Abstract

AZIDEHAK, ALI. Design of Fault-tolerant Controller for Modular Multi-level Converters. (Under the direction of Dr. Subhashish Bhattacharya.)

Almost all power electronic circuits are using a digital controller to drive switching devices, handle faults, measure signals and communicate to other systems. The controller for most applications is required to have high degree of robustness, since the failure can cost money or may cause serious injuries to people. Therefore it is necessary to find out if the controller is functioning correctly and if it is not, required procedures must be done to handle the failure.
The first fault-tolerant computer was designed in 1950 and its application was limited to stationary systems such as nuclear plant control system. By 1970, microprocessor developments made it easy to embed chips into every device. There was a big problem with such microprocessors when used in critical applications like aerospace. A failure in the controller could fail the whole project; therefore it needed some layers of protection against faults. Lots of efforts have been done to design a fault-tolerant controller which is the base for this research and will be discussed in the following chapters.
Application of fault-tolerant controllers is not limited to high-end products and even in commercial products like servers, fault-tolerant and non-stop machine is required for packet processing, data storage and computation. With advancement in power electronic systems and control theories, microprocessors become popular in designs. There are more difficulties in power electronic circuits than other systems. The operating frequency of power circuits can exceed hundreds of kilo hertz; therefore real-time response to failure is one of the challenges in the design. Any fault in the controller should be diagnosed instantly; otherwise it is possible to get destructive overshoots in voltage or current. There are also different converter topologies and some need specific controller architecture. It is not possible to use one controller in every design. One of

the new topologies is Modular Multi-level Converter (MMC) that is going under lots of test and development. The biggest benefit associated with this topology is modularity and ability to work even when a fault has happened in one of the modules [8]. To get the most of this converter, controllers should also be fault proof and if a fault happens to one of the controllers, it should be diagnosed and other controller should take the responsibility. That means having no single point of failure in the system and single fault can never interrupt the functionality of the system. The goal of this report is to review researches that have been done for designing fault-tolerant systems and use it for power electronic controllers. In the first chapter, an introduction to the research will be presented. It will define the motivation for this research and the contribution of this dissertation. In second chapter, the techniques for designing a distributed controller system will be investigated. The techniques presented will be about control and synchronization of distributed controllers using a supervisory controller. Third chapter would focus on the architecture of the fault-tolerant controller and failure in the controller systems. Fourth chapter focus on reliability assessment which is a process in fault-tolerant controllers to reveal the perfectness of the functionality and Markov chain is the mathematical tool to model each controller's reliability. Fifth chapter is about firmware design and techniques to avoid failure in the system as much as it is possible. Sixth chapter presents experimental result for two types of converter including modular multi-level converter (MMC) and cascaded h-bridge converter (CHB). Seventh chapter makes the final conclusion and proposes the future researches in this field.

Design of Fault-tolerant Controller for Modular Multi-level Converters

by
Ali Azidehak

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2017

APPROVED BY:

Dr. Alexander Dean	Dr. Srdjan Lukic
Dr. Xiangwu Zhang	Dr. Subhashish Bhattacharya Chair of Advisory Committee

Dedication

To whom, we are waiting for his appearance.

تقدیم به آنکه انتظار ظهورش را می کشیم.

*****

To my parents, Fakhri Shahin & Asghar Azidehak.

تقدیم به پدر و مادر عزیزم.

*****

To my siblings, Farnaz, Amir & Faranak.

تقدیم به خواهران و برادر عزیزم، فرناز، امیر و فرانک.

Biography

Ali Azidehak was born in Esfahan, Iran. During his life, experiencing new things and researching about new subjects, was one of the priorities in the life for him. It started with a junior robotic team which participated in international competitions. He continued his dream and received his B.Sc in Electrical Engineering from Islamic Azad University at Qazvin, Iran. During B.Sc, He has worked in the Mechatronics Research Lab for 4 years and has participated in several Robocup international competitions. In 2012, he started M.Sc in Electrical Engineering at North Carolina State University in Raleigh, NC. He started his research under supervision of Dr.Bhattacharya at Future Renewable Electric Energy Delivery and Management(FREEDM) systems center. During this period, he has worked on broad area of power electronics systems, as well as embedded controller circuits. After completing the M.Sc, He continued PhD under the same advisor and co-advising of Dr.Alexander Dean with focus on the fault-tolerant controller architectures that can be used in power electronics. He has also worked as intern for period of six months in Robert Bosch research center in Charlotte, NC and did research on DC micro-grid systems. The dissertation presented here is based on his research in FREEDM research center about fault-tolerant controller architectures with focus on power electronics application.

Acknowledgements

I would like to thank my parents for their moral and financial support. They made it possible for me to get my degree from NCSU.

***

I would like to thank Dr. Subhashish Bhattacharya for funding and supporting me academically.

***

I would like to thank Dr. Alexander Dean for co-advising and Dr. Srdjan Lukic and Dr. Xianwu Zhang as committee members.

***

FREEDM research center and its employees made it possible for students to reach their goals. Thank you for your help.

***

Special thanks to employees of Robert Bosch LLC in Charlotte, NC whom we've had great time and learned a lot together.

***

I'd like to thank my friends who helped me during my education. Specially Mark Hwang, Rajat Agarwal and Prasanth Gomatam who helped me preparing the experimental setup.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Research Background

Facing the challenge of designing fault-tolerant control systems goes back to invention of the microprocessors. Microprocessor functionality is based on physical principles to execute instructions and proceed to the desired result of programmer. The problem with all physical devices is that they are obliged to fatigue and aging during their life time that can cause the device failure. Even if they don't get damaged, they can get affected by transient physical disturbances and give wrong results. All these hard and soft failures in the controller chip can cause the system failure. In some systems, the failure is not tolerated and the harm the failure can cause, motivates the engineers to design a system that can handle the failures. Power electronic systems are a good example for demand of such controllers. All electronic devices need power to function and without it, they are dead. Therefore, failure in power section always leads to a failure in the whole system. In order to design a fault-tolerant controller for power electronic systems, it is good to look in to other controllers that have been designed in the past. Server computers, space shuttle controllers and re-configurable VLSI chips are some of the examples.

1.1.1 Tandem Computer

Server computers should process the data correctly and reliably. They should always be available to process the data coming in and going out. In a research done for the risk of data corruption, one error per month was seen in 10000 computers running continuously [40]. Although this is a small error rate, the effect can be huge when it is related to money exchange or stock market. One of the most used computer for non-stop server applications is Tandem computer®[48] [10] [11] . The development of this computer has been started in 1974 and is continuing through different brands and corporations. This computer is based on dual or triple module redundancy (DMR or TMR) [12] and the computing power of each module is evolving with time. Figure 1.1 shows the architecture of early Tandem computer that consisted of two to 16 independent processors connected by a pair of interprocessor buses collectively referred to as the Dynabus. Each processor had its own memory and ran its own copy of the operating system. Each processor also had an I/O bus. Each dual-ported I/O controller was connected to two processors'I/O buses, and had internal logic that selected which port was currently the primary path. If a processor or its I/O bus were to fail, the controllers whose primary paths were currently configured to use that I/O bus would switch ownership to their backup paths. Controller configuration was flexible enough that the workload of a failing processor could be spread over multiple surviving processors rather than all being taken over by a single processor. The general design principle was that there be at least two of everything, including power supplies and fans as well as the more obvious processors, controllers, and peripherals. Dual-ported controllers and dual-ported peripherals provided four paths to each device. For disks, the use of host-based RAID-1 mirrored pairs of drives provided eight paths to the data and offered improved data integrity. All of these precaution were done to proof the system from single point of failure (page 4 of [11]).

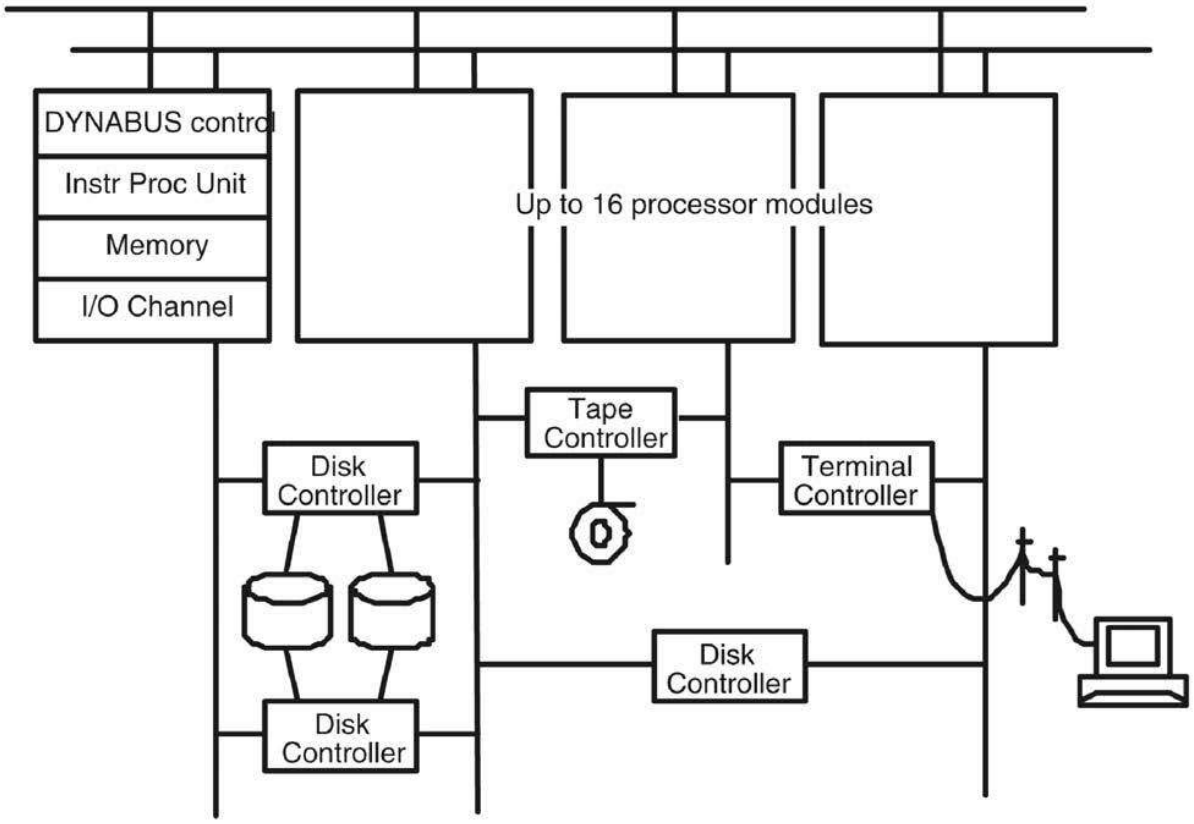


Figure 1.1: Original Architecture of Tandem Computer (1976)

