ms. Due to scheduling properties of the system, the minimum latency constraint cannot be met as Controller will dispatch simultaneously to the completion of Tester. The sixth fault injection is a change of the period of Controller, which still renders the system schedulable but contradicts latency properties of connections. The last fault injection is an added assignment (transition action) in Tester to a shared data component. Since the data component is not unlocked until the completion of Tester, the controller thread, which uses the same data component and which deadline will be met prior to the unlocking of it, will be subjected to starvation.

Table 15. Fault injections

Fault injection	**Original model**	**Mutant model**	**Fault location**	**Fault type(s)**
1	OpenLtr2 [0]-[not S_LtrCd]->Return { A_LtrTs := true, state_LtrTsSq := Ready} ;	OpenLtr2 [0]-[S_LtrCd]->Return { A_LtrTs := true, state_LtrTsSq := Ready} ;	transition guard of LtrTsSq.Impl	1
2	CloseLtr [0]-[S_LtrCd and Dy]->Return {B_OpLtr := true; state_LtrTsSq := OpenLtr2};	CloseLtr [0]-[S_LtrCd and Dy]->Return {B_OpLtr := true; state_LtrTsSq := OpenLtr1};	transition action of LtrTsSq.Impl	2
3	connection3: port relayController.DHWOMG_S_LtrCd ->relayTester.DHWOMG_S_LtrCd{Timing =>Immediate; Latency =>0ms .. 2ms;};		port connection declaration of LineTripSoftware.Impl	3
4	parameter DHSSMG_B_LtrHwOpFl ->dcu2_line_trip.enable;		parameter connection declaration of Controller.Impl	3
5	connection2: port relayTester.DHSSMG_B_CdLtr ->relayController.DHSSMG_B_CdLtr{Timing =>Immediate; Latency =>0ms .. 1ms;};	connection2: port relayTester.DHSSMG_B_CdLtr ->relayController.DHSSMG_B_CdLtr{Timing =>Immediate; Latency =>1ms .. 2ms;};	connection property of LineTripSoftware.Impl	4,6
6	Period =>4 ms;	Period =>8 ms;	scheduling property of Controller	6,4
7	state0 -[on dispatch]->state1 { LtrTsSq(...)};	state0 -[on dispatch]->state1 {MCU_LT_ON := true; LtrTsSq(...)};	transition action of Tester.Impl	5,7

12.2 Results

The results of the selection process of each fault injection are presented in Table 47–53 in Appendix C. Illustration of independent observers extraction for each modification is presented in Table 54–59 in Appendix D. The slices with respect to each modification are presented in Fig. 24–29 in Appendix E. With respect to these results, the results of selective regression verification in conjunction with re-run all regression verification of each fault injection are presented in Table 16–22. Descriptive statistics for time consumption, memory consumption, and trace size are listen in Table 23– 25. The overhead in terms of selecting verification sequences through ADG generation and slicing is included for each change (fault injection). Finally, the results of running the test suite generated in the first stage of the case study on each mutated version of the model are presented in Table 26.

Table 16. Fault injection 1: model checking results (regression verification selections bold)

Observer	**Verdict**	**Time cons. (sec)**	**Memory cons. (MB)**	**Trace size (KB)**
InternalTester1	satisfied	1	39	144
InternalLtrTsSq1	satisfied	1	38	35
InternalLtrTsSq2	satisfied	1	40	35
InternalLtrTsSq3	satisfied	1	42	472
InternalLtrTsSq4	satisfied	9	69	4066
InternalLtrTsSq5	satisfied	12	149	4435
InternalLtrTsSq6	satisfied	12	150	4435
InternalLtrTsSq7	satisfied	19	251	4883
InternalLtrTsSq8_updated	not satisfied	36	589	
InternalLtrTsSq9	satisfied	1	37	471
InternalLtrTsSq10	satisfied	1	38	471
InternalLtrTsSq11	satisfied	1	39	471
InternalLtrTsSq12	satisfied	2	61	913
InternalLtrTsSq13	satisfied	11	151	4434
InternalLtrTsSq14	satisfied	18	252	4881
InternalLtrTsSq15	satisfied	18	225	4881
InternalLtrTsSq16	satisfied	23	350	5329
InternalLtrTsSq17	not satisfied	35	587	
InternalLtrTsSq18	satisfied	2	63	913
InternalLtrTsSq19	satisfied	11	147	4434
InternalController1	satisfied	1	38	31
InternalLtrInt1	satisfied	1	40	22
Internaldcu2_line_trip1	satisfied	1	44	607
Internaldcu2_line_trip2	satisfied	1	44	607
Internaldcu2_line_trip3	satisfied	1	45	607
Internaldcu2_line_trip4	satisfied	1	46	607
Internaldcu2_line_trip5	satisfied	1	45	29
Internaldcu2_line_trip6	satisfied	1	47	607
Internaldcu2_line_trip7	satisfied	1	47	607
Internaldcu2_line_trip8	satisfied	1	47	29
Internaldcu2_line_trip9	satisfied	1	47	58
Internaldcu2_line_trip10	satisfied	1	48	438
Internaldcu2_line_trip11	satisfied	1	48	438
Internaldcu2_line_trip12	satisfied	1	48	58
Direct1	satisfied	1	49	33
Direct2	satisfied	1	49	8
Direct3	satisfied	1	49	23
Direct4	satisfied	1	50	150
Direct5	satisfied	1	50	150
Direct6	satisfied	1	52	454
Indirect1	satisfied	1	53	175
Indirect2	satisfied	1	53	175
Indirect3	satisfied	13	160	4699
Indirect4	satisfied	13	166	4699
Indirect5	satisfied	9	89	4237
Indirect6	satisfied	9	88	4237
Indirect7	satisfied	9	89	4237
Indirect8	satisfied	9	89	4237
Indirect9	satisfied	9	90	4237
Indirect10	satisfied	9	91	4237
Indirect11	satisfied	12	117	4223
Indirect12	satisfied	12	116	4223
Indirect13	satisfied	13	192	4686
Indirect14	satisfied	13	193	4686
Indirect15	satisfied	13	190	4686
Indirect16_updated	not satisfied	62	1029	
Indirect17_updated	not satisfied	62	1030	

Table 17. Fault injection 2: model checking results (regression verification selections bold)

Observer	**Verdict**	**Time cons. (sec)**	**Memory cons. (MB)**	**Trace size (KB)**
InternalTester1	satisfied	1	36	144
InternalLtrTsSq1	satisfied	1	36	35
InternalLtrTsSq2	satisfied	1	37	35
InternalLtrTsSq3	satisfied	1	40	472
InternalLtrTsSq4	satisfied	8	72	4066
InternalLtrTsSq5	satisfied	11	151	4435
InternalLtrTsSq6_updated	satisfied	11	152	4435
InternalLtrTsSq7	not satisfied	35	569	
InternalLtrTsSq8	not satisfied	35	570	
InternalLtrTsSq9	satisfied	1	49	472
InternalLtrTsSq10	satisfied	1	50	471
InternalLtrTsSq11	satisfied	1	51	471
InternalLtrTsSq12	satisfied	2	72	913
InternalLtrTsSq13	satisfied	11	159	4434
InternalLtrTsSq14	satisfied	18	258	4881
InternalLtrTsSq15_updated	not satisfied	34	574	
InternalLtrTsSq16	not satisfied	34	575	
InternalLtrTsSq17	not satisfied	34	576	
InternalLtrTsSq18	satisfied	2	75	913
InternalLtrTsSq19	satisfied	11	164	4434
InternalController1	satisfied	1	55	31
InternalLtrInt1	satisfied	1	55	22
Internaldcu2_line_trip1	satisfied	1	57	607
Internaldcu2_line_trip2	satisfied	1	58	607
Internaldcu2_line_trip3	satisfied	1	60	607
Internaldcu2_line_trip4	satisfied	1	60	607
Internaldcu2_line_trip5	satisfied	1	60	29
Internaldcu2_line_trip6	satisfied	1	61	607
Internaldcu2_line_trip7	satisfied	1	62	607
Internaldcu2_line_trip8	satisfied	1	38	29
Internaldcu2_line_trip9	satisfied	1	39	58
Internaldcu2_line_trip10	satisfied	1	41	438
Internaldcu2_line_trip11	satisfied	1	42	438
Internaldcu2_line_trip12	satisfied	1	42	58
Direct1	satisfied	1	42	33
Direct2	satisfied	1	42	8
Direct3	satisfied	1	43	23
Direct4	satisfied	1	43	150
Direct5	satisfied	1	44	150
Direct6	satisfied	1	46	454
Indirect1	satisfied	1	47	175
Indirect2	satisfied	1	47	175
Indirect3_updated	satisfied	12	161	4699
Indirect4_updated	satisfied	11	149	4699
Indirect5	satisfied	8	71	4237
Indirect6	satisfied	8	72	4237
Indirect7	satisfied	8	72	4237
Indirect8	satisfied	8	73	4237
Indirect9	satisfied	8	74	4237
Indirect10	satisfied	8	74	4237
Indirect11	satisfied	12	95	4223
Indirect12	satisfied	12	96	4223
Indirect13_updated	satisfied	13	173	4686
Indirect14_updated	satisfied	13	173	4686
Indirect15_updated	satisfied	13	173	4686
Indirect16	not satisfied	59	960	
Indirect17	not satisfied	59	962	

Table 18. Fault injection 3: model checking results (regression verification selections bold)

Observer	**Verdict**	**Time cons. (sec)**	**Memory cons. (MB)**	**Trace size (KB)**
InternalTester1	satisfied	1	38	144
InternalLtrTsSq1	satisfied	1	39	35
InternalLtrTsSq2	satisfied	1	40	35
InternalLtrTsSq3	satisfied	1	42	471
InternalLtrTsSq4	satisfied	8	68	4054
InternalLtrTsSq5	satisfied	11	130	4422
InternalLtrTsSq6	not satisfied	30	468	
InternalLtrTsSq7	not satisfied	30	468	
InternalLtrTsSq8	not satisfied	30	469	
InternalLtrTsSq9	satisfied	1	44	471
InternalLtrTsSq10	satisfied	1	45	470
InternalLtrTsSq11	satisfied	1	46	470
InternalLtrTsSq12	satisfied	2	67	910
InternalLtrTsSq13	satisfied	11	136	4421
InternalLtrTsSq14	satisfied	16	214	4867
InternalLtrTsSq15	not satisfied	30	474	
InternalLtrTsSq16	not satisfied	30	475	
InternalLtrTsSq17	not satisfied	30	476	
InternalLtrTsSq18	satisfied	2	72	910
InternalLtrTsSq19	not satisfied	30	477	
InternalController1	satisfied	1	52	31
InternalLtrInt1	satisfied	1	52	22
Internaldcu2_line_trip1	satisfied	1	57	605
Internaldcu2_line_trip2	satisfied	1	58	605
Internaldcu2_line_trip3	satisfied	1	59	605
Internaldcu2_line_trip4	satisfied	1	59	605
Internaldcu2_line_trip5	satisfied	1	59	29
Internaldcu2_line_trip6	satisfied	1	60	606
Internaldcu2_line_trip7	satisfied	1	60	606
Internaldcu2_line_trip8	satisfied	1	61	29
Internaldcu2_line_trip9	satisfied	1	61	58
Internaldcu2_line_trip10	satisfied	1	61	437
Internaldcu2_line_trip11	satisfied	1	62	437
Internaldcu2_line_trip12	satisfied	1	62	58
Direct1	satisfied	1	62	33
Direct2	satisfied	1	63	8
Direct3	satisfied	1	63	23
Direct4	satisfied	1	63	149
Direct5	satisfied	1	64	149
Direct6	not valid			
Indirect1	satisfied	1	64	174
Indirect2	satisfied	1	64	174
Indirect3	not satisfied	30	491	
Indirect4	not satisfied	30	491	
Indirect5	satisfied	8	97	4225
Indirect6	satisfied	8	97	4225
Indirect7	satisfied	8	98	4225
Indirect8	satisfied	8	99	4225
Indirect9	satisfied	8	100	4225
Indirect10	satisfied	8	100	4225

Table 19. Fault injection 4: model checking results (regression verification selections bold)

Observer	**Verdict**	**Time cons. (sec)**	**Memory cons. (MB)**	**Trace size (KB)**
InternalTester1	satisfied	1	47	144
InternalLtrTsSq1	satisfied	1	47	35
InternalLtrTsSq2	satisfied	1	47	35
InternalLtrTsSq3	satisfied	1	49	471
InternalLtrTsSq4	satisfied	7	67	4054
InternalLtrTsSq5	satisfied	10	109	4422
InternalLtrTsSq6	satisfied	10	110	4422
InternalLtrTsSq7	satisfied	15	172	4869
InternalLtrTsSq8	satisfied	15	173	4869
InternalLtrTsSq9	satisfied	1	44	471
InternalLtrTsSq10	satisfied	1	44	471
InternalLtrTsSq11	satisfied	1	45	470
InternalLtrTsSq12	satisfied	1	50	910
InternalLtrTsSq13	satisfied	10	114	4421
InternalLtrTsSq14	satisfied	15	176	4867
InternalLtrTsSq15	satisfied	15	177	4867
InternalLtrTsSq16	satisfied	20	246	5314
InternalLtrTsSq17	satisfied	20	247	5314
InternalLtrTsSq18	satisfied	1	54	910
InternalLtrTsSq19	satisfied	10	121	4421
InternalController1	satisfied	1	54	31
InternalLtrInt1	satisfied	1	54	22
Internaldcu2_line_trip1	not satisfied	53	744	
Internaldcu2_line_trip2	not satisfied	53	745	
Internaldcu2_line_trip3	not satisfied	53	745	
Internaldcu2_line_trip4	not satisfied	53	746	
Internaldcu2_line_trip5	not satisfied	53	748	
Internaldcu2_line_trip6	satisfied	1	61	606
Internaldcu2_line_trip7	satisfied	1	62	606
Internaldcu2_line_trip8	satisfied	1	62	29
Internaldcu2_line_trip9	not satisfied	53	767	
Internaldcu2_line_trip10	not satisfied	53	767	
Internaldcu2_line_trip11	not satisfied	53	768	
Internaldcu2_line_trip12	satisfied	1	65	58
Direct1	satisfied	1	65	33
Direct2	satisfied	1	65	8
Direct3_updated	satisfied	1	66	23
Direct4	satisfied	1	66	149
Direct5	satisfied	1	66	149
Direct6	satisfied	1	67	452
Indirect1	not satisfied	51	776	
Indirect2	satisfied	1	70	174
Indirect3	not satisfied	90	1135	
Indirect4	satisfied	11	138	4686
Indirect5	not satisfied	51	776	
Indirect6	not satisfied	51	777	
Indirect7	not satisfied	51	778	
Indirect8	not satisfied	51	778	
Indirect9	satisfied	8	104	4225
Indirect10	satisfied	7	103	4225
Indirect11	not satisfied	51	780	
Indirect12	satisfied	8	102	4211
Indirect13	not satisfied	52	781	
Indirect14	not satisfied	52	781	
Indirect15	satisfied	11	148	4676
Indirect16	not satisfied	51	782	
Indirect17	satisfied	15	211	5132

Table 20. Fault injection 5: model checking results (regression verification selections bold)

Observer	**Verdict**	**Time cons. (sec)**	**Memory cons. (MB)**	**Trace size (KB)**
InternalTester1	satisfied	1	35	144
InternalLtrTsSq1	satisfied	1	34	35
InternalLtrTsSq2	satisfied	1	36	35
InternalLtrTsSq3	satisfied	1	39	472
InternalLtrTsSq4	satisfied	8	79	4066
InternalLtrTsSq5	satisfied	11	149	4435
InternalLtrTsSq6	satisfied	11	149	4435
InternalLtrTsSq7	satisfied	18	251	4883
InternalLtrTsSq8	satisfied	18	251	4883
InternalLtrTsSq9	satisfied	1	45	472
InternalLtrTsSq10	satisfied	1	45	471
InternalLtrTsSq11	satisfied	1	46	471
InternalLtrTsSq12	satisfied	1	66	913
InternalLtrTsSq13	satisfied	11	150	4434
InternalLtrTsSq14	satisfied	18	266	4881
InternalLtrTsSq15	satisfied	18	256	4881
InternalLtrTsSq16	satisfied	24	380	5329
InternalLtrTsSq17	satisfied	24	380	5329
InternalLtrTsSq18	satisfied	2	74	913
InternalLtrTsSq19	satisfied	11	162	4434
InternalController1	satisfied	1	53	31
InternalLtrInt1	satisfied	1	54	22
Internaldcu2_line_trip1	satisfied	1	55	607
Internaldcu2_line_trip2	satisfied	1	56	607
Internaldcu2_line_trip3	satisfied	1	57	607
Internaldcu2_line_trip4	satisfied	1	57	607
Internaldcu2_line_trip5	satisfied	1	58	29
Internaldcu2_line_trip6	satisfied	1	60	607
Internaldcu2_line_trip7	satisfied	1	60	607
Internaldcu2_line_trip8	satisfied	1	60	29
Internaldcu2_line_trip9	satisfied	1	61	58
Internaldcu2_line_trip10	satisfied	1	61	438
Internaldcu2_line_trip11	satisfied	1	61	438
Internaldcu2_line_trip12	satisfied	1	62	58
Direct1	satisfied	1	62	33
Direct2	satisfied	1	62	8
Direct3	satisfied	1	62	23
Direct4	satisfied	1	63	150
Direct5_updated	not satisfied	1	63	
Direct6	satisfied	1	65	454
Indirect1	satisfied	1	65	175
Indirect2	satisfied	1	65	175
Indirect3	satisfied	12	175	4699
Indirect4	satisfied	12	174	4699
Indirect5_updated	not satisfied	9	91	
Indirect6_updated	not satisfied	9	90	
Indirect7_updated	not satisfied	9	92	
Indirect8_updated	not satisfied	9	92	
Indirect9_updated	not satisfied	9	93	
Indirect10_updated	not satisfied	9	94	
Indirect11	satisfied	11	120	4223
Indirect12	satisfied	11	122	4223
Indirect13	satisfied	13	197	4686
Indirect14	satisfied	13	198	4686
Indirect15	satisfied	13	198	4686
Indirect16	satisfied	30	440	5149
Indirect17	satisfied	30	441	5146

Table 21. Fault injection 6: model checking results (regression verification selections bold)

Observer	**Verdict**	**Time cons. (sec)**	**Memory cons. (MB)**	**Trace size (KB)**
InternalTester1	satisfied	1	40	65
InternalLtrTsSq1	satisfied	1	40	35
InternalLtrTsSq2	satisfied	1	41	35
InternalLtrTsSq3	satisfied	1	44	258
InternalLtrTsSq4	satisfied	6	46	2079
InternalLtrTsSq5	satisfied	8	86	2267
InternalLtrTsSq6	satisfied	8	87	2267
InternalLtrTsSq7	satisfied	11	109	2495
InternalLtrTsSq8	satisfied	11	110	2495
InternalLtrTsSq9	satisfied	1	41	258
InternalLtrTsSq10	satisfied	1	41	257
InternalLtrTsSq11	satisfied	1	41	257
InternalLtrTsSq12	satisfied	2	46	482
InternalLtrTsSq13	satisfied	8	89	2265
InternalLtrTsSq14	satisfied	11	114	2493
InternalLtrTsSq15	satisfied	11	114	2493
InternalLtrTsSq16	satisfied	14	158	2721
InternalLtrTsSq17	satisfied	14	159	2721
InternalLtrTsSq18	satisfied	2	51	482
InternalLtrTsSq19	satisfied	8	94	2266
InternalController1	satisfied	1	48	31
InternalLtrInt1	satisfied	1	49	22
Internaldcu2_line_trip1	satisfied	1	50	314
Internaldcu2_line_trip2	satisfied	1	51	314
Internaldcu2_line_trip3	satisfied	1	51	314
Internaldcu2_line_trip4	satisfied	1	52	314
Internaldcu2_line_trip5	satisfied	1	52	29
Internaldcu2_line_trip6	satisfied	1	54	314
Internaldcu2_line_trip7	satisfied	1	53	314
Internaldcu2_line_trip8	satisfied	1	54	29
Internaldcu2_line_trip9	satisfied	1	54	58
Internaldcu2_line_trip10	satisfied	1	54	171
Internaldcu2_line_trip11	satisfied	1	54	171
Internaldcu2_line_trip12	satisfied	1	55	58
Direct1	satisfied	1	55	33
Direct2	satisfied	1	55	8
Direct3	satisfied	1	56	23
Direct4	not satisfied	1	57	
Direct5	not satisfied	1	57	
Direct6	not satisfied	1	62	
Indirect1	not satisfied	4	61	
Indirect2	not satisfied	4	61	
Indirect3	not satisfied	18	231	
Indirect4	not satisfied	18	231	
Indirect5	not satisfied	8	67	
Indirect6	not satisfied	8	68	
Indirect7	not satisfied	8	68	
Indirect8	not satisfied	8	69	
Indirect9	not satisfied	8	69	
Indirect10	not satisfied	8	70	
Indirect11	not satisfied	43	664	
Indirect12	not satisfied	43	665	
Indirect13	not satisfied	30	409	
Indirect14	not satisfied	30	409	
Indirect15	not satisfied	27	369	
Indirect16	not satisfied	43	667	
Indirect17	not satisfied	43	668	

Table 22. Fault injection 7: model checking results (regression verification selections bold)

Observer	**Verdict**	**Time cons. (sec)**	**Memory cons. (MB)**	**Trace size (KB)**
InternalTester1_updated	satisfied	1	42	42
InternalLtrTsSq1	satisfied	1	42	35
InternalLtrTsSq2	satisfied	1	43	35
InternalLtrTsSq3	not satisfied	1	44	
InternalLtrTsSq4	not satisfied	1	44	
InternalLtrTsSq5	not satisfied	1	45	
InternalLtrTsSq6	not satisfied	1	45	
InternalLtrTsSq7	not satisfied	1	46	
InternalLtrTsSq8	not satisfied	1	46	
InternalLtrTsSq9	not satisfied	1	47	
InternalLtrTsSq10	not satisfied	1	48	
InternalLtrTsSq11	not satisfied	1	48	
InternalLtrTsSq12	not satisfied	1	49	
InternalLtrTsSq13	not satisfied	1	49	
InternalLtrTsSq14	not satisfied	1	50	
InternalLtrTsSq15	not satisfied	1	50	
InternalLtrTsSq16	not satisfied	1	51	
InternalLtrTsSq17	not satisfied	1	52	
InternalLtrTsSq18	not satisfied	1	52	
InternalLtrTsSq19	not satisfied	1	53	
InternalController1	satisfied	1	53	31
InternalLtrInt1	satisfied	1	54	22
Internaldcu2_line_trip1	satisfied	1	59	34
Internaldcu2_line_trip2	satisfied	1	39	34
Internaldcu2_line_trip3	satisfied	1	40	34
Internaldcu2_line_trip4	satisfied	1	40	34
Internaldcu2_line_trip5	satisfied	1	39	34
Internaldcu2_line_trip6	satisfied	1	41	34
Internaldcu2_line_trip7	satisfied	1	41	34
Internaldcu2_line_trip8	satisfied	1	41	34
Internaldcu2_line_trip9	not satisfied	1	40	
Internaldcu2_line_trip10	not satisfied	1	41	
Internaldcu2_line_trip11	not satisfied	1	41	
Internaldcu2_line_trip12	not satisfied	1	41	
Direct1	satisfied	1	40	33
Direct2	satisfied	1	40	8
Direct3	satisfied	1	41	23
Direct4	not satisfied	1	41	
Direct5	not satisfied	1	41	
Direct6	not satisfied	1	41	
Direct7	not satisfied	1	41	
Indirect1	not satisfied	1	40	
Indirect2	not satisfied	1	40	
Indirect3	not satisfied	1	40	
Indirect4	not satisfied	1	41	
Indirect5	not satisfied	1	41	
Indirect6	not satisfied	1	41	
Indirect7	not satisfied	1	42	
Indirect8	not satisfied	1	42	
Indirect9	not satisfied	1	42	
Indirect10	not satisfied	1	42	
Indirect11	not satisfied	1	42	
Indirect12	not satisfied	1	40	
Indirect13	not satisfied	1	42	
Indirect14	not satisfied	1	41	
Indirect15	not satisfied	1	41	
Indirect16	not satisfied	1	42	
Indirect17	not satisfied	1	42	

Table 23. Descriptive statistics for model checking – time consumption

Fault injection	Method	Mean(sec)	Median(sec)	SD(sec)	Min(sec)	Max(sec)	Sum(sec)	N	Overhead(sec)
1	all	9	1	13	1	62	504	57	-
1	sel	49	49	15	35	62	195	4	1
2	all	9	1	14	1	59	527	57	-
2	sel	28	34	17	11	59	363	13	1
3	all	8	1	11	1	30	396	49	-
3	sel	15	8	12	1	30	372	25	1
4	all	21	10	24	1	90	1204	57	-
4	sel	35	51	25	1	90	1041	30	1
5	all	7	1	8	1	30	403	57	-
5	sel	10	9	7	1	24	234	24	1
6	all	9	2	12	1	43	492	57	-
6	sel	-	-	-	-	-	-	-	-
7	all	1	1	0	1	1	58	58	-
7	sel	1	1	0	1	1	50	50	1

Table 24. Descriptive statistics for model checking – memory consumption

Fault injection	Method	Mean(MB)	Median(MB)	SD(MB)	Min(MB)	Max(MB)	Sum(MB)	N	Overhead(MB)
1	all	142	53	205	37	1030	8085	57	-
1	sel	809	809	255	587	1030	3235	4	1
2	all	153	62	215	36	962	8728	57	-
2	sel	444	569	301	149	962	5767	13	1
3	all	145	63	161	38	491	7127	49	-
3	sel	230	100	192	42	491	5744	25	1
4	all	315	114	331	44	1135	17942	57	-
4	sel	512	746	351	61	1135	15366	30	1
5	all	120	65	102	34	441	6832	57	-
5	sel	142	93	102	39	380	3403	24	1
6	all	131	57	170	40	668	7470	57	-
6	sel	-	-	-	-	-	-	-	-
7	all	44	42	5	39	59	2542	58	-
7	sel	44	42	4	39	59	2189	50	1

Table 25. Descriptive statistics for model checking – trace size

Fault injection	Method	Mean(KB)	Median(KB)	SD(KB)	Min(KB)	Max(KB)	Sum(KB)	N
1	all	2055	607	2103	8	5329	108940	53
1	sel	0	0	0	0	0	0	0
2	all	1877	607	2029	8	4881	93848	50
2	sel	4649	4686	105	4435	4699	27891	6
3	all	1312	471	1774	8	4867	52473	40
3	sel	2926	4225	1860	470	4867	46816	16
4	all	2186	606	2217	8	5314	85252	39
4	sel	2388	2409	2255	23	5132	28651	12
5	all	2078	607	2193	8	5329	103876	50
5	sel	3277	4434	2052	471	5329	55702	17
6	all	843	314	1049	8	2721	31208	37
6	sel	843	314	1049	8	2721	31208	37
7	all	31	34	8	8	42	501	16
7	sel	35	34	3	33	42	279	8

