Journal of Electronic Testing (2024) 40:509–523
https://doi.org/10.1007/s10836-024-06126-6

Formal Verification of a Dependable State Machine-Based Hardware Architecture for Safety-Critical Cyber-Physical Systems: Analysis, Design, and Implementation

Shawkat Sabah Khairullah[1]

Received: 24 March 2024 / Accepted: 25 June 2024 / Published online: 5 July 2024

Abstract
With the increasing interest in embedding digital devices in safety-critical *cyber-physical systems* (CPSs), such as industrial automation, aerospace, and automotive industries, attention has been directed toward proposing verifiable and reliable architectures. Prominent levels of formal verification and fault-tolerance are a requirement in dependable CPS systems to ensure system design meet the specifications and verify safety properties. In this paper, a novel formal verifiable and fault-tolerant hardware architecture uses the concepts of state machine, verification, and fault-tolerance as a foundation is developed. It is divided into four models: analysis model includes the functional requirements defined by the user, design model, the finite state machine is utilized to model the systems behavior which is tested by the NuSMV checker tool, implementation model simulates test cases on waveforms to validate the design against the requirements and verification model verifies functional and critical properties using mathematical formal linear time and computation tree logic to prove compliance with requirements and detect errors. The system uses temporal logic to formulate the required properties for a railway interlocking system (RIS) as a case study and symbolic model verifier (SMV) to demonstrate the correct execution. From the simulation results, the effectiveness of the architecture is proved for verifying critical properties and detecting design faults through majority voting circuits. The proposed architecture has been synthesized in the Altera FPGA programmable chip with logic elements 33%, 52% area overhead, and frequency as 100 MHz. The system does meet its reliability requirements with the lowest reliability 91.333687 x 10^{-2} and failure rate 0.2 failure per hour at time 60 min. Finally, we think that adopting this architecture will enhance the trustworthiness and certification of CPS systems.

Keywords Safety · Verifiable · Fault-tolerance · Railway interlocking system (RIS) · Formal method · Trustworthiness

1 Introduction

Cyber-physical systems (CPSs) have been in use for more than a decade in safety-critical applications [1], including medical health-care devices, instrumentation and control systems of nuclear power generation plants, railway interlocking critical infrastructure, and industrial internet of things [2, 3]. Many of these systems have a direct influence on the safety and security of their users and the public. Such systems are often referred to as safety-critical CPS systems, which depend on the correct operation of the electronic hardware architectures that execute a set of instructions and realize logic, sequential and arithmetic functions [4]. The hardware architecture is typically comprised of combinational and sequential logic circuits, which use flip flops, latches, and basic logic gates interconnected to execute safety-critical tasks [5]. The internal structure of these components is prone to the effects of development software errors and hardware operational faults, which can strike the system and lead to catastrophic consequences and a violation of critical state operation [6, 7]. This introduction will discuss the essential dependable techniques required in the design and modeling of safety-critical cyber physical applications and embedded electronic hardware architecture.

Responsible Editor: A. Ivanov.

✉ Shawkat Sabah Khairullah
shawkat.sabah@uomosul.edu.iq

1 Department of Computer Engineering, College of Engineering, University of Mosul, Mosul, Iraq

Examples of these techniques are formal verification and validation (V&V) methodologies, model-based design, and fault tolerance methods.

1.1 Safety-Critical Cyber-Physical Systems

Ensuring the reliability and resilience of safety-critical CPS applications is extremely important and complex task. In recent years, formal verification, electronic testing, model checking, error correction codes and fault tolerance methods have become more prevalent to augment the dependability of safety-critical systems [8–10]. Safety-critical CPSs are a class of systems that are complex in its structure and are often comprised of a diverse set of hardware and software processing components: a sensing electronic module, which collect the data from the physical environment, an actuating electronic module to affect the environment and achieve objectives during the execution, and a computational module that interact via a communication network [11–13]as it is shown in Fig. 1. Furthermore, the safety-critical cyber-physical system is typically consisting of multiple processing electronic components whose operation must be continuously monitored for the correct execution and controlled by resilient, fault-tolerant, and runtime verification techniques [14–16]. Nowadays, the digital components can consist of hardware units, software, or hardware/software co-design interconnections to be used as platforms for the realization of real-time and dependable digital devices.

1.2 Formal Verification

In general, verification and validation (V&V) are essential methods of ensuring the integrity and dependability of digital CPS systems [17]. Moreover, digital system design meets a set of requirements set by the user (e.g., did we design the digital electronic system correctly). However, validation is a method to evaluate whether the digital embedded system meets the user expectations (e.g., did we implement the correct digital system) [17]. Different Formal verification and model checking techniques have been used in the literature [18]to prove a diverse set of time related properties that can be liveness or safety, and requirements for the physical hardware layers or software layers of safety-critical CPS applications [19]. For example, a liveness property indicates that something positive will occur during the system execution, whereas a safety property indicates that nothing terrible occurs during the execution which is bounded and unbounded [20]. Both safety and liveness properties can constitute system invariants using a formal verification tool checker called NuSMV. This tool compares the digital system output with its specific properties that can be defined using a mathematical model represented by temporal logic [21]. An example of symbolic modeling verification language is SMV which works as a formal modeling language supporting the specifications of finite state machines and determines the architectural concepts of digital systems. The input for the SMV model is a textual description defined by linear temporal logic (LTL)-based property or computation tree logic (CTL)-based properties [19]. The basic SMV module includes a group of submodules consists of three subsections: variable, assign, and specification [22–24].

1.3 Fault-Tolerance

Figure 2 shows the basic diagram of a highly verifiable and safety-critical cyber-physical system that can be formally verified through a hybrid method of model-based design and traditional verified fault-tolerance techniques [25, 26]. The correct operation of CPS electronic system demand functional safety regulations, high reliability, and must be

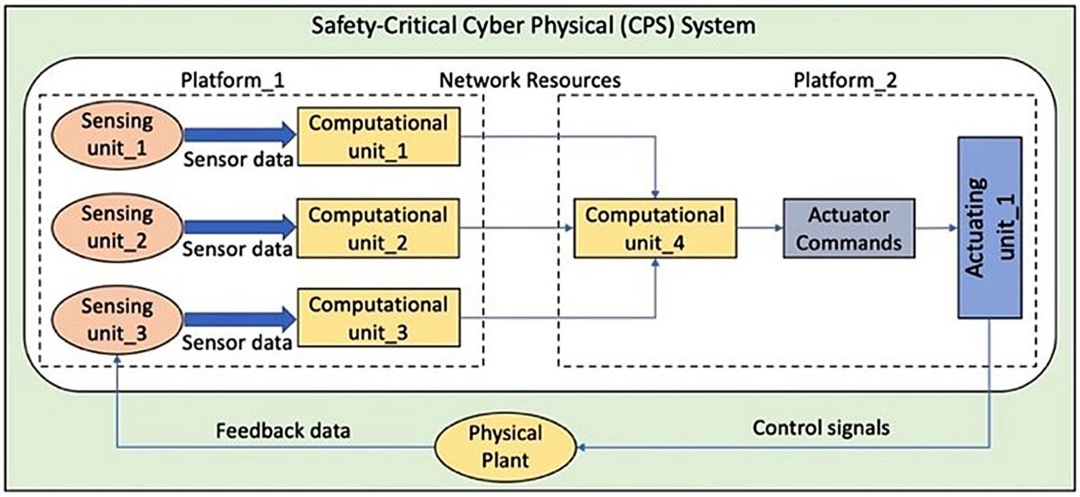


Fig. 1 Safety-critical CPS comprising of multiple design units

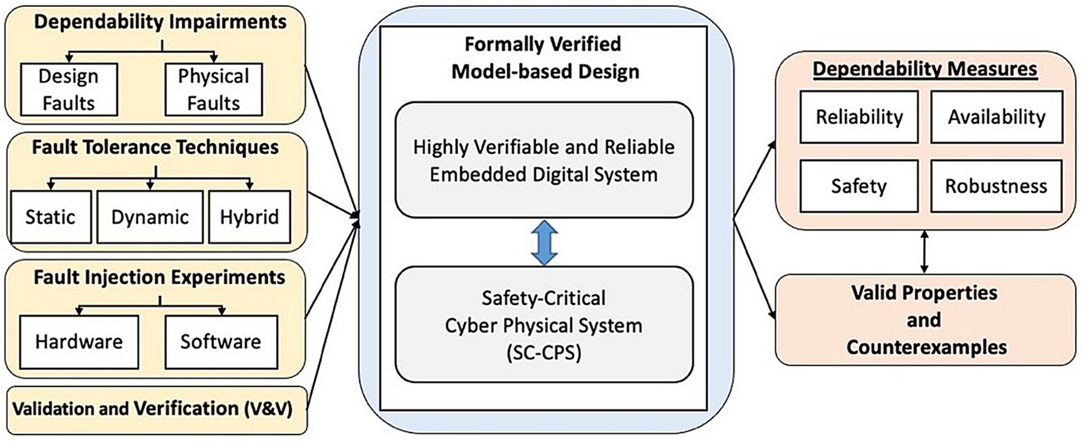

Dependability Impairments
Design Faults
Physical Faults
Fault Tolerance Techniques
Static
Dynamic
Hybrid
Fault Injection Experiments
Hardware
Software
Validation and Verification (V&V)
Formally Verified Model-based Design
Highly Verifiable and Reliable Embedded Digital System
Safety-Critical Cyber Physical System (SC-CPS)
Dependability Measures
Reliability
Availability
Safety
Robustness
Valid Properties and Counterexamples

Fig. 2 General architecture of verifiable safety-critical CPS system

immune against different types of failure modes: physically occurring hardware faults due to ionizing radiation [27]and design software errors [28]. The hardware faults can occur and be of type single event upset (SEU), multiple bit upsets (MBU), stuck-at-0 and stuck-at-1 faults caused by alpha particles or electromagnetic field [29, 30]. For example, the main consequence of SEU fault is bit flips in the flip flop (D, J-K, T, or S-R) memory devices of the CPS digital system. On the other hand, software errors include overflow in the data, dead logic, an array access violation, and division by zero. All these types of faults and errors must be addressed in the design and verification of safety-critical CPS systems [31]. All these methods have been used in the design and analysis of dependable safety-critical digital systems, which require high V&V, formal certification, robustness, and dependability against cyber threats, errors and faults [32]. In addition, Towards the dependability, robustness and formal verification objectives, a novel formal verifiable hardware architecture is proposed targeting the design and analysis of safety-critical applications whose failure may result in human life loss or damage in the critical infrastructure [33].

