Symbiotic Controller Design Using a Memory-Based FSM Model

Su-Fu Kuo and Cheng-Wen Wu
Department of Electrical Engineering
National Tsing Hua University
Hsinchu 30013, TAIWAN

Abstract—The main obstacles in promoting IOT applications are cost and energy consumption constraints of the devices and systems. As a step forward in improving the reliability and reducing the cost and energy consumption of IOT devices and systems, we had proposed previously a high-level model for efficient design-space exploration, which is called the *symbiotic system* (SS) model. In this work, as an example for SS-based design, we show that it is possible to develop symbiotic controllers that achieve higher reliability, thus longer lifetime and lower cost. The proposed approach target FSM-based controllers that are key components in almost all digital systems. The experimental result for a small traffic light controller shows that the reliability is improved by about 1.75 times, assuming only hard faults. If the system encounters more soft errors, the lifetime can be greatly extended, e.g., for 50% soft errors, the reliability improvement is about 8.4 times.

Keywords — error detection and correction (EDAC), built-in self-repair (BISR), on-line testing, resilient system, symbiotic computing, symbiotic system, fault tolerance

I. Introduction

The *internet of things* (IOT) has long been expected as the driver for many new industries, but there are grand challenges yet to be solved. The first grand challenge is cost. As the global GDP in the foreseeable years to come will remain quite flat [1], it will not be reasonable to expect a dramatic jump in IOT-related markets, without sacrificing others. Energy is another grand challenge. It has been a great concern that the entire world is over-using energy, and the world energy supply is not likely to have significant growth in the next few decades, either [2]. Therefore, the growth of IOT will be under the constraints of world economy scale and total energy supply.

The *symbiotic system* (SS) model was proposed to model and design reliable and cost-effective IOT devices and systems [3-5]. An SS is a twin system, consisting of the *primary (functional) system* and *secondary (test) system*. There is a relationship between the mutually dependent primary and secondary systems, where part of the inputs of one system come from the outputs of the other system, and vice versa. The relationship is known as *symbiotic relationship* (SR) [3]. Based on existing technology, the secondary (test) system can be an on-line test and diagnosis scheme, and changing the state of the primary system by the secondary system is considered a healing or repair mechanism. Results from [4] show that when the failure rate (λ) is, e.g., $4{\cdot}10^{-6}$, the SS with 20% repair rate can maintain the reliability at 95% or higher, after 28,000 hours of operation. However, the reliability of the original functional system (without repair) drops to 29% after the same amount of operation time. Therefore, by the SS model, it is possible to design devices with higher reliability and lifetime, thus lower cost.

In this work, we target digital controllers modeled by finite-state-machine (FSM), which are simply called the *controllers*. We use the FSM model to illustrate the states and state transitions of a controller. For ease of discussion, we assume that the controller's outputs are its states, though in reality there may be a simple mapping between the states and outputs. In a *highly reliable system*, the system will always remain in states that are all clearly specified and safe, i.e., the system will not enter unspecified states that can result in hazardous system failure. We denote a state that is specified as a *legal state*, and a state that is unspecified as an *illegal state*. In normal operation, the controller should always operate in legal states and produce legal outputs. An ordinary controller only processes the functional part of the system (the *primary system* [3-5]), i.e., it is designed to meet the functional specifications only. To build a low-cost and high-performance controller, *logic optimization* [6-8] is normally performed during the synthesis stage in the design flow, which exploits the design space, taking advantage of don't-care (X) terms, assuming that the don't-cares will not affect the system's functional behavior. This is true only when the system is fault free in its normal operation. However, logic optimization only guarantees a better hardware as specified. When faults occur, the system may reach illegal states, possibly causing hazards. Hence, to build a highly reliable system with low cost, the symbiotic system model can be applied, where the *secondary (test) system* needs to be designed concurrently with the primary (functional) system [3-5].