1.1.2 NASA Shuttle Guidance System

In space shuttles, the degree of robustness is much higher and penta module redundancy is the common controller [88] [71] [80]. Figure 1.2 [99] demonstrate one of the controllers used by NASA during 1970's in different shuttle projects. It consists of five processor unit based on IBM AP-101 which is coupled with I/O processor (IOP) to process the data transfer between different units through data bus. All subsystems on the spacecraft are connected redundantly to at least a pair of data buses. There are 24 of these buses, and the subsystems share them, using multiplexers to control the sharing. Eight of the 24 are "flight-critical data buses" that help fly the vehicle; 5 are used for intercomputer communication among the five general-purpose

computers; 4 connect to the four display units; 2 run to the twin mass memory units; 2 more are "launch data buses," and connect to the Launch Processing System; 2 are used for payloads, and the final pair for instrumentation [71]. All these precautions are necessary to gain high rate of reliability in aerospace projects.

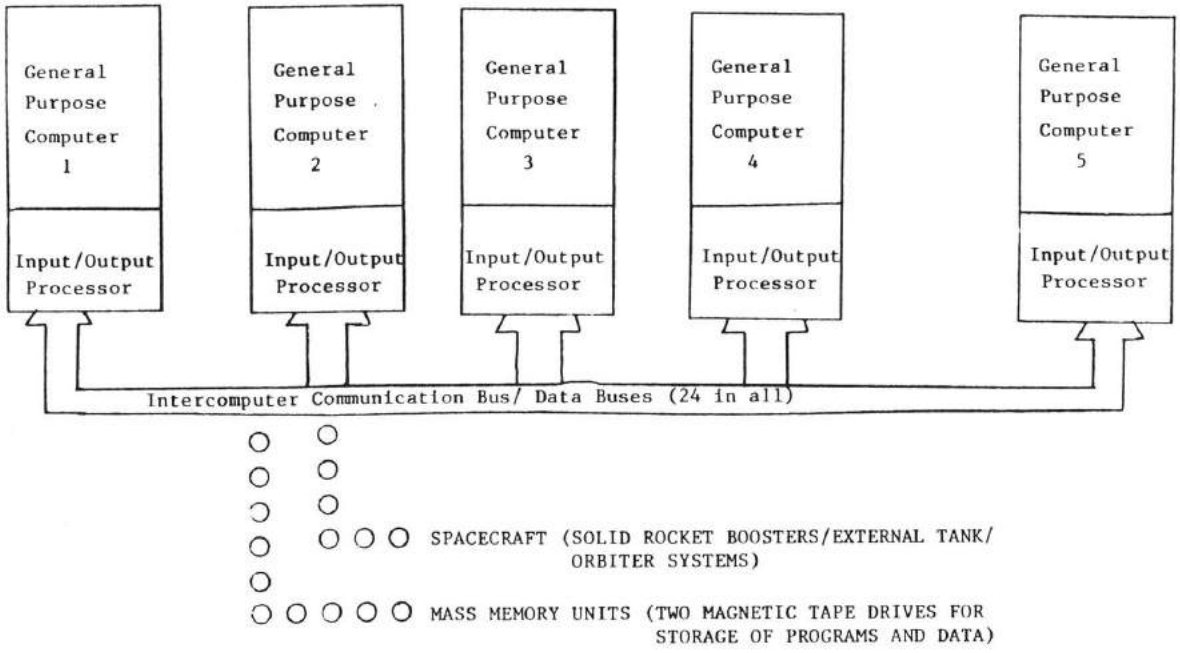


Figure 1.2: Block Diagram of NASA Shuttle Data Processing System

1.1.3 Automotive Control Units

In today's vehicles, as much as 70 micro-controllers may exist for handling the control and safety tasks[56]. Some control tasks like anti-lock braking system (ABS), engine control unit (ECU) and traction control unit (TCU) are directly connected to the safety of the passengers and the design must be fail-safe to decrease the possibility of injuries. Unlike the space and aviation

technology in which designing systems with high availability is desired, in automotive industry the focus is on fail-safe systems. Systems with high availability require much more resources and would increase the cost of the final product. That is not possible in automotive industry with high competition between car manufacturers in decreasing the cost of the vehicle. Instead, the design must be fail-safe and lots of validation tests must be done to make sure there is no bug in design and fabrication stages. In the firmware design, specific standards like MISRA-C have been defined to address the problems in computer programming[20]. These standards help reducing the common error that programmers make in development stages.

The new microprocessors for the automotive industry are much sophisticated than before. Figure 1.3 shows the architecture of a tri-core processor on a single chip. Using multiple core to perform a task will enables the firmware development group to design a fault-tolerant controller architecture. Since all the hardware is integrated, the overall cost won't change very much.

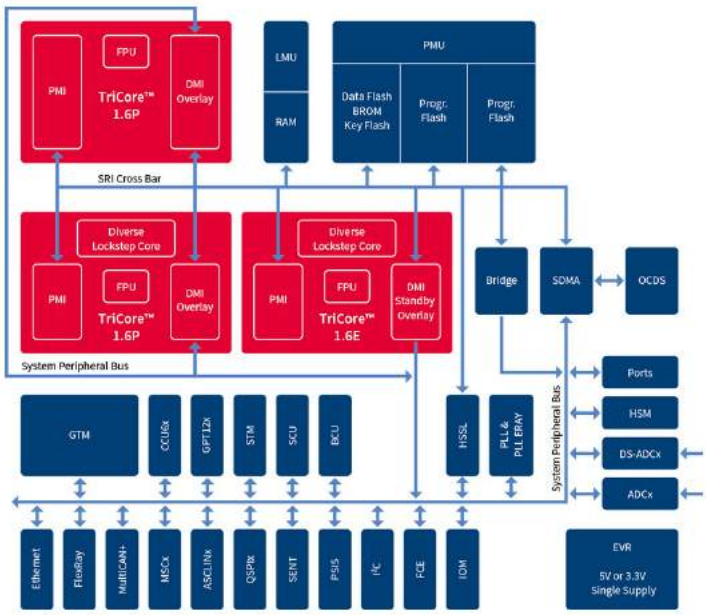


Figure 1.3: Architecture of Tricore^{TM} ver-1.6 Automotive Micro-controller[2]

1.1.4 Fault-tolerant WSI/VLSI Architecture

In most of the digital signal processing (DSP) applications, a huge amount of data is processed to get the final result. One of the good examples of fault tolerancy in DSP systems is the re-configurable processor array that has been designed by a team of researchers in Stanford university [98]. As it is shown in Figure 1.4, an array of processing elements (PE) have been connected to each other through a set of configurable data bus. Many computations in matrix algebra can be conveniently carried out on an array of identical processing elements. VLSI technology provides an inexpensive approach to building such arrays. However, during the fabrication process or during operation, some of the processing elements in a large array are inevitably going to be faulty. Spare PEs and extra routing hardware are often provided so that a fault-free array can be constructed; such reconfiguration capability can be used to increase the yield, and to guarantee fault tolerance in applications where failure is not permissible.

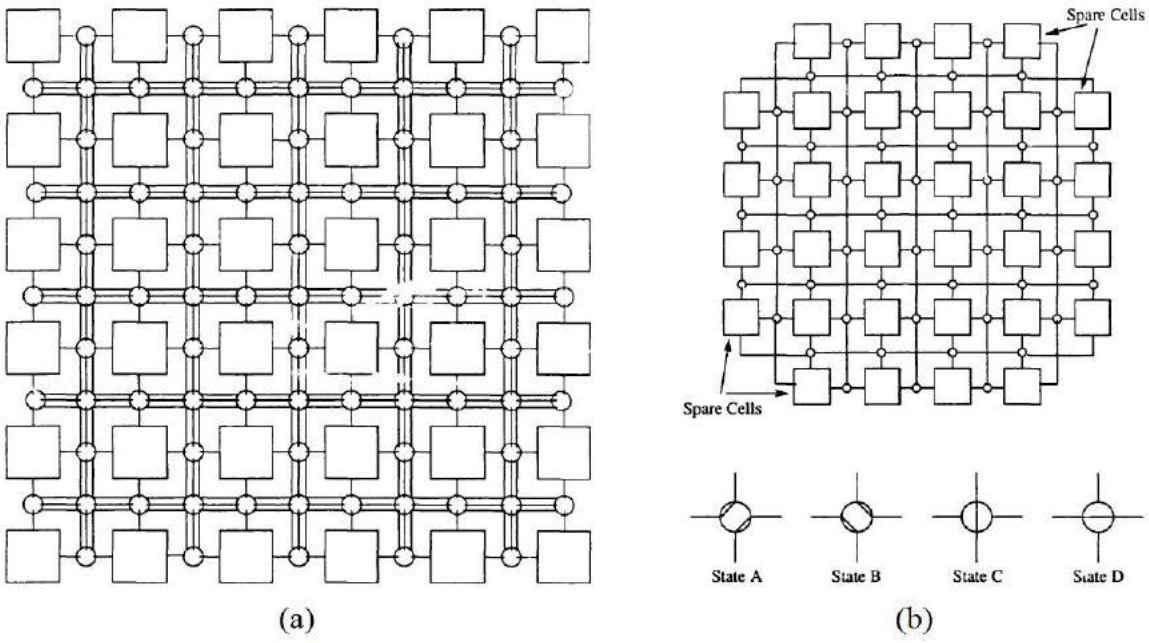


Figure 1.4: Re-configurable Processor Array (a)Before Configuration (b)After Configuration

There are difficulty in hardware and software design of the system. A systematic design procedures for a class of structured algorithms that often encountered in signal processing applications has been developed by that team which is called regular iterative algorithm (RIA). Once a Regular Iterative Algorithm is designed for a given problem, then one can use the systematic design theory to generate efficient processor arrays. The general models that has been explored consist of a set of identical processors embedded in a flexible interconnection structure that is configured in the form of a rectangular grid. This grid can be configured to implement the desired DSP algorithms. In each column of the grid, one extra PE is available to form a N(N+1) matrix of PE(1.5). In the case of failure, the faulty module will be bypassed and the restructured array processor is able to perform the same function like the time before failure.