1.4 The Main Contribution and Outline

The paper presents many contributions in the topics of verification, fault-toerance and dependable embedded systems, and they can be summarized as follows:

- Designing a novel formal verification and fault-tolerant hardware architecture utilizes the concepts of Mealy type finite state machine, verification and validation (V&V), and triple modular redundancy (TMR) fault-tolerance as a foundation in the design, analysis, and implementation.
- It uses formal verification linear temporal logic (LTL) and computation tree logic (CTL) at the development phase to express families of input and output behaviors and formulate the required properties for a simulated safety-critical Railway Interlocking System (RIS) application.
- Symbolic model verifier (SMV) and NuSMV model checker design tool was used to demonstrate the correct execution of essential safety properties and detect design errors.
- The effectiveness of the proposed architecture for verifying critical properties and detecting design faults through majority voting logic circuits have been demonstrated through Quartus FPGA timing diagrams.
- Markov-based reliability analysis with various values of failure rates and composite Karnaugh map were used as primary techniques in the design of the state machine architecture.
- The architecture has been synthesized in the Altera FPGA programmable chip with logic elements 33%, 52% area overhead and frequency as 100 MHz. Furthermore, the system does meet its reliability requirements with the lowest reliability 91.333687 x 10^{-2} and failure rate 0.2 failure per hour at time 60 min.
- Adopting this new method of formal verification and fault-tolerance in which multiple modules are interconnected generating test cases for models, hardware, and software code in the design and analysis of modern digital devices.
- The state machine architecture supports the formal verification and simulation for the design properties and requirements proving or violations whether it is valid or falsified objectives.

The remainder of the research work is structured into four sections. Section 2 presents the research works related to different types of verification approaches, architectural principles and fault-tolerance techniques for digital automation systems. Section 3 describes the schema of the proposed state machine-based hardware architecture and its LTL/CTL modeling that is used in the modeling and formal verification of safety-critical applications. In addition, the requirements analysis, architectural principles and fault-tolerant methods are presented in this section. The effect of varied failure rate on the reliability, FPGA implementation results of experimental safety properties checking using NuSMV model checker and Quartus design tools are discussed in Sect. 4. Section 5 concludes the article and gives some future works for this ongoing research work.

2 Related Work

To identify the specific gaps that this manuscript aims to address in this research, this section provides a comprehensive literature review of formal verification and testing of a CPS design against design faults as well as errors vulnerabilities during the development phase and runtime execution [9, 34]. Furthermore, fault-tolerance is a pervasive error detection approach that is used in hardware digital systems for which a diverse set of faults and errors are expected to cause an unacceptable service failure [35–37]. This section highlights several previous research works on the topics of formal verification and fault-tolerance techniques aiming at achieving high levels of resilience and dependability. The different verification and fault-tolerance approaches developed by the previous researchers are categorized as it is illustrated in Fig. 3:

These approaches are discussed in detail as follows: the authors [34] have developed a verification approach of a complex and safety-critical industrial railway train system by using the Logica Di Sicurvexz (LDS) software. The LDS software executes logic and sequential functions that are specified by human operators who write C++related programming language to prepare train tracks and routes. In [15], a formally verified rule-based approach of state-based control module in a CPS manufacturing automation system is proposed. The logic control module is transferred into a verifiable and synthesizable VHDL module so it can be implemented on FPGA device. The researchers in [38], suggested a formal verification approach targeting the behavior principles of electronic components embedded inside an electric power system. These components with safety and liveness properties are formally described by interpreted petri nets. A verifiable model is generated by applying a rule-based logic module, which can be verified with model checking XuXmv tool. Furthermore, the previous research work presented in [39], develops a model-based software verification approach of a medical device. This approach analyzes the user requirements, models the medical system, and verifies the model in a formal way. In addition, the verification approach is generating code that can be executed in the hardware of a patient healthcare analgesic pump to support the qualification process against different failure modes. Ref. [40] described the architectural concept of a process control application, logic-based controller that combines the control interpreted Petri Nets verification concepts with the hardware components built based on unified modeling language (UML) activity diagrams. A train control system was modeled using the NuSMV verification language in [41] and implemented on ARM Cortex processor to ensure the correctness of the desired safety properties in safe speed

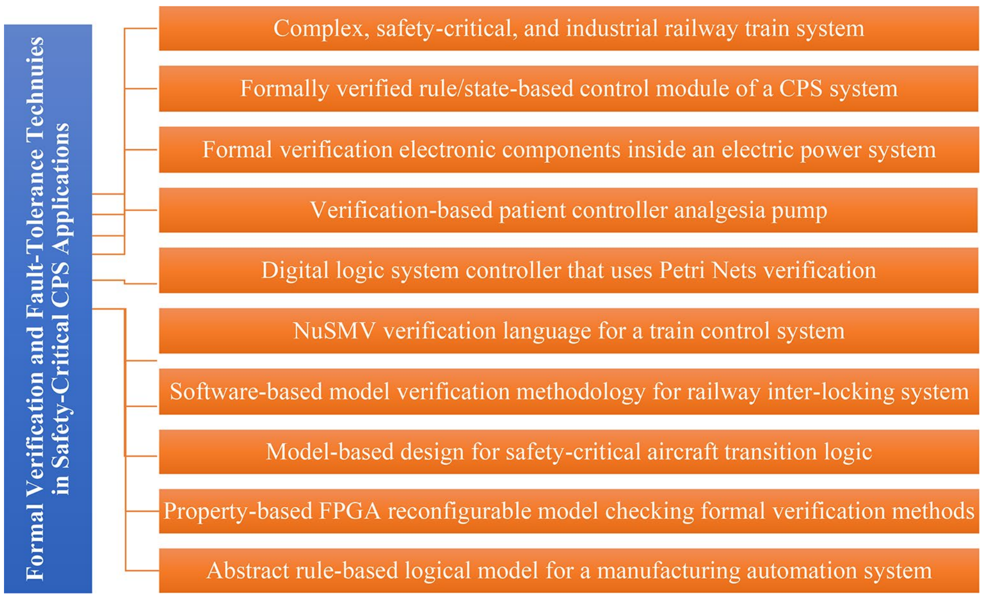

Formal Verification and Fault-Tolerance Technuies in Safety-Critical CPS Applications
Complex, safety-critical, and industrial railway train system
Formally verified rule/state-based control module of a CPS system
Formal verification electronic components inside an electric power system
Verification-based patient controller analgesia pump
Digital logic system controller that uses Petri Nets verification
NuSMV verification language for a train control system
Software-based model verification methodology for railway inter-locking system
Model-based design for safety-critical aircraft transition logic
Property-based FPGA reconfigurable model checking formal verification methods
Abstract rule-based logical model for a manufacturing automation system

Fig. 3 The different formal verification and fault-tolerance techniques in safety-critical CPS applications

application. In [42], the authors suggested a programmable logic controller-based verified and secured home system using ladder diagram programming model. The reliable controller system consists of input, processing, and output hardware modules which operate in real time to read the input data from external sensors (e.g., proximity or motion sensor), execute the ladder program, and send commands to the output modules. In [31], model checking concept is used as a formal technique in verifying LTL/CTL properties of safety-critical instrumentation and control application, a model of reactor nuclear protection system. They combined the communication delays with the hardware failure occurrence in the proposed model using the NuSMV design tool. Concentrating on the object-oriented devices, the researchers suggested a software-based model verification methodology for the railway inter-locking system [43]. Their proposed work focused on the combination of Unified Model Language (UML) generated diagrams and software-based finite state machine models to verify the safety requirements of the system. Using Petri nets property-based fault injection and model checking to test the global specifications of a representative safety-critical system, the authors in [44] have described a novel formal verification technique that can exchange the implemented verified modules in the running system. Furthermore, the rule-based logic models were. Used for the reconfiguration of programmable logic controller realized in FPGA device. Finally, in [45], a vertical mode of the transition mode logic properties found in modeling the safety-critical autopilot design is discussed. They used a formal verification method using NuSMV tool and Matlab in generating test case reports. In addition, they have proved its complete correctness and improved the system dependability.