To address part of the grand challenges, we propose a memory-based approach for designing *symbiotic controllers*, hoping to help build reliable and cost-effective devices in the future. We use the memory model for symbiotic controller design, since memory can be used as a look-up table that contains the truth-table of the digital design [9]. With the memory model, the *error detection and correction* (EDAC) mechanism can be applied to memory for reliability improvement [10-12]. Based on the memory model, we propose a symbiotic controller architecture corresponding to the memory model, i.e., the EDAC mechanism for the memory

978-1-5386-3705-0/18/$31.00 ©2018 IEEE

model helps us correct the faults in the controller output. We also propose an illegal state detection function to check whether the current output state of the system is legal or not. If the output state is illegal, the remap mechanism will remap the state back to a pre-defined legal state for guaranteeing safe control outputs and system operation.

Experimental result for a small traffic light controller shows that the reliability is improved by 1.75 times, assuming only hard faults. If the system encounters more soft errors, the lifetime can be greatly extended. If the ratio of soft errors among all errors and faults is 50%, simulation results show that the reliability improvement for our approach is 8.4 times. For the small symbiotic traffic light controller, the area overhead is about 57%, including the parity storage of check bit mechanism. Note that the hardware overhead decreases as we apply the approach to larger controllers, i.e., those with more variables.

II. Memory-Based Model

We use a finite state machine (FSM) to describe the states and state transitions of the system. The states that operate correctly and produce safe control signals are called *legal states*, and those that produce incorrect (unspecified) controller outputs are *illegal states*.

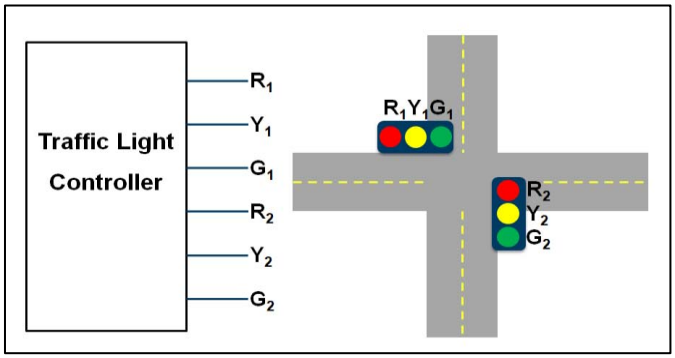

Fig. 1. Outputs of the traffic light controller.

Figure 1 shows the outputs of a simple traffic light controller. Assume we are dealing with a simple crossroad, i.e., a crossroad of vertical and horizontal roads, as shown in the figure. In each direction there are three traffic lights, denoted as R_x, Y_x, and G_x. The subscript x can be 1 or 2. Subscript 1 denotes the traffic lights for the vertical direction, and subscript 2 denotes the traffic lights for the horizontal direction. For ease of discussion, a logical value True indicates that the corresponding traffic light is ON, and a logical value False indicates the light is OFF. Apparently, for the traffic lights on the vertical road, we may regard "$R_1Y_1G_1$"=100 as the *red light* (R), "010" as the *yellow light* (Y), and "001" as the *green light* (G). Since there are two directions in a single traffic light system, we specify the state of all the six traffic lights from the two directions together.

We assume that the controller's outputs are its states. For a traffic light controller with 6 outputs, there are 2^6 possible output combinations, or states. Among all the possible states, there are only five legal states, i.e., R_1R_2, R_1G_2, R_1Y_2, G_1R_2, and Y_1R_2. In normal operation, they always follow the state sequence: $R_1R_2 \rightarrow R_1G_2 \rightarrow R_1Y_2 \rightarrow R_1R_2 \rightarrow G_1R_2 \rightarrow Y_1R_2 \rightarrow R_1R_2$, and repeat indefinitely, as shown in Fig 2. The rest of the states are illegal states since they may produce erroneous or even hazardous signals, such as those shown at the bottom of the figure. The total number of illegal states for the traffic light controller is $2^6 - 5 = 59$.

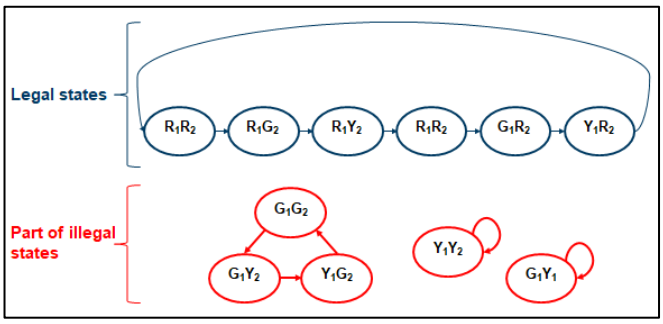

Fig. 2. State diagram of traffic light controller.