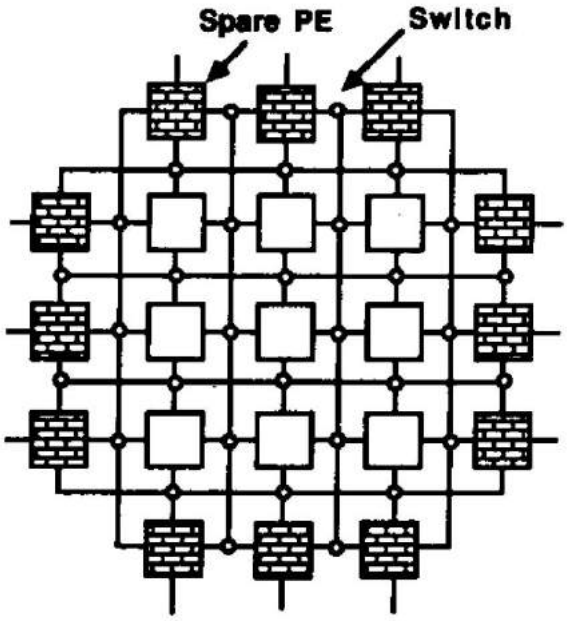


Figure 1.5: Architecture of Array Processor with Spare Processing Element at the Edges

The similarity between array processors and modular multi-level converter is that several modules are available and the architecture of these modules is changeable. In the following chapters, the architecture of fault-tolerant array processor will be investigated in detail and will be modified to match the criteria of controller for modular multi-level converter.

1.2 Motivation

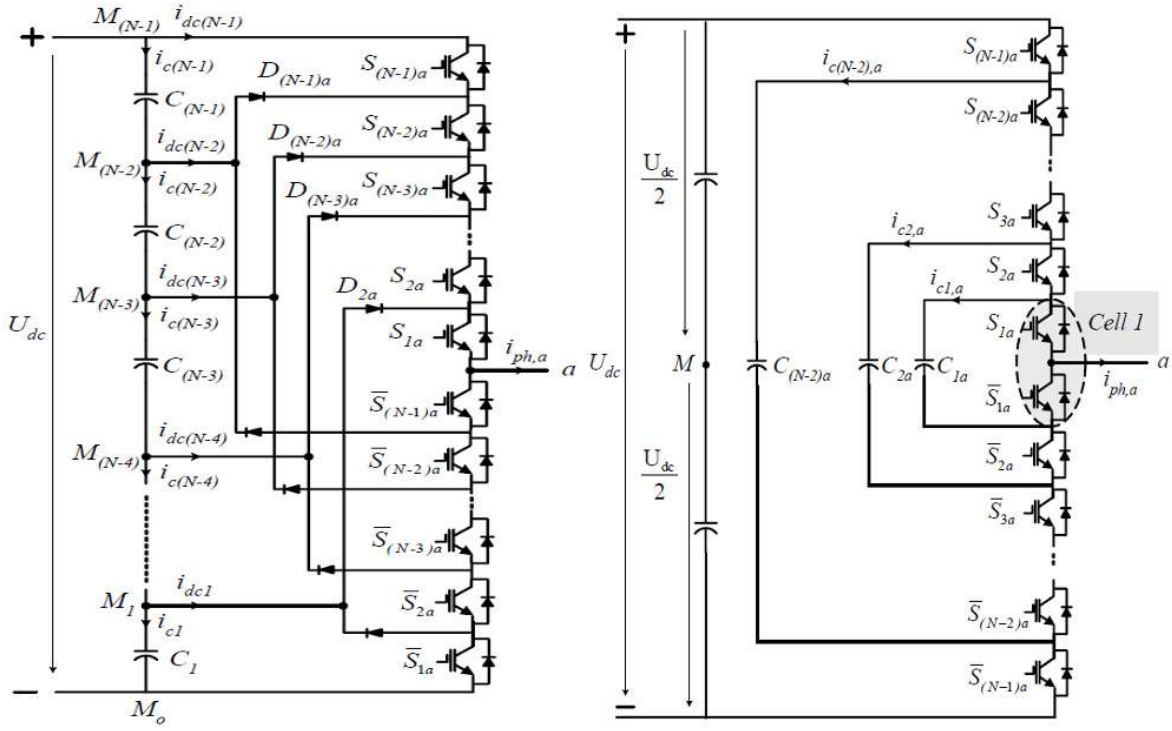
Distributed controllers are much complicated than centralized controllers, hence, there should be justification for using such controllers in the system. In multi-level converters, the are several power modules connected to each other to convert energy from one form to another form. This type of converter is distributed and using distributed controllers doesn't make big change in the system. In the following sections different architectures of multi-level converters will be reviewed and best converter for distributed controller will be selected. After that, application and benefits of such converter systems using fault-tolerant controller will be evaluated.

1.2.1 Comparison of Multi-level Converters and Feasibility of Using Distributed Fault-tolerant Controller

Multi-level converter is a power electronics converter that synthesizes the modulating waveform (usually a sinusoid) using multiple voltage levels. These voltage levels are created using capacitor voltage division network. Figure **??** and **??** shows four different multi-level converters. By turning on the IGBTs, it is possible to connect to capacitors and change the output voltage[89, 26].

In clamped diode multi-level converter (figure 1.6a), it is necessary to turn on the IGBTs that are in the conduction path to the phase output in order get the voltage level of that capacitor. Therefore the sink and sourcing current don't pass a symmetrical line. In flying capacitor multi-level converter, the output voltage is synthesized by arranging the connection style of the floating capacitors. The number of capacitor in series and the polarity of them defines the output voltage. These two type of multi-level converters are **not** modular and there are not the best options for implementing the fault-tolerant controllers.

Figure **??** demonstrate another category of multi-level converters in which the power modules



(a) Clamped Diode Multi-level Converter (b) Flying Capacitor Multi-level Converter

Figure 1.6: Diagram of Two Non-modular Multi-level Converters

can be scaled up easily by adding them in series. Modular multi-level converter (MMC) as well as Cascaded H-bridge converter synthesize the output voltage by changing the capacitors arrangement from DC grid to the output phase (details in Appendix A and B). Each sub-module is separable as a whole block, therefore if one module fails, it is possible to bypass the failed module and converter can continue its operation. By taking advantage of this capability and using distributed controllers, it is possible to form fault-tolerant controller as well as fault-tolerant converter.

1.2.2 Performance and Reliability Enhancement in Power Delivery using Modular Multi-level Converter with Distributed Fault-tolerant Controller

Along with the big changes in power generation, transmission and distribution, the need for intelligent controllers has increased. The new systems need high speed signal processors to handle the tasks. Renewable energy is one of the areas that require intelligent converters. Figure 1.7 shows the renewable energy usage of each country in 2010 [54] [5].

Energy conversion in renewable sources is not as same as the traditional fossil sources. In most

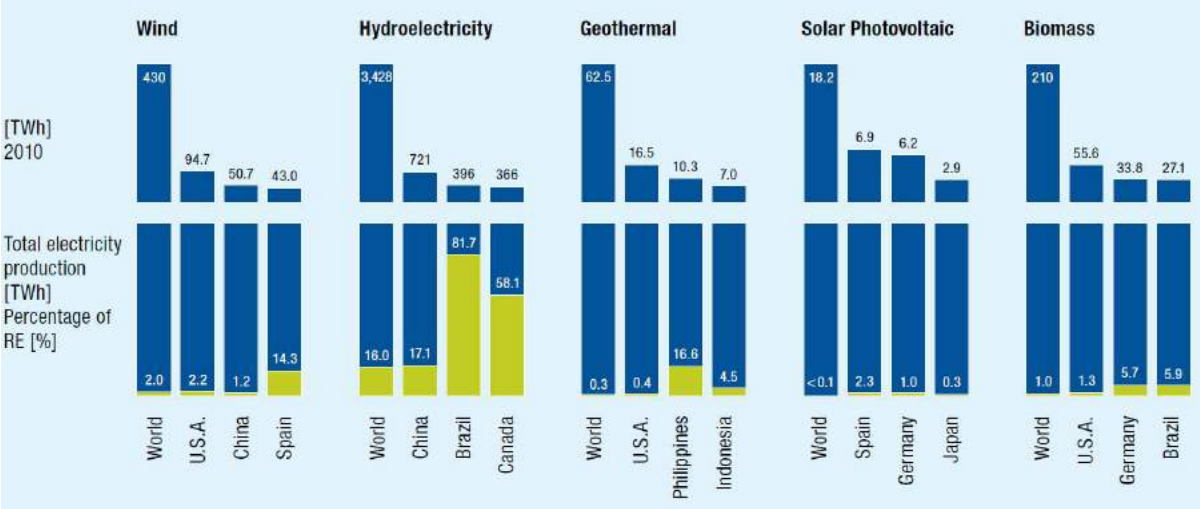
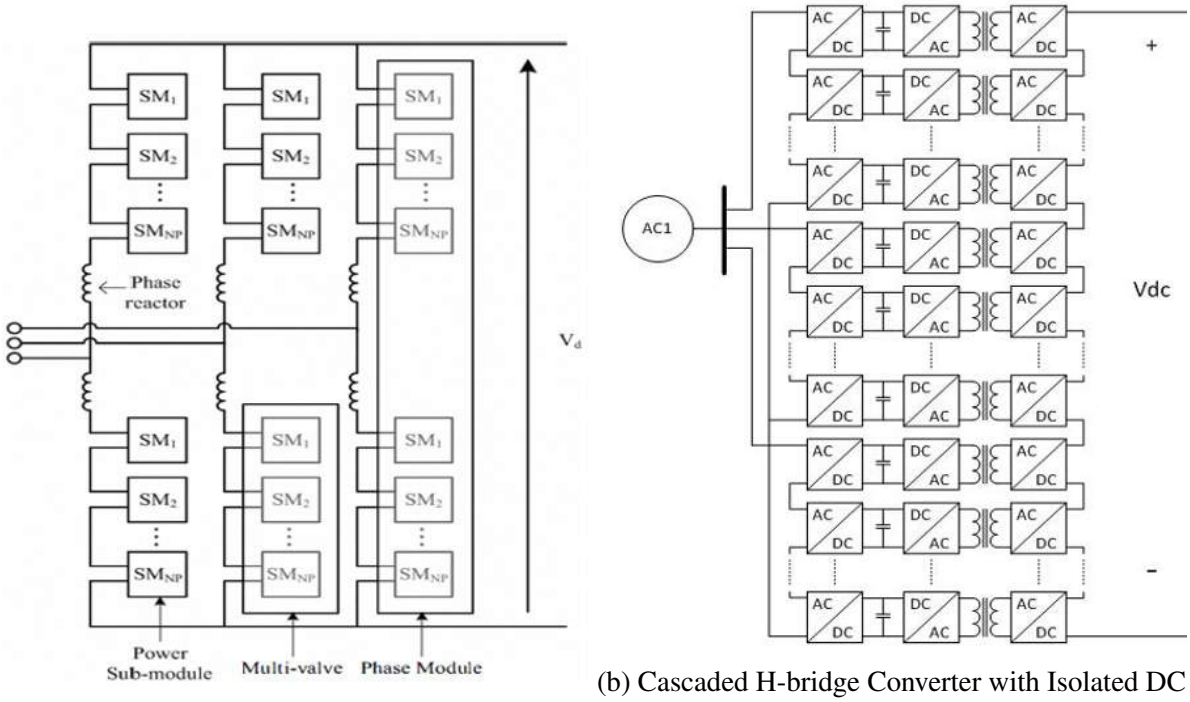