3 Proposed Dependable State Machine-Based Hardware Architecture for Safety-Critical Applications

The architectural schema of the proposed state machine hardware architecture and its formal verification technique is presented in this section. The dependable architecture, which is divided into four basic models: analysis, design, simulation, and implementation (S&I), and verification and validation (V&V) is illustrated in Fig. 4. The analysis module includes the system functional requirements and its critical properties specifications using temporal logic defined by the designer for the safety-critical CPS system. Additionally, Model-based system and its finite state machine-based architectural concept are integrated into the I&C application using NuSMV model checker tool in the design model. The S&I module performs fault-tolerance test cases on real-time hardware platform e.g., FPGA fabric to validate the design against the user requirements using Quartus software. Finally, the V&V module verifies functional, and safety critical properties using formal methods as LTL/CTL

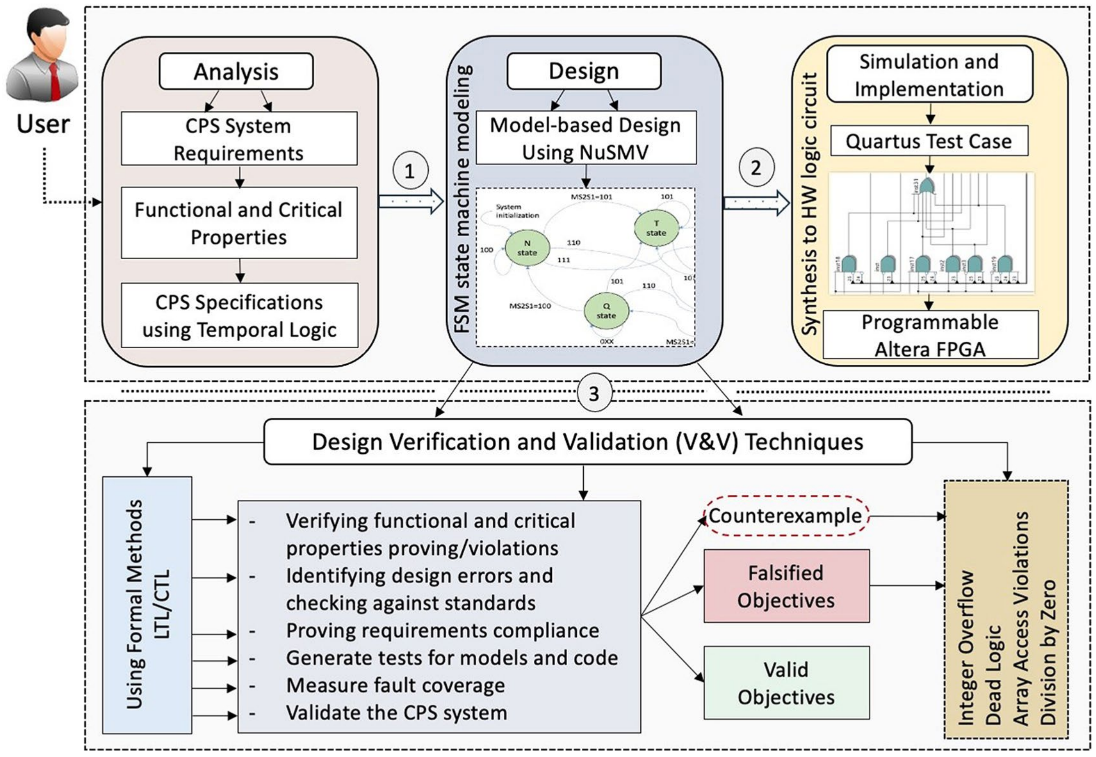

User
Analysis
CPS System Requirements
Functional and Critical Properties
CPS Specifications using Temporal Logic
1
FSM state machine modeling
Design
Model-based Design Using NuSMV
System initialization
MS251=101
N state
T state
Q state
MS251=100
2
Synthesis to HW logic circuit
Simulation and Implementation
Quartus Test Case
Programmable Altera FPGA
3
Design Verification and Validation (V&V) Techniques
Using Formal Methods LTL/CTL
- Verifying functional and critical properties proving/violations
- Identifying design errors and checking against standards
- Proving requirements compliance
- Generate tests for models and code
- Measure fault coverage
- Validate the CPS system
Counterexample
Falsified Objectives
Valid Objectives
Integer Overflow
Dead Logic
Array Access Violations
Division by Zero

Fig. 4 A schema of the proposed formal verification, design, and implementation of a highly dependable state machine architecture

and uses the NuSMV model checker to prove the design requirements compliance and detect deign errors. Transition 1 in Fig. 3 refers to translating the system specifications into a SMV programming code using NuSMV. However, transition 2 refers to synthesizing the hardware architecture from HDL Code or schematic design using Quartus Prime integrated system tool. The verification module refers to examining the timing analysis for the safety-critical CPS application being implemented on the hardware architecture at the high level of verification.

3.1 Analysis of a Railway Interlocking CPS System Requirements

A railway interlocking CPS system was adopted in this work as a safety-critical driverless or assisted train model. Figure 5 presents the main outlines describing the five different case scenarios in which the proposed railway system monitors the status of sensors continuously and sends commands to the actuators. Also, the design and verification of the railway architectural approach is illustrated and the modelling of its software and hardware components is formally verified. The RIS interlocking system is a safety critical application that monitors and controls the objects signaling of the train traffic in order to prevent a conflicting movement [46]. The software system is fulfilled with the indeed properties formally, which describe wanted and unwanted properties using temporal logic language as linear LTL or CTL. Using symbolic model checkers such as the NuSMV tool, the formal properties can be verified visually [34]. If there was a violation property in the model execution, a counterexample will be generated and then the model can be refined. The proposed railway crossing architecture using NuSMV model checker works as it is shown in the following algorithm:

1. When the train is coming and arrive the sensing unit region called sensor1 which is highlighted with the yellow and dotted red color as shown in Fig. 5, the sensing unit will immediately be triggered and activated causing the two gates: gate1 and gate2 highlighted with the black color to be closed, the traffic green lights highlighted with the green color to be on, and the whistle to start working.
2. When the train is determined to be in the safety-critical interlocking zone which is represented by the distance between the sensor1 and sensor2 units, the two gates will remain in a closed status, and the red alarm lights will start flashing.
3. When the train arrives the sensing unit called sensor2, it will immediately be triggered on, the train considers coming out the safety-critical interlocking zone, the two gates will remain closed, the red alarm lights stop flashing, and the whistle remains on.
4. When the two sensing units sensor1 and sensor2 are disabled, that indicates that there is no train in the railway interlocking station, thus, the finite state machine will transition into a safe state in which the two gates will be opened, the green alarm lights are on, and the whistle becomes off.
5. All these automated processes can be presented in a clearer way as shown in Fig. 6.

3.2 Modeling of a Railway Interlocking CPS System Using Sate Machine and UML Model-based Design

Model-based design was used in the realization and simulation of safety-critical digital embedded devices in industrial automation systems, avionics, and instrumentation devices of nuclear power plants [32, 47]. In this subsection, the state

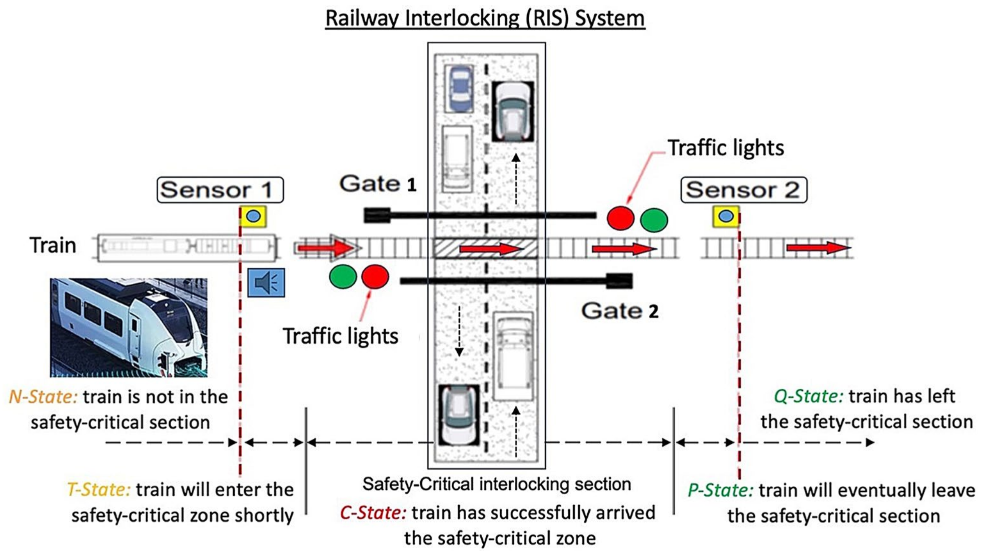

Railway Interlocking (RIS) System
Traffic lights
Sensor 1
Gate 1
Sensor 2
Train
Traffic lights
Gate 2
N-State: train is not in the safety-critical section
Q-State: train has left the safety-critical section
T-State: train will enter the safety-critical zone shortly
Safety-Critical interlocking section
C-State: train has successfully arrived the safety-critical zone
P-State: train will eventually leave the safety-critical section

Fig. 5 High-level model of a case study example of a railway interlocking safety-critical cyber-physical system

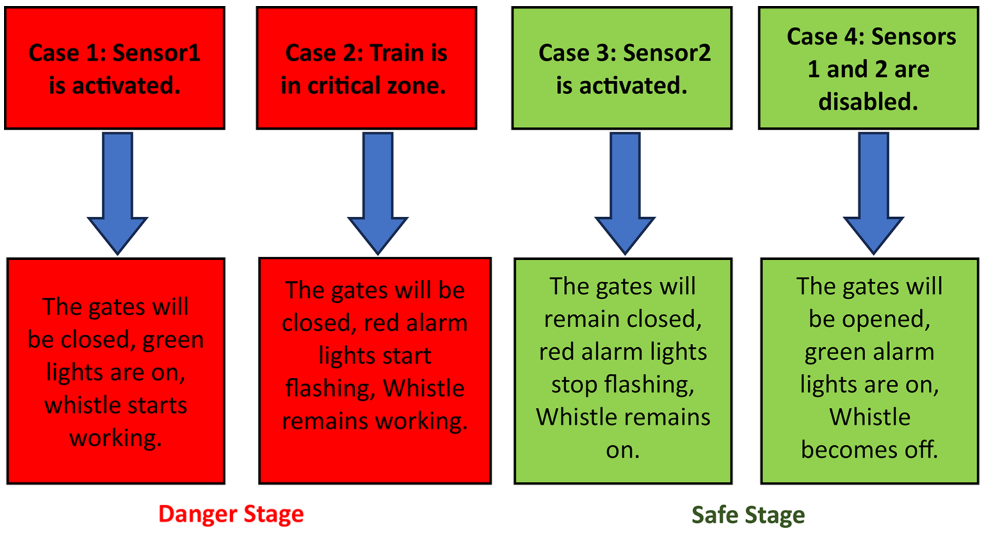

Case 1: Sensor1 is activated.
Case 2: Train is in critical zone.
Case 3: Sensor2 is activated.
Case 4: Sensors 1 and 2 are disabled.
The gates will be closed, green lights are on, whistle starts working.
The gates will be closed, red alarm lights start flashing, Whistle remains working.
The gates will remain closed, red alarm lights stop flashing, Whistle remains on.
The gates will be opened, green alarm lights are on, Whistle becomes off.
Danger Stage
Safe Stage