Table 26. Model-based testing results (O=passed and X=failed)

Test case/Fault injection	**1**	**2**	**3**	**4**	**5**	**6**	**7**
TestcaseInternalTester1	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq1	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq2	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq3	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq4	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq5	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq6	O	O	X	O	O	O	X
TestcaseInternalLtrTsSq7	O	X	X	O	O	O	X
TestcaseInternalLtrTsSq8	X	X	X	O	O	O	X
TestcaseInternalLtrTsSq9	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq10	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq11	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq12	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq13	O	O	X	O	O	O	X
TestcaseInternalLtrTsSq14	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq15	X	X	X	O	O	O	X
TestcaseInternalLtrTsSq16	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq17	X	X	X	O	O	O	X
TestcaseInternalLtrTsSq18	O	O	O	O	O	O	X
TestcaseInternalLtrTsSq19	O	O	X	O	O	O	X
TestcaseInternalController1	O	O	X	O	O	O	O
TestcaseInternalLtrInt1	O	O	O	O	O	O	O
TestcaseInternaldcu2_line_trip1	O	O	O	X	O	O	O
TestcaseInternaldcu2_line_trip2	O	O	O	X	O	O	O
TestcaseInternaldcu2_line_trip3	O	O	O	X	O	O	O
TestcaseInternaldcu2_line_trip4	O	O	O	X	O	O	O
TestcaseInternaldcu2_line_trip5	O	O	O	X	O	O	O
TestcaseInternaldcu2_line_trip6	O	O	O	O	O	O	O
TestcaseInternaldcu2_line_trip7	O	O	O	O	O	O	O
TestcaseInternaldcu2_line_trip8	O	O	O	O	O	O	O
TestcaseInternaldcu2_line_trip9	O	O	O	X	O	O	O
TestcaseInternaldcu2_line_trip10	O	O	O	O	O	X	X
TestcaseInternaldcu2_line_trip11	O	O	O	X	O	X	X
TestcaseInternaldcu2_line_trip12	O	O	O	O	O	O	O
TestcaseDirect1	O	O	O	O	O	O	X
TestcaseDirect2	O	O	O	O	O	O	O
TestcaseDirect3	O	O	O	X	O	O	O
TestcaseDirect4	O	O	O	O	O	O	X
TestcaseDirect5	O	O	O	O	O	O	X
TestcaseDirect6	O	O	O	O	O	O	X
TestcaseIndirect1	O	O	X	X	O	O	X
TestcaseIndirect2	O	O	X	O	O	O	X
TestcaseIndirect3	O	O	X	X	O	O	X
TestcaseIndirect4	O	O	X	O	O	O	X
TestcaseIndirect5	O	O	X	X	O	O	X
TestcaseIndirect6	O	O	X	X	O	O	X
TestcaseIndirect7	O	O	X	X	O	O	X
TestcaseIndirect8	O	O	X	X	O	O	X
TestcaseIndirect9	O	O	X	O	O	O	X
TestcaseIndirect10	O	O	X	O	O	O	X
TestcaseIndirect11	O	O	X	X	O	X	X
TestcaseIndirect12	O	O	X	O	O	O	X
TestcaseIndirect13	O	O	X	X	O	X	X
TestcaseIndirect14	O	O	X	X	O	X	X
TestcaseIndirect15	O	O	X	O	O	X	X
TestcaseIndirect16	X	X	X	X	O	X	X
TestcaseIndirect17	X	X	X	O	O	O	X

13 Summary and conclusion

The complete study is summarized in Table 27. The results conform to the expectations except in two cases. First, fault No. five was not detected by the test suite. In retrospect, the result is not a surprise as the fault is an inconsistent latency property, which in the model does not affect the execution but impose an analysis constraint on it. Thus, it is not sound to treat the faulty model as a faulty implementation in this case, since the inconsistent property must be manifested in the execution to be a realistic implementation fault. Second, fault No. 6 corresponds to a changed scheduling property which has no relation to the AFG, consequently, no slicing can be performed.

By considering the resource consumption of a re-run all approach, faults (changes) have the ability to both significantly reduce as well as increase the resource consumption of observers satisfiability checking. There are mainly two parameters that determine the outcome. First, a fault (change) may add or remove states of the timed automata model, which may exponentially reduce or increase the state space. Thus, possibly lengthens or shortens the state space search by a significant amount. Fault No. seven produced the lowest recorded resource consumption. The cause is a significantly reduced state space since Controller will miss its deadline (whereupon the scheduler deadlocks) shortly after the dispatch of Tester. Fault No. four produced the highest recorded resource consumption. This is however not caused by a significantly increased state space. In fact, the fault reduces the state space by preventing parts of the model to be reached. Instead, the significant increase of necessary resources is caused by the unreachable paths. Unreachability can only be determined by searching the complete state space, which still, in this case, is relatively large.

Table 27. Results summary

		model checking and sel. regr. ver. effect.			**TOT time**		**TOT mem.**		**Sel. efficiency**		**Testing effectiveness**
Fault	**No V-seqs**	**No sel.**	**No unsat. sel. Obs.**	**No unsat. Obs.**	**Sel.**	**All**	**Sel.**	**All**	**Time**	**Mem.**	**No failed TCs (of 57)**
n/a	57	n/a	n/a	0	n/a	855	n/a	9327	n/a	n/a	0
1	57	4	4	4	196	504	3236	8085	61%	60%	5
2	57	13	7	7	364	527	5768	8728	31%	34%	6
3	49	25	9	9	373	396	5745	7127	6%	19%	25
4	57	30	18	18	1042	1204	15367	17942	13%	14%	18
5	57	24	7	7	235	403	3404	6832	42%	50%	0
6	57	n/a	n/a	20	n/a	492	n/a	7470	0%	0%	7
7	58	50	42	42	51	58	2190	2542	12%	14%	43
	56					**555**		**8507**	**24%**	**27%**	

On average, it took 555 seconds and 8507 MB to check satisfiability of 56 observers. Seven out of seven design faults were detected and, by disregarding fault five at the implementation-level, six out of six implementation faults. The selective approach, on average, reduced the time and memory consumption of regression verification by 24% and 27% respectively. The time complexity of slicing is linear and, in this study, the additional expense slicing brings is close to negligible with respect to the savings except for time consumption of fault injection No 7. No verification sequence that reveals a fault in the modified design was not selected.

Acknowledgments

This research is supported by the Swedish Foundation for Strategic Research (SSF) project SYNOPSIS.

References

1. A. Avizienis, J.-C. Laprie, and B Randell. Dependability and its threats - A taxonomy. pages 91–120, 2004. IFIP Congress Topical Sessions.
2. Nancy G. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems).* The MIT Press, January 2012.
3. RTI. The Economic Impacts of Inadequate Infrastructure for Software Testing. NIST Planning report 02-3, Washington, DC, 2002.
4. Barry Boehm, Ricardo Valerdi, and Eric Honour. The roi of systems engineering: Some quantitative results for software-intensive systems. *Syst. Eng.*, 11(3):221–234, August 2008.
5. Joseph Elm, Dennis Goldenson, Khaled El Emam, Nichole Donitelli, Angelica Neisa, and NDIA SE Effectiveness Committee. A Survey of Systems Engineering Effectiveness: Initial Results (CMU/SEI-2007-SR-014). Technical report, Software Engineering Institute, Carnegie Mellon University, 2007.
6. As-2 Embedded Computing Systems Committee SAE. Architecture Analysis & Design Language (AADL). SAE Standards n^o AS5506A, 2009.
7. Andreas Johnsen and Kristina Lundqvist. Developing Dependable Software-intensive Systems: AADL vs. EAST-ADL. In *Proceedings of the 16th Ada-Europe International Conference on Reliable Software Technologies*, Ada-Europe'11, pages 103–117, Berlin, Heidelberg, 2011. Springer-Verlag.
8. C.M. Holloway. Why engineers should consider formal methods. In *Digital Avionics Systems Conference, 1997. 16th DASC., AIAA/IEEE*, volume 1, pages 1.3–16–22 vol.1, Oct 1997.
9. Holger Kienle, Daniel Sundmark, Kristina Lundqvist, and Andreas Johnsen. Liability for Software in Safety-Critical Mechatronic Systems: An Industrial Questionnaire. In *Proceedings of the 2nd International Workshop on Software Engineering for Embedded Systems*, June 2012.
10. Gordon Fraser, Franz Wotawa, and Paul E. Ammann. Testing with Model Checkers: A Survey. *Softw. Test. Verif. Reliab.*, 19(3):215–261, September 2009.
11. R.N. Taylor, N. Medvidovic, and E. Dashofy. *Software Architecture: Foundations, Theory, and Practice.* Wiley, 2009.
12. Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.
13. As-2 Embedded Computing Systems Committee SAE. Architecture Analysis & Design Language (AADL). SAE Standards n^o AS5506, November 2004.

14. Ricardo Bedin Franca, Jean-Paul Bodeveix, Mamoun Filali, Jean-Francois Rolland, David Chemouil, and Dave Thomas. The AADL behaviour annex – experiments and roadmap. In *ICECCS '07: Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems*, pages 377–382, Washington, DC, USA, 2007. IEEE Computer Society.
15. UP4ALL International AB. The UPPAAL Model-checking Tool. http://www.uppaal.com, May 2013.
16. Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on Uppaal. pages 200–236. Springer, 2004.
17. Nancy S. Eickelmann and Debra J. Richardson. What makes one software architecture more testable than another? In *ISAW '96: Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96) on SIGSOFT '96 workshops*, pages 65–67, New York, NY, USA, 1996. ACM.
18. ISO/DIS 26262 - Road vehicles - Functional safety. Technical report, Geneva, Switzerland.
19. Johan Blom, Anders Hessel, Bengt Jonsson, and Paul Pettersson. Specifying and Generating Test Cases Using Observer Automata. In *Proc. 4 th International Workshop on Formal Approaches to Testing of Software 2004 (FATES'04), volume 3395 of Lecture Notes in Computer Science*, pages 125–139. Springer-Verlag, 2005.
20. S. Horwitz, T. Reps, and D. Binkley. Interprocedural slicing using dependence graphs. In *Proceedings of the ACM SIGPLAN 1988 conference on Programming Language design and Implementation*, PLDI '88, pages 35–46, New York, NY, USA, 1988. ACM.
21. Zhenyi Jin and Jeff Offutt. Deriving Tests From Software Architectures. In *ISSRE '01: Proceedings of the 12th International Symposium on Software Reliability Engineering*, page 308, Washington, DC, USA, 2001. IEEE Computer Society.
22. Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi. Schedulability analysis of fixed-priority systems using timed automata. *Theor. Comput. Sci.*, 354:301–317, March 2006.
23. Juei Chang and Debra J. Richardson. Static and dynamic specification slicing. In *In Proceedings of the Fourth Irvine Software Symposium*, 1994.
24. Mark Weiser. Program slicing. In *Proceedings of the 5th international conference on Software engineering*, ICSE '81, pages 439–449, Piscataway, NJ, USA, 1981. IEEE Press.
25. Karl J. Ottenstein and Linda M. Ottenstein. The program dependence graph in a software development environment. *SIGPLAN Not.*, 19(5):177–184, April 1984.

Appendix A Verification sequences

Verification sequences: component internal paths

Tester.Impl:

1. ENTRY Tester.Impl (on dispatch) $\rightarrow_{cT}$ LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr, DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$ EXIT Tester.Impl

LtrTsSq.Impl:

1. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cT}$ NX_LtrSaSq := 0 $\rightarrow_{c}$ (not L_EnLtrSv) and C_LtrTs $\rightarrow_{cT}$ state_LtrTsSq := Ready $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr, DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)
2. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cT}$ NX_LtrSaSq := 0 $\rightarrow_{c}$ (not L_EnLtrSv) and C_LtrTs $\rightarrow_{cF}$ L_EnLtrSv and C_LtrTs $\rightarrow_{cT}$ B_OpLtr := true $\rightarrow_{c}$ state_LtrTsSq := OpenLtr1 $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr, DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)
3. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$ NX_LtrSaSq := 1 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cT}$ state_LtrTsSq := Start $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr, DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)
4. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$ NX_LtrSaSq := 1 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_{c}$ Dy := true $\rightarrow_{c}$ state_LtrTsSq := CloseLtr $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr, DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)
   - $\langle$B_CdLtr := true $\rightarrow_{c}$ Dy := true, $\{Delay(512ms)\}\rangle$
5. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$ state_LtrTsSq = CloseLtr $\rightarrow_{cT}$ NX_LtrSaSq := 2 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cT}$ state_LtrTsSq := Start $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr, DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)

6. ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$
   state_LtrTsSq = CloseLtr $\rightarrow_{cT}$ NX_LtrSaSq := 2 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ S_LtrCd and Dy $\rightarrow_{cT}$
   B_OpLtr := true $\rightarrow_c$ state_LtrTsSq := OpenLtr2 $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$
   LtrTsSq(DCUIMG_C_LtrTs,
   DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp,
   DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr,
   DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)
7. ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$
   state_LtrTsSq = CloseLtr $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr2 $\rightarrow_{cT}$ NX_LtrSaSq := 3 $\rightarrow_c$
   B_LtrFl $\rightarrow_{cT}$ state_LtrTsSq := Start $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(DCUIMG_C_LtrTs,
   DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp,
   DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr,
   DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)
8. ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$
   state_LtrTsSq = CloseLtr $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr2 $\rightarrow_{cT}$ NX_LtrSaSq := 3 $\rightarrow_c$
   B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ A_LtrTs := true $\rightarrow_c$ state_LtrTsSq := Ready $\rightarrow_c$
   EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(DCUIMG_C_LtrTs,
   DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp,
   DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr,
   DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)
9. ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$
   state_LtrTsSq = CloseLtr $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr2 $\rightarrow_{cF}$ state_LtrTsSq = Ready $\rightarrow_{cT}$
   NX_LtrSaSq := 4 $\rightarrow_c$ S_DCUNtRdy $\rightarrow_{cT}$ state_LtrTsSq := Start $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$
   LtrTsSq(DCUIMG_C_LtrTs,
   DHSSMG_B_LtrFl,DHWOMG_S_LtrCd,DIGIMG_S_LtrOp,
   DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn,DHSSMG_B_OpLtr,
   DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1,temp2)
10. state_LtrTsSq := Ready $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = Ready
11. state_LtrTsSq := OpenLtr1 $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = OpenLtr1
12. state_LtrTsSq := Start $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = Start
13. state_LtrTsSq := CloseLtr $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = CloseLtr
14. state_LtrTsSq := Start $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = Start
15. state_LtrTsSq := OpenLtr2 $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = OpenLtr2
16. state_LtrTsSq := Start $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = Start
17. state_LtrTsSq := Ready $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = Ready
18. state_LtrTsSq := Start $\rightarrow_d$ Internal_connection2 := state $\rightarrow_{d-in}$
    State := internal_connection2 $\rightarrow_d$ state_LtrTsSq = Start

19. Dy := true $\rightarrow_d$ Internal_connection1 := Dy $\rightarrow_{d-in}$ Dy := internal_connection1 $\rightarrow_d$ S_LtrCd and Dy

Controller.Impl:

1. ENTRY Controller.Impl (on dispatch) $\rightarrow_{cT}$ LtrInt(. . .) $\rightarrow_c$ dcu2_line_trip(. . .) $\rightarrow_c$ EXIT Controller.Impl

LtrInt.Imp:

1. ENTRY LtrInt.Imp $\rightarrow_c$ temp1 := NX_SqSt >= 3 $\rightarrow_c$ WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_c$ temp3 := NX_SqSt >= 38 $\rightarrow_c$ temp4 := NX_SqSt = 30 $\rightarrow_c$ temp5 := NX_SqSt = 31 $\rightarrow_c$ temp6 := B_RqPrSd and temp1 $\rightarrow_c$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_c$ temp8 := temp2 or temp3 $\rightarrow_c$ F_TRIG(temp4,temp9) $\rightarrow_c$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_c$ R_TRIG(temp8,temp11) $\rightarrow_c$ temp12 := temp9 or temp5 $\rightarrow_c$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_c$ temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_c$ temp15 := temp12 and temp13 $\rightarrow_c$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_c$ RS(temp7,temp16,temp17) $\rightarrow_c$ B_ClLtr := temp14 and temp17 $\rightarrow_c$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(. . .)

dcu2_line_trip.Impl:

1. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cT}$ not fb $\rightarrow_{cT}$ btemp := true $\rightarrow_c$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .)
2. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cT}$ not fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .)
3. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cT}$ btemp := true $\rightarrow_c$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .)
4. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .)
5. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_c$

fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .)

6. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cT}$ btemp := true $\rightarrow_c$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .)
7. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cF}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .)
8. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .)
9. GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_d$ Controller-Controller := GPIO_OUT $\rightarrow_{d-in}$ GPIO_OUT := Controller-Controller $\rightarrow_d$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N
10. GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_d$ Controller-Controller := GPIO_OUT $\rightarrow_{d-in}$ GPIO_OUT := Controller-Controller $\rightarrow_d$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N
11. GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_d$ Controller-Controller := GPIO_OUT $\rightarrow_{d-in}$ GPIO_OUT := Controller-Controller $\rightarrow_d$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N
12. GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_d$ Controller-Controller := GPIO_OUT $\rightarrow_{d-in}$ GPIO_OUT := Controller-Controller $\rightarrow_d$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N

Verification sequences: direct paths

1. LtrTsSq(DCUIMG_C_LtrTs,DHSSMG_B_LtrFl,DHWOMG_S_LtrCd, DIGIMG_S_LtrOp,DCUIMG_S_DcuNtRdy,PARAGP_L_LtrSvEn, DHSSMG_B_OpLtr,DHSSMG_B_CdLtr,DHSSMG_NX_LtrSaSq,temp1, temp2) $\rightarrow_{call}$ ENTRY LtrTsSq.Impl
   (i) C_LtrTs_in := DCUIMG_C_LtrTs $\rightarrow_{d-in}$ C_LtrTs := C_LtrTs_in
   (ii) B_LtrFl_in := DHSSMG_B_LtrFl $\rightarrow_{d-in}$ B_LtrFl := B_LtrFl_in
   (iii) S_LtrCd_in := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ S_LtrCd := S_LtrCd_in
   (iv) S_LtrOp_in := DIGIMG_S_LtrOp $\rightarrow_{d-in}$ S_LtrOp := S_LtrOp_in
   (v) S_DCUNtRdy_in := DCUIMG_S_DcuNtRdy $\rightarrow_{d-in}$ S_DCUNtRdy := S_DCUNtRdy_in
   (vi) L_EnLtrSv_in := PARAGP_L_LtrSvEn $\rightarrow_{d-in}$ L_EnLtrSv := L_EnLtrSv_in
   (vii) B_OpLtr_out := B_OpLtr $\rightarrow_{d-out}$ DHSSMG_B_OpLtr := B_OpLtr_out
   (viii) B_CdLtr_out := B_CdLtr $\rightarrow_{d-out}$ DHSSMG_B_CdLtr := B_CdLtr_out
   (ix) NX_LtrSaSq_out := NX_LtrSaSq $\rightarrow_{d-out}$ DHSSMG_NX_LtrSaSq := NX_LtrSaSq_out
   (x) A_LtrTs_out := A_LtrTs $\rightarrow_{d-out}$ temp1 := A_LtrTs_out
   (xi) A_LtrOpVd_out := A_LtrOpVd $\rightarrow_{d-out}$ temp2 := A_LtrOpVd_out

2. LtrInt(PRASMZ_B_RqPrSd,APSIMZ_B_OpLtr,SSSCMZ_NX_MnSqSt, PCTHMZ_A_PctMo,PLTTMG_B_OpLtr,DHSSMG_B_OpLtr, DHSSMG_B_CdLtr,APSIMZ_B_EnCdLnTrpSlt,PARAGP_L_CnfHpp, DIGOMG_B_CdLtr) $\rightarrow_{call}$ ENTRY LtrInt.Impl
   - (i) B_RqPrSd_in := PRASMZ_B_RqPrSd $\rightarrow_{d-in}$ B_RqPrSd := B_RqPrSd_in
   - (ii) B_OpLtr_AppSpec_in := APSIMZ_B_OpLtr $\rightarrow_{d-in}$ B_OpLtr_AppSpec := B_OpLtr_AppSpec_in
   - (iii) NX_SqSt_in := SSSCMZ_NX_MnSqSt $\rightarrow_{d-in}$ NX_SqSt := NX_SqSt_in
   - (iv) A_PctMo_in := PCTHMZ_A_PctMo $\rightarrow_{d-in}$ A_PctMo := A_PctMo_in
   - (v) B_LtrTsOpLtr_in := PLTTMG_B_OpLtr $\rightarrow_{d-in}$ B_LtrTsOpLtr := B_LtrTsOpLtr_in
   - (vi) B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ B_OpLtr_LtrTs := B_OpLtr_LtrTs_in
   - (vii) B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
   - (viii) B_EnCdLnTrpSlt_in := APSIMZ_B_EnCdLnTrpSlt $\rightarrow_{d-in}$ B_EnCdLnTrpSlt := B_EnCdLnTrpSlt_in
   - (ix) L_CnfHpp_in := PARAGP_L_CnfHpp $\rightarrow_{d-in}$ L_CnfHpp := L_CnfHpp_in
   - (x) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
3. dcu2_line_trip(not DHSSMG_B_LtrHwOpFl,temp0,DHWOMG_S_LtrCd, DHWOMG_B_FpgaLtrOn,DHWOMG_B_DcuLtrFl) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl
   - (i) enable_in := not DHSSMG_B_LtrHwOpFl $\rightarrow_{d-in}$ enable := enable_in
   - (ii) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
   - (iii) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out
   - (iv) fpga2_on_out := fpga2_on $\rightarrow_{d-out}$ DHWOMG_B_FpgaLtrOn := fpga2_on_out
   - (v) fb_ne_out := fb_ne $\rightarrow_{d-out}$ DHWOMG_B_DcuLtrFl := fb_ne_out
4. connection1 := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ DHSSMG_B_OpLtr := connection1
   - ⟨connection1 := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ DHSSMG_B_OpLtr := connection1, $\{Latency => Xms..Xms\}$⟩
5. connection2 := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ DHSSMG_B_CdLtr := connection2
   - ⟨connection2 := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ DHSSMG_B_CdLtr := connection2, $\{Latency => Xms..Xms\}$⟩
6. connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3
   - ⟨connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3, $\{Latency => Xms..Xms\}$⟩

Verification sequences: indirect paths
LtrTsSq.Impl → Tester.Impl → Controller.Impl → dcu2_line_trip.Impl

1. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cT}$ NX_LtrSaSq := 0 $\rightarrow_{c}$ (not L_EnLtrSv) and C_LtrTs $\rightarrow_{cF}$ L_EnLtrSv and C_LtrTs $\rightarrow_{cT}$ B_OpLtr := true $\rightarrow_{c}$ tate_LtrTsSq := OpenLtr1 $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(. . .) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$ EXIT Tester.Impl(connection1 := DHSSMG_B_OpLtr) $\rightarrow_{d-in}$ ENTRY Controller.Impl(on dispatch(DHSSMG_B_OpLtr := connection1)) $\rightarrow_{cT}$ LtrInt(. . .) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_{c}$ temp1 := NX_SqSt >= 3 $\rightarrow_{c}$ WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_{c}$ temp3 := NX_SqSt >= 38 $\rightarrow_{c}$

temp4 := NX_SqSt = 30 $\rightarrow_c$ temp5 := NX_SqSt = 31 $\rightarrow_c$ temp6 := B_RqPrSd and temp1 $\rightarrow_c$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_c$ temp8 := temp2 or temp3 $\rightarrow_c$ F_TRIG(temp4,temp9) $\rightarrow_c$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_c$ R_TRIG(temp8,temp11) $\rightarrow_c$ temp12 := temp9 or temp5 $\rightarrow_c$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_c$ temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_c$ temp15 := temp12 and temp13 $\rightarrow_c$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_c$ RS(temp7,temp16,temp17) $\rightarrow_c$ B_ClLtr := temp14 and temp17 $\rightarrow_c$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(. . .) $\rightarrow_c$ dcu2_line_trip(. . .) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_c$ EXIT Controller.Impl

(i) B_OpLtr := true $\rightarrow_d$ B_OpLtr_out := B_OpLtr
(ii) B_OpLtr_out := B_OpLtr $\rightarrow_{d-out}$ DHSSMG_B_OpLtr := B_OpLtr_out
(iii) DHSSMG_B_OpLtr := B_OpLtr_out $\rightarrow_d$ connection1 := DHSSMG_B_OpLtr
(iv) DHSSMG_B_OpLtr := connection1 $\rightarrow_d$ B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr
(v) B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ B_OpLtr_LtrTs := B_OpLtr_LtrTs_in
(vi) B_OpLtr_LtrTs := B_OpLtr_LtrTs_in $\rightarrow_d$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec
(vii) B_ClLtr := temp14 and temp17 $\rightarrow_d$ B_ClLtr_out := B_ClLtr
(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_d$ act_in := DIGOMG_B_CdLtr
(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
(xi) act := act_in $\rightarrow_d$ act
(xii) act := act_in $\rightarrow_d$ enable and act and fpga2_on
(xiii) fb := temp and LT_RELAY_FB $\rightarrow_d$ fb_out := fb
(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

- ⟨connection1 := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ DHSSMG_B_OpLtr := connection1, $\{Latency => Xms..Xms\}$⟩

2. ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cT}$ NX_LtrSaSq := 0 $\rightarrow_c$ (not L_EnLtrSv) and C_LtrTs $\rightarrow_{cF}$ L_EnLtrSv and C_LtrTs $\rightarrow_{cT}$ B_OpLtr := true $\rightarrow_c$ tate_LtrTsSq := OpenLtr1 $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(. . .) $\rightarrow_c$ temp3 := temp1 and temp2 $\rightarrow_c$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_c$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_c$ EXIT Tester.Impl(connection1 := DHSSMG_B_OpLtr) $\rightarrow_{d-in}$ ENTRY Controller.Impl(on dispatch(DHSSMG_B_OpLtr := connection1)) $\rightarrow_{cT}$ LtrInt(. . .) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_c$ temp1 := NX_SqSt >= 3 $\rightarrow_c$ WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_c$ temp3 := NX_SqSt >= 38 $\rightarrow_c$ temp4 := NX_SqSt = 30 $\rightarrow_c$ temp5 := NX_SqSt = 31 $\rightarrow_c$ temp6 := B_RqPrSd and temp1 $\rightarrow_c$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_c$ temp8 := temp2 or temp3 $\rightarrow_c$ F_TRIG(temp4,temp9) $\rightarrow_c$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_c$ R_TRIG(temp8,temp11) $\rightarrow_c$ temp12 := temp9 or temp5 $\rightarrow_c$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_c$ temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_c$ temp15 := temp12 and temp13 $\rightarrow_c$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_c$ RS(temp7,temp16,temp17) $\rightarrow_c$ B_ClLtr := temp14 and temp17 $\rightarrow_c$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(. . .) $\rightarrow_c$

dcu2_line_trip(...) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$ temp = false $\rightarrow_{c}$ btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_{c}$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_{c}$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$ fb := temp and LT_RELAY_FB $\rightarrow_{c}$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(...) $\rightarrow_{c}$ EXIT Controller.Impl