In normal operation, the controller always operates in legal states. However, if any fault occurs, as shown in Fig. 3, the system may reach an illegal state, producing hazardous signals. In the example as shown in the figure, the actual faulty transition is $R_1G_2 \rightarrow Y_1Y_2$, while the fault-free transition should be $R_1G_2 \rightarrow R_1Y_2$. This is the situation we must avoid.

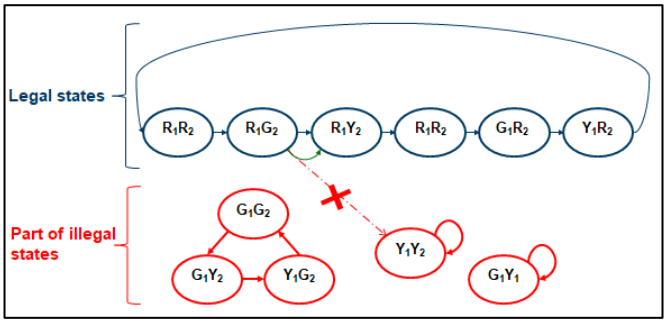

Fig. 3. Illegal state transitions of the traffic light controller.

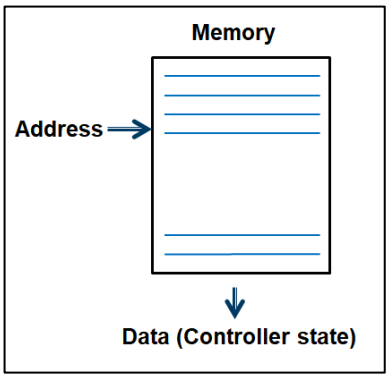

Fig. 4. Memory-based model.

Figure 4 shows a memory that models the hardware controller. The memory stores the controller states (outputs) as its data, and a data word will become the address (pointer) of the next state, providing a sequence of accesses to the memory. As the controller outputs are the data stored in the memory, the error correction capability for the memory can effectively correct the controller state (output) as well. Therefore, with the

memory model, we may apply and reuse the EDAC mechanism for reliability improvement of the controller.

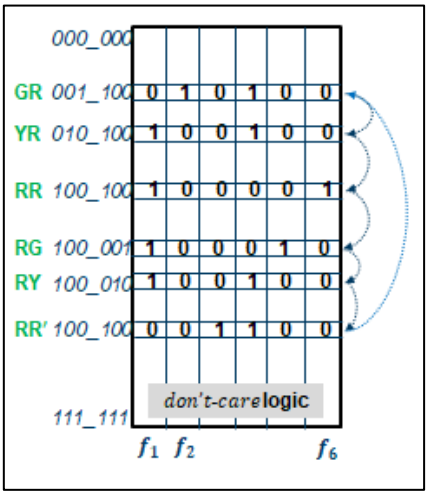

Fig. 5. Traffic light controller modeled by memory.

The traffic light controller as modeled by memory is shown in Fig. 5. There are no inputs in this example, and the outputs are directly taken from the state bits. The finite state machine (FSM) for the traffic light controller is quite simple. Again, in normal operation, it always follow the state transition sequence: $R_1R_2 \rightarrow R_1G_2 \rightarrow R_1Y_2 \rightarrow R_1R_2 \rightarrow G_1R_2 \rightarrow Y_1R_2 \rightarrow R_1R_2$, and repeat indefinitely. Note the correspondence between the state diagram in Fig. 2 and the memory model. The output states correspond to a word in the memory, and the state transitions correspond to a sequence of memory accesses. There are 5 legal states in the state diagram, except that the state RR is duplicated to simplify implementation.