Figure 1.7: Renewable Energy Usage by Each Country

cases, the energy harvested from renewable sources can't be used directly by customers and need conversion by power electronic systems. The voltage and current rating of the generated energy is high and simple converters can't be implemented for this application. Modular multi-level converter (MMC) is one of the converters that can handle this power rating (Figure 1.8a). The

problem with this converter is its complexity in control and drive. Tens or hundreds of modules are connected together and each one should be triggered separately.



(a) Double-Stat-Configured MMC Converter

(b) Cascaded H-bridge Converter with Isolated DC Capacitor

Figure 1.8: Diagram of Modular Multi-level Converters [54]

One of the biggest advantage of this converter is that the total rating of all the modules are higher than the generated power and it is possible to take out some of the modules without interrupting its functionality [25]. To achieve this benefit, controllers must have this capability too. If controllers are distributed just like the power modules, then it is possible to have fault tolerant controllers for the converter.

The dissertation is trying to implement distributed control of the MMC and convert the designed system with fault-tolerant controllers. If failure happens in one of the modules (either in the

controller or the power electronic circuit), it would be detected and isolated from the system. This will increase the availability and reliability of the converter by a large factor because the faulty module can be replaced meanwhile the converter system is operating. From the economical point of view, repairing modular multi-level converters might take a lot of time and in conventional controlled system, power converters should be turned off for the repair period. Down time will increase the cost of repair and it is not desirable. These improvements as well as the economical benefits are the main motivation for the proposed architecture in the dissertation.

1.3 Research Contributions

This dissertation contains many contributions to fault-tolerant controller design for multi-level converter. Some of these contributions are listed as follows:

- Defining different architectures for distributed control of multi-level converters
- Proposing synchronization methods for distributed controllers in multi-level converters
- Applying the control method to cascaded H-bridge converter (CHB) and modular multi-level converter (MMC)
- Reviewing literature related to fault origins in hardware controllers and available solutions in controller architectures and apply it to the proposed controller
- Proposing a new controller architecture for multi-level converters with fault-toleration capability
- Proposing second generation of fault-tolerant controller for modular multi-level converters and focus on synchronization methods related to this type of controller.
- Categorizing fault-detection methods and implementing them for hardware controller and power devices
- Simulating the proposed controller for cascaded H-bridge multi-level converter
- Estimating the lifetime for the controller card and components that have been used inside the controller card
- Modeling the controller architecture using Markov chain, estimating the availability of the proposed controller and assessing the reliability of the proposed controller

- Researching fault-tolerant firmware design methods and techniques to limit failure in software and techniques to bypass those failures
- Implementing proposed controller architecture in hardware in the loop (HIL) simulation for CHB and MMC converter and designing a real CHB converter to test the proposed controller

Chapter 2

Synchronization and Architectures of Distributed Controllers for Modular Multi-level Converters

2.1 Introduction

Due to limited breakdown voltage and nominal current of controlled switches (SCRs or IGBTs), several devices must be connected in series and parallel to tolerate the required line voltage and current. In some applications, designers are forced to utilize several converter modules working together to deliver the electrical power. As a result, several controllers might be required to generate the switching signals of the controlled switches. These controllers must be synchronized with each other in order to work properly. To achieve this goal, proper control architecture is required to synchronize all the controllers and determine how each controller communicates with other controllers. In this chapter, different control architectures including series (daisy chain), parallel and parallel-daisy are explored and compared. There are different problems

associated with designing such converters including synchronizing the distributed controllers to the observatory controller and timing consideration in communication. The focus of this chapter would be on architectures that fit for the design of a fault-tolerant controller.

2.2 Architectures of Distributed Controllers

There are different ways to connect distributed controllers together and to the master controller. Architectures of distributed controllers for modular multi-level converters can be categorized in three sections.

In **daisy chain** architecture, each controller is connected directly to the adjacent controller (Figure 2.1). The nearby controllers can communicate fast, but since the entire process is being controlled by a supervisory controller (master controller), this benefit cannot be utilized in the system. A data packet from master should pass through all modules to reach the last controller. This can cause data latency and can lead to instability in some cases. To overcome this issue, the data packet must be transferred faster compared to control time step (using fiber optic) and be sent bit by bit without being buffered in each module. The latency in data arrival is more critical when pulse width modulation (PWM) signal synchronization is required to be done. Due to harmonics considerations, each PWM signal in any controller should be synchronized to a reference time stamp and the resolution of time difference should be in the order of tens of nano seconds.

Beside all these cons, the major benefit of this architecture is its ability to be implemented by fiber optic based decoupled hardware. Since data line is point to point, fiber optic communication is possible and this will give a high degree of robustness to the entire design. The other benefit is that adjacent controllers can share data with each other to assure correct functionality. This would be helpful in the design of fault-tolerant controller.

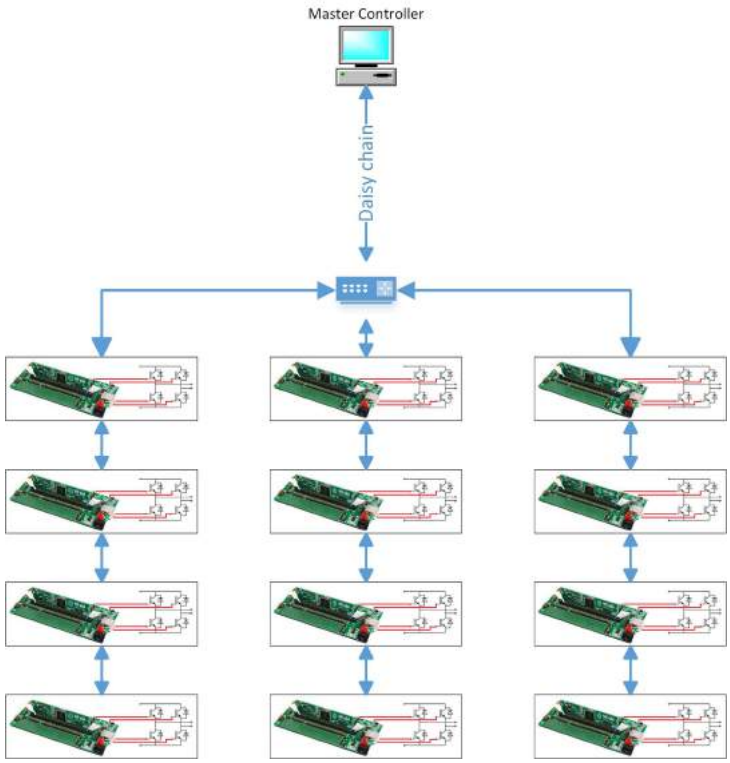


Figure 2.1: Daisy-Chain Distributed Controller Architecture

In **parallel architecture** (Figure 2.2), all controllers share a common communication link and they are able to send and receive data through the same link. However, the permission is being controlled by the master controller and slave controllers are not able to communicate directly with each other. When a data packet is sent, it will propagate to all controllers at the same time and they can read the packet. This will make the synchronization process much easier but bandwidth would be limited due to the fact that all controllers share the same data link. This architecture can easily be implemented using differential pair signals. Good example of this type of communication is RS-485 standard. This standard enables us to have up to 256 devices in the system with baud rate up to 10 Mbps. The communication is half duplex i.e. only one device can transmit data at any time. The pay off in this architecture is the ability of master

controller to send data packets to all of the controllers simultaneously. Electrical precautions must be considered in the design and twisted wire pair should be used in transmission to avoid interference and noise issues.