(i) B_OpLtr := true $\rightarrow_{d}$ B_OpLtr_out := B_OpLtr
(ii) B_OpLtr_out := B_OpLtr $\rightarrow_{d-out}$ DHSSMG_B_OpLtr := B_OpLtr_out
(iii) DHSSMG_B_OpLtr := B_OpLtr_out $\rightarrow_{d}$ connection1 := DHSSMG_B_OpLtr
(iv) DHSSMG_B_OpLtr := connection1 $\rightarrow_{d}$ B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr
(v) B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ B_OpLtr_LtrTs := B_OpLtr_LtrTs_in
(vi) B_OpLtr_LtrTs := B_OpLtr_LtrTs_in
$\rightarrow_{d}$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec
(vii) B_ClLtr := temp14 and temp17 $\rightarrow_{d}$ B_ClLtr_out := B_ClLtr
(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_{d}$ act_in := DIGOMG_B_CdLtr
(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
(xi) act := act_in $\rightarrow_{d}$ act
(xii) act := act_in $\rightarrow_{d}$ enable and act and fpga2_on
(xiii) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb
(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out
- ⟨connection1 := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ DHSSMG_B_OpLtr := connection1, $\{Latency => Xms..Xms\}$⟩

3. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$ state_LtrTsSq = CloseLtr $\rightarrow_{cT}$ NX_LtrSaSq := 2 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cF}$ S_LtrCd and Dy $\rightarrow_{cT}$ B_OpLtr := true $\rightarrow_{c}$ state_LtrTsSq := OpenLtr2 $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(...) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$
EXIT Tester.Impl(connection1 := DHSSMG_B_OpLtr) $\rightarrow_{d-in}$
ENTRY Controller.Impl(on dispatch(DHSSMG_B_OpLtr := connection1)) $\rightarrow_{cT}$
LtrInt(...) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_{c}$
temp1 := NX_SqSt >= 3 $\rightarrow_{c}$ WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_{c}$
temp3 := NX_SqSt >= 38 $\rightarrow_{c}$ temp4 := NX_SqSt = 30 $\rightarrow_{c}$ temp5 := NX_SqSt = 31 $\rightarrow_{c}$
temp6 := B_RqPrSd and temp1 $\rightarrow_{c}$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_{c}$
temp8 := temp2 or temp3 $\rightarrow_{c}$ F_TRIG(temp4,temp9) $\rightarrow_{c}$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_{c}$
R_TRIG(temp8,temp11) $\rightarrow_{c}$ temp12 := temp9 or temp5 $\rightarrow_{c}$
temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_{c}$ temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_{c}$
temp15 := temp12 and temp13 $\rightarrow_{c}$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_{c}$
RS(temp7,temp16,temp17) $\rightarrow_{c}$ B_ClLtr := temp14 and temp17 $\rightarrow_{c}$ EXIT LtrInt.Impl $\rightarrow_{call}$
LtrInt(...) $\rightarrow_{c}$ dcu2_line_trip(...) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$
temp = false $\rightarrow_{c}$ btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cT}$
GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_{c}$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_{c}$
fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$ fb := temp and LT_RELAY_FB $\rightarrow_{c}$
enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$
dcu2_line_trip(...) $\rightarrow_{c}$ EXIT Controller.Impl

(i) B_OpLtr := true $\rightarrow_{d}$ B_OpLtr_out := B_OpLtr

(ii) B_OpLtr_out := B_OpLtr $\rightarrow_{d-out}$ DHSSMG_B_OpLtr := B_OpLtr_out

(iii) DHSSMG_B_OpLtr := B_OpLtr_out $\rightarrow_{d}$ connection1 := DHSSMG_B_OpLtr

(iv) DHSSMG_B_OpLtr := connection1 $\rightarrow_{d}$ B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr

(v) B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ B_OpLtr_LtrTs := B_OpLtr_LtrTs_in

(vi) B_OpLtr_LtrTs := B_OpLtr_LtrTs_in
$\rightarrow_{d}$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec

(vii) B_ClLtr := temp14 and temp17 $\rightarrow_{d}$ B_ClLtr_out := B_ClLtr

(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out

(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_{d}$ act_in := DIGOMG_B_CdLtr

(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in

(xi) act := act_in $\rightarrow_{d}$ act

(xii) act := act_in $\rightarrow_{d}$ enable and act and fpga2_on

(xiii) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb

(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

- ⟨connection1 := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ DHSSMG_B_OpLtr := connection1, $\{Latency => Xms..Xms\}$⟩

4. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$ state_LtrTsSq = CloseLtr $\rightarrow_{cT}$ NX_LtrSaSq := 2 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cF}$ S_LtrCd and Dy $\rightarrow_{cT}$ B_OpLtr := true $\rightarrow_{c}$ state_LtrTsSq := OpenLtr2 $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(. . .) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$
EXIT Tester.Impl(connection1 := DHSSMG_B_OpLtr) $\rightarrow_{d-in}$
ENTRY Controller.Impl(on dispatch(DHSSMG_B_OpLtr := connection1)) $\rightarrow_{cT}$
LtrInt(. . .) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_{c}$ temp1 := NX_SqSt >= 3 $\rightarrow_{c}$
WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_{c}$ temp3 := NX_SqSt >= 38 $\rightarrow_{c}$
temp4 := NX_SqSt = 30 $\rightarrow_{c}$ temp5 := NX_SqSt = 31 $\rightarrow_{c}$ temp6 := B_RqPrSd and temp1 $\rightarrow_{c}$
temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_{c}$ temp8 := temp2 or temp3 $\rightarrow_{c}$
F_TRIG(temp4,temp9) $\rightarrow_{c}$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_{c}$ R_TRIG(temp8,temp11) $\rightarrow_{c}$
temp12 := temp9 or temp5 $\rightarrow_{c}$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_{c}$
temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_{c}$ temp15 := temp12 and temp13 $\rightarrow_{c}$
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_{c}$ RS(temp7,temp16,temp17) $\rightarrow_{c}$
B_ClLtr := temp14 and temp17 $\rightarrow_{c}$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(. . .) $\rightarrow_{c}$
dcu2_line_trip(. . .) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$ temp = false $\rightarrow_{c}$
btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_{c}$
act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_{c}$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$
fb := temp and LT_RELAY_FB $\rightarrow_{c}$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$
fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_{c}$
EXIT Controller.Impl

(i) B_OpLtr := true $\rightarrow_{d}$ B_OpLtr_out := B_OpLtr

(ii) B_OpLtr_out := B_OpLtr $\rightarrow_{d-out}$ DHSSMG_B_OpLtr := B_OpLtr_out

(iii) DHSSMG_B_OpLtr := B_OpLtr_out $\rightarrow_{d}$ connection1 := DHSSMG_B_OpLtr

(iv) DHSSMG_B_OpLtr := connection1 $\rightarrow_{d}$ B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr

(v) B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ B_OpLtr_LtrTs := B_OpLtr_LtrTs_in

(vi) B_OpLtr_LtrTs := B_OpLtr_LtrTs_in $\rightarrow_{d}$
temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec

(vii) B_ClLtr := temp14 and temp17 $\rightarrow_{d}$ B_ClLtr_out := B_ClLtr

(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_{d}$ act_in := DIGOMG_B_CdLtr
(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
(xi) act := act_in $\rightarrow_{d}$ act
(xii) act := act_in $\rightarrow_{d}$ enable and act and fpga2_on
(xiii) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb
(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

- ⟨connection1 := DHSSMG_B_OpLtr $\rightarrow_{d-in}$ DHSSMG_B_OpLtr := connection1, $\{Latency => Xms..Xms\}$⟩

5. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$ NX_LtrSaSq := 1 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_{c}$ Dy := true $\rightarrow_{c}$ state_LtrTsSq := CloseLtr $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(. . .) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$ EXIT Tester.Impl(connection2 := DHSSMG_B_CdLtr) $\rightarrow_{d-in}$ ENTRY Controller.Impl(on dispatch(DHSSMG_B_CdLtr := connection2)) $\rightarrow_{cT}$ LtrInt(. . .) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_{c}$ temp1 := NX_SqSt >= 3 $\rightarrow_{c}$ WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_{c}$ temp3 := NX_SqSt >= 38 $\rightarrow_{c}$ temp4 := NX_SqSt = 30 $\rightarrow_{c}$ temp5 := NX_SqSt = 31 $\rightarrow_{c}$ temp6 := B_RqPrSd and temp1 $\rightarrow_{c}$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_{c}$ temp8 := temp2 or temp3 $\rightarrow_{c}$ F_TRIG(temp4,temp9) $\rightarrow_{c}$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_{c}$ R_TRIG(temp8,temp11) $\rightarrow_{c}$ temp12 := temp9 or temp5 $\rightarrow_{c}$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_{c}$ temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_{c}$ temp15 := temp12 and temp13 $\rightarrow_{c}$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_{c}$ RS(temp7,temp16,temp17) $\rightarrow_{c}$ B_ClLtr := temp14 and temp17 $\rightarrow_{c}$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(. . .) $\rightarrow_{c}$ dcu2_line_trip(. . .) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$ temp = false $\rightarrow_{c}$ btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_{c}$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_{c}$ FPGA_LTRCR := temp $\rightarrow_{c}$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$ fb := temp and LT_RELAY_FB $\rightarrow_{c}$ enable and act and fpga2_on $\rightarrow_{cT}$ not fb $\rightarrow_{cT}$ btemp := true $\rightarrow_{c}$ fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_{c}$ EXIT Controller.Impl

(i) B_CdLtr := true $\rightarrow_{d}$ B_CdLtr_out := B_CdLtr
(ii) B_CdLtr_out := B_CdLtr $\rightarrow_{d-out}$ DHSSMG_B_CdLtr := B_CdLtr_out
(iii) DHSSMG_B_CdLtr := B_CdLtr_out $\rightarrow_{d}$ connection2 := DHSSMG_B_CdLtr
(iv) DHSSMG_B_CdLtr := connection2 $\rightarrow_{d}$ B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr
(v) B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
(vi) B_CdLtr_LtrTs := B_CdLtr_LtrTs_in $\rightarrow_{d}$
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15
(vii) B_ClLtr := temp14 and temp17 $\rightarrow_{d}$ B_ClLtr_out := B_ClLtr
(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_{d}$ act_in := DIGOMG_B_CdLtr
(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
(xi) act := act_in $\rightarrow_{d}$ act
(xii) act := act_in $\rightarrow_{d}$ enable and act and fpga2_on
(xiii) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb
(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

  - ⟨B_CdLtr := true $\rightarrow_c$ Dy := true, $\{Delay(512ms)\}$⟩
  - ⟨connection2 := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ DHSSMG_B_CdLtr := connection2, $\{Latency => Xms..Xms\}$⟩

6. ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$ NX_LtrSaSq := 1 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_c$ Dy := true $\rightarrow_c$ state_LtrTsSq := CloseLtr $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(...) $\rightarrow_c$ temp3 := temp1 and temp2 $\rightarrow_c$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_c$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_c$ EXIT Tester.Impl(connection2 := DHSSMG_B_CdLtr) $\rightarrow_{d-in}$ ENTRY Controller.Impl(on dispatch(DHSSMG_B_CdLtr := connection2)) $\rightarrow_{cT}$ LtrInt(...) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_c$ temp1 := NX_SqSt >= 3 $\rightarrow_c$ WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_c$ temp3 := NX_SqSt >= 38 $\rightarrow_c$ temp4 := NX_SqSt = 30 $\rightarrow_c$ temp5 := NX_SqSt = 31 $\rightarrow_c$ temp6 := B_RqPrSd and temp1 $\rightarrow_c$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_c$ temp8 := temp2 or temp3 $\rightarrow_c$ F_TRIG(temp4,temp9) $\rightarrow_c$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_c$ R_TRIG(temp8,temp11) $\rightarrow_c$ temp12 := temp9 or temp5 $\rightarrow_c$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_c$ temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_c$ temp15 := temp12 and temp13 $\rightarrow_c$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_c$ RS(temp7,temp16,temp17) $\rightarrow_c$ B_ClLtr := temp14 and temp17 $\rightarrow_c$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(...) $\rightarrow_c$ dcu2_line_trip(...) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cT}$ not fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(...) $\rightarrow_c$ EXIT Controller.Impl

   (i) B_CdLtr := true $\rightarrow_d$ B_CdLtr_out := B_CdLtr
   (ii) B_CdLtr_out := B_CdLtr $\rightarrow_{d-out}$ DHSSMG_B_CdLtr := B_CdLtr_out
   (iii) DHSSMG_B_CdLtr := B_CdLtr_out $\rightarrow_d$ connection2 := DHSSMG_B_CdLtr
   (iv) DHSSMG_B_CdLtr := connection2 $\rightarrow_d$ B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr
   (v) B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
   (vi) B_CdLtr_LtrTs := B_CdLtr_LtrTs_in $\rightarrow_d$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15
   (vii) B_ClLtr := temp14 and temp17 $\rightarrow_d$ B_ClLtr_out := B_ClLtr
   (viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
   (ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_d$ act_in := DIGOMG_B_CdLtr
   (x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
   (xi) act := act_in $\rightarrow_d$ act
   (xii) act := act_in $\rightarrow_d$ enable and act and fpga2_on
   (xiii) fb := temp and LT_RELAY_FB $\rightarrow_d$ fb_out := fb
   (xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

  - ⟨B_CdLtr := true $\rightarrow_c$ Dy := true, $\{Delay(512ms)\}$⟩
  - ⟨connection2 := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ DHSSMG_B_CdLtr := connection2, $\{Latency => Xms..Xms\}$⟩

7. ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$ NX_LtrSaSq := 1 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_c$

Dy := true $\rightarrow_{c}$ state_LtrTsSq := CloseLtr $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$
LtrTsSq(. . .) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$
SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$
EXIT Tester.Impl(connection2 := DHSSMG_B_CdLtr) $\rightarrow_{d-in}$
ENTRY Controller.Impl(on dispatch(DHSSMG_B_CdLtr := connection2)) $\rightarrow_{cT}$
LtrInt(. . .) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_{c}$ temp1 := NX_SqSt >= 3 $\rightarrow_{c}$
WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_{c}$ temp3 := NX_SqSt >= 38 $\rightarrow_{c}$
temp4 := NX_SqSt = 30 $\rightarrow_{c}$ temp5 := NX_SqSt = 31 $\rightarrow_{c}$ temp6 := B_RqPrSd and temp1 $\rightarrow_{c}$
temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_{c}$ temp8 := temp2 or temp3 $\rightarrow_{c}$
F_TRIG(temp4,temp9) $\rightarrow_{c}$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_{c}$ R_TRIG(temp8,temp11) $\rightarrow_{c}$
temp12 := temp9 or temp5 $\rightarrow_{c}$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_{c}$
temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_{c}$ temp15 := temp12 and temp13 $\rightarrow_{c}$
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_{c}$ RS(temp7,temp16,temp17) $\rightarrow_{c}$
B_ClLtr := temp14 and temp17 $\rightarrow_{c}$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(. . .) $\rightarrow_{c}$
dcu2_line_trip(. . .) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$ temp = false $\rightarrow_{c}$
btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_{c}$
act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_{c}$ FPGA_LTRCR := temp $\rightarrow_{c}$
fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$ fb := temp and LT_RELAY_FB $\rightarrow_{c}$
enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cT}$ btemp := true $\rightarrow_{c}$ fb_ne := btemp $\rightarrow_{c}$
EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_{c}$ EXIT Controller.Impl

(i) B_CdLtr := true $\rightarrow_{d}$ B_CdLtr_out := B_CdLtr
(ii) B_CdLtr_out := B_CdLtr $\rightarrow_{d-out}$ DHSSMG_B_CdLtr := B_CdLtr_out
(iii) DHSSMG_B_CdLtr := B_CdLtr_out $\rightarrow_{d}$ connection2 := DHSSMG_B_CdLtr
(iv) DHSSMG_B_CdLtr := connection2 $\rightarrow_{d}$ B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr
(v) B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
(vi) B_CdLtr_LtrTs := B_CdLtr_LtrTs_in $\rightarrow_{d}$
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15
(vii) B_ClLtr := temp14 and temp17 $\rightarrow_{d}$ B_ClLtr_out := B_ClLtr
(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_{d}$ act_in := DIGOMG_B_CdLtr
(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
(xi) act := act_in $\rightarrow_{d}$ act
(xii) act := act_in $\rightarrow_{d}$ enable and act and fpga2_on
(xiii) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb
(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

- ⟨B_CdLtr := true $\rightarrow_{c}$ Dy := true, $\{Delay(512ms)\}$⟩
- ⟨connection2 := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ DHSSMG_B_CdLtr := connection2, $\{Latency => Xms..Xms\}$⟩

8. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$
NX_LtrSaSq := 1 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_{c}$
Dy := true $\rightarrow_{c}$ state_LtrTsSq := CloseLtr $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$
LtrTsSq(. . .) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$
SR(temp3,false,DHSSMG_S_LtrOpVd
$\rightarrow_{c}$ EXIT Tester.Impl(connection2 := DHSSMG_B_CdLtr) $\rightarrow_{d-in}$
ENTRY Controller.Impl(on dispatch(DHSSMG_B_CdLtr := connection2)) $\rightarrow_{cT}$
LtrInt(. . .) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_{c}$ temp1 := NX_SqSt >= 3 $\rightarrow_{c}$

WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_{c}$ temp3 := NX_SqSt >= 38 $\rightarrow_{c}$
temp4 := NX_SqSt = 30 $\rightarrow_{c}$ temp5 := NX_SqSt = 31 $\rightarrow_{c}$ temp6 := B_RqPrSd and temp1 $\rightarrow_{c}$
temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_{c}$ temp8 := temp2 or temp3 $\rightarrow_{c}$
F_TRIG(temp4,temp9) $\rightarrow_{c}$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_{c}$ R_TRIG(temp8,temp11) $\rightarrow_{c}$
temp12 := temp9 or temp5 $\rightarrow_{c}$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_{c}$
temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_{c}$ temp15 := temp12 and temp13 $\rightarrow_{c}$
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_{c}$ RS(temp7,temp16,temp17) $\rightarrow_{c}$
B_ClLtr := temp14 and temp17 $\rightarrow_{c}$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(...) $\rightarrow_{c}$
dcu2_line_trip(...) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$ temp = false $\rightarrow_{c}$
btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_{c}$
act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_{c}$ FPGA_LTRCR := temp $\rightarrow_{c}$
fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$ fb := temp and LT_RELAY_FB $\rightarrow_{c}$
enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$
dcu2_line_trip(...) $\rightarrow_{c}$ EXIT Controller.Impl

(i) B_CdLtr := true $\rightarrow_{d}$ B_CdLtr_out := B_CdLtr
(ii) B_CdLtr_out := B_CdLtr $\rightarrow_{d-out}$ DHSSMG_B_CdLtr := B_CdLtr_out
(iii) DHSSMG_B_CdLtr := B_CdLtr_out $\rightarrow_{d}$ connection2 := DHSSMG_B_CdLtr
(iv) DHSSMG_B_CdLtr := connection2 $\rightarrow_{d}$ B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr
(v) B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
(vi) B_CdLtr_LtrTs := B_CdLtr_LtrTs_in $\rightarrow_{d}$
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15
(vii) B_ClLtr := temp14 and temp17 $\rightarrow_{d}$ B_ClLtr_out := B_ClLtr
(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_{d}$ act_in := DIGOMG_B_CdLtr
(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
(xi) act := act_in $\rightarrow_{d}$ act
(xii) act := act_in $\rightarrow_{d}$ enable and act and fpga2_on
(xiii) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb
(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

- ⟨B_CdLtr := true $\rightarrow_{c}$ Dy := true, $\{Delay(512ms)\}$⟩
- ⟨connection2 := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ DHSSMG_B_CdLtr := connection2, $\{Latency => Xms..Xms\}$⟩

9. ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$
NX_LtrSaSq := 1 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_{c}$
Dy := true $\rightarrow_{c}$ state_LtrTsSq := CloseLtr $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$
LtrTsSq(...) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$
SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$
EXIT Tester.Impl(connection2 := DHSSMG_B_CdLtr) $\rightarrow_{d-in}$
ENTRY Controller.Impl(on dispatch(DHSSMG_B_CdLtr := connection2)) $\rightarrow_{cT}$
LtrInt(...) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_{c}$ temp1 := NX_SqSt >= 3 $\rightarrow_{c}$
WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_{c}$ temp3 := NX_SqSt >= 38 $\rightarrow_{c}$
temp4 := NX_SqSt = 30 $\rightarrow_{c}$ temp5 := NX_SqSt = 31 $\rightarrow_{c}$ temp6 := B_RqPrSd and temp1 $\rightarrow_{c}$
temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_{c}$ temp8 := temp2 or temp3 $\rightarrow_{c}$
F_TRIG(temp4,temp9) $\rightarrow_{c}$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_{c}$ R_TRIG(temp8,temp11) $\rightarrow_{c}$
temp12 := temp9 or temp5 $\rightarrow_{c}$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_{c}$
temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_{c}$ temp15 := temp12 and temp13 $\rightarrow_{c}$

temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_c$ RS(temp7,temp16,temp17) $\rightarrow_c$ B_ClLtr := temp14 and temp17 $\rightarrow_c$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(. . .) $\rightarrow_c$ dcu2_line_trip(. . .) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cT}$ btemp := true $\rightarrow_c$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_c$ EXIT Controller.Impl

(i) B_CdLtr := true $\rightarrow_d$ B_CdLtr_out := B_CdLtr
(ii) B_CdLtr_out := B_CdLtr $\rightarrow_{d-out}$ DHSSMG_B_CdLtr := B_CdLtr_out
(iii) DHSSMG_B_CdLtr := B_CdLtr_out $\rightarrow_d$ connection2 := DHSSMG_B_CdLtr
(iv) DHSSMG_B_CdLtr := connection2 $\rightarrow_d$ B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr
(v) B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
(vi) B_CdLtr_LtrTs := B_CdLtr_LtrTs_in $\rightarrow_d$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15
(vii) B_ClLtr := temp14 and temp17 $\rightarrow_d$ B_ClLtr_out := B_ClLtr
(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_d$ act_in := DIGOMG_B_CdLtr
(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
(xi) act := act_in $\rightarrow_d$ act
(xii) act := act_in $\rightarrow_d$ enable and act and fpga2_on
(xiii) fb := temp and LT_RELAY_FB $\rightarrow_d$ fb_out := fb
(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

- ⟨B_CdLtr := true $\rightarrow_c$ Dy := true, $\{Delay(512ms)\}$⟩
- ⟨connection2 := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ DHSSMG_B_CdLtr := connection2, $\{Latency => Xms..Xms\}$⟩

10. ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$ NX_LtrSaSq := 1 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_c$ Dy := true $\rightarrow_c$ state_LtrTsSq := CloseLtr $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(. . .) $\rightarrow_c$ temp3 := temp1 and temp2 $\rightarrow_c$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_c$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_c$ EXIT Tester.Impl(connection2 := DHSSMG_B_CdLtr) $\rightarrow_{d-in}$ ENTRY Controller.Impl(on dispatch(DHSSMG_B_CdLtr := connection2)) $\rightarrow_{cT}$ LtrInt(. . .) $\rightarrow_{call}$ ENTRY LtrInt.Imp $\rightarrow_c$ temp1 := NX_SqSt >= 3 $\rightarrow_c$ WITHIN_I(true,NX_SqSt,27,4,temp2) $\rightarrow_c$ temp3 := NX_SqSt >= 38 $\rightarrow_c$ temp4 := NX_SqSt = 30 $\rightarrow_c$ temp5 := NX_SqSt = 31 $\rightarrow_c$ temp6 := B_RqPrSd and temp1 $\rightarrow_c$ temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec $\rightarrow_c$ temp8 := temp2 or temp3 $\rightarrow_c$ F_TRIG(temp4,temp9) $\rightarrow_c$ F_TRIG(B_OpLtr_AppSpec,temp10) $\rightarrow_c$ R_TRIG(temp8,temp11) $\rightarrow_c$ temp12 := temp9 or temp5 $\rightarrow_c$ temp13 := L_CnfHpp or B_EnCdLnTrpSlt $\rightarrow_c$ temp14 := not (A_PctMo and B_LtrTsOpLtr) $\rightarrow_c$ temp15 := temp12 and temp13 $\rightarrow_c$ temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15 $\rightarrow_c$ RS(temp7,temp16,temp17) $\rightarrow_c$ B_ClLtr := temp14 and temp17 $\rightarrow_c$ EXIT LtrInt.Impl $\rightarrow_{call}$ LtrInt(. . .) $\rightarrow_c$ dcu2_line_trip(. . .) $\rightarrow_{call}$ ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$

enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_{c}$ EXIT Controller.Impl

(i) B_CdLtr := true $\rightarrow_{d}$ B_CdLtr_out := B_CdLtr
(ii) B_CdLtr_out := B_CdLtr $\rightarrow_{d-out}$ DHSSMG_B_CdLtr := B_CdLtr_out
(iii) DHSSMG_B_CdLtr := B_CdLtr_out $\rightarrow_{d}$ connection2 := DHSSMG_B_CdLtr
(iv) DHSSMG_B_CdLtr := connection2 $\rightarrow_{d}$ B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr
(v) B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
(vi) B_CdLtr_LtrTs := B_CdLtr_LtrTs_in $\rightarrow_{d}$
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15
(vii) B_ClLtr := temp14 and temp17 $\rightarrow_{d}$ B_ClLtr_out := B_ClLtr
(viii) B_ClLtr_out := B_ClLtr $\rightarrow_{d-out}$ DIGOMG_B_CdLtr := B_ClLtr_out
(ix) DIGOMG_B_CdLtr := B_ClLtr_out $\rightarrow_{d}$ act_in := DIGOMG_B_CdLtr
(x) act_in := DIGOMG_B_CdLtr $\rightarrow_{d-in}$ act := act_in
(xi) act := act_in $\rightarrow_{d}$ act
(xii) act := act_in $\rightarrow_{d}$ enable and act and fpga2_on
(xiii) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb
(xiv) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