Fig. 6 The followed algorithm of the proposed railway interlocking safety-critical cyber-physical system using NuSMV model checker

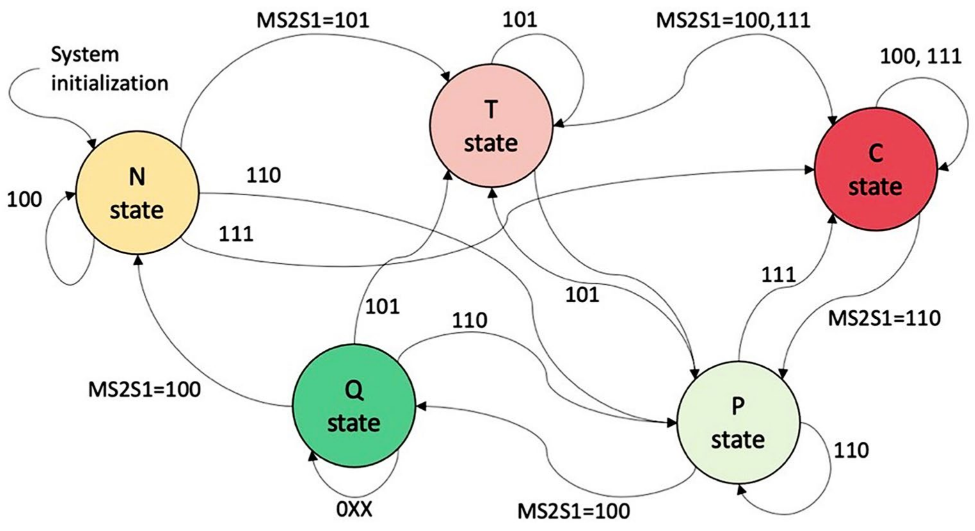


Fig. 7 A high-level state machine diagram for the safety-critical railway interlocking system

machine for the safety-critical railway interlocking system is described based on the model-based design. Given that the basic parameters for the design model as follows: input sensors: S1, S2: [0, 1], Main switch: [0, 1], output gates, gate1, gate2: G1, G2: [0,1], output alarm lights: L1, L2: [0, 1], and output alarm sound, whistle: B: [0, 1]. The three input signals are represented by three Boolean variables "sensor1", "sensor2", and "switch" and the five output signals are represented by three Boolean variables: gate, light alarm and whistle alarm.

The finite state machine (FSM) diagram presented in Fig. 7, for the proposed railway interlocking system has five states that are described as follows:

- *N-state*, defined in NuSMV program as enumeration "train_not_CS" means that the train is not in the safety-critical railway interlocking section.
- *T-state*, defined as enumeration "train_tries_CS" means that the train will enter the safety-critical zone shortly.
- *C-state*, defined as enumeration "train_in_CS" means that the train has successfully arrived the safety-critical zone.
- *P-state*, defined as enumeration "train_away_CS" means that the train will eventually leave the safety-critical section.
- *Q-state*, defined as enumeration "train_out_CS" means that the train has left the safety-critical section.

Referring to Fig. 7, the RIS system state is initially in "train_not_CS" state, when the train arrives at sensor1, then sensor1 will be triggered and the state becomes the "train_tries_CS" state which causes the gates to be closed, the green lights are on, and the train whistle starts working. The state machine does not transit to "train_away_CS" or "train_out_CS" states until the train arrives at sensor2 zone. Otherwise, the state remains in the safety-critical "train_in_CS" state. The flowchart in Fig. 8 illustrates the procedure of how the required safety-critical properties are

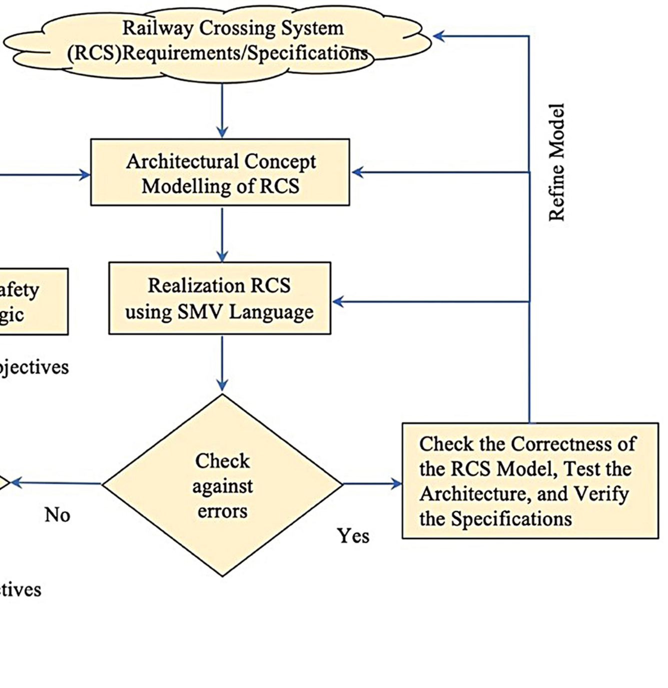

Railway Crossing System (RCS)Requirements/Specifications
Architectural Concept Modelling of RCS
Realization RCS using SMV Language
Modify the Model and Check Safety Properties using Temporal Logic
Refine Model
Falsified Objectives
Checking RCS Model via NuSMV
Check against errors
No
Yes
Check the Correctness of the RCS Model, Test the Architecture, and Verify the Specifications
Valid Objectives
End

Fig. 8 Property proving and verification workflow for safety-critical railway crossing system using the NuSMV model checker

formalized and formally verified. The verification process using NuSMV tool is performed at the first stage, then, modeling the behavior of safety-critical railway interlocking system, which was molded in UML state diagram model. After that, transforming UML state diagram system to SMV code for the railway interlocking and controllable system. The system has been applied to several test cases to prove the correctness of functional and critical properties that have been defined for the digital safety-critical application whether they are valid or falsified. If they are falsified and show some kind of integer overflow, division by zero, dead logic, or array access violations, a test case harness is executed to verify its correct execution.

3.3 Formal Modeling and Verification of LTL/CTL Properties

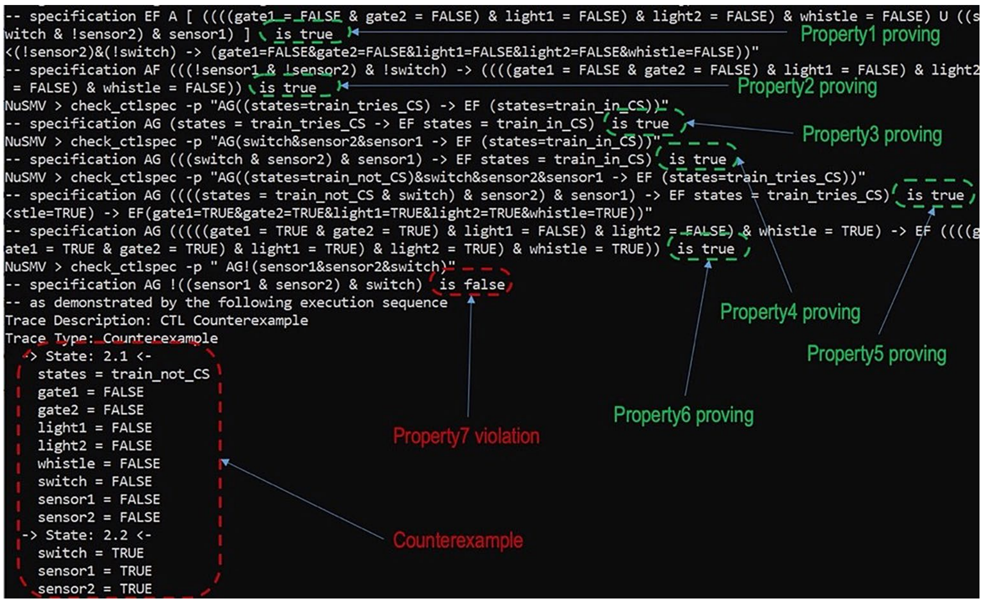
To determine if an embedded digital system is operating correctly or incorrectly, it is necessary to compare the system's design to its requirements. Hence, the specifications are translated, and the functionality execution of the application is simulated. To characterize the digital system's behavior mathematically, formal logic is employed. Following the development of the digital system's mathematical model, several specifications, requirements, and properties are assessed and evaluated to determine whether the system complies with them or not. The formal verification tools for the proposed state machine architectural approach are NuSMV model checker and Quartus software.