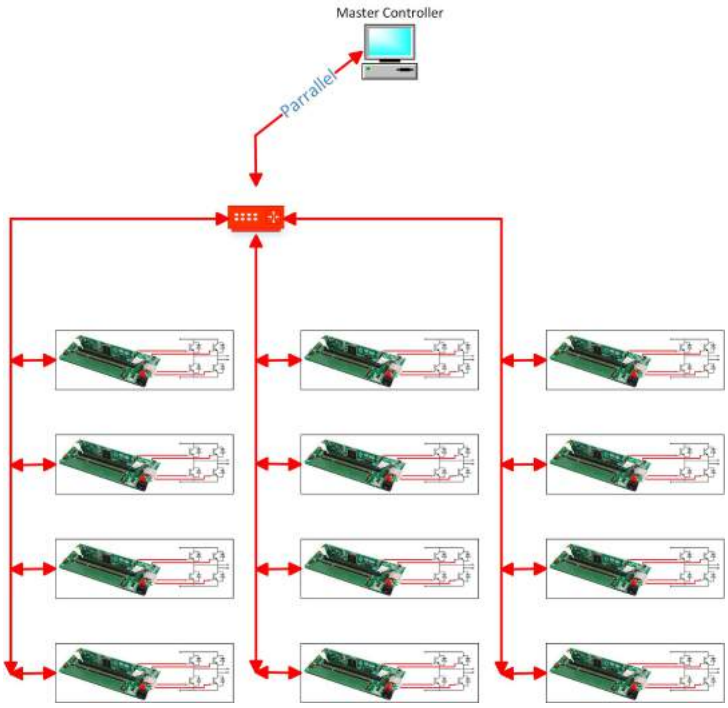


Figure 2.2: Parallel Distributed Controller Architecture

To overcome limitations of two architectures, we can combine both of the typologies together in the modular multi-level converter. In **parallel-daisy** architecture (Figure 2.3), adjacent module can communicate with each other just like the daisy chain architecture. This communication like is used to check the status each controller and in the case of failure in the controller, its function can be done by the neighbor controller. There is also a parallel communication link that connects all slave controllers to the master controller and it would be

used to synchronize the controllers (like parallel architecture). This architecture would be used in other chapter as the base for designing fault-tolerant distributed controller.

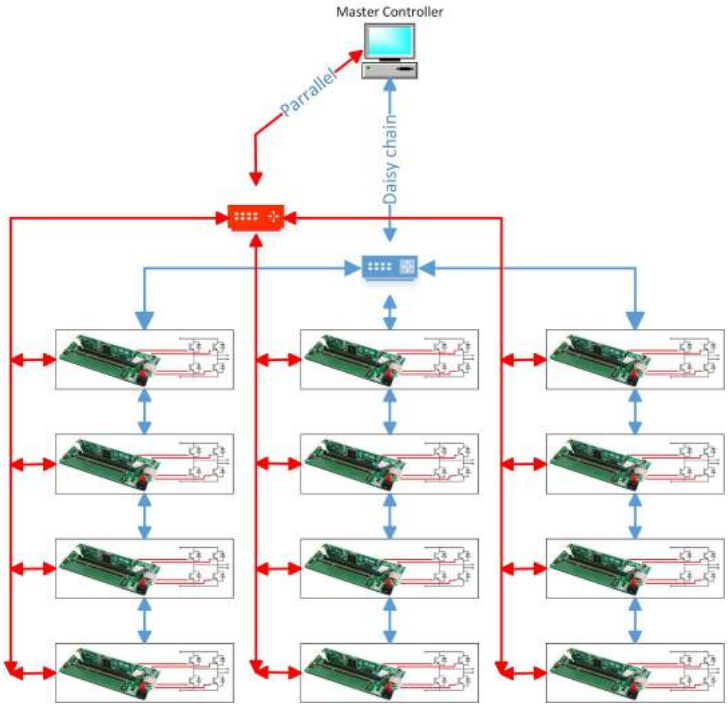


Figure 2.3: Parallel-Daisy Distributed Controller Architecture

The comparison between all architectures can be summarized in table 2.1.

Table 2.1: Comparison between Architectures of Different Distributed Controller Systems

Architecture	Network Realization	Advantages	Disadvantages
Daisy Chain	Any type of serial communication including UART, SPI, Fiber Optic	Short signal path, Scalability, Fiber optic compatibility	Data latency (from master to slave), Vulnerable data link (one connection shortage can stop the system)
Parallel	One to many protocols (RS-422/485, CAN, I2C)	Scalability, Broadcasting ability	Vulnerable data link (short circuit in data link can stop system functionality)
Parallel-Daisy	Combination of serial protocols	Scalability, Broadcasting ability, Fault tolerancy	Complex Packet processing

2.3 Synchronization of Distributed Controllers in Packet Switched Networks

In distributed systems, slave controllers must be synchronized with the master controller [59]. The synchronization process helps the controllers to accomplish tasks in a timely manner and follow the schedule. Frequency (rate of oscillation), phase (start of the oscillation) and time (number of oscillations) are three important parameters that must be passed during synchronization from the master to the slaves. All of these variables are function of the oscillator frequency and any frequency difference may cause error in the system. These error can be formulated as given below:

$$\begin{aligned} f_e &= f_m - f_s \Rightarrow \omega_e = 2\pi f_e \\ t_e &= \tfrac{f_e}{f_m} \times t \\ \varphi_e &= \omega_e \times t + \theta_{e_0} \end{aligned} \tag{2.1}$$

In the above equations, the frequency of the master controller oscillator (f_m) is assumed as the reference frequency and its difference with the slave controller oscillator frequency (f_s) is the error (f_e). The time error (t_e) and phase error (φ_e) are function of the oscillator frequency as well.

One of the most important synchronization parameters in network controlled systems is the phase. Since the control tasks are scheduled to be done in sequence, the phase difference may introduce error in task management. It also introduces interference in communications if any time division multiple access (TDMA) method has been used.

The error in timing parameters may also be introduced during the synchronization process. The

synchronized time of the slave clock $k \in \{1,2,3,...,K\}$ by the master is as below:

$$t_{s,k} = t_m + \theta(t_m) \tag{2.2}$$

In this formula, $\theta(t_m)$ is the offset of the slave controller time compared to the master controller reference time. The offset is equal to:

$$\theta(t_m) = \gamma_{s,k} \cdot t_m + \omega_{s,k}(t_m) + \theta^0_{s,k} \tag{2.3}$$

where $\gamma_{s,k}$ is the deterministic skew, $\omega_{s,k}(t_m)$ is the variable deviation relatively to deterministic skew and $\theta^0_{s,k}$ is the initial offset between the master controller to the slave controller.
In order to synchronize slave clocks to a master clock, the master clock time (t_m) must be transmitted to the slave clock passing through the communication link. This will introduce an error between the slave and master clock as given below:

$$t_{s,k} \leftarrow t_m + \bar{d} + \theta \tag{2.4}$$

Since the data must be passed through packet-switched network, the round-trip time (RTT) must be calculated which is base on the delay between the master to slave ($D_{m\to s}$) and slave to master ($D_{s\to m}$) and can define the delay error as below:

$$\bar{d} = \frac{RTT}{2} = \frac{D_{m\to s} + D_{s\to m}}{2} \tag{2.5}$$

Delay error always exist in the packet-switched networks and will decrease the accuracy of the synchronization by a great factor. In recent network synchronization protocols like Synchronous Ethernet (Sync-E), the synchronization channel has been separated from the data channel and

there is no need for packet processing to synchronize the slave controllers. This can increase the synchronization accuracy up to $\pm 4.6 ppm$.

2.4 Implementation of Clock Synchronization in First Generation of Fault-tolerant Controllers Using TI-Concerto^{TM} Micro controller

In network controlled systems, all controllers must be synchronized to a time reference. In the proposed fault-tolerant controller for modular multi-level converters, the synchronization between slave controllers and the master controller must be implemented in the following cases:

- **Scheduling** the control tasks and communication data flow.
- **Acquisition** of data and variables at the desired time (start of the control procedure).
- **Phase alignment** of the switching signals in comparison to each other.

Control tasks in the slave and master controller must follow a sequence in order to obtain the desired result. It is necessary to synchronize all the slave controllers at each time interval to calibrate and compensate the effect of time drift. In the implemented circuitry for synchronization, a dual processor controller card (28M36) which contains an ARM processor (M3) and a C28 processor (power electronic controller) has been used in both master and slave controllers. In normal operation, data is being transferred from master to slaves in a periodic manner after sampling and computation of the control loops in highest level. Figure 2.4 shows the data flow between controllers. In the supervisory controller, the control loop runs based on the switching frequency of the converter (when analog signals have been converted to digital at start of PWM signal). The result is being transferred from the control subsystem (C28) to the master subsystem (m3). This processor is responsible to form a packet frame and send the data to the slave controllers.

The Slave controller use the same process to send/receive data and analyzing them. The differ-

ence is that the data from master is being broadcast to all slaves , which means sending only one packet is enough to synchronize all the slave controllers, but from slaves to master is not the same and the channel is being shared between controllers. To divide the communication channel between slave controllers, two principle solutions are available:

- **Time Division Multiple Access**: The receive channel can be divided based on time division multiple access (TDMA) technique. Whenever the master sends request to slave to get updated data, a scheduler will start counting and based on the module number (assigned automatically or by installer), each module sends out its status and variables. The biggest problem in this method is the error in crystal oscillators of slave controllers and the probability of data frame overlapping. Therefore, a dead time must be allowed between data packets to avoid this problem.

- **Carrier Sense Multiple Access**: In this mode, all the slaves listen to the master controller and when the master requests for data from a slave controller, then slave starts to send data to the master controller. The other slaves can listen to the transmitting slave and when it ends the process, the next slave can start sending data immediately.

Due to the difference in the operating frequency of the crystal oscillators in each module and desynchronization of controller after each hardware fault, periodic module synchronization is necessary for switching signals. For synchronizing the PWM waveform, a fast trigger signal must be applied to all slave controllers to inform them about the reference phase (start of PWM signal). The easiest way is to consider a separate signal line to all the modules, but it is costly and similar to having a data link again.