- ⟨B_CdLtr := true $\rightarrow_{c}$ Dy := true, $\{Delay(512ms)\}$⟩
- ⟨connection2 := DHSSMG_B_CdLtr $\rightarrow_{d-in}$ DHSSMG_B_CdLtr := connection2, $\{Latency => Xms..Xms\}$⟩

dcu2_line_trip.Impl → Controller.Impl → Tester.Impl → LtrTsSq.Impl

11. ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$ temp = false $\rightarrow_{c}$ btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_{c}$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_{c}$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$ fb := temp and LT_RELAY_FB $\rightarrow_{c}$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_{c}$ EXIT Controller.Impl (connection3 := DHWOMG_S_LtrCd) $\rightarrow_{d-in}$ ENTRY Tester.Impl(DHWOMG_S_LtrCd := connection3) $\rightarrow_{c}$ LtrTsSq(. . .) $\rightarrow_{call}$ ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$ NX_LtrSaSq := 1 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_{c}$ Dy := true $\rightarrow_{c}$ state_LtrTsSq := CloseLtr $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(. . .) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$ EXIT Tester.Impl

(i) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb
(ii) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out
(iii) DHWOMG_S_LtrCd := fb_out $\rightarrow_{d}$ connection3 := DHWOMG_S_LtrCd
(iv) DHWOMG_S_LtrCd := connection3 $\rightarrow_{d}$ S_LtrCd_in := DHWOMG_S_LtrCd
(v) S_LtrCd_in := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ S_LtrCd := S_LtrCd_in
(vi) S_LtrCd := S_LtrCd_in $\rightarrow_{d}$ not S_LtrCd

- ⟨B_CdLtr := true $\rightarrow_{c}$ Dy := true, $\{Delay(512ms)\}$⟩
- ⟨connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3, $\{Latency => Xms..Xms\}$⟩

12. ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$ temp = false $\rightarrow_{c}$ btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_{c}$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_{c}$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$ fb := temp and LT_RELAY_FB $\rightarrow_{c}$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$

dcu2_line_trip(. . .) $\rightarrow_c$ EXIT Controller.Impl (connection3 := DHWOMG_S_LtrCd) $\rightarrow_{d-in}$
ENTRY Tester.Impl(DHWOMG_S_LtrCd := connection3) $\rightarrow_c$ LtrTsSq(. . .) $\rightarrow_{call}$
ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cT}$
NX_LtrSaSq := 1 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ B_CdLtr := true $\rightarrow_c$
Dy := true $\rightarrow_c$ state_LtrTsSq := CloseLtr $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$
LtrTsSq(. . .) $\rightarrow_c$ temp3 := temp1 and temp2 $\rightarrow_c$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_c$
SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_c$ EXIT Tester.Impl

(i) fb := temp and LT_RELAY_FB $\rightarrow_d$ fb_out := fb
(ii) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out
(iii) DHWOMG_S_LtrCd := fb_out $\rightarrow_d$ connection3 := DHWOMG_S_LtrCd
(iv) DHWOMG_S_LtrCd := connection3 $\rightarrow_d$ S_LtrCd_in := DHWOMG_S_LtrCd
(v) S_LtrCd_in := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ S_LtrCd := S_LtrCd_in
(vi) S_LtrCd := S_LtrCd_in $\rightarrow_d$ not S_LtrCd

- $\langle$B_CdLtr := true $\rightarrow_c$ Dy := true, $\{Delay(512ms)\}\rangle$
- $\langle$connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3, $\{Latency => Xms..Xms\}\rangle$

13. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$
GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$
FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$
fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cT}$ not fb $\rightarrow_{cF}$
fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_c$
EXIT Controller.Impl (connection3 := DHWOMG_S_LtrCd) $\rightarrow_{d-in}$
ENTRY Tester.Impl(DHWOMG_S_LtrCd := connection3) $\rightarrow_c$ LtrTsSq(. . .) $\rightarrow_{call}$
ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$
state_LtrTsSq = CloseLtr $\rightarrow_{cT}$ NX_LtrSaSq := 2 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ S_LtrCd and Dy $\rightarrow_{cT}$
B_OpLtr := true $\rightarrow_c$ state_LtrTsSq := OpenLtr2 $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$
LtrTsSq(. . .) $\rightarrow_c$ temp3 := temp1 and temp2 $\rightarrow_c$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_c$
SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_c$ EXIT Tester.Impl

(i) fb := temp and LT_RELAY_FB $\rightarrow_d$ fb_out := fb
(ii) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out
(iii) DHWOMG_S_LtrCd := fb_out $\rightarrow_d$ connection3 := DHWOMG_S_LtrCd
(iv) DHWOMG_S_LtrCd := connection3 $\rightarrow_d$ S_LtrCd_in := DHWOMG_S_LtrCd
(v) S_LtrCd_in := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ S_LtrCd := S_LtrCd_in
(vi) S_LtrCd := S_LtrCd_in $\rightarrow_d$ S_LtrCd and Dy

- $\langle$connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3, $\{Latency => Xms..Xms\}\rangle$

14. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$
GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$
FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$
fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cT}$
btemp := true $\rightarrow_c$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(. . .) $\rightarrow_c$
EXIT Controller.Impl (connection3 := DHWOMG_S_LtrCd) $\rightarrow_{d-in}$
ENTRY Tester.Impl(DHWOMG_S_LtrCd := connection3) $\rightarrow_c$ LtrTsSq(. . .) $\rightarrow_{call}$
ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$
state_LtrTsSq = CloseLtr $\rightarrow_{cT}$ NX_LtrSaSq := 2 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ S_LtrCd and Dy $\rightarrow_{cT}$

B_OpLtr := true $\rightarrow_c$ state_LtrTsSq := OpenLtr2 $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(...) $\rightarrow_c$ temp3 := temp1 and temp2 $\rightarrow_c$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_c$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_c$ EXIT Tester.Impl

(i) fb := temp and LT_RELAY_FB $\rightarrow_d$ fb_out := fb
(ii) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out
(iii) DHWOMG_S_LtrCd := fb_out $\rightarrow_d$ connection3 := DHWOMG_S_LtrCd
(iv) DHWOMG_S_LtrCd := connection3 $\rightarrow_d$ S_LtrCd_in := DHWOMG_S_LtrCd
(v) S_LtrCd_in := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ S_LtrCd := S_LtrCd_in
(vi) S_LtrCd := S_LtrCd_in $\rightarrow_d$ S_LtrCd and Dy

- ⟨connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3, $\{Latency => Xms..Xms\}$⟩

15. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cT}$ temp := temp or MCU_LT_ON $\rightarrow_c$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cT}$ btemp := true $\rightarrow_c$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(...) $\rightarrow_c$ EXIT Controller.Impl (connection3 := DHWOMG_S_LtrCd) $\rightarrow_{d-in}$ ENTRY Tester.Impl(DHWOMG_S_LtrCd := connection3) $\rightarrow_c$ LtrTsSq(...) $\rightarrow_{call}$ ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$ state_LtrTsSq = CloseLtr $\rightarrow_{cT}$ NX_LtrSaSq := 2 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ S_LtrCd and Dy $\rightarrow_{cT}$ B_OpLtr := true $\rightarrow_c$ state_LtrTsSq := OpenLtr2 $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(...) $\rightarrow_c$ temp3 := temp1 and temp2 $\rightarrow_c$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_c$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_c$ EXIT Tester.Impl

(i) fb := temp and LT_RELAY_FB $\rightarrow_d$ fb_out := fb
(ii) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out
(iii) DHWOMG_S_LtrCd := fb_out $\rightarrow_d$ connection3 := DHWOMG_S_LtrCd
(iv) DHWOMG_S_LtrCd := connection3 $\rightarrow_d$ S_LtrCd_in := DHWOMG_S_LtrCd
(v) S_LtrCd_in := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ S_LtrCd := S_LtrCd_in
(vi) S_LtrCd := S_LtrCd_in $\rightarrow_d$ S_LtrCd and Dy

- ⟨connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3, $\{Latency => Xms..Xms\}$⟩

16. ENTRY dcu2_line_trip.Impl $\rightarrow_c$ temp = false $\rightarrow_c$ btemp = false $\rightarrow_c$ enable $\rightarrow_{cT}$ GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N $\rightarrow_c$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_c$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_c$ fb := temp and LT_RELAY_FB $\rightarrow_c$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_c$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(...) $\rightarrow_c$ EXIT Controller.Impl (connection3 := DHWOMG_S_LtrCd) $\rightarrow_{d-in}$ ENTRY Tester.Impl(DHWOMG_S_LtrCd := connection3) $\rightarrow_c$ LtrTsSq(...) $\rightarrow_{call}$ ENTRY LtrTsSq.Impl $\rightarrow_c$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$ state_LtrTsSq = CloseLtr $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr2 $\rightarrow_{cT}$ NX_LtrSaSq := 3 $\rightarrow_c$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ A_LtrTs := true $\rightarrow_c$ state_LtrTsSq := Ready $\rightarrow_c$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(...) $\rightarrow_c$ temp3 := temp1 and temp2 $\rightarrow_c$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_c$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_c$ EXIT Tester.Impl

(i) fb := temp and LT_RELAY_FB $\rightarrow_d$ fb_out := fb
(ii) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out

(iii) DHWOMG_S_LtrCd := fb_out $\rightarrow_{d}$ connection3 := DHWOMG_S_LtrCd
(iv) DHWOMG_S_LtrCd := connection3 $\rightarrow_{d}$ S_LtrCd_in := DHWOMG_S_LtrCd
(v) S_LtrCd_in := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ S_LtrCd := S_LtrCd_in
(vi) S_LtrCd := S_LtrCd_in $\rightarrow_{d}$ not S_LtrCd

- $\langle$connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3, $\{Latency => Xms..Xms\}\rangle$

17. ENTRY dcu2_line_trip.Impl $\rightarrow_{c}$ temp = false $\rightarrow_{c}$ btemp = false $\rightarrow_{c}$ enable $\rightarrow_{cF}$ GPIO_OUT:= GPIO_OUT or LTRIP_EN_N $\rightarrow_{c}$ act $\rightarrow_{cF}$ FPGA_LTRCR := temp $\rightarrow_{c}$ fpga2_on := temp and FPGA2_LT_ON $\rightarrow_{c}$ fb := temp and LT_RELAY_FB $\rightarrow_{c}$ enable and act and fpga2_on $\rightarrow_{cF}$ fb $\rightarrow_{cF}$ fb_ne := btemp $\rightarrow_{c}$ EXIT dcu2_line_trip.Impl $\rightarrow_{call}$ dcu2_line_trip(...) $\rightarrow_{c}$ EXIT Controller.Impl (connection3 := DHWOMG_S_LtrCd) $\rightarrow_{d-in}$ ENTRY Tester.Impl(DHWOMG_S_LtrCd := connection3) $\rightarrow_{c}$ LtrTsSq(...) $\rightarrow_{call}$ ENTRY LtrTsSq.Impl $\rightarrow_{c}$ State_LtrTsSq = Start $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr1 $\rightarrow_{cF}$ state_LtrTsSq = CloseLtr $\rightarrow_{cF}$ state_LtrTsSq = OpenLtr2 $\rightarrow_{cT}$ NX_LtrSaSq := 3 $\rightarrow_{c}$ B_LtrFl $\rightarrow_{cF}$ not S_LtrCd $\rightarrow_{cT}$ A_LtrTs := true $\rightarrow_{c}$ state_LtrTsSq := Ready $\rightarrow_{c}$ EXIT LtrTsSq.Impl $\rightarrow_{call}$ LtrTsSq(...) $\rightarrow_{c}$ temp3 := temp1 and temp2 $\rightarrow_{c}$ SR(temp1,false,DHSSMG_S_LtrTsRdy) $\rightarrow_{c}$ SR(temp3,false,DHSSMG_S_LtrOpVd $\rightarrow_{c}$ EXIT Tester.Impl

(i) fb := temp and LT_RELAY_FB $\rightarrow_{d}$ fb_out := fb
(ii) fb_out := fb $\rightarrow_{d-out}$ DHWOMG_S_LtrCd := fb_out
(iii) DHWOMG_S_LtrCd := fb_out $\rightarrow_{d}$ connection3 := DHWOMG_S_LtrCd
(iv) DHWOMG_S_LtrCd := connection3 $\rightarrow_{d}$ S_LtrCd_in := DHWOMG_S_LtrCd
(v) S_LtrCd_in := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ S_LtrCd := S_LtrCd_in
(vi) S_LtrCd := S_LtrCd_in $\rightarrow_{d}$ not S_LtrCd

- $\langle$connection3 := DHWOMG_S_LtrCd $\rightarrow_{d-in}$ DHWOMG_S_LtrCd := connection3, $\{Latency => Xms..Xms\}\rangle$

Appendix B Test suite

Table 28: Generated test cases

InternalTester1	InternalLtrTsSq1	InternalLtrTsSq2
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 1
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 1	some_connection13 = 1
some_connection14 = 1	some_connection14 = 0	some_connection14 = 1
d=20	*d=20*	*d=20*
connection1 = 1	**B_OpLtr_out = 0**	**B_OpLtr_out = 1**
connection2 = 0	**B_CdLtr_out = 0**	**B_CdLtr_out = 0**
some_connection15 = 0	**NX_LtrSaSq_out = 0**	**NX_LtrSaSq_out = 0**
some_connection16 = 0	**A_LtrTs_out = 0**	**A_LtrTs_out = 0**
some_connection17 = 0	**A_LtrOpVd_out = 0**	**A_LtrOpVd_out = 0**

Table 29: Generated test cases

InternalLtrTsSq3	InternalLtrTsSq4	InternalLtrTsSq5
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1

some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 1	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 1	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=24	*d=536*	*d=576*
B_OpLtr_out = 0	**B_OpLtr_out = 0**	some_connection1 = 0
B_CdLtr_out = 0	**B_CdLtr_out = 1**	some_connection2 = 0
NX_LtrSaSq_out = 1	**NX_LtrSaSq_out = 1**	some_connection3 = 0
A_LtrTs_out = 0	**A_LtrTs_out = 0**	some_connection4 = 0
A_LtrOpVd_out = 0	**A_LtrOpVd_out = 0**	some_connection5 = 0
		some_connection6 = 0
		some_connection7 = 0
		some_connection8 = 1
		FPGA2_LT_ON = 0
		LT_RELAY_FB = 1
		some_connection9 = 1
		some_connection10 = 1
		some_connection12 = 0
		some_connection13 = 1
		some_connection14 = 1
		d=24
		B_OpLtr_out = 0
		B_CdLtr_out = 0
		NX_LtrSaSq_out = 2
		A_LtrTs_out = 0
		A_LtrOpVd_out = 0

Table 30: Generated test cases

InternalLtrTsSq6	InternalLtrTsSq7	InternalLtrTsSq8

some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=576	*d=576*	*d=576*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1

d=24	*d=64*	*d=64*
B_OpLtr_out = 1	some_connection1 = 0	some_connection1 = 0
B_CdLtr_out = 0	some_connection2 = 0	some_connection2 = 0
NX_LtrSaSq_out = 2	some_connection3 = 0	some_connection3 = 0
A_LtrTs_out = 0	some_connection4 = 0	some_connection4 = 0
A_LtrOpVd_out = 0	some_connection5 = 0	some_connection5 = 0
	some_connection6 = 0	some_connection6 = 0
	some_connection7 = 0	some_connection7 = 0
	some_connection8 = 1	some_connection8 = 1
	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
	LT_RELAY_FB = 1	LT_RELAY_FB = 1
	some_connection9 = 1	some_connection9 = 1
	some_connection10 = 1	some_connection10 = 0
	some_connection12 = 0	some_connection12 = 0
	some_connection13 = 1	some_connection13 = 0
	some_connection14 = 1	some_connection14 = 1
	d=24	*d=24*
	B_OpLtr_out = 0	**B_OpLtr_out = 0**
	B_CdLtr_out = 0	**B_CdLtr_out = 0**
	NX_LtrSaSq_out = 3	**NX_LtrSaSq_out = 3**
	A_LtrTs_out = 0	**A_LtrTs_out = 1**
	A_LtrOpVd_out = 0	**A_LtrOpVd_out = 0**

Table 31: Generated test cases

InternalLtrTsSq9	InternalLtrTsSq10	InternalLtrTsSq11
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 1	some_connection13 = 1	some_connection13 = 0
some_connection14 = 0	some_connection14 = 0	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0

some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 1	some_connection10 = 1	some_connection10 = 1
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 1	some_connection13 = 1	some_connection13 = 1
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=24	*d=24*	*d=24*
B_OpLtr_out = 0	**B_OpLtr_out = 0**	**B_OpLtr_out = 0**
B_CdLtr_out = 0	**B_CdLtr_out = 0**	**B_CdLtr_out = 0**
NX_LtrSaSq_out = 4	**NX_LtrSaSq_out = 4**	**NX_LtrSaSq_out = 1**
A_LtrTs_out = 0	**A_LtrTs_out = 0**	**A_LtrTs_out = 0**
A_LtrOpVd_out = 0	**A_LtrOpVd_out = 0**	**A_LtrOpVd_out = 0**

Table 32: Generated test cases

InternalLtrTsSq12	InternalLtrTsSq13	InternalLtrTsSq14
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0

some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 1	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 1	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=64	*d=576*	*d=576*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 1
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 1
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=24	*d=24*	*d=64*
B_OpLtr_out = 1	**B_OpLtr_out = 1**	some_connection1 = 0
B_CdLtr_out = 0	**B_CdLtr_out = 0**	some_connection2 = 0
NX_LtrSaSq_out = 0	**NX_LtrSaSq_out = 2**	some_connection3 = 0
A_LtrTs_out = 0	**A_LtrTs_out = 0**	some_connection4 = 0
A_LtrOpVd_out = 0	**A_LtrOpVd_out = 0**	some_connection5 = 0
		some_connection6 = 0
		some_connection7 = 0
		some_connection8 = 1
		FPGA2_LT_ON = 0
		LT_RELAY_FB = 1
		some_connection9 = 1
		some_connection10 = 0
		some_connection12 = 0
		some_connection13 = 0
		some_connection14 = 1
		d=24
		B_OpLtr_out = 1
		B_CdLtr_out = 0
		NX_LtrSaSq_out = 0
		A_LtrTs_out = 0
		A_LtrOpVd_out = 0

Table 33: Generated test cases

InternalLtrTsSq15	InternalLtrTsSq16	InternalLtrTsSq17
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=576	*d=576*	*d=576*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1

some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=64	*d=64*	*d=64*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 1	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 1	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=24	*d=64*	*d=64*
B_OpLtr_out = 0	some_connection1 = 0	some_connection1 = 0
B_CdLtr_out = 0	some_connection2 = 0	some_connection2 = 0
NX_LtrSaSq_out = 3	some_connection3 = 0	some_connection3 = 0
A_LtrTs_out = 1	some_connection4 = 0	some_connection4 = 0
A_LtrOpVd_out = 0	some_connection5 = 0	some_connection5 = 0
	some_connection6 = 0	some_connection6 = 0
	some_connection7 = 0	some_connection7 = 0
	some_connection8 = 1	some_connection8 = 1
	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
	LT_RELAY_FB = 1	LT_RELAY_FB = 1
	some_connection9 = 1	some_connection9 = 1
	some_connection10 = 0	some_connection10 = 1
	some_connection12 = 0	some_connection12 = 0
	some_connection13 = 0	some_connection13 = 1
	some_connection14 = 1	some_connection14 = 1
	d=24	*d=24*
	B_OpLtr_out = 1	**B_OpLtr_out = 0**
	B_CdLtr_out = 0	**B_CdLtr_out = 0**
	NX_LtrSaSq_out = 0	**NX_LtrSaSq_out = 4**
	A_LtrTs_out = 0	**A_LtrTs_out = 0**
	A_LtrOpVd_out = 0	**A_LtrOpVd_out = 0**

Table 34: Generated test cases

InternalLtrTsSq18	InternalLtrTsSq19	InternalController1
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0

some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
GPIO_OUT = 0
LTRIP_EN_N = 1
MCU_LT_ON = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 1
some_connection14 = 0
d=60
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 1
some_connection12 = 0
some_connection13 = 1
some_connection14 = 1
d=64
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=24

some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
GPIO_OUT = 0
LTRIP_EN_N = 1
MCU_LT_ON = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=60
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=576
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=24

some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
GPIO_OUT = 0
LTRIP_EN_N = 1
MCU_LT_ON = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=2
connection3 = 0
some_connection18 = 0
some_connection20 = 0
some_connection21 = 0
GPIO_OUT = 1
FPGA_LTRCR = 0

B_OpLtr_out = 1	**B_OpLtr_out = 1**
B_CdLtr_out = 0	**B_CdLtr_out = 0**
NX_LtrSaSq_out = 0	**NX_LtrSaSq_out = 2**
A_LtrTs_out = 0	**A_LtrTs_out = 0**
A_LtrOpVd_out = 0	**A_LtrOpVd_out = 0**

Table 35: Generated test cases

InternalLtrInt1	Internaldcu2_line_trip1	Internaldcu2_line_trip2
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 31	some_connection3 = 31
some_connection4 = 0	some_connection4 = 1	some_connection4 = 1
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 1	some_connection6 = 1
some_connection7 = 0	some_connection7 = 1	some_connection7 = 1
some_connection8 = 1	some_connection8 = 0	some_connection8 = 0
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 1	FPGA2_LT_ON = 1
LT_RELAY_FB = 1	LT_RELAY_FB = 0	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 1	some_connection10 = 1
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 1	some_connection13 = 1
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=2	*d=2*	*d=2*
B_ClLtr_out = 0	**GPIO_OUT = 0**	**GPIO_OUT = 0**
	FPGA_LTRCR = 1	**FPGA_LTRCR = 1**
	fb_out = 0	**fb_out = 1**
	fpga2_on_out = 1	**fpga2_on_out = 1**
	fb_ne_out = 1	**fb_ne_out = 0**

Table 36: Generated test cases

Internaldcu2_line_trip3	Internaldcu2_line_trip4	Internaldcu2_line_trip5
some_connection1 = 0	some_connection1 = 0	some_connection1 = 1
some_connection2 = 0	some_connection2 = 0	some_connection2 = 1
some_connection3 = 31	some_connection3 = 31	some_connection3 = 31
some_connection4 = 1	some_connection4 = 1	some_connection4 = 1
some_connection5 = 0	some_connection5 = 0	some_connection5 = 1
some_connection6 = 1	some_connection6 = 1	some_connection6 = 1
some_connection7 = 1	some_connection7 = 1	some_connection7 = 1

some_connection8 = 0	some_connection8 = 0	some_connection8 = 0
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 1
LT_RELAY_FB = 1	LT_RELAY_FB = 0	LT_RELAY_FB = 0
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 1	some_connection10 = 1	some_connection10 = 1
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 1	some_connection13 = 1	some_connection13 = 1
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=2	*d=2*	*d=2*
GPIO_OUT = 0	**GPIO_OUT = 0**	**GPIO_OUT = 0**
FPGA_LTRCR = 1	**FPGA_LTRCR = 1**	**FPGA_LTRCR = 0**
fb_out = 1	**fb_out = 0**	**fb_out = 0**
fpga2_on_out = 0	**fpga2_on_out = 0**	**fpga2_on_out = 0**
fb_ne_out = 1	**fb_ne_out = 0**	**fb_ne_out = 0**

Table 37: Generated test cases

Internaldcu2_line_trip6	Internaldcu2_line_trip7	Internaldcu2_line_trip8
some_connection1 = 0	some_connection1 = 0	some_connection1 = 1
some_connection2 = 0	some_connection2 = 0	some_connection2 = 1
some_connection3 = 31	some_connection3 = 31	some_connection3 = 31
some_connection4 = 1	some_connection4 = 1	some_connection4 = 1
some_connection5 = 0	some_connection5 = 0	some_connection5 = 1
some_connection6 = 1	some_connection6 = 1	some_connection6 = 1
some_connection7 = 1	some_connection7 = 1	some_connection7 = 1
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 1	FPGA2_LT_ON = 1	FPGA2_LT_ON = 1
LT_RELAY_FB = 1	LT_RELAY_FB = 0	LT_RELAY_FB = 0
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 1	some_connection10 = 1	some_connection10 = 1
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 1	some_connection13 = 1	some_connection13 = 1
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=2	*d=2*	*d=2*
GPIO_OUT = 1	**GPIO_OUT = 1**	**GPIO_OUT = 1**
FPGA_LTRCR = 1	**FPGA_LTRCR = 1**	**FPGA_LTRCR = 0**
fb_out = 1	**fb_out = 0**	**fb_out = 0**
fpga2_on_out = 1	**fpga2_on_out = 1**	**fpga2_on_out = 0**
fb_ne_out = 1	**fb_ne_out = 0**	**fb_ne_out = 0**

Table 38: Generated test cases

Internaldcu2_line_trip9	Internaldcu2_line_trip10	Internaldcu2_line_trip11
some_connection1 = 1	some_connection1 = 0	some_connection1 = 0
some_connection2 = 1	some_connection2 = 0	some_connection2 = 0
some_connection3 = 31	some_connection3 = 0	some_connection3 = 0
some_connection4 = 1	some_connection4 = 0	some_connection4 = 0
some_connection5 = 1	some_connection5 = 0	some_connection5 = 0
some_connection6 = 1	some_connection6 = 0	some_connection6 = 0
some_connection7 = 1	some_connection7 = 0	some_connection7 = 0
some_connection8 = 0	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 1	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 0	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 1	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 1	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=6	*d=60*	*d=60*
GPIO_OUT = 0	some_connection1 = 0	some_connection1 = 0
FPGA_LTRCR = 0	some_connection2 = 0	some_connection2 = 0
fb_out = 0	some_connection3 = 0	some_connection3 = 0
fpga2_on_out = 0	some_connection4 = 0	some_connection4 = 0
fb_ne_out = 0	some_connection5 = 0	some_connection5 = 0
	some_connection6 = 0	some_connection6 = 0
	some_connection7 = 0	some_connection7 = 0
	some_connection8 = 0	some_connection8 = 0
	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
	LT_RELAY_FB = 1	LT_RELAY_FB = 1
	some_connection9 = 1	some_connection9 = 1
	some_connection10 = 1	some_connection10 = 0
	some_connection12 = 0	some_connection12 = 0
	some_connection13 = 1	some_connection13 = 0
	some_connection14 = 1	some_connection14 = 1
	d=64	*d=2*
	some_connection1 = 0	**GPIO_OUT = 0**
	some_connection2 = 0	**FPGA_LTRCR = 0**
	some_connection3 = 0	**fb_out = 0**
	some_connection4 = 0	**fpga2_on_out = 0**
	some_connection5 = 0	**fb_ne_out = 0**
	some_connection6 = 0	
	some_connection7 = 0	
	some_connection8 = 1	
	FPGA2_LT_ON = 0	
	LT_RELAY_FB = 1	
	some_connection9 = 1	

		some_connection10 = 0
		some_connection12 = 0
		some_connection13 = 0
		some_connection14 = 1
		d=2
		GPIO_OUT = 1
		FPGA_LTRCR = 0
		fb_out = 0
		fpga2_on_out = 0
		fb_ne_out = 0