The CTL safety critical properties presented in the Table 1 were inserted in the "spec" section of the SMV model for the proposed system. After checking the behavior of the SMV model using the NuSMV tool, one of the properties which was (P.7) was falsified. Consequently, the SMV programing code was modified. The reason is that sensor1 and sensor2 were not dependent on each other and they operated at the same time because the NuSMV tool would try all possibilities. Therefore, the code must be refined by making the sensor1 and sensor2 not operating at the same time by adding "sensor1" and "sensor2" as enumeration definition, Thus, the properties were reformatted. After running the NuSMV tool, all properties were satisfied, and they were matching with the model. Then, using CTL temporal logic, the safety properties were specified and formulated, which have expressed them in Table 1.

Using LTL Formalization: Consider the following English descriptions of safety properties and their corresponding semantic LTL formalizations as an example: Property 8: "*whenever the train is arriving the sensor1, eventually it will enter the critical section*". Assume T denote the condition in which the train is arriving the sensor1, and C denote the condition that the train will enter the critical section. Therefore, this safety property can be formalized in LTL as follows: G (T ==> FC). Property9: "*whenever the sensor1 signal is asserted the railway interlocking state machine should move immediately to the "train-in-CS" state and remain there until the sensor1 signal is de-asserted*". Let T be true when the sensor1 signal is asserted, and C be true when the FSM is transitioning into the critical section.

Table 1 CTL safety properties for the safety-critical railway interlocking system

NO.	CTL property and its SMV translation	Description
Property1	EF(A[gate1 = FALSE&gate2 = FALSE&light1 = FALSE&light2 = FALSE whistle = FALSE U switch&!Sensor2&sensor1])	The gates remain open, green lights are on, and whistle of the train is off until train arrives sensor1 and main switch is on
Property2	AF ((! Sensor1) & (! Sensor2) & (! Switch) -> (gate1 = FALSE&gate2 = FALSE&light1 = FALSE&light2 = FALSE&whistle = FALSE))	Always in all paths, when sensor1 is off and sensor2 is off, then gates are open, alarm lights are off, and whistle is off
Property3	AG((states = train_tries_CS) -> EF (states = train_in_CS))	In all paths, whenever sensor1 and main switch are active at the same time, it will eventually take the FSM into C state
Property4	AG (((switch&sensor2) &sensor1) -> EF states = train_in_CS)	In all paths, whenever sensor1, sensor2, and main switch are active at the same time, it will eventually take the FSM into C state
Property5	AG((states = train_not_CS)&switch&sensor2&sensor1 -> EF (states = train_tries_CS))	In all paths, the FSM machine is in state N and sensor1, sensor2, and main switch are active, it will eventually be taken into a T state
Property6	AG(((((gate1 = TRUE&gate2 = TRUE) &light1 = FALSE) &light2 = FALSE) &whistle = TRUE) -> EF ((((gate1 = TRUE&gate2 = TRUE) &light1 = TRUE) &light2 = TRUE) &whistle = TRUE))	In all paths, the FSM machine is in state T, it will eventually be taken into a C state
Property7	AG! (sensor1 & sensor2 & switch)	The safety property which is important to verify is that sensor1, sensor2, and switch always must never be active at the same time
Property8	G (T ==> F C)	Whenever the train is arriving the sensor1, eventually it will enter the critical zone
Property9	G (T==> X(C U Q))	Whenever the sensor1 signal is asserted, the railway interlocking state machine should move immediately to C state and remain there until the sensor1 signal is de-asserted



```
-- specification EF A [ ((((gate1 = FALSE & gate2 = FALSE) & light1 = FALSE) & light2 = FALSE) & whistle = FALSE) U ((s
witch & !sensor2) & sensor1) ] is true
<(!sensor2)&(!switch) -> (gate1=FALSE&gate2=FALSE&light1=FALSE&light2=FALSE&whistle=FALSE))"
-- specification AF (((!sensor1 & !sensor2) & !switch) -> ((((gate1 = FALSE & gate2 = FALSE) & light1 = FALSE) & light2
 = FALSE) & whistle = FALSE)) is true
NuSMV > check_ctlspec -p "AG((states=train_tries_CS) -> EF (states=train_in_CS))"
-- specification AG (states = train_tries_CS -> EF states = train_in_CS) is true
NuSMV > check_ctlspec -p "AG(switch&sensor2&sensor1 -> EF (states=train_in_CS))"
-- specification AG (((switch & sensor2) & sensor1) -> EF states = train_in_CS) is true
NuSMV > check_ctlspec -p "AG((states=train_not_CS)&switch&sensor2&sensor1 -> EF (states=train_tries_CS))"
-- specification AG ((((states = train_not_CS & switch) & sensor2) & sensor1) -> EF states = train_tries_CS) is true
<stle=TRUE) -> EF(gate1=TRUE&gate2=TRUE&light1=TRUE&light2=TRUE&whistle=TRUE))"
-- specification AG (((((gate1 = TRUE & gate2 = TRUE) & light1 = FALSE) & light2 = FALSE) & whistle = TRUE) -> EF ((((g
ate1 = TRUE & gate2 = TRUE) & light1 = TRUE) & light2 = TRUE) & whistle = TRUE)) is true
NuSMV > check_ctlspec -p " AG!(sensor1&sensor2&switch)"
-- specification AG !((sensor1 & sensor2) & switch) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 2.1 <-
    states = train_not_CS
    gate1 = FALSE
    gate2 = FALSE
    light1 = FALSE
    light2 = FALSE
    whistle = FALSE
    switch = FALSE
    sensor1 = FALSE
    sensor2 = FALSE
  -> State: 2.2 <-
    switch = TRUE
    sensor1 = TRUE
    sensor2 = TRUE
```

Fig. 9 Generated counterexample for the safety property7 violation and proving results for the safety properties: property1 to property6

Then, the English property can be formalized in LTL as: G (T ==> X(CUQ)).

Then, the proving results for the safety-critical properties were obtained as shown in Fig. 9. The results of checking all above properties are true except the P.7. This property means that sensor1 and sensor2 always and globally do not operate one another. But this property is false after checking it on the model, and the reason can be traced by the counterexample. The false has happened because of the sensor1 and sensor2, each one was not dependent on the other and possibly both are true at the same time. Therefore, to satisfy this property the model was refined. The safety property which is important to be verified was "AG! (event = sensor1 & event = sensor2)", which means that sensor1 and sensor2 always must be never both "on" at the same time. After running the model by the NuSMV tool, all properties were satisfied, proved the requirements compliance with simulation results for the safety- critical application and they were matching with the model that contains the results.

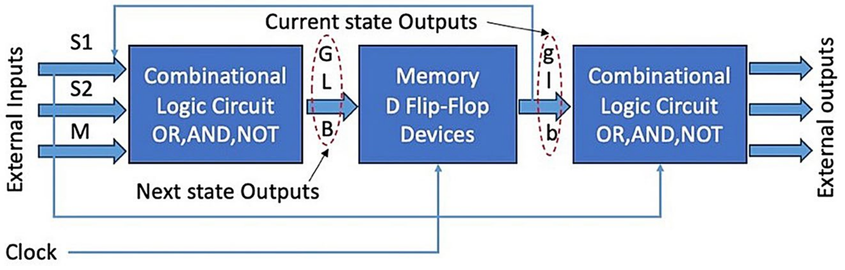
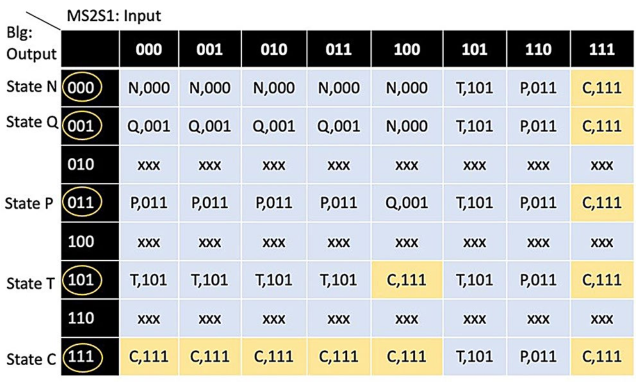

Current state Outputs
S1
S2
M
External Inputs
Combinational Logic Circuit OR,AND,NOT
G
L
B
Memory D Flip-Flop Devices
g
l
b
Combinational Logic Circuit OR,AND,NOT
External outputs
Next state Outputs
Clock

Fig. 10 A high level of Mealy type state machine hardware model of RIS system



Blg: Output \ MS2S1: Input		000	001	010	011	100	101	110	111
State N	000	N,000	N,000	N,000	N,000	N,000	T,101	P,011	C,111
State Q	001	Q,001	Q,001	Q,001	Q,001	N,000	T,101	P,011	C,111
	010	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx
State P	011	P,011	P,011	P,011	P,011	Q,001	T,101	P,011	C,111
	100	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx
State T	101	T,101	T,101	T,101	T,101	C,111	T,101	P,011	C,111
	110	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx
State C	111	C,111	C,111	C,111	C,111	C,111	T,101	P,011	C,111

Fig. 11 Composite Karnaugh map for the realized hardware architecture

3.4 Architecting Principles of the Proposed Fault-Tolerant Hardware Architecture

The fault-tolerant hardware architecture is designed to avoid CPS system failure in the presence of human-made and natural hardware faults that are assumed to be continuous in time or bounded in its interval [48]. Figure 10 presents a high level of the proposed CPS system structure which has been built using the concept of mealy type synchronous state machine (SM). The respective values of the input and output signals for the composite Karnaugh map of the realized hardware architecture are presented in Fig. 11.