The most efficient way is to using the data link to send the reference signal, but the problem is the baud rate of the data link. Transferring single byte would take several microseconds and this accuracy is not good enough for phase synchronization (the complete packet has several bytes

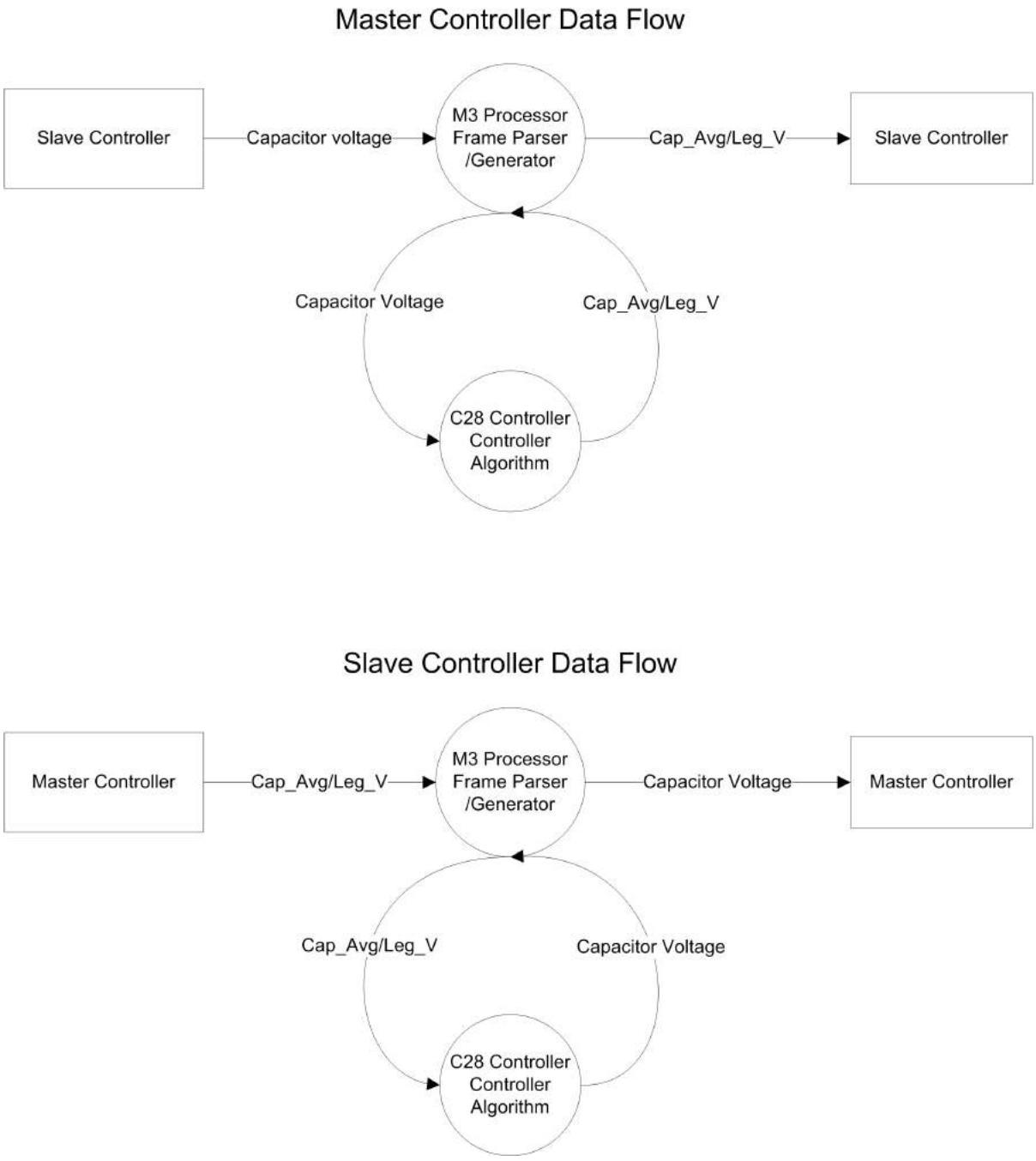


Figure 2.4: Example of Data Flow Diagram (Capacitor Voltage) between Master and Slave Controllers

and it takes even more time). The solution is to send a data packet and make the controllers ready to sense a transition in Rx signal and then doing the phase adjustment procedure. Figure 2.5 represent a simple hardware that can do this in the easiest way.

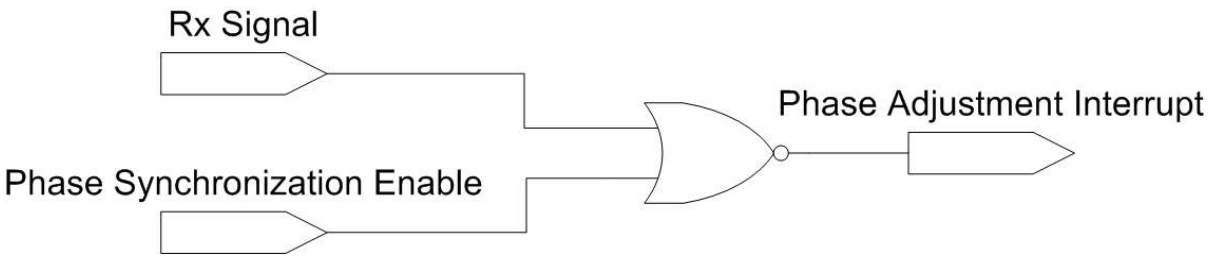


Figure 2.5: PWM Phase Synchronization Hardware for Slave Controllers

Whenever the slave controllers receive ready to synchronize (ROS) packet (which makes them ready to adjust the PWM phase), each controller enables the phase adjustment input (low level) and wait to receive interrupt signal to load the phase value (based on module number) to the PWM counter register. When control function is called in the master controller (which is synchronized to internal oscillator and triggers at specific times), a dummy data is sent to the serial port. The change in communication link will trigger the entire slave controller and therefore synchronize them to the reference phase value. Figure 2.6 show the flow chart implemented in controllers to handle signal synchronization.

The proposed synchronization method has been implemented in the system and the result can be seen in figure 2.7 and 2.8. In figure 2.7, there is a phase difference between the controllers which has been accumulated after 100 control cycles. By reaching this time, the

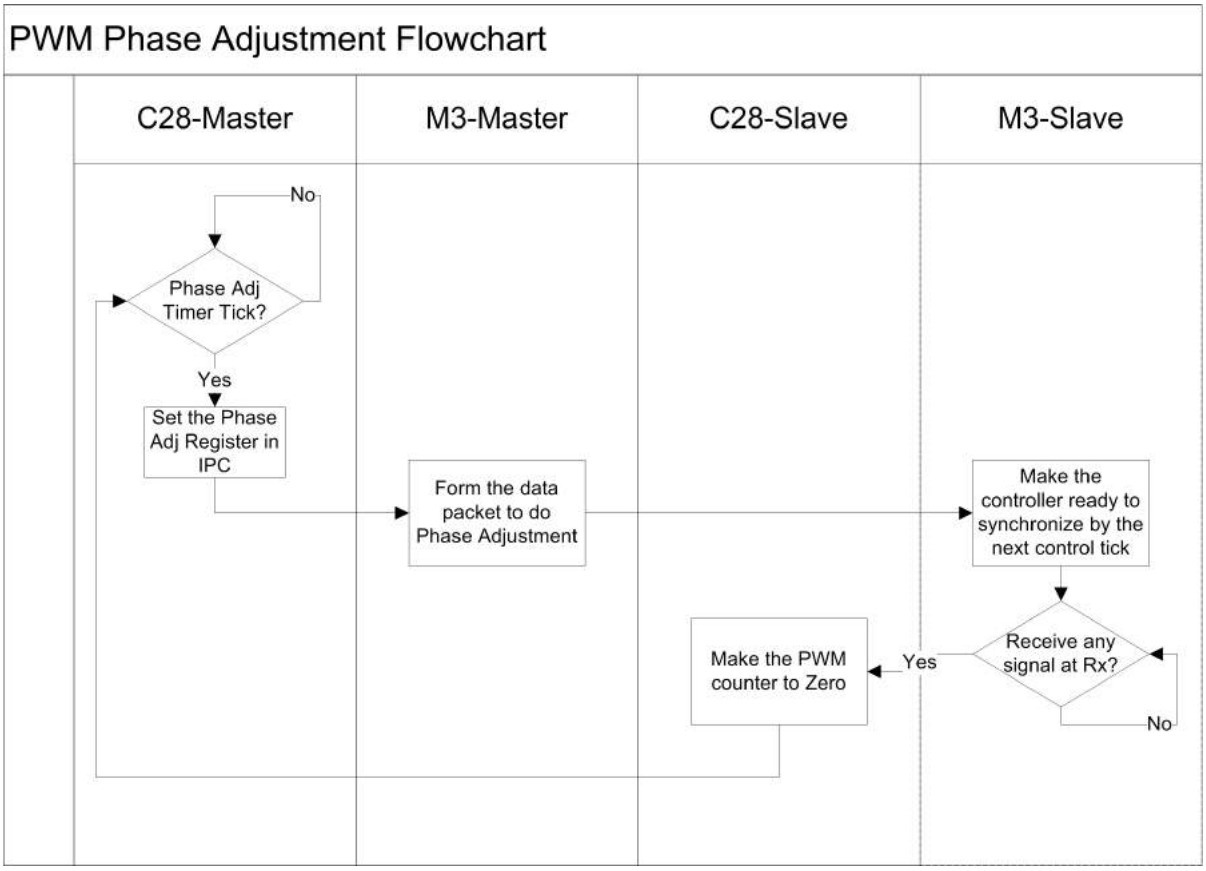


Figure 2.6: PWM Phase Synchronization Flowchart

master controller sends data packet to prepare the slave controllers for the synchronization process. While controller is in this mode, all communication tasks would be halted (not to interfere with synchronization process). Master controller waits until the reference phase become zero, then it signals all slave controller with a dummy byte. This will set the phase register of the controllers to zero and synchronize all controllers together. Figure 2.8 shows the synchronized controllers after the completion of this procedure.