Table 39: Generated test cases

Internaldcu2_line_trip12	Direct1	Direct2
some_connection1 = 1	some_connection1 = 0	some_connection1 = 0
some_connection2 = 1	some_connection2 = 0	some_connection2 = 0
some_connection3 = 31	some_connection3 = 0	some_connection3 = 0
some_connection4 = 1	some_connection4 = 0	some_connection4 = 0
some_connection5 = 1	some_connection5 = 0	some_connection5 = 0
some_connection6 = 1	some_connection6 = 0	some_connection6 = 0
some_connection7 = 1	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 1	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 0	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 1	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 1	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=6	*d=20*	*d=2*
GPIO_OUT = 1	**C_LtrTs_in = 1**	**B_RqPrSd_in = 0**
FPGA_LTRCR = 0	**B_LtrFl_in = 0**	**B_OpLtr_AppSpec_in = 0**
fb_out = 0	**S_LtrCd_in = 0**	**NX_SqSt_in = 0**
fpga2_on_out = 0	**S_LtrOp_in = 0**	**A_PctMo_in = 0**
fb_ne_out = 0	**S_DCUNtRdy_in = 0**	**B_LtrTsOpLtr_in = 0**
	L_EnLtrSv_in = 1	**B_OpLtr_LtrTs_in = 0**
		B_CdLtr_LtrTs_in = 0
		B_EnCdLnTrpSlt_in = 0
		L_CnfHpp_in = 0

Table 40: Generated test cases

Direct3	Direct4	Direct5
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=2	*d=20*	*d=20*
enable_in = 0	**DHSSMG_B_OpLtr = 1**	**DHSSMG_B_CdLtr = 0**
act_in = 0		

Table 41: Generated test cases

Direct6	Indirect1	Indirect2
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 0	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 1	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=20*	*d=20*
some_connection1 = 0	**some_connection15 = 0**	**some_connection15 = 0**

some_connection2 = 0	**some_connection16 = 0**	**some_connection16 = 0**
some_connection3 = 0	**some_connection17 = 0**	**some_connection17 = 0**
some_connection4 = 0	*d=2*	*d=2*
some_connection5 = 0	**connection3 = 0**	**connection3 = 0**
some_connection6 = 0	**some_connection18 = 0**	**some_connection18 = 0**
some_connection7 = 0	**some_connection20 = 0**	**some_connection20 = 0**
some_connection8 = 1	**some_connection21 = 0**	**some_connection21 = 0**
FPGA2_LT_ON = 0	**GPIO_OUT = 0**	**GPIO_OUT = 1**
LT_RELAY_FB = 1	**FPGA_LTRCR = 0**	**FPGA_LTRCR = 0**
some_connection9 = 1		
some_connection10 = 0		
some_connection12 = 0		
some_connection13 = 0		
some_connection14 = 1		
d=4		
DHWOMG_S_LtrCd = 0		

Table 42: Generated test cases

Indirect3	Indirect4	Indirect5
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 0

FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 1
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 0
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=576	*d=576*	*d=536*
some_connection1 = 0	some_connection1 = 0	**some_connection15 = 1**
some_connection2 = 0	some_connection2 = 0	**some_connection16 = 0**
some_connection3 = 0	some_connection3 = 0	**some_connection17 = 0**
some_connection4 = 0	some_connection4 = 0	*d=2*
some_connection5 = 0	some_connection5 = 0	**connection3 = 0**
some_connection6 = 0	some_connection6 = 0	**some_connection18 = 1**
some_connection7 = 0	some_connection7 = 0	**some_connection20 = 1**
some_connection8 = 0	some_connection8 = 1	**some_connection21 = 1**
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	**GPIO_OUT = 0**
LT_RELAY_FB = 1	LT_RELAY_FB = 1	**FPGA_LTRCR = 1**
some_connection9 = 1	some_connection9 = 1	
some_connection10 = 0	some_connection10 = 0	
some_connection12 = 0	some_connection12 = 0	
some_connection13 = 0	some_connection13 = 0	
some_connection14 = 1	some_connection14 = 1	
d=24	*d=24*	
some_connection15 = 2	**some_connection15 = 2**	
some_connection16 = 0	**some_connection16 = 0**	
some_connection17 = 0	**some_connection17 = 0**	
d=2	*d=2*	
connection3 = 0	**connection3 = 0**	
some_connection18 = 0	**some_connection18 = 0**	
some_connection20 = 0	**some_connection20 = 0**	
some_connection21 = 0	**some_connection21 = 0**	
GPIO_OUT = 0	**GPIO_OUT = 1**	
FPGA_LTRCR = 0	**FPGA_LTRCR = 0**	

Table 43: Generated test cases

Indirect6	Indirect7	Indirect8
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0

LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 0	some_connection8 = 0	some_connection8 = 0
FPGA2_LT_ON = 1	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 0
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=536	*d=536*	*d=536*
some_connection15 = 1	**some_connection15 = 1**	**some_connection15 = 1**
some_connection16 = 0	**some_connection16 = 0**	**some_connection16 = 0**
some_connection17 = 0	**some_connection17 = 0**	**some_connection17 = 0**
d=2	*d=2*	*d=2*
connection3 = 1	**connection3 = 1**	**connection3 = 0**
some_connection18 = 1	**some_connection18 = 1**	**some_connection18 = 1**
some_connection20 = 1	**some_connection20 = 0**	**some_connection20 = 0**
some_connection21 = 0	**some_connection21 = 1**	**some_connection21 = 0**
GPIO_OUT = 0	**GPIO_OUT = 0**	**GPIO_OUT = 0**
FPGA_LTRCR = 1	**FPGA_LTRCR = 1**	**FPGA_LTRCR = 1**

Table 44: Generated test cases

Indirect9	Indirect10	Indirect11
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0

some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 0
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 0	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=536	*d=536*	*d=2*
some_connection15 = 1	**some_connection15 = 1**	**connection3 = 0**
some_connection16 = 0	**some_connection16 = 0**	**some_connection18 = 0**
some_connection17 = 0	**some_connection17 = 0**	**some_connection20 = 0**
d=2	*d=2*	**some_connection21 = 0**
connection3 = 1	**connection3 = 0**	**GPIO_OUT = 0**
some_connection18 = 1	**some_connection18 = 1**	**FPGA_LTRCR = 0**
some_connection20 = 0	**some_connection20 = 0**	*d=534*
some_connection21 = 1	**some_connection21 = 0**	**some_connection15 = 1**
GPIO_OUT = 1	**GPIO_OUT = 1**	**some_connection16 = 0**
FPGA_LTRCR = 1	**FPGA_LTRCR = 1**	**some_connection17 = 0**

Table 45: Generated test cases

Indirect12	Indirect13	Indirect14
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0

some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
GPIO_OUT = 0
LTRIP_EN_N = 1
MCU_LT_ON = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=60
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=2
connection3 = 0
some_connection18 = 0
some_connection20 = 0
some_connection21 = 0
GPIO_OUT = 1
FPGA_LTRCR = 0
d=534
some_connection15 = 1
some_connection16 = 0
some_connection17 = 0

some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
GPIO_OUT = 0
LTRIP_EN_N = 1
MCU_LT_ON = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=60
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=576
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 0
FPGA2_LT_ON = 1
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=2
connection3 = 1
some_connection18 = 1
some_connection20 = 1
some_connection21 = 0

some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
GPIO_OUT = 0
LTRIP_EN_N = 1
MCU_LT_ON = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=60
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 1
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=576
some_connection1 = 0
some_connection2 = 0
some_connection3 = 0
some_connection4 = 0
some_connection5 = 0
some_connection6 = 0
some_connection7 = 0
some_connection8 = 0
FPGA2_LT_ON = 0
LT_RELAY_FB = 1
some_connection9 = 1
some_connection10 = 0
some_connection12 = 0
some_connection13 = 0
some_connection14 = 1
d=2
connection3 = 1
some_connection18 = 1
some_connection20 = 0
some_connection21 = 1

	GPIO_OUT = 0	**GPIO_OUT = 0**
	FPGA_LTRCR = 1	**FPGA_LTRCR = 1**
	d=22	*d=22*
	some_connection15 = 2	**some_connection15 = 2**
	some_connection16 = 0	**some_connection16 = 0**
	some_connection17 = 0	**some_connection17 = 0**

Table 46: Generated test cases

Indirect15	Indirect16	Indirect17
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
GPIO_OUT = 0	GPIO_OUT = 0	GPIO_OUT = 0
LTRIP_EN_N = 1	LTRIP_EN_N = 1	LTRIP_EN_N = 1
MCU_LT_ON = 1	MCU_LT_ON = 1	MCU_LT_ON = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=60	*d=60*	*d=60*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0
some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 0	some_connection8 = 1	some_connection8 = 0
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=576	*d=576*	*d=576*
some_connection1 = 0	some_connection1 = 0	some_connection1 = 0
some_connection2 = 0	some_connection2 = 0	some_connection2 = 0

some_connection3 = 0	some_connection3 = 0	some_connection3 = 0
some_connection4 = 0	some_connection4 = 0	some_connection4 = 0
some_connection5 = 0	some_connection5 = 0	some_connection5 = 0
some_connection6 = 0	some_connection6 = 0	some_connection6 = 0
some_connection7 = 0	some_connection7 = 0	some_connection7 = 0
some_connection8 = 1	some_connection8 = 1	some_connection8 = 1
FPGA2_LT_ON = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
LT_RELAY_FB = 1	LT_RELAY_FB = 1	LT_RELAY_FB = 1
some_connection9 = 1	some_connection9 = 1	some_connection9 = 1
some_connection10 = 0	some_connection10 = 0	some_connection10 = 0
some_connection12 = 0	some_connection12 = 0	some_connection12 = 0
some_connection13 = 0	some_connection13 = 0	some_connection13 = 0
some_connection14 = 1	some_connection14 = 1	some_connection14 = 1
d=2	*d=64*	*d=64*
connection3 = 1	some_connection1 = 0	some_connection1 = 0
some_connection18 = 1	some_connection2 = 0	some_connection2 = 0
some_connection20 = 0	some_connection3 = 0	some_connection3 = 0
some_connection21 = 1	some_connection4 = 0	some_connection4 = 0
GPIO_OUT = 1	some_connection5 = 0	some_connection5 = 0
FPGA_LTRCR = 1	some_connection6 = 0	some_connection6 = 0
d=22	some_connection7 = 0	some_connection7 = 0
some_connection15 = 2	some_connection8 = 0	some_connection8 = 1
some_connection16 = 0	FPGA2_LT_ON = 0	FPGA2_LT_ON = 0
some_connection17 = 0	LT_RELAY_FB = 1	LT_RELAY_FB = 1
	some_connection9 = 1	some_connection9 = 1
	some_connection10 = 0	some_connection10 = 0
	some_connection12 = 0	some_connection12 = 0
	some_connection13 = 0	some_connection13 = 0
	some_connection14 = 1	some_connection14 = 1
	d=2	*d=2*
	connection3 = 0	**connection3 = 0**
	some_connection18 = 0	**some_connection18 = 0**
	some_connection20 = 0	**some_connection20 = 0**
	some_connection21 = 0	**some_connection21 = 0**
	GPIO_OUT = 0	**GPIO_OUT = 1**
	FPGA_LTRCR = 0	**FPGA_LTRCR = 0**
	d=22	*d=22*
	some_connection15 = 3	**some_connection15 = 3**
	some_connection16 = 1	**some_connection16 = 1**
	some_connection17 = 1	**some_connection17 = 1**

Appendix C Selection

Table 47. Fault injection 1: Selection results

Modification		V		A	
AFG'\AFG		{S_LtrCd}		{}	
VS'old	**vs cov. mod.**	**Indep. VS'old**	**V'aff**	**VS'old cov. V'aff**	**VS'new**
InternalTester1	InternalLtrTsSq8	InternalTester1	Fig. 24	InternalLtrTsSq8_updated	
InternalLtrTsSq1	Indirect16	InternalLtrTsSq1		InternalLtrTsSq17	
InternalLtrTsSq2	Indirect17	InternalLtrTsSq2		Indirect16_updated	
InternalLtrTsSq3		InternalLtrTsSq3		Indirect17_updated	
InternalLtrTsSq4		InternalLtrTsSq4			
InternalLtrTsSq5		InternalLtrTsSq5			
InternalLtrTsSq6		InternalLtrTsSq6			
InternalLtrTsSq7		InternalLtrTsSq7			
InternalLtrTsSq8_updated		InternalLtrTsSq9			
InternalLtrTsSq9		InternalLtrTsSq10			
InternalLtrTsSq10		InternalLtrTsSq11			
InternalLtrTsSq11		InternalLtrTsSq12			
InternalLtrTsSq12		InternalLtrTsSq13			
InternalLtrTsSq13		InternalLtrTsSq14			
InternalLtrTsSq14		InternalLtrTsSq15			
InternalLtrTsSq15		InternalLtrTsSq16			
InternalLtrTsSq16		InternalLtrTsSq18			
InternalLtrTsSq17		InternalLtrTsSq19			
InternalLtrTsSq18		InternalController1			
InternalLtrTsSq19		InternalLtrInt1			
InternalController1		Internaldcu2_line_trip1			
InternalLtrInt1		Internaldcu2_line_trip2			
Internaldcu2_line_trip1		Internaldcu2_line_trip3			
Internaldcu2_line_trip2		Internaldcu2_line_trip4			
Internaldcu2_line_trip3		Internaldcu2_line_trip5			
Internaldcu2_line_trip4		Internaldcu2_line_trip6			
Internaldcu2_line_trip5		Internaldcu2_line_trip7			
Internaldcu2_line_trip6		Internaldcu2_line_trip8			
Internaldcu2_line_trip7		Internaldcu2_line_trip9			
Internaldcu2_line_trip8		Internaldcu2_line_trip10			
Internaldcu2_line_trip9		Internaldcu2_line_trip11			
Internaldcu2_line_trip10		Internaldcu2_line_trip12			
Internaldcu2_line_trip11		Direct1			
Internaldcu2_line_trip12		Direct2			
Direct1		Direct3			
Direct2		Direct4			
Direct3		Direct5			
Direct4		Direct6			
Direct5		Indirect1			
Direct6		Indirect2			
Indirect1		Indirect3			
Indirect2		Indirect4			
Indirect3		Indirect5			
Indirect4		Indirect6			
Indirect5		Indirect7			
Indirect6		Indirect8			
Indirect7		Indirect9			
Indirect8		Indirect10			
Indirect9		Indirect11			
Indirect10		Indirect12			
Indirect11		Indirect13			
Indirect12		Indirect14			
Indirect13		Indirect15			
Indirect14		(Table 54)			
Indirect15					
Indirect16_updated					

Table 48. Fault injection 2: Selection results

Modification	**V**			**A**	
AFG'\AFG	{state_LtrTsSq := OpenLtr1}			{}	
VS'old	**vs cov. mod.**	**Indep. VS'old**	**V'aff**	**VS'old cov. V'aff**	**VS'new**
InternalTester1	InternalLtrTsSq6	InternalTester1	Fig. 25	InternalLtrTsSq6_updated	
InternalLtrTsSq1	InternalLtrTsSq15	InternalLtrTsSq1		InternalLtrTsSq7	
InternalLtrTsSq2	Indirect3	InternalLtrTsSq2		InternalLtrTsSq8	
InternalLtrTsSq3	Indirect4	InternalLtrTsSq3		InternalLtrTsSq15_updated	
InternalLtrTsSq4	Indirect13	InternalLtrTsSq4		InternalLtrTsSq16	
InternalLtrTsSq5	Indirect14	InternalLtrTsSq5		InternalLtrTsSq17	
InternalLtrTsSq6_updated	Indirect15	InternalLtrTsSq9		Indirect3_updated	
InternalLtrTsSq7		InternalLtrTsSq10		Indirect4_updated	
InternalLtrTsSq8		InternalLtrTsSq11		Indirect13_updated	
InternalLtrTsSq9		InternalLtrTsSq12		Indirect14 _updated	
InternalLtrTsSq10		InternalLtrTsSq13		Indirect15_updated	
InternalLtrTsSq11		InternalLtrTsSq14		Indirect16	
InternalLtrTsSq12		InternalLtrTsSq18		Indirect17	
InternalLtrTsSq13		InternalLtrTsSq19			
InternalLtrTsSq14		InternalController1			
InternalLtrTsSq15_updated		InternalLtrInt1			
InternalLtrTsSq16		Internaldcu2_line_trip1			
InternalLtrTsSq17		Internaldcu2_line_trip2			
InternalLtrTsSq18		Internaldcu2_line_trip3			
InternalLtrTsSq19		Internaldcu2_line_trip4			
InternalController1		Internaldcu2_line_trip5			
InternalLtrInt1		Internaldcu2_line_trip6			
Internaldcu2_line_trip1		Internaldcu2_line_trip7			
Internaldcu2_line_trip2		Internaldcu2_line_trip8			
Internaldcu2_line_trip3		Internaldcu2_line_trip9			
Internaldcu2_line_trip4		Internaldcu2_line_trip10			
Internaldcu2_line_trip5		Internaldcu2_line_trip11			
Internaldcu2_line_trip6		Internaldcu2_line_trip12			
Internaldcu2_line_trip7		Direct1			
Internaldcu2_line_trip8		Direct2			
Internaldcu2_line_trip9		Direct3			
Internaldcu2_line_trip10		Direct4			
Internaldcu2_line_trip11		Direct5			
Internaldcu2_line_trip12		Direct6			
Direct1		Indirect1			
Direct2		Indirect2			
Direct3		Indirect5			
Direct4		Indirect6			
Direct5		Indirect7			
Direct6		Indirect8			
Indirect1		Indirect9			
Indirect2		Indirect10			
Indirect3_updated		Indirect11			
Indirect4_updated		Indirect12			
Indirect5		(Table 55)			
Indirect6					
Indirect7					
Indirect8					
Indirect9					
Indirect10					
Indirect11					
Indirect12					
Indirect13_updated					
Indirect14 _updated					
Indirect15_updated					
Indirect16					
Indirect17					

Table 49. Fault injection 3: Selection results

Modification	V	A
AFG\AFG'	{connection3 := DHWOMG_S_LtrCd(v_1)}	{$v_1 \rightarrow_{d-in}$DHWOMG_S_LtrCd := connection3}
AFG'\AFG	{DHWOMG_S_LtrCd := DHWOMG_S_LtrCd}	{}

VS'old	**vs cov. mod.**	**Indep. VS'old**	**V'aff**	**VS'old cov. V'aff**	**VS'new**
InternalTester1	Direct6	InternalTester1	Fig. 26	InternalLtrTsSq3	
InternalLtrTsSq1	Indirect11	InternalLtrTsSq1		InternalLtrTsSq4	
InternalLtrTsSq2	Indirect12	InternalLtrTsSq2		InternalLtrTsSq5	
InternalLtrTsSq3	Indirect13	InternalController1		InternalLtrTsSq6	
InternalLtrTsSq4	Indirect14	InternalLtrInt1		InternalLtrTsSq7	
InternalLtrTsSq5	Indirect15	Internaldcu2_line_trip1		InternalLtrTsSq8	
InternalLtrTsSq6	Indirect16	Internaldcu2_line_trip2		InternalLtrTsSq9	
InternalLtrTsSq7	Indirect17	Internaldcu2_line_trip3		InternalLtrTsSq10	
InternalLtrTsSq8		Internaldcu2_line_trip4		InternalLtrTsSq11	
InternalLtrTsSq9		Internaldcu2_line_trip5		InternalLtrTsSq12	
InternalLtrTsSq10		Internaldcu2_line_trip6		InternalLtrTsSq13	
InternalLtrTsSq11		Internaldcu2_line_trip7		InternalLtrTsSq14	
InternalLtrTsSq12		Internaldcu2_line_trip8		InternalLtrTsSq15	
InternalLtrTsSq13		Internaldcu2_line_trip9		InternalLtrTsSq16	
InternalLtrTsSq14		Internaldcu2_line_trip12		InternalLtrTsSq17	
InternalLtrTsSq15		Direct1		InternalLtrTsSq18	
InternalLtrTsSq16		Direct2		InternalLtrTsSq19	
InternalLtrTsSq17		Direct3		Indirect3	
InternalLtrTsSq18		Direct4		Indirect4	
InternalLtrTsSq19		Direct5		Indirect5	
InternalController1		Indirect1		Indirect6	
InternalLtrInt1		Indirect2		Indirect7	
Internaldcu2_line_trip1		(Table 56)		Indirect8	
Internaldcu2_line_trip2				Indirect9	
Internaldcu2_line_trip3				Indirect10	
Internaldcu2_line_trip4					
Internaldcu2_line_trip5					
Internaldcu2_line_trip6					
Internaldcu2_line_trip7					
Internaldcu2_line_trip8					
Internaldcu2_line_trip9					
Internaldcu2_line_trip10					
Internaldcu2_line_trip11					
Internaldcu2_line_trip12					
Direct1					
Direct2					
Direct3					
Direct4					
Direct5					
Indirect1					
Indirect2					
Indirect3					
Indirect4					
Indirect5					
Indirect6					
Indirect7					
Indirect8					
Indirect9					
Indirect10					

Table 50. Fault injection 4: Selection results

Modification	V	A
AFG\AFG'	{enable_in := not DHSSMG_B_LtrHwOpFl(v_1)}	{$v_1 \rightarrow_{d-in}$enable := enable_in}
AFG'\AFG	{enable := enable}	{}

VS'old	vs cov. mod.	Indep. VS'old	V'aff	VS'old cov. V'aff	VS'new
InternalTester1	Internaldcu2_line_trip1	InternalLtrInt1	Fig. 27	Internaldcu2_line_trip1	
InternalLtrTsSq1	Internaldcu2_line_trip2	Direct2		Internaldcu2_line_trip2	
InternalLtrTsSq2	Internaldcu2_line_trip3	(Table 57)		Internaldcu2_line_trip3	
InternalLtrTsSq3	Internaldcu2_line_trip4			Internaldcu2_line_trip4	
InternalLtrTsSq4	Internaldcu2_line_trip5			Internaldcu2_line_trip5	
InternalLtrTsSq5	Internaldcu2_line_trip6			Internaldcu2_line_trip6	
InternalLtrTsSq6	Internaldcu2_line_trip7			Internaldcu2_line_trip7	
InternalLtrTsSq7	Internaldcu2_line_trip8			Internaldcu2_line_trip8	
InternalLtrTsSq8	Direct3_updated			Internaldcu2_line_trip9	
InternalLtrTsSq9				Internaldcu2_line_trip10	
InternalLtrTsSq10				Internaldcu2_line_trip11	
InternalLtrTsSq11				Internaldcu2_line_trip12	
InternalLtrTsSq12				Direct3_updated	
InternalLtrTsSq13				Indirect1	
InternalLtrTsSq14				Indirect2	
InternalLtrTsSq15				Indirect3	
InternalLtrTsSq16				Indirect4	
InternalLtrTsSq17				Indirect5	
InternalLtrTsSq18				Indirect6	
InternalLtrTsSq19				Indirect7	
InternalController1				Indirect8	
InternalLtrInt1				Indirect9	
Internaldcu2_line_trip1				Indirect10	
Internaldcu2_line_trip2				Indirect11	
Internaldcu2_line_trip3				Indirect12	
Internaldcu2_line_trip4				Indirect13	
Internaldcu2_line_trip5				Indirect14	
Internaldcu2_line_trip6				Indirect15	
Internaldcu2_line_trip7				Indirect16	
Internaldcu2_line_trip8				Indirect17	
Internaldcu2_line_trip9					
Internaldcu2_line_trip10					
Internaldcu2_line_trip11					
Internaldcu2_line_trip12					
Direct1					
Direct2					
Direct3_updated					
Direct4					
Direct5					
Direct6					
Indirect1					
Indirect2					
Indirect3					
Indirect4					
Indirect5					
Indirect6					
Indirect7					
Indirect8					
Indirect9					
Indirect10					
Indirect11					
Indirect12					
Indirect13					
Indirect14					
Indirect15					
Indirect16					
Indirect17					

Table 51. Fault injection 5: Selection results

Modification		**V**		**A**
	AFG'\AFG	{DHSSMG_B_CdLtr := connection2(v_1)}		{connection2 := DHSSMG_B_CdLtr$\rightarrow_{d-in}v_1$
VS'old	**vs cov. mod.**	**Indep. VS'old**	**V'aff**	**VS'old cov. V'aff VS'new**
InternalTester1	Direct5	InternalTester1	Fig. 28	InternalLtrTsSq3
InternalLtrTsSq1	Indirect5	InternalLtrTsSq1		InternalLtrTsSq4
InternalLtrTsSq2	Indirect6	InternalLtrTsSq2		InternalLtrTsSq5
InternalLtrTsSq3	Indirect7	InternalController1		InternalLtrTsSq6
InternalLtrTsSq4	Indirect8	InternalLtrInt1		InternalLtrTsSq7
InternalLtrTsSq5	Indirect9	Internaldcu2_line_trip1		InternalLtrTsSq8
InternalLtrTsSq6	Indirect10	Internaldcu2_line_trip2		InternalLtrTsSq9
InternalLtrTsSq7		Internaldcu2_line_trip3		InternalLtrTsSq10
InternalLtrTsSq8		Internaldcu2_line_trip4		InternalLtrTsSq11
InternalLtrTsSq9		Internaldcu2_line_trip5		InternalLtrTsSq12
InternalLtrTsSq10		Internaldcu2_line_trip6		InternalLtrTsSq13
InternalLtrTsSq11		Internaldcu2_line_trip7		InternalLtrTsSq14
InternalLtrTsSq12		Internaldcu2_line_trip8		InternalLtrTsSq15
InternalLtrTsSq13		Internaldcu2_line_trip9		InternalLtrTsSq16
InternalLtrTsSq14		Internaldcu2_line_trip12		InternalLtrTsSq17
InternalLtrTsSq15		Direct1		InternalLtrTsSq18
InternalLtrTsSq16		Direct2		InternalLtrTsSq19
InternalLtrTsSq17		Direct3		Direct5_updated
InternalLtrTsSq18		Direct4		Indirect5_updated
InternalLtrTsSq19		(Table 58)		Indirect6_updated
InternalController1				Indirect7_updated
InternalLtrInt1				Indirect8_updated
Internaldcu2_line_trip1				Indirect9_updated
Internaldcu2_line_trip2				Indirect10_updated
Internaldcu2_line_trip3				
Internaldcu2_line_trip4				
Internaldcu2_line_trip5				
Internaldcu2_line_trip6				
Internaldcu2_line_trip7				
Internaldcu2_line_trip8				
Internaldcu2_line_trip9				
Internaldcu2_line_trip10				
Internaldcu2_line_trip11				
Internaldcu2_line_trip12				
Direct1				
Direct2				
Direct3				
Direct4				
Direct5_updated				
Direct6				
Indirect1				
Indirect2				
Indirect3				
Indirect4				
Indirect5_updated				
Indirect6_updated				
Indirect7_updated				
Indirect8_updated				
Indirect9_updated				
Indirect10_updated				
Indirect11				
Indirect12				
Indirect13				
Indirect14				
Indirect15				
Indirect16				
Indirect17				