The next state equations of the clocked synchronous state machine depend on the current state of the machine and on the current inputs. The next state equations obtained from the composite Karnaugh map shown in Fig. 10, are presented in Eq. 1, Eq. 2, and Eq. 3.

$$\begin{aligned} \mathrm{G} = {} & \bar{b}\,\bar{l}\,MS_2 + \bar{b}\,\bar{l}\,MS_1 + g\,\bar{M} + gS_2 \\ & + \bar{b}\,lg\,\bar{S}_2 + bg\,\bar{S}_2 \end{aligned} \tag{1}$$

$$\begin{aligned} \mathrm{L} = {} & \bar{b}\,\bar{l}\,MS_2 + b\,\bar{l}\,gMS_2 + bg\mathrm{M}\,\bar{S}_2\,S_1 \\ & + blg\,\bar{S}_2\,\bar{S}_1 + lgS_2 + lg\,\bar{\mathrm{M}} \end{aligned} \tag{2}$$

$$\begin{aligned} \mathrm{B} = {} & \bar{b}\,\bar{l}\,MS_1 + gMS_1 + bg\,\bar{\mathrm{M}}\,S_2 \\ & + bgS_1 + bg\,\bar{S}_2 \end{aligned} \tag{3}$$

The fault-tolerant RIS safety-critical system was hosted on the proposed FPGA state-machine hardware architecture. Referring to Fig. 12, the hardware architecture is comprised

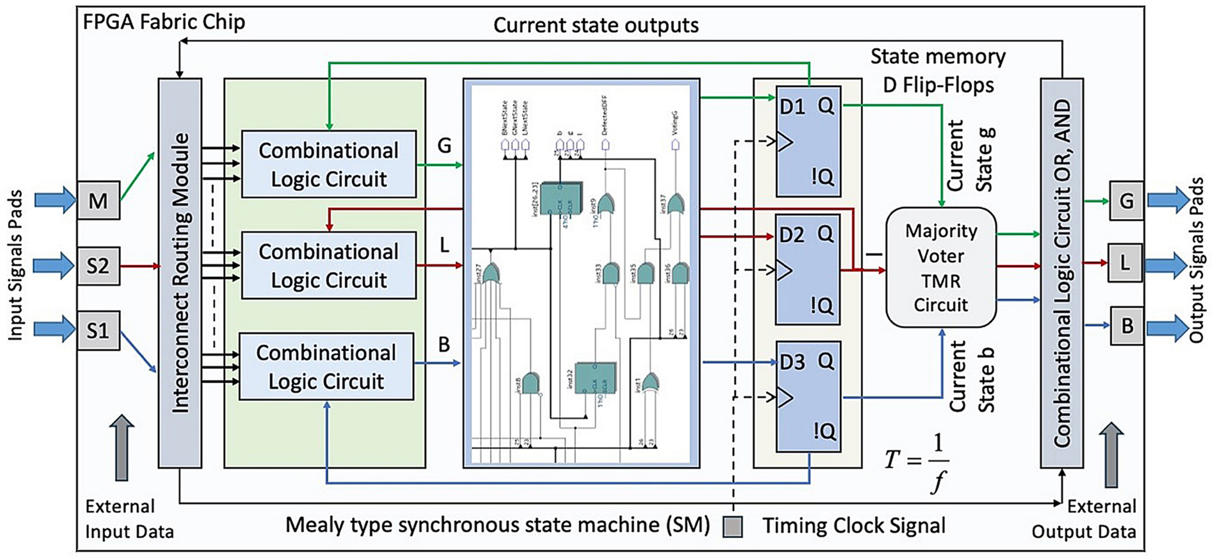

FPGA Fabric Chip
Current state outputs
State memory D Flip-Flops
Input Signals Pads
M
S2
S1
Interconnect Routing Module
Combinational Logic Circuit
Combinational Logic Circuit
Combinational Logic Circuit
G
L
B
D1 Q !Q
D2 Q !Q
D3 Q !Q
Current State g
Current State b
Majority Voter TMR Circuit
Combinational Logic Circuit OR, AND
G
L
B
Output Signals Pads
$T = \frac{1}{f}$
External Input Data
Mealy type synchronous state machine (SM)
Timing Clock Signal
External Output Data

Fig. 12 The implemented fault-tolerant hardware architecture using Mealy type state machine

Table 2 State description of the Markov reliability model in fig. E

State number (0-F)	Description of state status
0	All combinational and sequential logic circuits are working correctly (RIS system is fully operational).
1	One of the biostable memory devices (D Flip-Flop) is affected by a permanent fault. The failure effect is tolerated using a majority voting circuit (RIS system is failed-operational.
F	Two memory devices are (D Flip-Flop) are shutdown and affected by permanent faults or three D Flip-Flops are down and defected by faults (RIS system is failed).

of three basic digital logic circuits: the combinational logic circuit, synchronous Mealy machine sequential logic circuit, and a majority voting logic circuit that uses the concept of triple modular redundancy in order to mask the effect of failed D Flip-Flops. Flip flops are clocked binary storage components that store either 0 or 1. The output value of flip flop device changes based on the transition event of the clock. The internal structure of the flip flop is prone to the effects of transient and permanent faults. In the designed architecture, combinational logic circuit represented by a network of logic gates: AND, OR, and NOT receives the input data of sensor1, sensor2, and main switch for the railway RIS system through three input signals pads. Furthermore, the synchronous Mealy state machine circuit operates on the input data and provides output data results in the output signals pads.

4 Results and Discussion

The discussion of the results section is divided into two parts. The first part explains the effect of different values of failure rates on the reliability calculation using Markov chain analysis for the proposed RIS architecture. The other part shows the FPGA-based timing simulation results related to fault-tolerance and verification in terms of injecting faults into D Flip Flop memory devices and transitioning among the operational states.

4.1 Effect of Varied Failure rate on the Reliability

It is important to analyze the correct behavior of the safety-critical system and predict the failure occurrence that can defect software and hardware critical components [49]. To provide sufficient quantitative data and assess the reliable behavior of the proposed Mealy-based state machine hardware architecture, a Markov reliability model [50], consisting of three states is presented in Fig. 12. and the states description are illustrated in Table 2. The reliability model includes three states: fully operational RIS state, failed operational system state, or the system failed state,

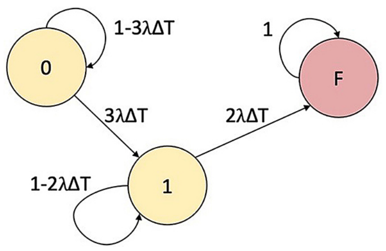


Fig. 13 State transition and Markov reliability diagram for the triple modular redundancy system

Table 3 Effect of failure rate on reliability

Time in Minutes (t)	λ 1 = 0.1 f/hr	λ 2 = 0.2 f/hr	λ 3 = 0.01 f/hr	λ 4 = 0.001 f/hr
1	0.99999169	0.99335551	0.99999992	1
2	0.99996685	0.99986814	0.99999968	1
3	0.99992562	0.99970495	0.99999924	0.99999999
4	0.99986814	0.99947837	0.99999868	0.9996667
5	0.99979454	0.99918945	0.99999792	0.99999998
10	0.99918945	0.99684612	0.99999169	0.99999992
20	0.99684613	0.98805845	0.99996685	0.99999968
30	0.99309835	0.974555582	0.99992562	0.99999925
40	0.98805845	0.95714497	0.99986765	0.99999867
50	0.9818436	0.93653293	0.99979454	0.99999792
60	0.97455582	0.91333687	0.99970495	0.99999709

that have been assumed in the evaluation of the proposed system. State 0, which is described as fully operation RIS systems; represent the case where all the combinational and sequential logic circuits of the digital system are working correctly. State 1 which is called failed-operational system, represents the RIS system in which one of the biostable memory devices D Flip-Flop is affected by a permanent fault and the failure effect is tolerated using a majority voting circuit. State F, which is called system failed represents the system in which two memory D- Flip-Flop devices are shutdown and affected by permanent faults or three D Flip-Flop devices are down and defected by faults. The reliable behavior of the RIS system was modeled based on the Markov chain model presented in Fig. 13.

The results of the reliability analysis for the state machine hardware architecture over 60 min are shown in Table 3 and in Fig. 14. There are some important observations that can be seen based on the data presented. In Fig. 13, it can be observed that the reliability R(t) level decreases as both the time of testing in minutes and the failure rate are increasing. In addition, various values of the failure rate (λ) were assumed and the reliability was calculated for each value at different times ranging from 1 min to 60 min. As can be seen in Fig. 13, the reliability calculated for the two failure rates λ 3 = 0.01 f/hr, and λ 4 = 0.001 f/hr does not decrease significantly over time and remains within a high reliability level ranging between 0.99 and 1. However, in both cases, λ

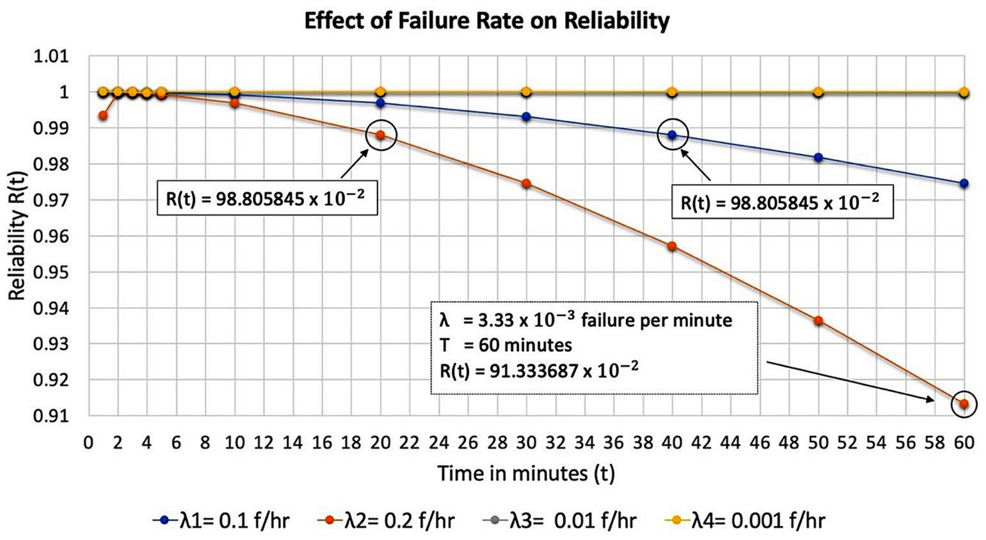


Fig. 14 Impact in the reliability of the different failure rates for the safety-critical railway interlocking system