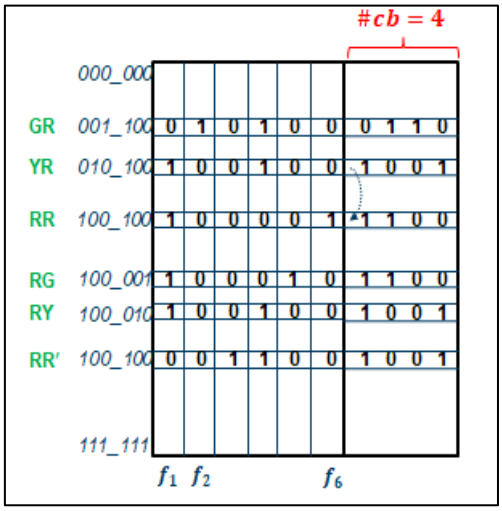

Fig. 6. Memory model with EDAC for traffic light controller.

When a fault occurs and the system reaches an illegal state, in the memory model it means we are accessing a word with don't-cares (X's). Note that in reality, when designing a digital controller, the synthesis tool is free to assign any X to 0 or 1, normally for optimization purposes. In such case, a transition to the unspecified region can be hazardous since the output state is illegal.

III. Memory-Based Symbiotic Controller Design

Our memory-based approach for coping with the reliability issue of the controller is composed of: 1) check bit correction, 2) illegal state detection, and 3) remapping logic for error correction, which will be described below.

A. Check Bit Correction

It is important to protect the system from illegal states, because an illegal state can be hazardous due to incorrect output control signals. In this work, we apply the idea of error detection and correction (EDAC) to the memory model to detect and correct the fault.

In Fig. 6, we add check bits to the traffic light controller memory model. In this example, the number of output bits are 6, so we need 4 check bits to fully correct any single bit error using the Hamming code. We can use more check bits to achieve a higher correction capability, e.g., in Fig. 7, we add an overall parity bit *p*, so that the *Hamming distance* (or *minimum distance*) of the extended Hamming code becomes 4, i.e., $d_m = 4$, which is a *single-error correction and double-error detection* (SEC-DED) code.

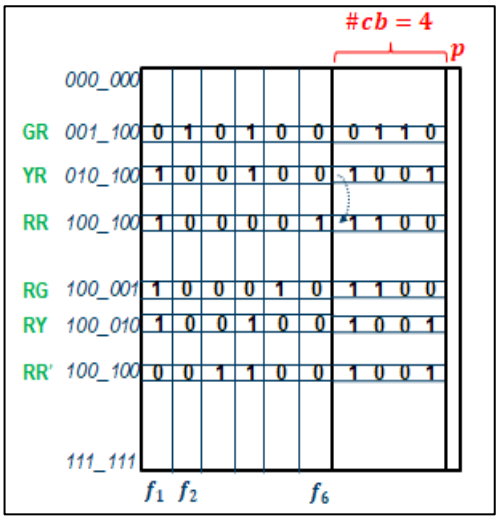

Fig. 7. Traffic light controller with an SEC-DED memory model.

B. Illegal State Detection

We need an illegal state detect mechanism to examine whether the output states are legal or not. As shown in Fig. 8, we use a look-up table, *F*, to illustrate our approach. The look-up table *F* takes the controller's output state as input. For legal states, *F* produces 1, and for illegal states, *F* produces 0. In the traffic light controller example, the number of legal states is 6 (including the duplicated RR state), i.e., there are only 6 words (out of 64) in the memory that *F* produces a 1. The illegal state detection plays an important role in our method, because it guards the system from reaching illegal states, i.e., if the illegal output/state is not corrected by the EDAC mechanism, the look-up table will produce a 0, indicating that the state is illegal.

C. Remapping Logic for Error Detection

Check bit correction and illegal state detection produce useful information for data correction and illegal state detection. For check bit correction, the syndrome points out the error location. As for illegal state detection, the result 1 indicates that the output/state is legal, and 0 indicates that it is illegal.