Table 52. Fault injection 6: Selection results

Modification	**V**			**A**		
	N/A			N/A		
VS'old	**vs cov. mod.**	**Indep. VS'old**	**V'aff**	**VS'old cov.**	**V'aff**	**VS'new**
InternalTester1	N/A	N/A	N/A	N/A		
InternalLtrTsSq1						
InternalLtrTsSq2						
InternalLtrTsSq3						
InternalLtrTsSq4						
InternalLtrTsSq5						
InternalLtrTsSq6						
InternalLtrTsSq7						
InternalLtrTsSq8						
InternalLtrTsSq9						
InternalLtrTsSq10						
InternalLtrTsSq11						
InternalLtrTsSq12						
InternalLtrTsSq13						
InternalLtrTsSq14						
InternalLtrTsSq15						
InternalLtrTsSq16						
InternalLtrTsSq17						
InternalLtrTsSq18						
InternalLtrTsSq19						
InternalController1						
InternalLtrInt1						
Internaldcu2_line_trip1						
Internaldcu2_line_trip2						
Internaldcu2_line_trip3						
Internaldcu2_line_trip4						
Internaldcu2_line_trip5						
Internaldcu2_line_trip6						
Internaldcu2_line_trip7						
Internaldcu2_line_trip8						
Internaldcu2_line_trip9						
Internaldcu2_line_trip10						
Internaldcu2_line_trip11						
Internaldcu2_line_trip12						
Direct1						
Direct2						
Direct3						
Direct4						
Direct5						
Direct6						
Indirect1						
Indirect2						
Indirect3						
Indirect4						
Indirect5						
Indirect6						
Indirect7						
Indirect8						
Indirect9						
Indirect10						
Indirect11						
Indirect12						
Indirect13						
Indirect14						
Indirect15						
Indirect16						
Indirect17						

Table 53. Fault injection 7: Selection results

Modification	**V**			**A**	
AFG'\AFG	{MCU_LT_ON :=s_w5(v_1),MCU_LT_ON := true (v_2), LtrTsSq()(v_3), T-C := MCU_LT_ON(v_4),T-C := MCU_LT_ON(v_5), temp := temp or MCU_LT_ON(v_6)}			$\{v_1 \rightarrow_d v_2\ , v_2 \rightarrow_c v_3 , v_2 \rightarrow_d v_4 , v_4 \rightarrow_{d-in} v_5 , v_5 \rightarrow_d v_6\}$	
VS'old	**vs cov. mod.**	**Indep. VS'old**	**V'aff**	**VS'old cov. V'aff**	**VS'new**
InternalTester1_updated	InternalTester1	InternalLtrTsSq1	Fig. 29	InternalTester1_updated	Direct7
InternalLtrTsSq1	Indirect11	InternalLtrTsSq2		InternalLtrTsSq3	
InternalLtrTsSq2	Indirect12	InternalController1		InternalLtrTsSq4	
InternalLtrTsSq3	Indirect13	InternalLtrInt1		InternalLtrTsSq5	
InternalLtrTsSq4	Indirect14	Internaldcu2_line_trip5		InternalLtrTsSq6	
InternalLtrTsSq5	Indirect15	Internaldcu2_line_trip8		InternalLtrTsSq7	
InternalLtrTsSq6	Indirect16	Direct2		InternalLtrTsSq8	
InternalLtrTsSq7	Indirect17	Direct3		InternalLtrTsSq9	
InternalLtrTsSq8		(Table 59)		InternalLtrTsSq10	
InternalLtrTsSq9				InternalLtrTsSq11	
InternalLtrTsSq10				InternalLtrTsSq12	
InternalLtrTsSq11				InternalLtrTsSq13	
InternalLtrTsSq12				InternalLtrTsSq14	
InternalLtrTsSq13				InternalLtrTsSq15	
InternalLtrTsSq14				InternalLtrTsSq16	
InternalLtrTsSq15				InternalLtrTsSq17	
InternalLtrTsSq16				InternalLtrTsSq18	
InternalLtrTsSq17				InternalLtrTsSq19	
InternalLtrTsSq18				Internaldcu2_line_trip1	
InternalLtrTsSq19				Internaldcu2_line_trip2	
InternalController1				Internaldcu2_line_trip3	
InternalLtrInt1				Internaldcu2_line_trip4	
Internaldcu2_line_trip1				Internaldcu2_line_trip6	
Internaldcu2_line_trip2				Internaldcu2_line_trip7	
Internaldcu2_line_trip3				Internaldcu2_line_trip9	
Internaldcu2_line_trip4				Internaldcu2_line_trip10	
Internaldcu2_line_trip5				Internaldcu2_line_trip11	
Internaldcu2_line_trip6				Internaldcu2_line_trip12	
Internaldcu2_line_trip7				Direct1	
Internaldcu2_line_trip8				Direct4	
Internaldcu2_line_trip9				Direct5	
Internaldcu2_line_trip10				Direct6	
Internaldcu2_line_trip11				Indirect1	
Internaldcu2_line_trip12				Indirect2	
Direct1				Indirect3	
Direct2				Indirect4	
Direct3				Indirect5	
Direct4				Indirect6	
Direct5				Indirect7	
Direct6				Indirect8	
Indirect1				Indirect9	
Indirect2				Indirect10	
Indirect3				Indirect11_updated	
Indirect4				Indirect12_updated	
Indirect5				Indirect13_updated	
Indirect6				Indirect14_updated	
Indirect7				Indirect15_updated	
Indirect8				Indirect16_updated	
Indirect9				Indirect17_updated	
Indirect10					
Indirect11_updated					
Indirect12_updated					
Indirect13_updated					
Indirect14_updated					
Indirect15_updated					
Indirect16_updated					
Indirect17_updated					

Appendix D Independent observers

Table 54. Fault injection 1: Independent observers (gray)

	InternalTester1	InternalLtrTsSq1	InternalLtrTsSq2	InternalLtrTsSq3	InternalLtrTsSq4	InternalLtrTsSq5	InternalLtrTsSq6	InternalLtrTsSq7	**InternalLtrTsSq8**	InternalLtrTsSq9	InternalLtrTsSq10	InternalLtrTsSq11	InternalLtrTsSq12	InternalLtrTsSq13	InternalLtrTsSq14	InternalLtrTsSq15	InternalLtrTsSq16	InternalLtrTsSq17	InternalLtrTsSq18	InternalLtrTsSq19	InternalController1	InternalLtrInt1	Internaldcu2_line_trip1	Internaldcu2_line_trip2	Internaldcu2_line_trip3	Internaldcu2_line_trip4	Internaldcu2_line_trip5	Internaldcu2_line_trip6	Internaldcu2_line_trip7	Internaldcu2_line_trip8	Internaldcu2_line_trip9	Internaldcu2_line_trip10	Internaldcu2_line_trip11	Internaldcu2_line_trip12	Direct1	Direct2	Direct3	Direct4	Direct5	Direct6	Indirect1	Indirect2	Indirect3	Indirect4	Indirect5	Indirect6	Indirect7	Indirect8	Indirect9	Indirect10	Indirect11	Indirect12	Indirect13	Indirect14	Indirect15	**Indirect16**	**Indirect17**
InternalTester1	x		x																		x	x								x				x	x	x	x																				
InternalLtrTsSq1		x																			x	x								x					x	x	x																				
InternalLtrTsSq2			x																		x	x								x					x	x	x																				
InternalLtrTsSq3	x		x	x								x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq4	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x															
InternalLtrTsSq5	x		x		x	x						x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq6	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq7	x		x		x		x	x				x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq8	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq9	x	x								x	x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq10	x	x									x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq11	x		x									x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq12	x		x	x								x	x								x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq13	x		x		x							x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq14	x		x		x	x						x		x	x						x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq15	x		x		x		x					x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq16	x		x		x		x	x				x		x		x	x			x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		
InternalLtrTsSq17	x		x		x		x		x			x		x		x		x		x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		x
InternalLtrTsSq18	x	x								x	x								x		x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq19	x		x		x							x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalController1																					x	x								x						x	x																				
InternalLtrInt1																						x														x																					
Internaldcu2_line_trip1																						x	x													x	x																				
Internaldcu2_line_trip2																						x		x												x	x																				
Internaldcu2_line_trip3																						x			x											x	x																				
Internaldcu2_line_trip4																						x				x										x	x																				
Internaldcu2_line_trip5																						x					x									x	x																				
Internaldcu2_line_trip6																						x						x								x	x																				
Internaldcu2_line_trip7																						x							x							x	x																				
Internaldcu2_line_trip8																						x								x						x	x																				
Internaldcu2_line_trip9			x																		x	x					x				x				x	x	x																				
Internaldcu2_line_trip10	x		x	x								x									x	x					x			x	x	x		x	x	x	x	x	x	x	x																
Internaldcu2_line_trip11	x	x								x	x										x	x					x			x	x		x	x	x	x	x	x	x	x																	
Internaldcu2_line_trip12			x																		x	x								x				x	x	x	x																				
Direct1																					x	x								x					x	x	x																				
Direct2																																				x																					
Direct3																						x														x	x																				
Direct4	x		x																		x	x								x				x	x	x	x	x																			
Direct5	x		x																		x	x								x				x	x	x	x		x																		
Direct6	x	x																			x	x								x				x	x	x	x	x	x	x																	
Indirect1	x		x																		x	x					x				x				x	x	x	x	x		x																
Indirect2	x		x																		x	x								x				x	x	x	x	x	x			x															
Indirect3	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x		x					x			x			x			
Indirect4	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect5	x		x		x							x									x	x	x				x				x				x	x	x	x	x	x	x				x						x						
Indirect6	x		x		x							x									x	x		x			x				x				x	x	x	x	x	x	x					x					x						
Indirect7	x		x		x							x									x	x			x		x				x				x	x	x	x	x	x	x						x				x						
Indirect8	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x							x			x						
Indirect9	x		x		x							x									x	x						x		x				x	x	x	x	x	x	x		x							x			x					
Indirect10	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x								x		x					
Indirect11	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x										x						
Indirect12	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x										x					
Indirect13	x		x		x		x					x		x						x	x	x		x		x	x				x				x	x	x	x	x	x	x							x			x		x				
Indirect14	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x							x			x			x			
Indirect15	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect16	x		x		x		x		x			x		x		x				x	x	x			x	x	x				x				x	x	x	x	x	x	x		x	x	x				x			x				x	x	
Indirect17	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		x

Table 55. Fault injection 2: Independent observers (gray)

	InternalTester1	InternalLtrTsSq1	InternalLtrTsSq2	InternalLtrTsSq3	InternalLtrTsSq4	InternalLtrTsSq5	**InternalLtrTsSq6**	InternalLtrTsSq7	InternalLtrTsSq8	InternalLtrTsSq9	InternalLtrTsSq10	InternalLtrTsSq11	InternalLtrTsSq12	InternalLtrTsSq13	InternalLtrTsSq14	**InternalLtrTsSq15**	InternalLtrTsSq16	InternalLtrTsSq17	InternalLtrTsSq18	InternalLtrTsSq19	InternalController1	InternalLtrInt1	Internaldcu2_line_trip1	Internaldcu2_line_trip2	Internaldcu2_line_trip3	Internaldcu2_line_trip4	Internaldcu2_line_trip5	Internaldcu2_line_trip6	Internaldcu2_line_trip7	Internaldcu2_line_trip8	Internaldcu2_line_trip9	Internaldcu2_line_trip10	Internaldcu2_line_trip11	Internaldcu2_line_trip12	Direct1	Direct2	Direct3	Direct4	Direct5	Direct6	Indirect1	Indirect2	**Indirect3**	**Indirect4**	Indirect5	Indirect6	Indirect7	Indirect8	Indirect9	Indirect10	Indirect11	Indirect12	**Indirect13**	**Indirect14**	**Indirect15**	Indirect16	Indirect17
InternalTester1	x		x																		x	x								x				x	x	x	x																				
InternalLtrTsSq1		x																			x	x								x					x	x	x																				
InternalLtrTsSq2			x																		x	x								x					x	x	x																				
InternalLtrTsSq3	x		x	x								x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq4	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x															
InternalLtrTsSq5	x		x		x	x						x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq6	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq7	x		x		x		x	x				x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq8	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq9	x	x								x	x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq10	x	x									x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq11	x		x									x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq12	x		x	x								x	x								x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq13	x		x		x							x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq14	x		x		x	x						x		x	x						x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq15	x		x		x		x					x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq16	x		x		x		x	x				x		x		x	x			x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		
InternalLtrTsSq17	x		x		x		x		x			x		x		x		x		x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		x
InternalLtrTsSq18	x	x								x	x								x		x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq19	x		x		x							x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalController1																					x	x								x						x	x																			
InternalLtrInt1																						x														x																				
Internaldcu2_line_trip1																						x	x													x	x																			
Internaldcu2_line_trip2																						x		x												x	x																			
Internaldcu2_line_trip3																						x			x											x	x																			
Internaldcu2_line_trip4																						x				x										x	x																			
Internaldcu2_line_trip5																						x					x									x	x																			
Internaldcu2_line_trip6																						x						x								x	x																			
Internaldcu2_line_trip7																						x							x							x	x																			
Internaldcu2_line_trip8																						x								x						x	x																			
Internaldcu2_line_trip9			x																		x	x					x				x				x	x	x																			
Internaldcu2_line_trip10	x		x	x								x									x	x					x			x	x	x		x	x	x	x	x	x	x	x																
Internaldcu2_line_trip11	x	x								x	x										x	x					x			x	x		x	x	x	x	x	x	x	x																	
Internaldcu2_line_trip12			x																		x	x								x				x	x	x	x																			
Direct1																					x	x								x					x	x	x																			
Direct2																																				x																				
Direct3																						x														x	x																			
Direct4	x		x																		x	x								x				x	x	x	x	x																		
Direct5	x		x																		x	x								x				x	x	x	x		x																	
Direct6	x	x																			x	x								x				x	x	x	x	x	x	x																	
Indirect1	x		x																		x	x					x				x				x	x	x	x	x		x																
Indirect2	x		x																		x	x								x				x	x	x	x	x	x			x															
Indirect3	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x		x					x			x			x			
Indirect4	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect5	x		x		x							x									x	x	x				x				x				x	x	x	x	x	x	x				x						x						
Indirect6	x		x		x							x									x	x		x			x				x				x	x	x	x	x	x	x					x					x						
Indirect7	x		x		x							x									x	x			x		x				x				x	x	x	x	x	x	x						x				x						
Indirect8	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x							x			x						
Indirect9	x		x		x							x									x	x							x		x				x	x	x	x	x	x	x		x							x			x					
Indirect10	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x								x		x					
Indirect11	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x										x						
Indirect12	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x										x					
Indirect13	x		x		x		x					x		x						x	x	x		x		x	x				x				x	x	x	x	x	x	x							x			x		x				
Indirect14	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x							x			x			x			
Indirect15	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect16	x		x		x		x		x			x		x		x				x	x	x			x	x	x				x				x	x	x	x	x	x		x	x	x				x			x				x	x	
Indirect17	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		x

Table 56. Fault injection 3: Independent observers (gray)

	InternalTester1	InternalLtrTsSq1	InternalLtrTsSq2	InternalLtrTsSq3	InternalLtrTsSq4	InternalLtrTsSq5	InternalLtrTsSq6	InternalLtrTsSq7	InternalLtrTsSq8	InternalLtrTsSq9	InternalLtrTsSq10	InternalLtrTsSq11	InternalLtrTsSq12	InternalLtrTsSq13	InternalLtrTsSq14	InternalLtrTsSq15	InternalLtrTsSq16	InternalLtrTsSq17	InternalLtrTsSq18	InternalLtrTsSq19	InternalController1	InternalLtrInt1	Internaldcu2_line_trip1	Internaldcu2_line_trip2	Internaldcu2_line_trip3	Internaldcu2_line_trip4	Internaldcu2_line_trip5	Internaldcu2_line_trip6	Internaldcu2_line_trip7	Internaldcu2_line_trip8	Internaldcu2_line_trip9	Internaldcu2_line_trip10	Internaldcu2_line_trip11	Internaldcu2_line_trip12	Direct1	Direct2	Direct3	Direct4	Direct5	**Direct6**	Indirect1	Indirect2	Indirect3	Indirect4	Indirect5	Indirect6	Indirect7	Indirect8	Indirect9	Indirect10	**Indirect11**	**Indirect12**	**Indirect13**	**Indirect14**	**Indirect15**	**Indirect16**	**Indirect17**
InternalTester1	x		x																			x	x								x				x	x	x	x																				
InternalLtrTsSq1		x																			x	x								x					x	x	x																				
InternalLtrTsSq2			x																		x	x								x					x	x	x																				
InternalLtrTsSq3	x		x	x								x									x	x								x				x	x	x	x	x	x	x		x																
InternalLtrTsSq4	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x																
InternalLtrTsSq5	x		x		x	x						x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq6	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq7	x		x		x		x	x				x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq8	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq9	x	x								x	x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq10	x	x									x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq11	x		x									x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq12	x		x	x								x	x								x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq13	x		x		x							x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq14	x		x		x	x						x		x	x						x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq15	x		x		x		x					x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq16	x		x		x		x	x				x		x		x	x			x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		
InternalLtrTsSq17	x		x		x		x		x			x		x		x		x		x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		x
InternalLtrTsSq18	x	x								x	x								x		x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq19	x		x		x							x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalController1																					x	x								x						x	x																			
InternalLtrInt1																						x														x																				
Internaldcu2_line_trip1																						x	x													x	x																			
Internaldcu2_line_trip2																						x		x												x	x																			
Internaldcu2_line_trip3																						x			x											x	x																			
Internaldcu2_line_trip4																						x				x										x	x																			
Internaldcu2_line_trip5																						x					x									x	x																			
Internaldcu2_line_trip6																						x						x								x	x																			
Internaldcu2_line_trip7																						x							x							x	x																			
Internaldcu2_line_trip8																						x								x						x	x																			
Internaldcu2_line_trip9			x																		x	x					x				x				x	x	x																			
Internaldcu2_line_trip10	x		x	x								x									x	x					x			x	x	x		x	x	x	x	x	x	x	x																
Internaldcu2_line_trip11	x	x								x	x										x	x					x			x	x		x	x	x	x	x	x	x	x																	
Internaldcu2_line_trip12			x																		x	x								x				x	x	x	x																				
Direct1																					x	x								x					x	x	x																				
Direct2																																				x																					
Direct3																						x														x	x																			
Direct4	x		x																		x	x								x					x	x	x	x																			
Direct5	x		x																		x	x								x					x	x	x		x																		
Direct6	x	x																			x	x								x					x	x	x	x	x	x																	
Indirect1	x		x																		x	x					x				x				x	x	x	x	x		x																
Indirect2	x		x																		x	x								x				x	x	x	x	x	x			x															
Indirect3	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x		x					x			x			x			
Indirect4	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect5	x		x		x							x									x	x	x				x				x				x	x	x	x	x	x	x				x						x						
Indirect6	x		x		x							x									x	x		x			x				x				x	x	x	x	x	x	x					x					x						
Indirect7	x		x		x							x									x	x			x		x				x				x	x	x	x	x	x	x						x				x						
Indirect8	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x							x			x						
Indirect9	x		x		x							x									x	x						x		x				x	x	x	x	x	x	x		x							x			x					
Indirect10	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x								x		x					
Indirect11	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x										x						
Indirect12	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x										x					
Indirect13	x		x		x		x					x		x						x	x	x		x		x	x				x				x	x	x	x	x	x	x							x			x		x				
Indirect14	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x							x			x			x			
Indirect15	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x			x		
Indirect16	x		x		x		x		x			x		x		x				x	x	x			x	x	x				x				x	x	x	x	x	x		x	x	x				x			x				x	x	
Indirect17	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		x

Table 57. Fault injection 4: Independent observers (gray)

	InternalTester1	InternalLtrTsSq1	InternalLtrTsSq2	InternalLtrTsSq3	InternalLtrTsSq4	InternalLtrTsSq5	InternalLtrTsSq6	InternalLtrTsSq7	InternalLtrTsSq8	InternalLtrTsSq9	InternalLtrTsSq10	InternalLtrTsSq11	InternalLtrTsSq12	InternalLtrTsSq13	InternalLtrTsSq14	InternalLtrTsSq15	InternalLtrTsSq16	InternalLtrTsSq17	InternalLtrTsSq18	InternalLtrTsSq19	InternalController1	InternalLtrInt1	**Internaldcu2_line_trip1**	**Internaldcu2_line_trip2**	**Internaldcu2_line_trip3**	**Internaldcu2_line_trip4**	**Internaldcu2_line_trip5**	**Internaldcu2_line_trip6**	**Internaldcu2_line_trip7**	**Internaldcu2_line_trip8**	Internaldcu2_line_trip9	Internaldcu2_line_trip10	Internaldcu2_line_trip11	Internaldcu2_line_trip12	Direct1	Direct2	**Direct3**	Direct4	Direct5	Direct6	Indirect1	Indirect2	Indirect3	Indirect4	Indirect5	Indirect6	Indirect7	Indirect8	Indirect9	Indirect10	Indirect11	Indirect12	Indirect13	Indirect14	Indirect15	Indirect16	Indirect17
InternalTester1	x		x																		x	x								x				x	x	x	x																				
InternalLtrTsSq1		x																			x	x								x					x	x	x																				
InternalLtrTsSq2			x																		x	x								x					x	x	x																				
InternalLtrTsSq3	x		x	x								x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq4	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x															
InternalLtrTsSq5	x		x		x	x						x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq6	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq7	x		x		x		x	x				x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq8	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq9	x	x								x	x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq10	x	x									x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq11	x		x									x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq12	x		x	x								x	x								x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq13	x		x		x							x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq14	x		x		x	x						x		x	x						x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq15	x		x		x		x					x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq16	x		x		x		x	x				x		x		x	x			x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		
InternalLtrTsSq17	x		x		x		x		x			x		x		x		x		x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		x
InternalLtrTsSq18	x	x								x	x								x		x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq19	x		x		x							x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalController1																					x	x								x						x	x																				
InternalLtrInt1																						x														x																					
Internaldcu2_line_trip1																						x	x													x	x																				
Internaldcu2_line_trip2																						x		x												x	x																				
Internaldcu2_line_trip3																						x			x											x	x																				
Internaldcu2_line_trip4																						x				x										x	x																				
Internaldcu2_line_trip5																						x					x									x	x																				
Internaldcu2_line_trip6																						x						x								x	x																				
Internaldcu2_line_trip7																						x							x							x	x																				
Internaldcu2_line_trip8																						x								x						x	x																				
Internaldcu2_line_trip9			x																		x	x					x				x				x	x	x																				
Internaldcu2_line_trip10	x		x	x								x									x	x					x			x	x	x		x	x	x	x	x	x	x	x																
Internaldcu2_line_trip11	x	x								x	x										x	x					x			x	x		x	x	x	x	x	x	x	x																	
Internaldcu2_line_trip12			x																		x	x								x				x	x	x	x																				
Direct1																					x	x								x					x	x	x																				
Direct2																																				x																					
Direct3																						x														x	x																				
Direct4	x		x																		x	x								x				x	x	x	x	x																			
Direct5	x		x																		x	x								x				x	x	x	x		x																		
Direct6	x	x																			x	x								x				x	x	x	x	x	x	x																	
Indirect1	x		x																		x	x					x				x				x	x	x	x	x		x																
Indirect2	x		x																		x	x								x				x	x	x	x	x	x			x															
Indirect3	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x		x					x			x			x			
Indirect4	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect5	x		x		x							x									x	x	x				x				x				x	x	x	x	x	x	x				x						x						
Indirect6	x		x		x							x									x	x		x			x				x				x	x	x	x	x	x	x					x					x						
Indirect7	x		x		x							x									x	x			x		x				x				x	x	x	x	x	x	x						x				x						
Indirect8	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x							x			x						
Indirect9	x		x		x							x									x	x						x		x				x	x	x	x	x	x	x		x							x			x					
Indirect10	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x								x		x					
Indirect11	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x										x						
Indirect12	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x										x					
Indirect13	x		x		x		x					x		x						x	x	x		x		x	x				x				x	x	x	x	x	x	x							x			x		x				
Indirect14	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x							x			x			x			
Indirect15	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect16	x		x		x		x		x			x		x		x				x	x	x			x	x	x				x				x	x	x	x	x	x		x	x	x				x			x				x	x	
Indirect17	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		x

Table 58. Fault injection 5: Independent observers (gray)