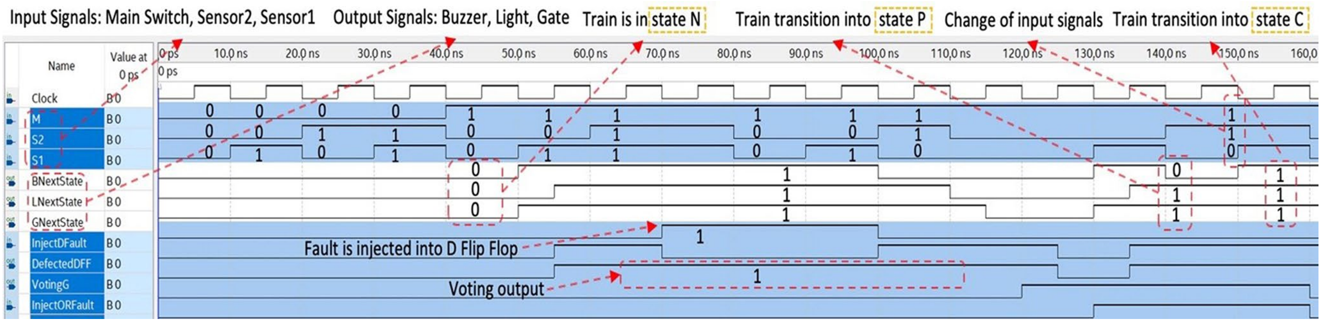

Input Signals: Main Switch, Sensor2, Sensor1
Output Signals: Buzzer, Light, Gate
Train is in state N
Train transition into state P
Change of input signals
Train transition into state C
Name
Value at 0 ps
Clock
M
S2
S1
BNextState
LNextState
GNextState
InjectDFault
DefectedDFF
VotingG
InjectORFault
Fault is injected into D Flip Flop
Voting output

Fig. 15 Timing simulation waveform of injecting a fault into D Flip Flop of the railway interlocking safety-critical CPS architecture

1 = 0.1 f/hr and λ 2 = 0.2 f/hr, the level of reliability dropped clearly below the level of 0.99, started from the two cases at the times, T = 20 min, and T = 40 min. The same levels of reliability were obtained in these two cases despite the difference in time and failure rate. The lowest reliability rate R(t) = 91.333687 x 10^{-2} obtained is in the case, λ 2 = 0.2 f/hr, T = 60 min which is shown in Fig. 14. As expected, the reliability improvements increase for lower failure rates parameters when three triplicated modules are independent and identical, R(t) to be the reliability of the triple modular redundancy (TMR) system, λ to be the constant failure rate. The reliability of the triple modular redundancy for the proposed hardware architecture is calculated as described in Eq. 4.

$$\mathrm{R(t)} = 3e^{-2\lambda T} - 2e^{-3\lambda T} \tag{4}$$

4.2 Simulation and Implementation Results Using FPGA-Based Quartus Prime Software

To demonstrate the correct operation of the proposed state machine hardware architecture, Quartus Prime Software version 17.1 from Altera has been running as an analysis and synthesis FPGA tool to realize the highly dependable digital system. The proposed architecture has been synthesized in the digital platform the Altera Max V FPGA programmable device 5M40ZE64C4 chip with total logic elements 33% and frequency as 100 MHz. Figure 15 presents the timing simulation results for the railway interlocking safety-critical CPS case study.The results demonstrates that the three next state outputs signals “BNextState”, “LNextState”, “GNextState” representing the code of the output signals: Buzzer, Light, Gate of the Mealy type of stat machine, can be changed over time based on the different values of the input sensors and the main switch “M”, “S2”, “S1”. The architecture generates the following codes: “000” representing the train is in state N, “011” train in state P, “111” train in state C, which are presented in Fig. 7 highlighting the design model. Furthermore, the internal structure of the proposed architecture is designed in a fault-tolerant method by using majority voting and comparator logic circuits which is capable of masking different types of faults. The internal structure is comprised of combinational and sequential logic circuits, flip-flops-based memory elements, and basic logic gates. In Fig. 15, whenever a one stuck at ‘1’ fault is injected at time

Table 4 Comparative analysis results between the proposed architecture and the standard method

	Standard Circuit	Proposed HW architecture	Area Overhead
Number of Flip-Flops	3 (D Flip-Flops)	5 (D Flip-Flops)	66.6667%
Number of Logic Gates	25 (20 AND gates, 5 OR gates)	33 (20 AND gates, 5 OR gates, 2 XOR gates, 4 AND gates, 2 OR gates)	32%
Number of Input/Output Pads	7	11	57.1429%

70 ns into a D1 flip-flop of the fault-tolerant hardware architecture presented in Fig. 12, fault is detected through the majority voter TMR circuit at time 70 ns by deserting the "DefectedDFF" signal value, the voting output "VotingG" which represents the voting output signal remains 1, and the failure effect on the RIS system is masked without defecting the correct value of the g output signal the next state output values which are "GnextState", "LnextState", and "BnextState" for the proposed state machine hardware architecture.

Table 4. shows the comparative analysis results between the proposed hardware architecture and the standard method. The design of the architecture is cost efficient in terms of the hardware area overhead. The hardware area overhead for the flip-flops, logic gates, and input/output pads are calculated based on the Eq. 5, which are illustrated in Table 4. The total area overhead the three hardware components, flip-flops, logic gates, and input/output pads is equal to 52%.

$$\text{Area Overhead} = \frac{Number\ of\ additional\ Hardware\ resources}{Number\ of\ statndard\ hardware\ resources} \times 100\% \tag{5}$$

5 Conclusion and Future Work

Accurate requirements and specifications for safety-critical CPS systems property proving or violations is a crucial stage in the realization of dependable systems. Using some modern design and synthesis tools, e.g., the NuSMV model checker tool and Altera Quartus Prime Software to formally prove the correct operation of critical properties and detect faults in real-time is an essential requirement in the implementation of highly reliable CPS systems. NuSMV model checker was used in the modeling of a railway interlocking system as a case study. The checker used temporal logic language to formulate properties and model system behavior in SMV language and it is known for its accuracy in the verification process because it uses temporal logic LTL/CTL visually. Though the complexity of temporal logic and tracing the counterexample is more difficult than Simulink design verifier, the verification results were highly accurate. The impact of various values of failure rate on the reliability of the system using markov chain analysis was presented. In addition, the proposed architecture was synthesized in the digital platform the Altera Max V FPGA programmable chip with total logic elements 33% and frequency as 100 MHz. It can be concluded that it is better to use the capabilities of the NuSMV model checker to detect generated counterexamples for the violated safety-critical properties before the hardware implementation of fault-tolerant digital systems.

Future works will focus on run time verification, self-checking fault detection, and fault tolerance of other types of critical properties for the proposed safety critical digital design that require higher levels of dependability. Other examples of the modern CPS systems will be utilized in the design and implementation of the proposed state machine hardware architecture, e.g., Xilinx reconfigurable field-programmable gate array (FPGA) fabric or Zynq-7000 All programmable system on chip (SoC) platform that can behave in a fault-tolerant, verifiable, and resilient manner.

Funding The author conforms that this research work has not received any funds, grants, or other support.

Data Availability The data sets obtained during the current experimental research study are available from the author upon reasonable request.

Declarations

Conflict of Interest The author declares that he neither has competing interests nor relevant financial or non-financial conflict interests to disclose.

References

1. Yaacoub J-PA, Salman O, Noura HN et al (2020) Cyber-physical systems security: limitations, issues and future trends. Microprocess Microsyst 77:103201. https://doi.org/10.1016/j.micpro.2020.103201
2. Knight J (2012) Fundamentals of dependable computing for software engineers. CRC, Boca Raton
3. Khan AH, Khan ZH, Weiguo Z (2014) Model-Based Verification and Validation of Safety-Critical Embedded Real-Time Systems: Formation and Tools. In: Proc. Embedded and Real Time System Development: A Software Engineering Perspective: Concepts, Methods and Principles. Springer, Berlin, Heidelberg, pp 153–183
4. Khairullah SS, Elks CR (2020) Self-repairing hardware architecture for safety-critical cyber-physical-systems. IET Cyber-Physical Systems: Theory Appl 5:92–99. https://doi.org/10.1049/iet-cps.2019.0022
5. Khairullah SS, Elks CR (2018) A Bio-inspired, Self-Healing, resilient Architecture for Digital Instrumentation and Control Systems and embedded devices. Nucl Technol 202:141–152. https://doi.org/10.1080/00295450.2018.1450014