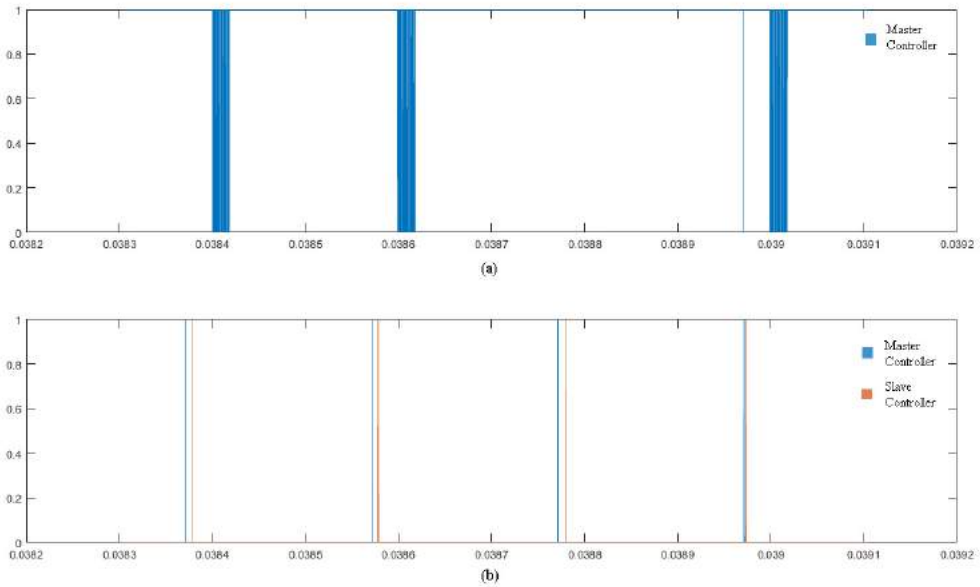


Figure 2.7: Synchronization Procedure in Proposed Controller (a)Data Packet Sent by the Master Controller (b)Phase Zero-crossing for Slave and Master Controllers

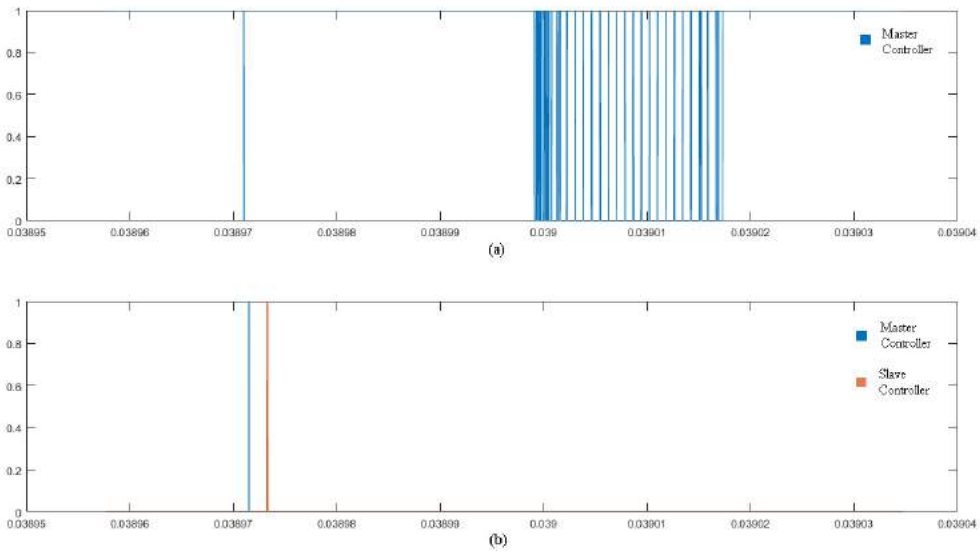


Figure 2.8: Magnified Synchronization Procedure after Initialization of Process by Master Controller

2.5 Second Generation of Fault-tolerant Controller and Dominant Output (DOMINO) Synchronization Algorithm

As depicted in figure 3.12, in the second generation of fault-tolerant controller there is no single controller as master (synchronizer). Instead, there is a group of the controllers which can measure global variables and can synchronize other slave controllers through serial communication. In case of failure in the master controller, another controller with synchronization capability will take the control and will synchronize all other controllers. In this architecture, communication links are point-to-point (full duplex), therefore fiber optic implementation is feasible. In each module, there are four serial ports (north, south, east and west) in which the controller can communicate to other controllers. Since data signal should pass through several controllers to reach the last controller, the latency must be minimum or synchronization may not be valid anymore (reference time between master controller and the slave controllers will be huge). The problem here is to find the shortest path between master controller and all other slave controllers. This will guarantee the least amount of latency in the synchronization process[51].

2.5.1 Shortest Synchronization Path in Second Generation Controller

The set of controllers in second generation architecture can be shown by a graph (figure 2.9). In this graph each controller is represented by a vertex (V) and the communication links are shown with edges (E). Therefore, the ordered pair $G = (V, E)$ represents the graph representation of the controller set. The weight of edges are different (due to the physical length of communication links). For horizontal edges, the communication delay is D_h and for vertical edges it is D_h.

The problem is to find the shortest path from the master controller (M) to all other slave controllers (S). In this problem, some slave controllers or communication links might be

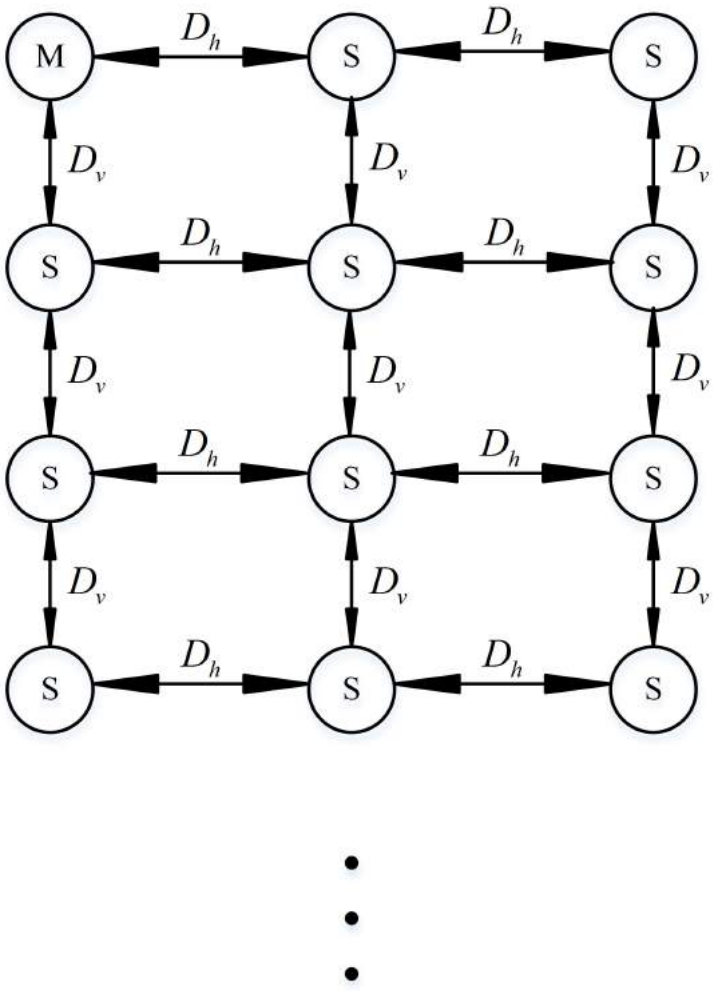


Figure 2.9: Graph Representation of the Second Generation Controller

unavailable. The problem can be classified as single source shortest path and there are different algorithms to find the shortest path from the source to other nodes [33, 34]. Among all the algorithms, Dijkstra's algorithm has been used in different problems and has good efficiency in comparison to other algorithms. Algorithm 1 demonstrate the method of find shortest path using Dijkstra's algorithm. In this algorithm, all the distance are initialized as infinity. After that, it starts from the source and finds the distance to all outgoing nodes. The table of distances would be update and the node with smallest distance would be chosen as the next node for

finding distance. If the distance from the node is in addition to the other distances are smaller that the value in the table, the table would be updated and the node with smallest distance would be chosen for the next iteration. This will be done for all nodes and the distance table at the end, represents the shortest path from the source to all other nodes[23].

```
Data: Graph and source as inputs
for each vertex v in Graph do
    dist[v] = infinity;
    previous[v] = undefined;
end
dist[source] = 0;
Q = the set of all nodes in Graph;
while Q is not empty do
    u = node in Q with smallest dist[ ];
    remove u from Q;
    for each neighbor v of u do
        alt = dist[u] + dist-between(u,v);
    end
    if alt < dist[v] then
        dist[v] = alt;
        previous[v] = u;
end
return previous[];
```

Algorithm 1: Pseudo-code of Dijkstra's algorithm

2.5.2 Hardware Synchronization Using Dominant Output (DOMINO) Algorithm

Dijkstra's algorithm is one of the greatest algorithms in finding shortest path. This algorithm is beneficial if the graph of controllers is available and there is enough time to process the algorithm. The time complexity of the Dijkstra's algorithm with Fibonacci heap is $O(E + VlogV)$. Therefore, it is not an optimistic solution of online applications[90]. The other problem is gathering data from all controllers to the master controller for the algorithm and sending back the data to slave controllers. This requires using huge amount of capacity from the data link in the system which is not possible on the proposed controller.