	InternalTester1	InternalLtrTsSq1	InternalLtrTsSq2	InternalLtrTsSq3	InternalLtrTsSq4	InternalLtrTsSq5	InternalLtrTsSq6	InternalLtrTsSq7	InternalLtrTsSq8	InternalLtrTsSq9	InternalLtrTsSq10	InternalLtrTsSq11	InternalLtrTsSq12	InternalLtrTsSq13	InternalLtrTsSq14	InternalLtrTsSq15	InternalLtrTsSq16	InternalLtrTsSq17	InternalLtrTsSq18	InternalLtrTsSq19	InternalController1	InternalLtrInt1	Internaldcu2_line_trip1	Internaldcu2_line_trip2	Internaldcu2_line_trip3	Internaldcu2_line_trip4	Internaldcu2_line_trip5	Internaldcu2_line_trip6	Internaldcu2_line_trip7	Internaldcu2_line_trip8	Internaldcu2_line_trip9	Internaldcu2_line_trip10	Internaldcu2_line_trip11	Internaldcu2_line_trip12	Direct1	Direct2	Direct3	Direct4	**Direct5**	Direct6	Indirect1	Indirect2	Indirect3	Indirect4	**Indirect5**	**Indirect6**	**Indirect7**	**Indirect8**	**Indirect9**	**Indirect10**	Indirect11	Indirect12	Indirect13	Indirect14	Indirect15	Indirect16	Indirect17
InternalTester1	x		x																		x	x								x				x	x	x	x																				
InternalLtrTsSq1		x																			x	x								x					x	x	x																				
InternalLtrTsSq2			x																		x	x								x					x	x	x																				
InternalLtrTsSq3	x		x	x								x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq4	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x															
InternalLtrTsSq5	x		x		x	x						x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq6	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq7	x		x		x		x	x				x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq8	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq9	x	x								x	x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq10	x	x									x										x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq11	x		x									x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq12	x		x	x								x	x								x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq13	x		x		x							x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq14	x		x		x	x						x		x	x						x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq15	x		x		x		x					x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq16	x		x		x		x	x				x		x		x	x			x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		
InternalLtrTsSq17	x		x		x		x		x			x		x		x		x		x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		x
InternalLtrTsSq18	x	x								x	x								x		x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq19	x		x		x							x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalController1																					x	x								x						x	x																			
InternalLtrInt1																						x														x																				
Internaldcu2_line_trip1																						x	x													x	x																			
Internaldcu2_line_trip2																						x		x												x	x																			
Internaldcu2_line_trip3																						x			x											x	x																			
Internaldcu2_line_trip4																						x				x										x	x																			
Internaldcu2_line_trip5																						x					x									x	x																			
Internaldcu2_line_trip6																						x						x								x	x																			
Internaldcu2_line_trip7																						x							x							x	x																			
Internaldcu2_line_trip8																						x								x						x	x																			
Internaldcu2_line_trip9			x																		x	x					x				x				x	x	x																				
Internaldcu2_line_trip10	x		x	x								x									x	x					x			x	x	x		x	x	x	x	x	x	x	x																
Internaldcu2_line_trip11	x	x								x	x										x	x					x			x	x		x	x	x	x	x	x	x	x																	
Internaldcu2_line_trip12			x																		x	x								x				x	x	x	x																				
Direct1																					x	x								x					x	x	x																				
Direct2																																				x																					
Direct3																						x														x	x																			
Direct4	x		x																		x	x								x				x	x	x	x	x																			
Direct5	x		x																		x	x								x				x	x	x	x		x																		
Direct6	x	x																			x	x								x				x	x	x	x	x	x	x																	
Indirect1	x		x																		x	x					x				x				x	x	x	x	x		x																
Indirect2	x		x																		x	x								x				x	x	x	x	x	x			x															
Indirect3	x		x		x		x					x		x						x	x	x				x	x	x				x				x	x	x	x	x	x	x		x					x			x			x			
Indirect4	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect5	x		x		x							x									x	x	x				x				x				x	x	x	x	x	x	x				x						x						
Indirect6	x		x		x							x									x	x		x			x				x				x	x	x	x	x	x	x					x					x						
Indirect7	x		x		x							x									x	x			x		x				x				x	x	x	x	x	x	x						x				x						
Indirect8	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x							x			x						
Indirect9	x		x		x							x									x	x							x		x				x	x	x	x	x	x	x		x							x			x					
Indirect10	x		x		x							x									x	x								x	x				x	x	x	x	x	x	x		x								x		x					
Indirect11	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x										x						
Indirect12	x		x		x							x									x	x								x	x				x	x	x	x	x	x	x		x										x					
Indirect13	x		x		x		x					x		x						x	x	x		x		x	x				x				x	x	x	x	x	x	x							x			x		x				
Indirect14	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x							x			x			x			
Indirect15	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x			x		
Indirect16	x		x		x		x		x			x		x		x				x	x	x			x	x	x				x				x	x	x	x	x	x		x	x	x				x			x				x	x	
Indirect17	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		x

Table 59. Fault injection 7: Independent observers (gray)

	InternalTester1	InternalLtrTsSq1	InternalLtrTsSq2	InternalLtrTsSq3	InternalLtrTsSq4	InternalLtrTsSq5	InternalLtrTsSq6	InternalLtrTsSq7	InternalLtrTsSq8	InternalLtrTsSq9	InternalLtrTsSq10	InternalLtrTsSq11	InternalLtrTsSq12	InternalLtrTsSq13	InternalLtrTsSq14	InternalLtrTsSq15	InternalLtrTsSq16	InternalLtrTsSq17	InternalLtrTsSq18	InternalLtrTsSq19	InternalController1	InternalLtrInt1	**Internaldcu2_line_trip1**	**Internaldcu2_line_trip2**	**Internaldcu2_line_trip3**	**Internaldcu2_line_trip4**	Internaldcu2_line_trip5	**Internaldcu2_line_trip6**	**Internaldcu2_line_trip7**	Internaldcu2_line_trip8	Internaldcu2_line_trip9	Internaldcu2_line_trip10	Internaldcu2_line_trip11	Internaldcu2_line_trip12	**Direct1**	Direct2	Direct3	Direct4	Direct5	Direct6	Indirect1	Indirect2	Indirect3	Indirect4	Indirect5	Indirect6	Indirect7	Indirect8	Indirect9	Indirect10	**Indirect11**	**Indirect12**	**Indirect13**	**Indirect14**	**Indirect15**	**Indirect16**	**Indirect17**
InternalTester1	x		x																		x	x								x				x	x	x	x																				
InternalLtrTsSq1		x																			x	x								x					x	x	x																			
InternalLtrTsSq2			x																		x	x								x					x	x	x																			
InternalLtrTsSq3	x		x	x								x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq4	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x															
InternalLtrTsSq5	x		x		x	x						x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq6	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq7	x		x		x		x	x				x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq8	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq9	x	x							x	x											x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq10	x	x								x											x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq11	x		x									x									x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq12	x		x	x								x	x								x	x								x				x	x	x	x	x	x	x		x															
InternalLtrTsSq13	x		x		x							x		x							x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq14	x		x		x	x						x		x	x						x	x							x	x				x	x	x	x	x	x	x		x								x		x					
InternalLtrTsSq15	x		x		x		x					x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
InternalLtrTsSq16	x		x		x		x	x				x		x		x	x			x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		
InternalLtrTsSq17	x		x		x		x		x			x		x		x		x		x	x	x						x		x				x	x	x	x	x	x	x		x		x					x	x		x			x		x
InternalLtrTsSq18	x	x							x	x									x		x	x								x				x	x	x	x	x	x	x																	
InternalLtrTsSq19	x		x		x							x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x					
InternalController1																					x	x								x						x	x																			
InternalLtrInt1																						x														x																				
Internaldcu2_line_trip1																						x	x													x	x																			
Internaldcu2_line_trip2																						x		x												x	x																			
Internaldcu2_line_trip3																						x			x											x	x																			
Internaldcu2_line_trip4																						x				x										x	x																			
Internaldcu2_line_trip5																						x					x									x	x																			
Internaldcu2_line_trip6																						x						x								x	x																			
Internaldcu2_line_trip7																						x							x							x	x																			
Internaldcu2_line_trip8																						x								x						x	x																			
Internaldcu2_line_trip9			x																		x	x					x				x				x	x	x																			
Internaldcu2_line_trip10	x		x	x								x									x	x					x			x	x	x		x	x	x	x	x	x	x	x																
Internaldcu2_line_trip11	x	x							x	x											x	x					x			x	x		x	x	x	x	x	x	x	x																	
Internaldcu2_line_trip12			x																		x	x								x				x	x	x	x																			
Direct1																					x	x								x					x	x	x																			
Direct2																																				x																					
Direct3																						x														x	x																			
Direct4	x		x																		x	x								x				x	x	x	x	x																		
Direct5	x		x																		x	x								x				x	x	x	x		x																	
Direct6	x	x																			x	x								x				x	x	x	x	x	x	x																	
Indirect1	x		x																		x	x					x				x				x	x	x	x	x		x																
Indirect2	x		x																		x	x								x				x	x	x	x	x	x			x															
Indirect3	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x		x					x			x			x			
Indirect4	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		
Indirect5	x		x		x							x									x	x	x				x				x				x	x	x	x	x	x	x				x						x						
Indirect6	x		x		x							x									x	x		x			x				x				x	x	x	x	x	x	x					x					x						
Indirect7	x		x		x							x									x	x			x		x				x				x	x	x	x	x	x	x						x				x						
Indirect8	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x							x			x						
Indirect9	x		x		x							x									x	x						x		x				x	x	x	x	x	x	x		x							x			x					
Indirect10	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x								x		x					
Indirect11	x		x		x							x									x	x				x	x				x				x	x	x	x	x	x	x										x						
Indirect12	x		x		x							x									x	x							x	x				x	x	x	x	x	x	x		x										x					
Indirect13	x		x		x		x					x		x						x	x	x		x		x	x				x				x	x	x	x	x	x	x							x			x		x				
Indirect14	x		x		x		x					x		x						x	x	x			x	x	x				x				x	x	x	x	x	x	x							x			x			x			
Indirect15	x		x		x		x					x		x						x	x	x						x	x	x				x	x	x	x	x	x	x		x								x		x			x		
Indirect16	x		x		x		x		x			x		x		x				x	x	x			x	x	x				x				x	x	x	x	x	x		x	x	x				x			x				x	x	
Indirect17	x		x		x		x		x			x		x		x				x	x	x						x	x	x				x	x	x	x	x	x	x		x		x						x		x			x		x

Appendix E Slices

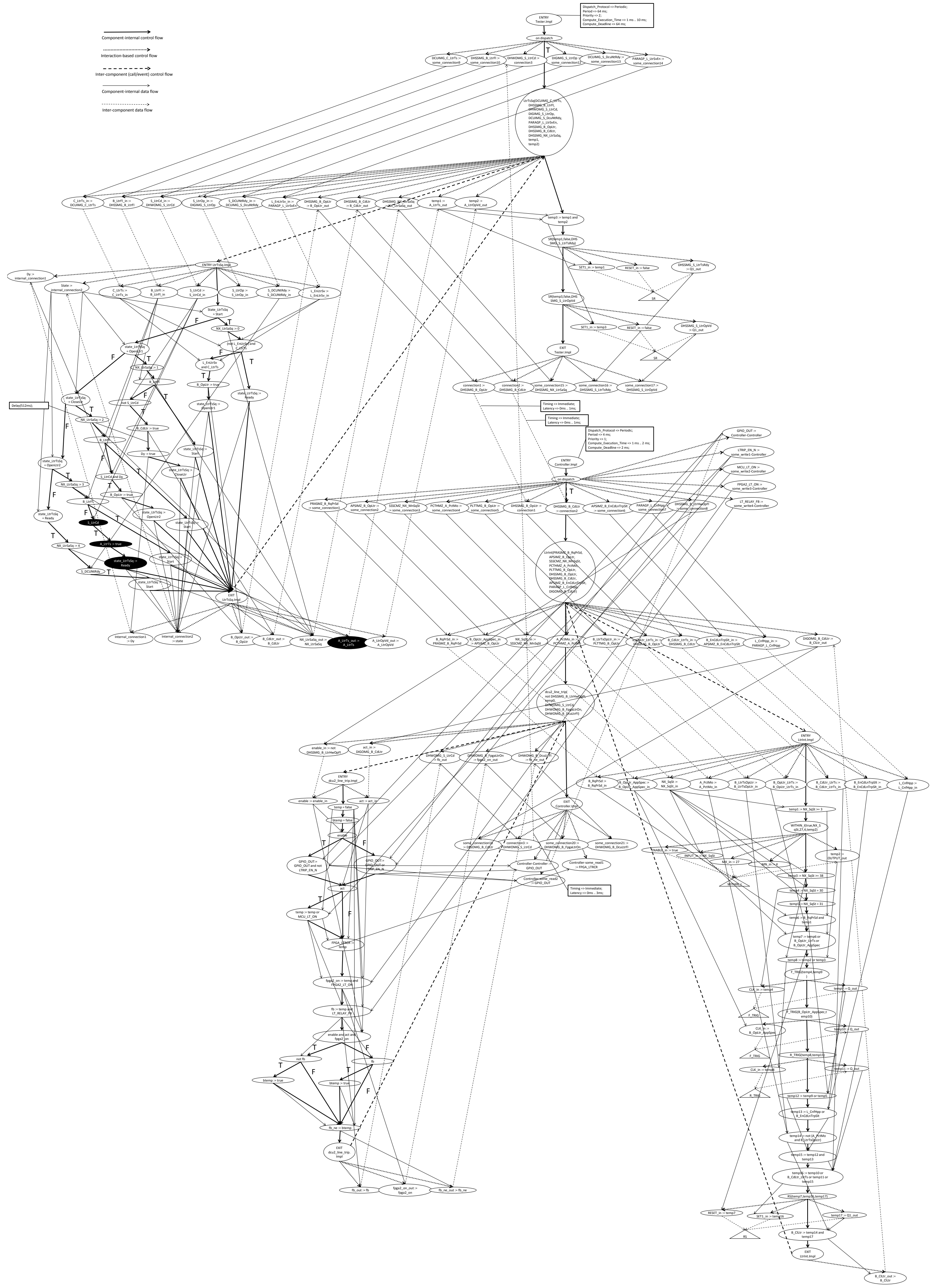

Component-internal control flow
Interaction-based control flow
Inter-component (call/event) control flow
Component-internal data flow
Inter-component data flow
ENTRY Tester.impl
Dispatch_Protocol => Periodic;
Period => 64 ms;
Priority => 2;
Compute_Execution_Time => 1 ms .. 10 ms;
Compute_Deadline => 64 ms;
on dispatch
EXIT Tester.impl
ENTRY LtrTsSq.impl
EXIT LtrTsSq.impl
Delay(512ms);
Timing => Immediate;
Latency => 0ms .. 1ms;
Dispatch_Protocol => Periodic;
Period => 4 ms;
Priority => 1;
Compute_Execution_Time => 1 ms .. 2 ms;
Compute_Deadline => 2 ms;
ENTRY Controller.impl
EXIT Controller.impl
ENTRY dcu2_line_trip.impl
EXIT dcu2_line_trip.impl
Timing => Immediate;
Latency => 0ms .. 3ms;
ENTRY Ltrint.impl
EXIT Ltrint.impl
SR
RS
F_TRIG
R_TRIG
T
F

Fig. 24. Fault injection 1: Slice

Fig. 25. Fault injection 2: Slice

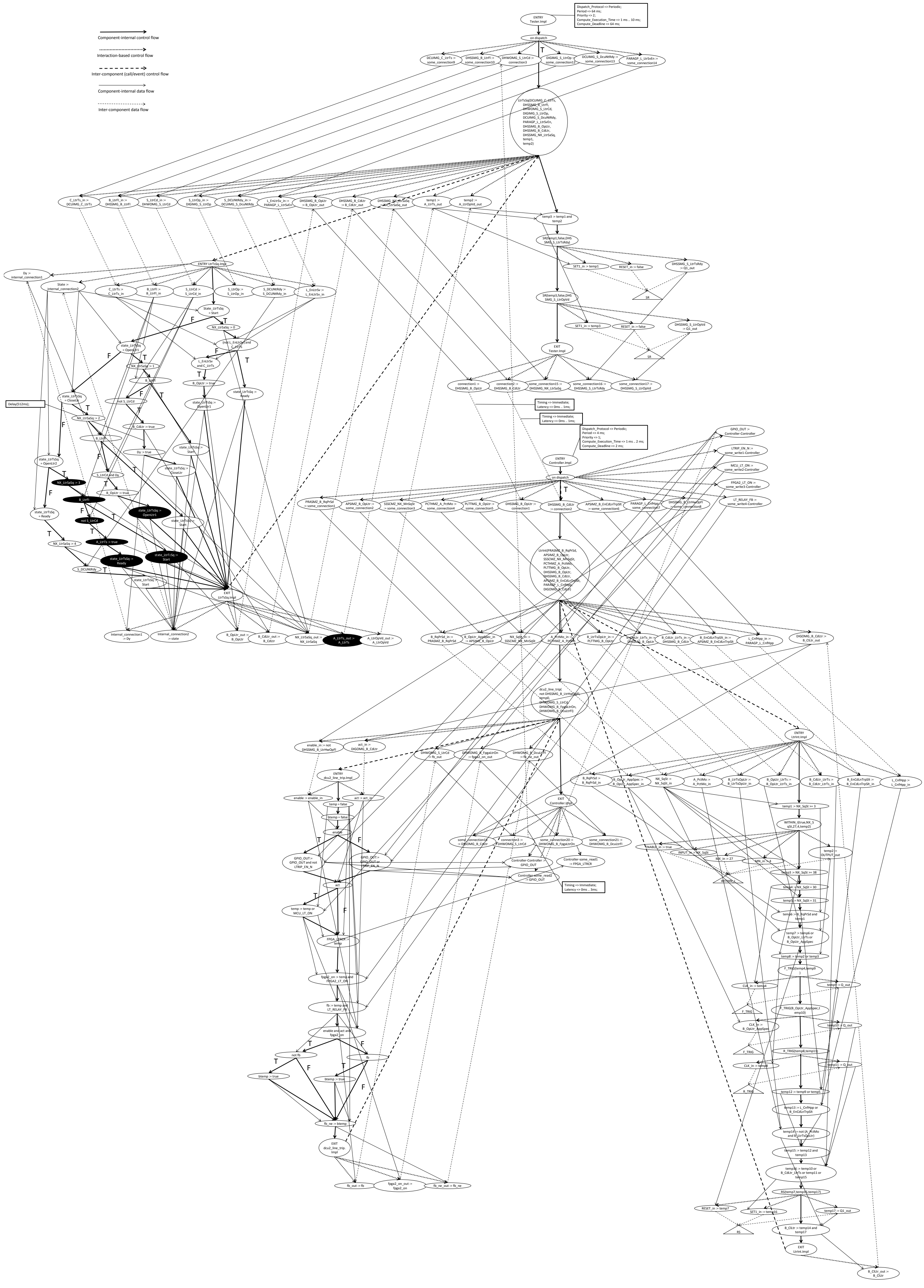

Component-internal control flow
Interaction-based control flow
Inter-component (call/event) control flow
Component-internal data flow
Inter-component data flow
Dispatch_Protocol => Periodic;
Period => 64 ms;
Priority => 2;
Compute_Execution_Time => 1 ms .. 10 ms;
Compute_Deadline => 64 ms;
ENTRY Tester.Impl
on dispatch
T
DCUIMG_C_LtrTs := some_connection9
DHSSMG_B_LtrFl := some_connection10
DHWOMG_S_LtrCd := connection3
DIGIMG_S_LtrOp := some_connection12
DCUIMG_S_DcuNtRdy := some_connection13
PARAGP_L_LtrSvEn := some_connection14
LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl, DHWOMG_S_LtrCd, DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy, PARAGP_L_LtrSvEn, DHSSMG_B_OpLtr, DHSSMG_B_CdLtr, DHSSMG_NX_LtrSaSq, temp1, temp2)
C_LtrTs_in := DCUIMG_C_LtrTs
B_LtrFl_in := DHSSMG_B_LtrFl
S_LtrCd_in := DHWOMG_S_LtrCd
S_LtrOp_in := DIGIMG_S_LtrOp
S_DCUNtRdy_in := DCUIMG_S_DcuNtRdy
L_EnLtrSv_in := PARAGP_L_LtrSvEn
DHSSMG_B_OpLtr := B_OpLtr_out
DHSSMG_B_CdLtr := B_CdLtr_out
DHSSMG_NX_LtrSaSq := NX_LtrSaSq_out
temp1 := A_LtrTs_out
temp2 := A_LtrOpVd_out
temp3 := temp1 and temp2
SR(temp1,false,DHSSMG_S_LtrTsRdy)
SET1_in := temp1
RESET_in := false
DHSSMG_S_LtrTsRdy := Q1_out
SR
SR(temp3,false,DHSSMG_S_LtrOpVd)
SET1_in := temp3
RESET_in := false
DHSSMG_S_LtrOpVd := Q1_out
SR
EXIT Tester.Impl
connection1 := DHSSMG_B_OpLtr
connection2 := DHSSMG_B_CdLtr
some_connection15 := DHSSMG_NX_LtrSaSq
some_connection16 := DHSSMG_S_LtrTsRdy
some_connection17 := DHSSMG_S_LtrOpVd
Timing => Immediate;
Latency => 0ms .. 1ms;
Timing => Immediate;
Latency => 0ms .. 1ms;
ENTRY LtrTsSq.Impl
Dy := internal_connection1
State := internal_connection2
C_LtrTs := C_LtrTs_in
B_LtrFl := B_LtrFl_in
S_LtrCd := S_LtrCd_in
S_LtrOp := S_LtrOp_in
S_DCUNtRdy := S_DCUNtRdy_in
L_EnLtrSv := L_EnLtrSv_in
State_LtrTsSq = Start
F
T
NX_LtrSaSq := 0
(not L_EnLtrSv) and C_LtrTs
state_LtrTsSq = OpenLtr1
L_EnLtrSv and C_LtrTs
NX_LtrSaSq := 1
B_LtrFl
B_OpLtr := true
state_LtrTsSq := Ready
Delay(512ms);
state_LtrTsSq = CloseLtr
not S_LtrCd
state_LtrTsSq := OpenLtr1
NX_LtrSaSq := 2
B_CdLtr := true
B_LtrFl
Dy := true
state_LtrTsSq := Start
state_LtrTsSq = OpenLtr2
state_LtrTsSq := CloseLtr
S_LtrCd and Dy
NX_LtrSaSq := 3
B_OpLtr := true
B_LtrFl
state_LtrTsSq := OpenLtr1
state_LtrTsSq = Ready
not S_LtrCd
state_LtrTsSq := Start
A_LtrTs := true
NX_LtrSaSq := 4
state_LtrTsSq := Ready
state_LtrTsSq := Start
S_DCUNtRdy
state_LtrTsSq := Start
EXIT LtrTsSq.Impl
Internal_connection1 := Dy
Internal_connection2 := state
B_OpLtr_out := B_OpLtr
B_CdLtr_out := B_CdLtr
NX_LtrSaSq_out := NX_LtrSaSq
A_LtrTs_out := A_LtrTs
A_LtrOpVd_out := A_LtrOpVd
Dispatch_Protocol => Periodic;
Period => 4 ms;
Priority => 1;
Compute_Execution_Time => 1 ms .. 2 ms;
Compute_Deadline => 2 ms;
ENTRY Controller.Impl
on dispatch
GPIO_OUT := Controller-Controller
LTRIP_EN_N := some_write1-Controller
MCU_LT_ON := some_write2-Controller
FPGA2_LT_ON := some_write3-Controller
LT_RELAY_FB := some_write4-Controller
PRASMZ_B_RqPrSd := some_connection1
APSIMZ_B_OpLtr := some_connection2
SSSCMZ_NX_MnSqSt := some_connection3
PCTHMZ_A_PctMo := some_connection4
PLTTMG_B_OpLtr := some_connection5
DHSSMG_B_OpLtr := connection1
DHSSMG_B_CdLtr := connection2
APSIMZ_B_EnCdLnTrpSlt := some_connection6
PARAGP_L_CnfHpp := some_connection7
DHSSMG_B_LtrHwOpFl := some_connection8
LtrInt(PRASMZ_B_RqPrSd, APSIMZ_B_OpLtr, SSSCMZ_NX_MnSqSt, PCTHMZ_A_PctMo, PLTTMG_B_OpLtr, DHSSMG_B_OpLtr, DHSSMG_B_CdLtr, APSIMZ_B_EnCdLnTrpSlt, PARAGP_L_CnfHpp, DIGOMG_B_CdLtr)
B_RqPrSd_in := PRASMZ_B_RqPrSd
B_OpLtr_AppSpec_in := APSIMZ_B_OpLtr
NX_SqSt_in := SSSCMZ_NX_MnSqSt
A_PctMo_in := PCTHMZ_A_PctMo
B_LtrTsOpLtr_in := PLTTMG_B_OpLtr
B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr
B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr
B_EnCdLnTrpSlt_in := APSIMZ_B_EnCdLnTrpSlt
L_CnfHpp_in := PARAGP_L_CnfHpp
DIGOMG_B_CdLtr := B_ClLtr_out
dcu2_line_trip(not DHSSMG_B_LtrHwOpFl, temp0, DHWOMG_S_LtrCd, DHWOMG_B_FpgaLtrOn, DHWOMG_B_DcuLtrFl)
enable_in := not DHSSMG_B_LtrHwOpFl
act_in := DIGOMG_B_CdLtr
DHWOMG_S_LtrCd := fb_out
DHWOMG_B_FpgaLtrOn := fpga2_on_out
DHWOMG_B_DcuLtrFl := fb_ne_out
ENTRY dcu2_line_trip.Impl
enable := enable_in
act := act_in
temp = false
btemp = false
enable
T
F
GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N
GPIO_OUT:= GPIO_OUT or LTRIP_EN_N
act
T
F
temp := temp or MCU_LT_ON
FPGA_LTRCR := temp
fpga2_on := temp and FPGA2_LT_ON
fb := temp and LT_RELAY_FB
enable and act and fpga2_on
T
F
not fb
fb
T
F
T
F
btemp := true
btemp := true
fb_ne := btemp
EXIT dcu2_line_trip.Impl
fb_out := fb
fpga2_on_out := fpga2_on
fb_ne_out := fb_ne
EXIT Controller.Impl
some_connection18 := DIGOMG_B_CdLtr
connection3 := DHWOMG_S_LtrCd
some_connection20 := DHWOMG_B_FpgaLtrOn
some_connection21 := DHWOMG_B_DcuLtrFl
Controller-Controller := GPIO_OUT
Controller-some_read1 := FPGA_LTRCR
Controller-some_read2 := GPIO_OUT
Timing => Immediate;
Latency => 0ms .. 3ms;
ENTRY LtrInt.Impl
B_RqPrSd := B_RqPrSd_in
B_OpLtr_AppSpec := B_OpLtr_AppSpec_in
NX_SqSt := NX_SqSt_in
A_PctMo := A_PctMo_in
B_LtrTsOpLtr := B_LtrTsOpLtr_in
B_OpLtr_LtrTs := B_OpLtr_LtrTs_in
B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
B_EnCdLnTrpSlt := B_EnCdLnTrpSlt_in
L_CnfHpp := L_CnfHpp_in
temp1 := NX_SqSt >= 3
WITHIN_I(true,NX_SqSt,27,4,temp2)
ENABLE_in := true
INPUT_in := NX_SqSt
MX_in := 27
MN_in := 4
temp2 := OUTPUT_out
WITHIN_I
temp3 := NX_SqSt >= 38
temp4 := NX_SqSt = 30
temp5 := NX_SqSt = 31
temp6 := B_RqPrSd and temp1
temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec
temp8 := temp2 or temp3
F_TRIG(temp4,temp9)
CLK_in := temp4
temp9 := Q_out
F_TRIG
F_TRIG(B_OpLtr_AppSpec,temp10)
CLK_in := B_OpLtr_AppSpec
temp10 := Q_out
F_TRIG
R_TRIG(temp8,temp11)
CLK_in := temp8
temp11 := Q_out
R_TRIG
temp12 := temp9 or temp5
temp13 := L_CnfHpp or B_EnCdLnTrpSlt
temp14 := not (A_PctMo and B_LtrTsOpLtr)
temp15 := temp12 and temp13
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15
RS(temp7,temp16,temp17)
RESET_in := temp7
SET1_in := temp16
temp17 := Q1_out
RS
B_ClLtr := temp14 and temp17
EXIT LtrInt.Impl
B_ClLtr_out := B_ClLtr