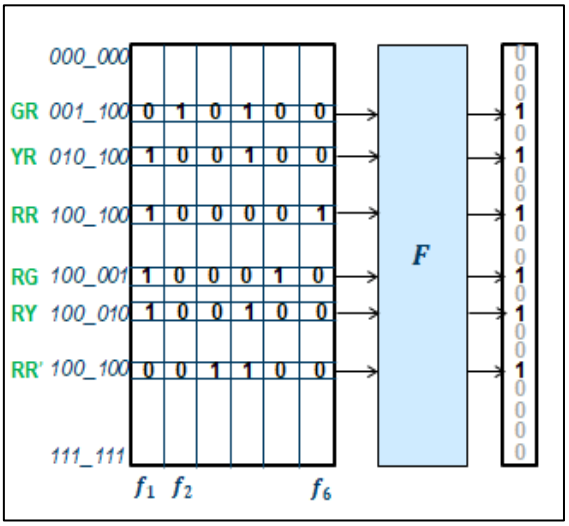

Fig. 8. Traffic light controller with look-up table *F*.

We now introduce the remapping table, which takes the error syndrome and the result of illegal state detection as inputs, and perform output state correction, as shown in Fig. 9. We separate the process into two stages: we first check if the output state is correctable, and then take the result of illegal state detection to check if the current state is legal. If the output state is illegal, the remapping logic can remap the state back to a pre-defined legal state, and make sure the system will then operate in legal states and produce safe output signals. For the traffic light controller example, we let the RR state to be the target remapping state, because RR is a legal state and it produces traffic light signals that is safe, i.e., no cars can pass through the cross-section and unnecessary accidents can be prevented.

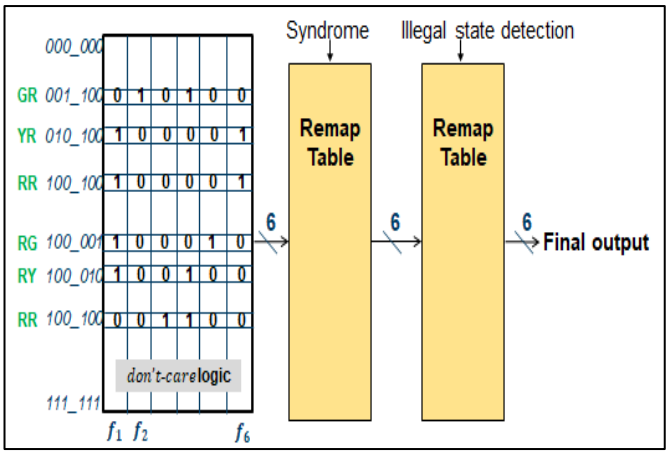

Fig. 9. Remapping logic modeled by memory.

The proposed mechanism is effective, because it can correct the output state or remap the state back to a legal state; therefore, soft errors can be solved by the remap approach.

D. Hardware Implementation

As to the translation of the memory model to hardware design, Fig. 10 shows a general architecture of the proposed symbiotic controller. The output state of the original functional controller is denoted as α, state after the check-bit error correction is denoted as β, and state after the illegal state detection module is denoted as γ, which is also the final output composed of the primary outputs (POs) and the next state vector. The current state vector from the flip-flops is denoted as δ. We assume that in the digital circuit, the number of inputs is n, number of outputs is k, and number of state bits is s.

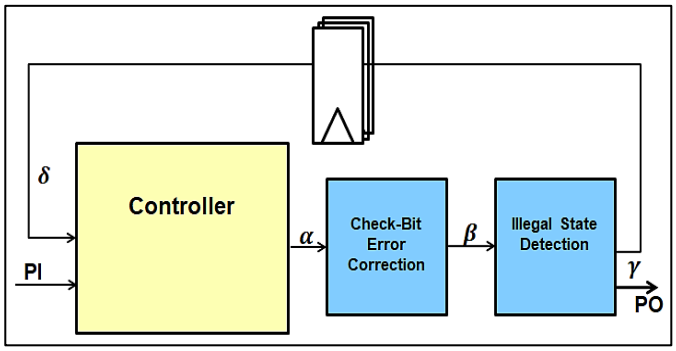

Fig. 10. A general architecture of the proposed approach.