6. Mallavalli S, Fekih A (2017) A fault tolerant control approach for a quadrotor UAV subject to time varying disturbances and actuator faults. In: Proc. 2017 IEEE Conference on Control Technology and Applications (CCTA). pp 596–601
7. Cai S, He B, Wang W et al (2020) Soft Error Reliability Evaluation of Nanoscale Logic Circuits in the Presence of multiple transient faults. J Electron Test 36:469–483. https://doi.org/10.1007/s10836-020-05898-x
8. Ali AT, Alneema DAF (2020) Design analysis of turbo decoder based on one MAP decoder using high level synthesis tool. Al-Rafidain Eng J (AREJ) 25:70–77. https://doi.org/10.33899/rengj.2020.126801.1022
9. Ibrahim H, Azmi H, El-Kharashi MW, Safar M (2024) Non-invasive hardware trojans modeling and insertion: a formal Verification Approach. J Electron Test 40:117–135. https://doi.org/10.1007/s10836-024-06100-2
10. Luteberget B, Johansen C (2018) Efficient verification of railway infrastructure designs against standard regulations. Formal Methods Syst Des 52:1–32. https://doi.org/10.1007/s10703-017-0281-z
11. Edward A, Lee (2017) Introduction to embedded systems: a cyber-physical systems approach, Second edition. The MIT Press, Cambridge, Massachusetts
12. Kumar P, Singh LK, Kumar C (2021) Model Based Verification of Safety-Critical Systems. In: Proc. 2021 2nd International Conference for Emerging Technology (INCET). IEEE, Belagavi, India, pp 1–9
13. Bolbot V, Theotokatos G, Bujorianu LM et al (2019) Vulnerabilities and safety assurance methods in Cyber-physical systems: a comprehensive review. Reliab Eng Syst Saf 182:179–193. https://doi.org/10.1016/j.ress.2018.09.004
14. Bartocci E, Falcone Y, Francalanza A, Reger G (2018) Introduction to Runtime Verification. In: Proc. Lectures on Runtime Verification. Springer International Publishing, Cham, pp 1–33
15. Grobelna I (2020) Formal Verification of Control Modules in Cyber-physical systems. Sensors 20:5154. https://doi.org/10.3390/s20185154
16. Alobaidy A, Abdul-Jabbar MA, Al-khayyt DJM SZ (2020) Faults diagnosis in robot systems: a review. Al-Rafidain Eng J (AREJ) 25:164–175. https://doi.org/10.33899/rengj.2020.127782.1051
17. Seceleanu C, Johansson M, Suryadevara J et al (2017) Analyzing a wind turbine system: from simulation to formal verification. Sci Comput Program 133:216–242. https://doi.org/10.1016/j.scico.2016.09.007
18. Bennion M, Habli I (2014) A candid industrial evaluation of formal software verification using model checking. In: Proc. Companion Proceedings of the 36th International Conference on Software Engineering. ACM, pp 175–184
19. Ljungkrantz O, Akesson K, Fabian M, Chengyin Y (2010) Formal specification and Verification of Industrial Control Logic Components. IEEE Trans Autom Sci Eng 7:538–548. https://doi.org/10.1109/TASE.2009.2031095
20. Pace GJ (2012) Classifying Relations. In: Proc. Pace GJ (ed) Mathematics of Discrete Structures for Computer Science. Springer, Berlin, Heidelberg, pp 141–155
21. Kang E-Y, Mu D, Huang L, Lan Q (2017) Verification and Validation of a Cyber-Physical System in the Automotive Domain. In: Proc. 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, Prague, Czech Republic, pp 326–333
22. Model-driven architecture based security analysis - Mili –2021 - Systems Engineering - Wiley Online Library. https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/sys.21581. Accessed 20 Jan 2024
23. Haqiq A, Bounabat B (2013) Verification of multi decisional reactive agent using SMV model checker. In: Proc. 2013 8th IEEE Design and Test Symposium. pp 1–6
24. Cimatti A, Pistore M, Roveri M, Traverso P (2003) Weak, strong, and strong cyclic planning via symbolic model checking. Artif Intell 147:35–84. https://doi.org/10.1016/S0004-3702(02)00374-0
25. Singh LK, Rajput H (2018) Dependability analysis of Safety critical real-Time systems by using Petri nets. IEEE Trans Control Syst Technol 26:415–426. https://doi.org/10.1109/TCST.2017.2669147
26. Utting M, Pretschner A, Legeard B (2012) A taxonomy of model-based testing approaches. Softw Test Verification Reliab 22:297–312. https://doi.org/10.1002/stvr.456
27. Li Z, Wang Y, Huang Z et al (2024) Ada-FA: a Comprehensive Framework for Adaptive Fault Tolerance and Ageing Mitigation in FPGAs. IEEE Internet Things J 11:17688–17699. https://doi.org/10.1109/JIOT.2024.3361036
28. Suryadevara J, Sapienza G, Seceleanu C et al (2014) Wind Turbine System: An Industrial Case Study in Formal Modeling and Verification. In: Proc. Artho C, Ölveczky PC (eds) Formal Techniques for Safety-Critical Systems. Springer International Publishing, Cham, pp 229–245
29. Kastensmidt FL (2006) Fault-tolerance techniques for SRAM-based FPGAs. Springer, Dordrecht
30. Wilkening M, Sridharan V, Li S et al (2014) Calculating Architectural Vulnerability Factors for Spatial Multi-Bit Transient Faults. In: Proc. 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture. pp 293–305
31. Pakonen A, Buzhinsky I (2019) Verification of fault-tolerant safety I&C systems using model checking. In: Proc. 2019 IEEE International Conference on Industrial Technology (ICIT). IEEE, Melbourne, Australia, pp 969–974
32. Saleh HA, Salim TM (2022) Design and implementation of model predictive controller. Al-Rafidain Eng J (AREJ) 27:219–230. https://doi.org/10.33899/rengj.2022.130477.1108
33. Hsiung P-A, Lin Y-H (2005) Modeling and Verification of Safety-Critical Systems Using Safecharts. In: Proc. Wang F (ed) Formal Techniques for Networked and Distributed Systems - FORTE 2005. Springer, Berlin, Heidelberg, pp 290–304
34. Cimatti A, Corvino R, Lazzaro A et al (2012) Formal Verification and Validation of ERTMS Industrial Railway Train Spacing System. In: Proc. Madhusudan P, Seshia SA (eds) Computer Aided Verification. Springer, Berlin, Heidelberg, pp 378–393
35. (2019) Protecting Against Common Cause Failures in Digital I&C Systems of Nuclear Power Plants. https://www.iaea.org/publications/8151/protecting-against-common-cause-failures-in-digital-ic-systems-of-nuclear-power-plants. Accessed 10 Sep 2019
36. Avizienis A, Laprie JC, Randell B, Landwehr C (2004) Basic concepts and taxonomy of dependable and secure computing. IEEE Trans Dependable Secur Comput 1:11–33. https://doi.org/10.1109/TDSC.2004.2
37. Khairullah SS, Qassabbashi FN, Kareem JA (2024) Design and analysis of fault-tolerant sequential logic circuits for safety-critical applications. Bull Electr Eng Inf 13:413–421. https://doi.org/10.11591/eei.v13i1.5713
38. Grobelna I, Szcześniak P (2022) Model checking Autonomous components within Electric Power Systems specified by Interpreted Petri Nets. Sensors 22:6936. https://doi.org/10.3390/s22186936
39. Murugesan A, Heimdahl MPE, Whalen MW et al (2017) From Requirements to Code: Model Based Development of a Medical Cyber Physical System. In: Proc. Huhn M, Williams L (eds) Software Engineering in Health Care. Springer International Publishing, Cham, pp 96–112

40. Grobelny M, Grobelna I, Adamski M (2012) Hardware behavioural modelling, Verification and Synthesis with UML 2.x activity diagrams. IFAC Proc Vol 45:134–139. https://doi.org/10.3182/20120523-3-CZ-3015.00028
41. Askari Hemmat MH, Mohamed OA, Boukadoum M (2015) Formal modeling, verification and implementation of a train control system. In: Proc. 2015 27th International Conference on Microelectronics (ICM). IEEE, Casablanca, Morocco, pp 134–137
42. Khairullah SS, Sharkawy A-N (2022) Design and implementation of a Reliable and Secure Controller for Smart Home Applications based on PLC. J Rob Control JRC 3:614–621. https://doi.org/10.18196/jrc.v3i5.15972
43. Ma W, Hei X (2012) An research for formal Verification of Safety-critical Software. Atlantis, pp 836–839
44. Grobelna I (2018) Model checking of reconfigurable FPGA modules specified by Petri nets. J Syst Architect 89:1–9. https://doi.org/10.1016/j.sysarc.2018.06.005
45. Shreya V, Nanda M (2016) Analysing MTL properties using NuSMV model checker. In: Proc. 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, Bangalore, India, pp 817–820
46. Busard S, Cappart Q, Limbrée C et al (2015) Verification of railway interlocking systems. Electron Proc Theoretical Comput Sci 184:19–31. https://doi.org/10.4204/EPTCS.184.2
47. Liebel G, Marko N, Tichy M et al (2018) Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. Softw Syst Model 17:91–113. https://doi.org/10.1007/s10270-016-0523-3
48. Khairullah SS, Mostafa AA (2020) Reliability and safety modeling of a digital feed-water control system. J Univ Babylon Pure Appl Sci 28
49. Chen L, Jiao J, Wei Q, Zhao T (2017) An improved formal failure analysis approach for safety-critical system based on MBSA. Eng Fail Anal 82:713–725. https://doi.org/10.1016/j.engfailanal.2017.06.034
50. Zhou Y, Lin C, Liu Y, Xu H (2018) Analytical Study on the reliability of Redundancy Architecture for Flight Control Computer based on homogeneous Markov process. IEEE Access 6:18290–18298. https://doi.org/10.1109/ACCESS.2018.2812819

Shawkat S. Khaiullah received the B.Sc. and M.Sc. degrees in computer engineering from University of Mosul, Mosul, Iraq, in 2006 and 2011 respectively, and the Ph.D. degree in Computer Engineering from Virginia Commonwealth University, Richmond, Virginia, USA in 2018. He is currently an Assistant Professor of computer engineering at the University of Mosul. He has published more than 11 articles in refereed international journals and conferences. His research interests include computer architecture and dependable systems, FPGA-based fault tolerant cyber-physical systems, and biologically inspired self-healing system design.