Fig. 26. Fault injection 3: Slice

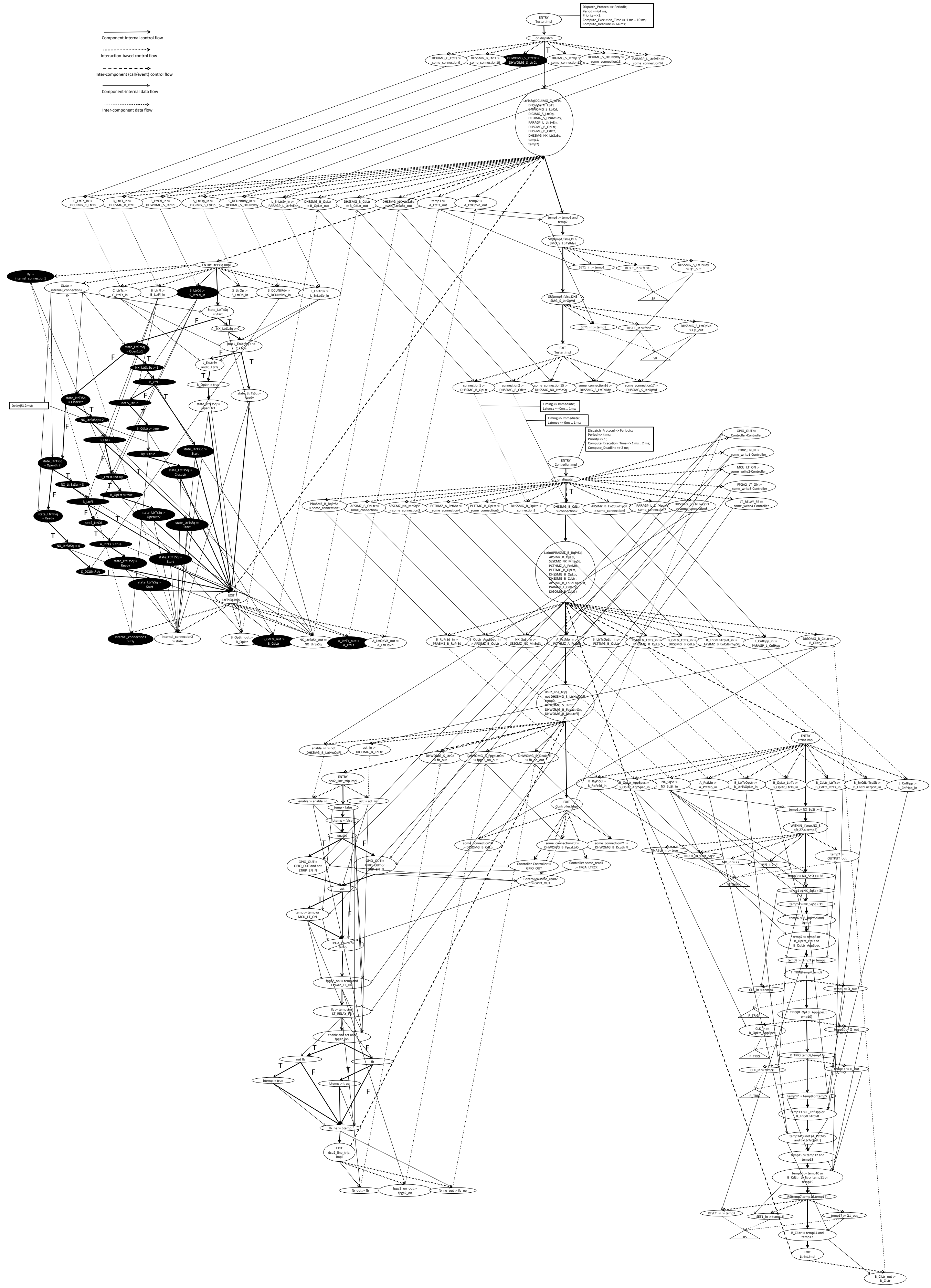

Component-internal control flow
Interaction-based control flow
Inter-component (call/event) control flow
Component-internal data flow
Inter-component data flow
Dispatch_Protocol => Periodic;
Period => 64 ms;
Priority => 2;
Compute_Execution_Time => 1 ms .. 10 ms;
Compute_Deadline => 64 ms;
ENTRY
Tester.impl
on dispatch
Delay(512ms);
Timing => Immediate;
Latency => 0ms .. 1ms;
Dispatch_Protocol => Periodic;
Period => 4 ms;
Priority => 1;
Compute_Execution_Time => 1 ms .. 2 ms;
Compute_Deadline => 2 ms;
ENTRY
Controller.impl
EXIT
Tester.impl
EXIT
Controller.impl
ENTRY LtrTsSq.impl
EXIT
LtrTsSq.impl
ENTRY
dcs2_line_trip.impl
EXIT
dcs2_line_trip.
impl
ENTRY
Ltrint.impl
EXIT
Ltrint.impl
SR
RS
F_TRIG
R_TRIG
T
F

Fig. 27. Fault injection 4: Slice

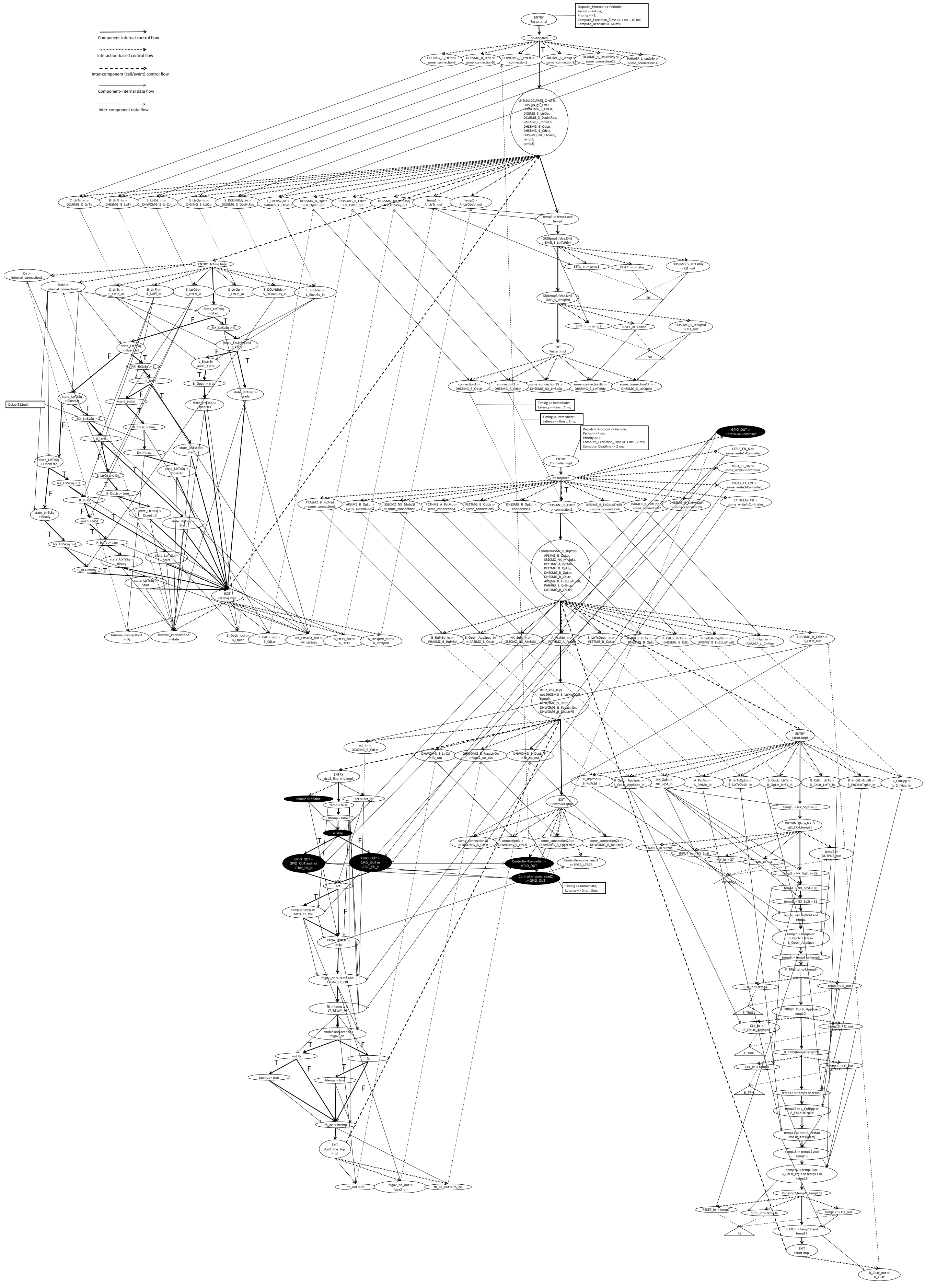

Component-internal control flow
Interaction-based control flow
Inter-component (call/event) control flow
Component-internal data flow
Inter-component data flow
ENTRY Tester.Impl
Dispatch_Protocol => Periodic; Period => 64 ms; Priority => 2; Compute_Execution_Time => 1 ms .. 10 ms; Compute_Deadline => 64 ms;
on dispatch
DCUIMG_C_LtrTs := some_connection9
DHSSMG_B_LtrFl := some_connection10
DHWOMG_S_LtrCd := connection3
DIGIMG_S_LtrOp := some_connection12
DCUIMG_S_DcuNtRdy := some_connection13
PARAGP_L_LtrSvEn := some_connection14
LtrTsSq(DCUIMG_C_LtrTs, DHSSMG_B_LtrFl, DHWOMG_S_LtrCd, DIGIMG_S_LtrOp, DCUIMG_S_DcuNtRdy, PARAGP_L_LtrSvEn, DHSSMG_B_OpLtr, DHSSMG_B_CdLtr, DHSSMG_NX_LtrSaSq, temp1, temp2)
C_LtrTs_in := DCUIMG_C_LtrTs
B_LtrFl_in := DHSSMG_B_LtrFl
S_LtrCd_in := DHWOMG_S_LtrCd
S_LtrOp_in := DIGIMG_S_LtrOp
S_DCUNtRdy_in := DCUIMG_S_DcuNtRdy
L_EnLtrSv_in := PARAGP_L_LtrSvEn
DHSSMG_B_OpLtr := B_OpLtr_out
DHSSMG_B_CdLtr := B_CdLtr_out
DHSSMG_NX_LtrSaSq := NX_LtrSaSq_out
temp1 := A_LtrTs_out
temp2 := A_LtrOpVd_out
temp3 := temp1 and temp2
SR(temp1,false,DHSSMG_S_LtrTsRdy)
SET1_in := temp1
RESET_in := false
DHSSMG_S_LtrTsRdy := Q1_out
SR
SR(temp3,false,DHSSMG_S_LtrOpVd)
SET1_in := temp3
RESET_in := false
DHSSMG_S_LtrOpVd := Q1_out
SR
EXIT Tester.Impl
connection1 := DHSSMG_B_OpLtr
connection2 := DHSSMG_B_CdLtr
some_connection15 := DHSSMG_NX_LtrSaSq
some_connection16 := DHSSMG_S_LtrTsRdy
some_connection17 := DHSSMG_S_LtrOpVd
Timing => Immediate; Latency => 0ms .. 1ms;
Timing => Immediate; Latency => 0ms .. 1ms;
ENTRY LtrTsSq.Impl
Dy := internal_connection1
State := internal_connection2
C_LtrTs := C_LtrTs_in
B_LtrFl := B_LtrFl_in
S_LtrCd := S_LtrCd_in
S_LtrOp := S_LtrOp_in
S_DCUNtRdy := S_DCUNtRdy_in
L_EnLtrSv := L_EnLtrSv_in
State_LtrTsSq = Start
NX_LtrSaSq := 0
(not L_EnLtrSv) and C_LtrTs
state_LtrTsSq = OpenLtr1
NX_LtrSaSq := 1
B_LtrFl
L_EnLtrSv and C_LtrTs
B_OpLtr := true
state_LtrTsSq := Ready
state_LtrTsSq := OpenLtr1
Delay(512ms);
state_LtrTsSq = CloseLtr
not S_LtrCd
NX_LtrSaSq := 2
B_CdLtr := true
B_LtrFl
Dy := true
state_LtrTsSq := Start
state_LtrTsSq := CloseLtr
state_LtrTsSq = OpenLtr2
S_LtrCd and Dy
NX_LtrSaSq := 3
B_OpLtr := true
B_LtrFl
state_LtrTsSq := OpenLtr2
state_LtrTsSq := Start
state_LtrTsSq = Ready
not S_LtrCd
NX_LtrSaSq := 4
A_LtrTs := true
state_LtrTsSq := Ready
state_LtrTsSq := Start
S_DCUNtRdy
state_LtrTsSq := Start
EXIT LtrTsSq.Impl
Internal_connection1 := Dy
Internal_connection2 := state
B_OpLtr_out := B_OpLtr
B_CdLtr_out := B_CdLtr
NX_LtrSaSq_out := NX_LtrSaSq
A_LtrTs_out := A_LtrTs
A_LtrOpVd_out := A_LtrOpVd
Dispatch_Protocol => Periodic; Period => 4 ms; Priority => 1; Compute_Execution_Time => 1 ms .. 2 ms; Compute_Deadline => 2 ms;
ENTRY Controller.Impl
on dispatch
GPIO_OUT := Controller-Controller
LTRIP_EN_N := some_write1-Controller
MCU_LT_ON := some_write2-Controller
FPGA2_LT_ON := some_write3-Controller
LT_RELAY_FB := some_write4-Controller
PRASMZ_B_RqPrSd := some_connection1
APSIMZ_B_OpLtr := some_connection2
SSSCMZ_NX_MnSqSt := some_connection3
PCTHMZ_A_PctMo := some_connection4
PLTTMG_B_OpLtr := some_connection5
DHSSMG_B_OpLtr := connection1
DHSSMG_B_CdLtr := connection2
APSIMZ_B_EnCdLnTrpSlt := some_connection6
PARAGP_L_CnfHpp := some_connection7
DHSSMG_B_LtrHwOpFl := some_connection8
LtrInt(PRASMZ_B_RqPrSd, APSIMZ_B_OpLtr, SSSCMZ_NX_MnSqSt, PCTHMZ_A_PctMo, PLTTMG_B_OpLtr, DHSSMG_B_OpLtr, DHSSMG_B_CdLtr, APSIMZ_B_EnCdLnTrpSlt, PARAGP_L_CnfHpp, DIGOMG_B_CdLtr)
B_RqPrSd_in := PRASMZ_B_RqPrSd
B_OpLtr_AppSpec_in := APSIMZ_B_OpLtr
NX_SqSt_in := SSSCMZ_NX_MnSqSt
A_PctMo_in := PCTHMZ_A_PctMo
B_LtrTsOpLtr_in := PLTTMG_B_OpLtr
B_OpLtr_LtrTs_in := DHSSMG_B_OpLtr
B_CdLtr_LtrTs_in := DHSSMG_B_CdLtr
B_EnCdLnTrpSlt_in := APSIMZ_B_EnCdLnTrpSlt
L_CnfHpp_in := PARAGP_L_CnfHpp
DIGOMG_B_CdLtr := B_ClLtr_out
dcu2_line_trip(not DHSSMG_B_LtrHwOpFl, temp0, DHWOMG_S_LtrCd, DHWOMG_B_FpgaLtrOn, DHWOMG_B_DcuLtrFl)
act_in := DIGOMG_B_CdLtr
DHWOMG_S_LtrCd := fb_out
DHWOMG_B_FpgaLtrOn := fpga2_on_out
DHWOMG_B_DcuLtrFl := fb_ne_out
EXIT Controller.Impl
some_connection18 := DIGOMG_B_CdLtr
connection3 := DHWOMG_S_LtrCd
some_connection20 := DHWOMG_B_FpgaLtrOn
some_connection21 := DHWOMG_B_DcuLtrFl
Controller-Controller := GPIO_OUT
Controller-some_read1 := FPGA_LTRCR
Controller-some_read2 := GPIO_OUT
Timing => Immediate; Latency => 0ms .. 3ms;
ENTRY dcu2_line_trip.Impl
enable := enable
act := act_in
temp = false
btemp = false
enable
GPIO_OUT:= GPIO_OUT and not LTRIP_EN_N
GPIO_OUT:= GPIO_OUT or LTRIP_EN_N
act
temp := temp or MCU_LT_ON
FPGA_LTRCR := temp
fpga2_on := temp and FPGA2_LT_ON
fb := temp and LT_RELAY_FB
enable and act and fpga2_on
not fb
fb
btemp := true
btemp := true
fb_ne := btemp
EXIT dcu2_line_trip. Impl
fb_out := fb
fpga2_on_out := fpga2_on
fb_ne_out := fb_ne
ENTRY LtrInt.Impl
B_RqPrSd := B_RqPrSd_in
B_OpLtr_AppSpec := B_OpLtr_AppSpec_in
NX_SqSt := NX_SqSt_in
A_PctMo := A_PctMo_in
B_LtrTsOpLtr := B_LtrTsOpLtr_in
B_OpLtr_LtrTs := B_OpLtr_LtrTs_in
B_CdLtr_LtrTs := B_CdLtr_LtrTs_in
B_EnCdLnTrpSlt := B_EnCdLnTrpSlt_in
L_CnfHpp := L_CnfHpp_in
temp1 := NX_SqSt >= 3
WITHIN_I(true,NX_SqSt,27,4,temp2)
ENABLE_in := true
INPUT_in := NX_SqSt
MX_in := 27
MN_in := 4
temp2 := OUTPUT_out
WITHIN_I
temp3 := NX_SqSt >= 38
temp4 := NX_SqSt = 30
temp5 := NX_SqSt = 31
temp6 := B_RqPrSd and temp1
temp7 := temp6 or B_OpLtr_LtrTs or B_OpLtr_AppSpec
temp8 := temp2 or temp3
F_TRIG(temp4,temp9)
CLK_in := temp4
temp9 := Q_out
F_TRIG
F_TRIG(B_OpLtr_AppSpec,temp10)
CLK_in := B_OpLtr_AppSpec
temp10 := Q_out
F_TRIG
R_TRIG(temp8,temp11)
CLK_in := temp8
temp11 := Q_out
R_TRIG
temp12 := temp9 or temp5
temp13 := L_CnfHpp or B_EnCdLnTrpSlt
temp14 := not (A_PctMo and B_LtrTsOpLtr)
temp15 := temp12 and temp13
temp16 := temp10 or B_CdLtr_LtrTs or temp11 or temp15
RS(temp7,temp16,temp17)
RESET_in := temp7
SET1_in := temp16
temp17 := Q1_out
RS
B_ClLtr := temp14 and temp17
EXIT LtrInt.Impl
B_ClLtr_out := B_ClLtr
T
F

Fig. 28. Fault injection 5: Slice

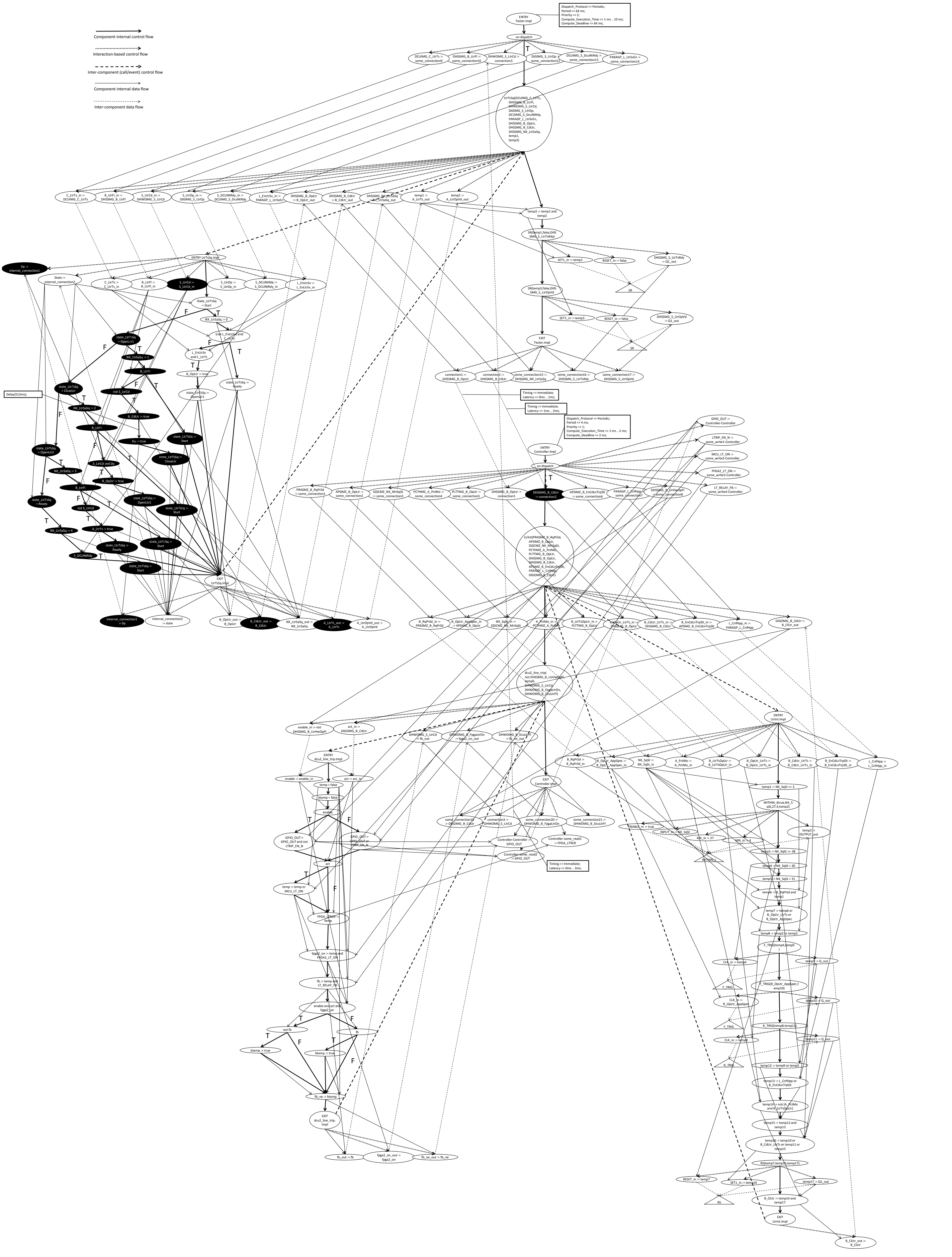
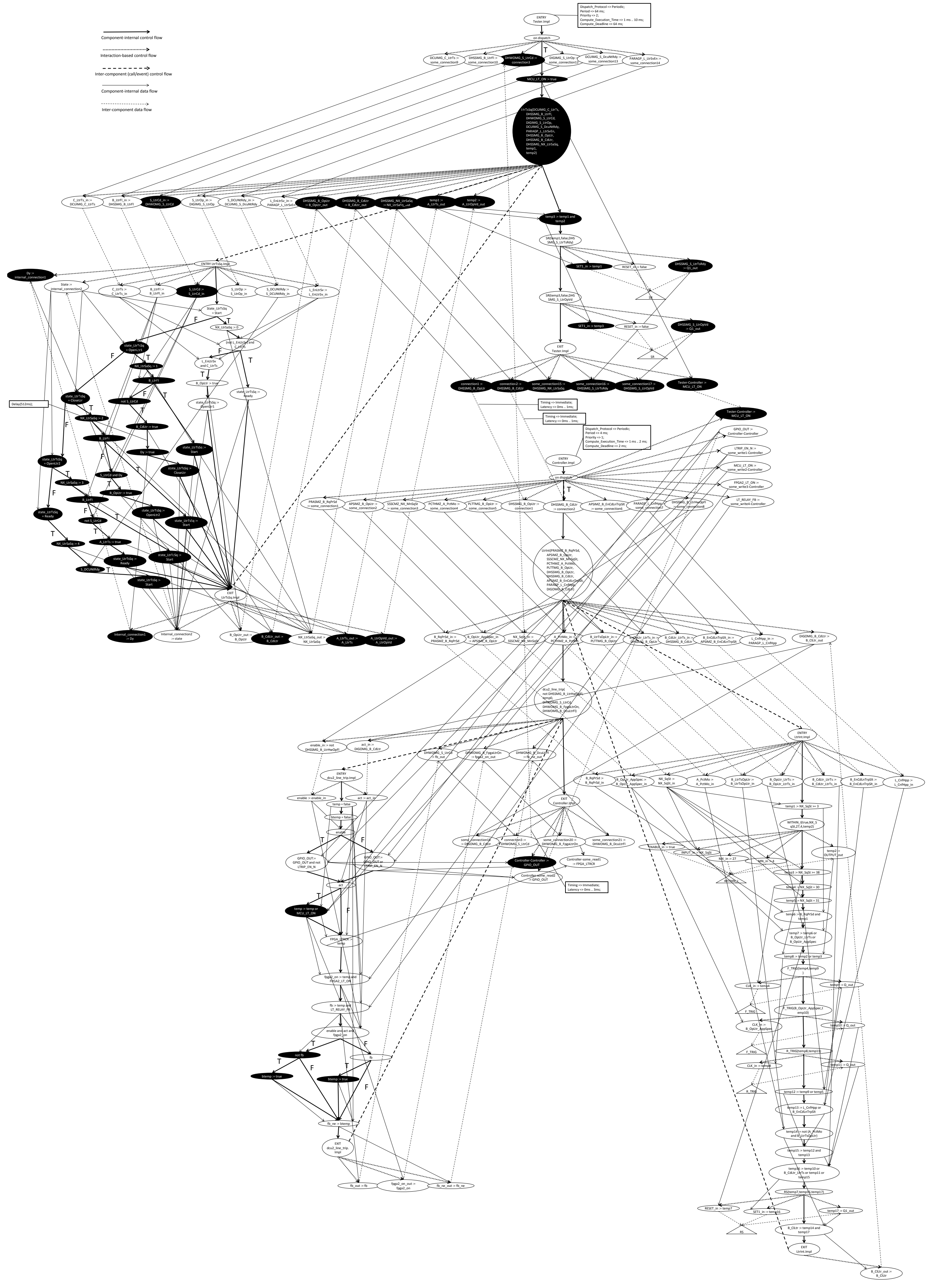


Fig. 29. Fault injection 7: Slice


Component-internal control flow
Interaction-based control flow
Inter-component (call/event) control flow
Component-internal data flow
Inter-component data flow
ENTRY
Tester.impl
Dispatch_Protocol => Periodic;
Period => 64 ms;
Priority => 2;
Compute_Execution_Time => 1 ms .. 10 ms;
Compute_Deadline => 64 ms;
on dispatch
MCU_LT_ON := true
temp3 := temp1 and temp2
EXIT
Tester.impl
ENTRY LtrTsSq.impl
Delay(512ms);
EXIT LtrTsSq.impl
Timing => Immediate;
Latency => 0ms .. 1ms;
Dispatch_Protocol => Periodic;
Period => 4 ms;
Priority => 1;
Compute_Execution_Time => 1 ms .. 2 ms;
Compute_Deadline => 2 ms;
ENTRY
Controller.impl
EXIT
Controller.impl
Tester-Controller := MCU_LT_ON
ENTRY
dcs2_line_trip.impl
EXIT
dcs2_line_trip.impl
ENTRY
Ltrint.impl
EXIT
Ltrint.impl
Timing => Immediate;
Latency => 0ms .. 3ms;
T
F
SR
RS
F_TRIG
R_TRIG