Figure 11 shows the block diagram of the check-bit correction module. The output state of the controller is denoted as Data, and the primary inputs and state vector that determine the check bits are denoted as Index. Similar to the received check bits and generated check bits in the EDAC scheme, the check bit correction module will generate two sets of check bit information: one is encoded from the state of the controller (Data) and the other is the pre-designed check bit function which is derived from Index.

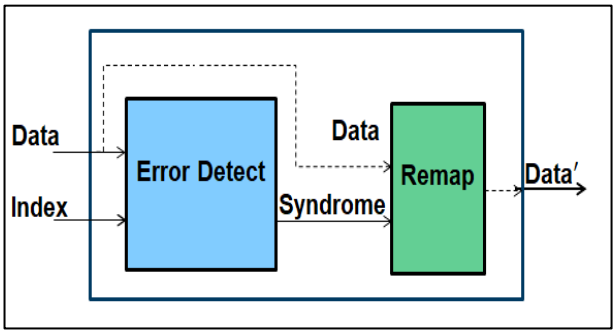

Fig. 11. Check-bit correction module.

Figure 12 shows an implementation of the check-bit correction module, corresponding to what shown in Fig. 11. As to the illegal state detection modules, they can be implemented mainly by XOR functions as well.

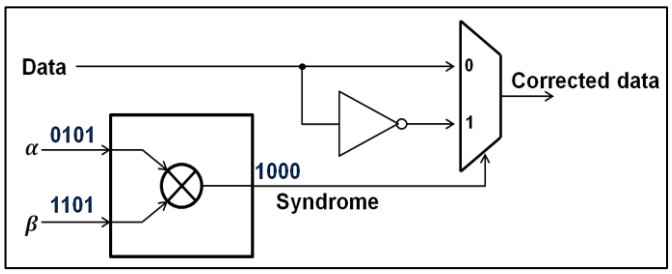

Fig. 12. Implementation of the Check-bit correction module.

Simple evaluation of the memory model-based controllers has been done for reliability improvement. Assume that there are N output states in the controller circuit, and errors occur randomly and independently. The reliability function is expressed as

$$R(t) = (e^{-\lambda t})^N,$$

where $R(t)$ is the reliability function, λ the constant failure rate, and N the number of states. We assume $\lambda=10^{-6}$/hr, and compare the proposed approach with one without repair, and another with the conventional TMR approach. The results are shown in Figs. 13 and 14, respectively, for $N = 10$ and 50.

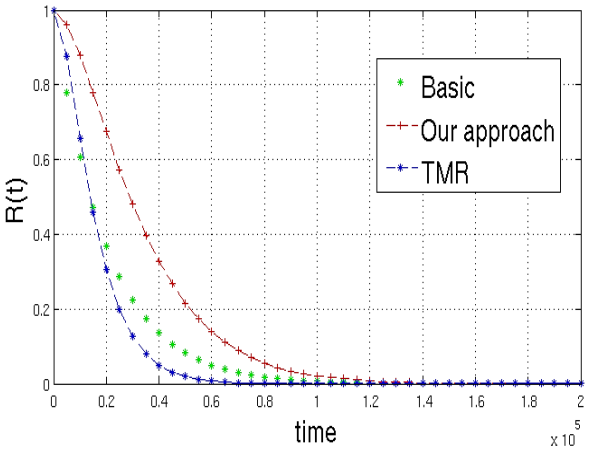

Fig. 13. Reliability functions (N=50).

The proposed approach shows clear advantage in terms of reliability, and thus lifetime. The improvement of MTTF is at least 1.75 times as compared with the original controller, depending on the amount of soft errors that may occur on the controller during operation. A simulator is built to simulate different ratio of hard faults and soft errors. The simulation result shows that for $N = 6$, and the soft error occurrence rate is 50% or higher, the reliability improvement by our approach is increased to 8.4 times. Finally, the area overhead is about 57%, including the check bit storage. Note that the hardware overhead decreases for larger controllers, i.e., for those controllers with more variables.

IV. Conclusions

In this work, we propose a memory-based approach for designing symbiotic controller. We adopted the memory model to design reliable digital controllers. With the memory model, we can apply the EDAC technique and illegal state detection method to the memory model to improve the reliability of the controller hardware, which is normally implemented with logic gates. We further show the method in transforming the memory model to the implemented logic circuit, which implements a practical symbiotic controller. An RTL model implementation of the traffic light controller example has been simulated,

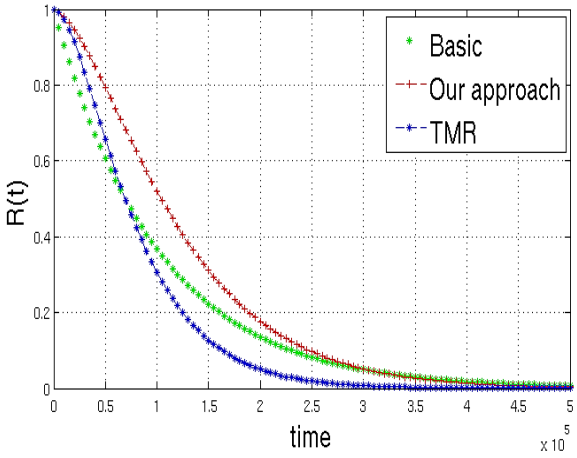

Fig. 14. Reliability functions (N=10).

followed by synthesis for gate-level hardware verification. Result shows that the hardware controller can tolerate specified faults as expected, and produce safe signals, which is important for safety-critical applications. The hardware cost is much lower than TMR. Reliability of the system is also evaluated. The improvement of MTTF is at least 1.75 times as compared with the original controller, depending on the amount of soft errors that occur. A simulator is thus built to simulate different ratio of hard faults and soft errors. The simulation result shows that for N=6, and the ratio of soft error occurrence is 50% or higher, the reliability improvement for our approach is 8.4 times. Thus our approach is good for products that demand high reliability. For the small symbiotic traffic light controller, the area overhead is about 57%, including the check bit storage. Note that the hardware overhead decreases as we apply the approach to larger controllers, i.e., those with more variables.

References

[1] International Monetary Fund (IMF), "World Economic Outlook: Subdued Demand: Symptoms and Remedies," Oct., 2016.

[2] U.S. Energy Information Administration (EIA), "International Energy Outlook 2016," May 11, 2016.

[3] C.-W. Wu, "Symbiotic-System Approach for IOT Devices," in *Proc*. 25th *IEEE Asian Test Symp.* (ATS), Hiroshima, Nov. 2016 (McCluskey Keynote Speech).

[4] C.-W. Wu, B.-Y. Lin, H.-W. Hung, S.-M. Tseng, and Chi Chen. "Symbiotic System Models for Efficient IOT System Design and Test," in *Proc*. 1st Int. *Test Conference in Asia* (ITC-Asia).

[5] B.-Y. Lin, H.-W. Hung, S.-M. Tseng, C. Chen, and C.-W. Wu, "Highly Reliable and Low-Cost Symbiotic IOT Devices and Systems," in *Proc*. IEEE Int. *Test Conf.* (ITC), Fort Worth, Texas, Oct. 2017. (Invited)

[6] R. Bergamaschi, D. Brand, L. Stok, "Efficient Use of Large Don't Cares in High-level and Logic Synthesis," in *Proc*. *International Conference on Computer-Aided Design* (ICCAD), Nov 1995.

[7] M. Turpin. "The Dangers of Living with an X (bugs hidden in your Verilog)." In *Proc*. *Synopsys Users Group* (SNUG), 2003.

[8] L. Piper and V. Vimjam. "X-propagation woes: Masking bugs at RTL and unnecessary debug at the netlist," in *Proc. Design and Verification Conference* (DVCon), 2012.

[9] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL Design*, 3rd Edition, McGraw Hill.

[10] P. K. Lala, Self-Checking and Fault-Tolerant Digital Design, CA, USA: Morgan Kaufmann, 2001.

[11] L.-T. Wang, C.-W. Wu, and X. Wen. VLSI Test Principles and Architectures: Design for Testability, Morgan Kaufmann, 2006.

[12] D. H. Yoon, M. Erez. "Memory Mapped ECC: Low-Cost Error Protection for Last Level Caches" in *Proc. Int. Symp. Computer Architecture* (ISCA), June, 2009.