The proposed solution is to use hardware method based on Dijkstra's algorithm for finding the shortest path which also implement fault-tolerant synchronization for the controllers. This algorithm may be called **Domin**ant **O**utput (DOMINO) synchronization algorithm. In this method, the master controller will start sending the bit-stream for the data packet. Whenever the first bit arrives, the internal circuit of the controller will use that serial port for propagating data to other serial ports. In order to decrease the latency, the received signal at serial port input of each module must be written to other serial outputs of the controller instantaneously, therefore it can propagate to the other controllers in the least amount of time. The arrival of the signal will be the reference time for the controllers. The pseudo code for the algorithm is shown in algorithm 2. By activating the synchronization circuit, the controller waits for the first coming bit on the receive input (RX) to set the direction flag. After locking to the first incoming signal, this signal will be routed to all other serial outputs (TX). This will propagate the data packet as well as synchronization signal to all controllers. This synchronization algorithm may be done

based on a time schedule which is related to the controller's oscillator precision.

```
Result: Synchronization of slave controllers to the master controller with minimum
        latency
Clear flags;
Enable Synchronization Circuit;
while (Synchronization Enable == True) do
    if (Direction flag != NULL) then
        TX[north, south, west, east] = RX[Direction flag];
    else
        if (first bit arrives from north direction == True) then
            Direction flag = north;
        else if (first bit arrives from south direction == True) then
            Direction flag = south;
        else if (first bit arrives from west direction == True) then
            Direction flag = west;
        else if (first bit arrives from east direction == True) then
            Direction flag = east;
end
```

Algorithm 2: Pseudo-code of Dominant Output (DOMINO) Synchronization Algorithm

The proposed algorithm may be synthesized with digital circuits to increase the speed and decrease the latency (figure 2.10). To show the effectiveness of the algorithm, a simulation based on Verilog has been done. In this case, a 10×3 matrix with $D_h = 7$ and $D_v = 5$ has been simulated under different fault conditions (figure **??**).

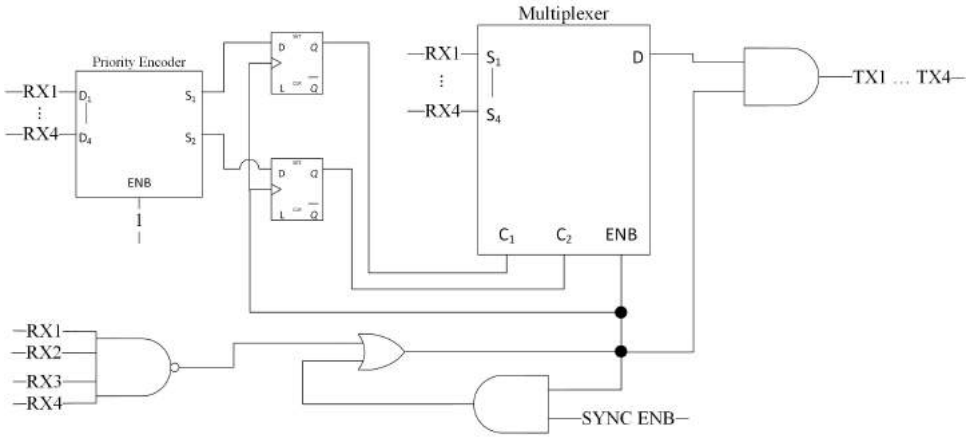


Figure 2.10: Hardware Implementation of DOMINO Synchronization Algorithm

	0	1	2
0	0	7	14
1	5	12	19
2	10	17	24
3	15	22	29
4	20	27	34
5	25	32	39
6	30	37	44
7	35	42	49
8	40	47	54
9	45	52	59

(a) No Fault

	0	1	2
0	0	OFF	24
1	5	12	19
2	10	17	24
3	15	22	29
4	20	27	34
5	25	32	39
6	30	37	44
7	35	42	49
8	40	47	54
9	45	52	59

(b) Single Fault

	0	1	2
0	0	7	14
1	OFF	OFF	19
2	38	31	24
3	43	36	29
4	48	41	34
5	53	46	39
6	58	51	44
7	63	56	49
8	68	61	54
9	73	66	59

(c) horizontal Fault 1

	0	1	2
0	0	7	14
1	OFF	OFF	19
2	38	31	24
3	43	OFF	OFF
4	48	55	62
5	53	60	67
6	58	65	72
7	63	70	77
8	68	75	82
9	73	80	87

(d) horizontal Fault 2

	0	1	2
0	0	7	14
1	OFF	12	19
2	OFF	17	24
3	OFF	22	29
4	OFF	27	34
5	39	32	39
6	44	37	44
7	49	42	49
8	54	47	54
9	59	52	59

(e) vertical Fault

	0	1	2
0	0	7	14
1	5	12	19
2	OFF	OFF	OFF
3	OFF	OFF	OFF
4	OFF	OFF	OFF
5	OFF	OFF	OFF
6	OFF	OFF	OFF
7	OFF	OFF	OFF
8	OFF	OFF	OFF
9	OFF	OFF	OFF

(f) Complete Failure

Figure 2.11: Simulation Result for DOMINO Algorithm

2.5.3 Proof of Minimal Time Synchronization for DOMINO Algorithm

By implementing DOMINO algorithm, all slave modules will be synchronized in the minimum time by the master controller (synchronizer). In this problem, each communication link (edge) has non-negative weight (propagation delay) and controllers (node) are placed in a net grid. DOMINO algorithm is a divergence of Dijkstra algorithm and it can be proved in the same manner.

The delay between the synchronizer (s) to controller at vertex v_i is called d_i. The set of controllers S can be chosen where[4]:

$$\forall v_i \in S, \quad \forall v_j \in V - S, \quad d_i \leq d_j \tag{2.6}$$

Second, for all slave controllers $v_j \in S$, there is a shortest delay from s to v_j using controllers of S as intermediates.

For controllers outside the S, delay $d_j \in V - S_k$ can be defined as:

$$d_j^{est} = \min d_i + \omega(e_{i,j}) \tag{2.7}$$

Lemma 1: The estimated delay d_j^{est} is the smallest delay from s to d_j, using only controllers in S as intermediates.

Proof: Any path from s to d_j using controllers of S as intermediates, consist of a smallest delay path from s to some $v_i \in S$ and then one more path from v_i to v_j. The last path is only choice for the synchronization and therefore the total delay is the minimum delay from s to v_j.

Lemma 2: Let v_m be an another controller in $V - S$ such that d_m^{est} is minimum. Then $d_m^{est} \leq d_j$ for all $j \in V - S$(i.e. if v_m is the first controller to get synchronized, it will have the smallest delay than to synchronizer that other un-synchronized controllers).

Proof: This can be proved by contradiction. Assume there is controller $v_j \in V - S$, with $d_j < d_m^{est}$. Since $d_m^{est} < d_j^{est}$, we would have $d_j < d_j^{est}$. Therefore, any shortest path P from s to v_j is shorter that the length of a shortest path using only controllers from S as intermediates. Therefore, P must use at least one controller from $V - S$ as intermediate. Let v_x be the first controller from $V - S$ as we go from s to v_j. Since v_x comes before v_j, then $d_x^{est} \leq d_j < d_m^{est}$. But v_m is defined to be a controller from $V - S$ such that d_m^{est} is minimum. This leads to a contradiction, so the assumption that there is some v_j that $d_j < d_m^{est}$ has to be wrong.

Now, it can be set that the new set of synchronized controllers S' has the two properties of previously synchronized controllers S.

Theorem: S' is a set of controllers nearest to s that:

$$\forall v_i \in S', \quad \forall v_j \in V - S', \quad d_i \leq d_j \tag{2.8}$$

Since the lemma is true for v_m as well as other controllers in $V - S$, we have $d_m^{est} \leq d_m \leq d_m^{est}$, that means $d_m^{est} = d_m$. So it is the path with smallest delay to v_m. Also, since $d_m^{est} = d_m \leq d_j$, for all $v_j \in V - S$, then it is true that $d_m \leq d_j$ for all $v_j \in V - S'$.

Therefore, based on this, the newly generated set of synchronized controllers will also have the smallest synchronization time from the s.

2.5.4 Synchronization Using Fiber Optics

Due to the use of controllers in high voltage applications, insulation is one of the problems in implementation of the controller circuits and it is desired that no electric connection exist between the controller cells. Fiber optic communication is suitable for this application because communication cables are made of non-conductive materials and the data rate is much higher than the copper wires. Fiber optic cables can carry data from point to point using light beams, hence they are much immune to external interference.

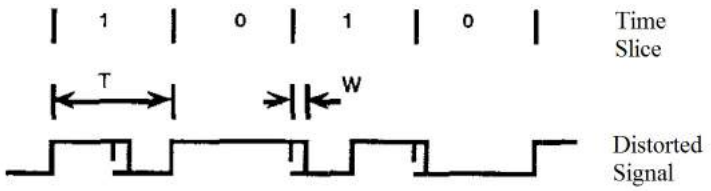


Figure 2.12: Pulse Width Distortion in Fiber Optic Output Signal at Receiver

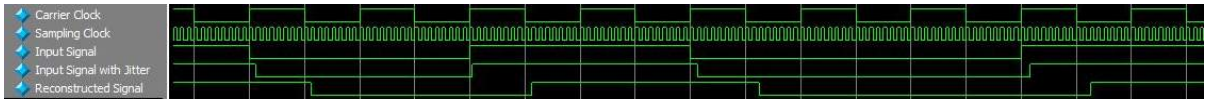
One of the biggest problems in using fiber optics for synchronization and data transfer is the pulse width distortion (w) caused at the optical to electrical conversion (figure 2.12). This type of error can be added up by repeating the conversion in the series controller modules in which data would be lost after passing several controller modules (second generation controller). One of the methods for eliminating this error is using voltage comparator at the receiver buffer and controlling the threshold voltage of the reference [67]. This method decreases the jitter at the receiver but it can't eliminate all the error.

The other method is re-sampling the received serial data and transmitting it with the delay equal to half of the transmission period. The re-sampling circuit is synchronized by the first falling edge (start of the transmission), then it waits for period of $T/2$. After that, it starts sampling

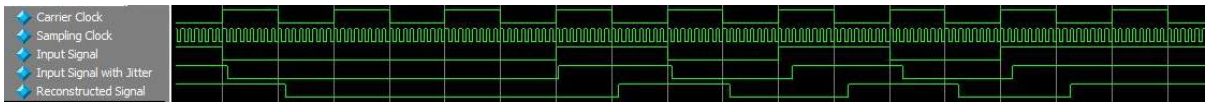
with period of T until the last bit of the data. Since the sampling is done in the middle of the data bit, this method guarantees signal reconstruction if pulse width distortion (w) is less than $T/2$.

Figures 2.13a and 2.13b show two cases in which the pulse width distortion (PWD) is less than $T/2$. The reconstructed signal has the same value of the received data but the distortion has been fixed and there is a delay of $T/2$ from the receiver to the transmitter.

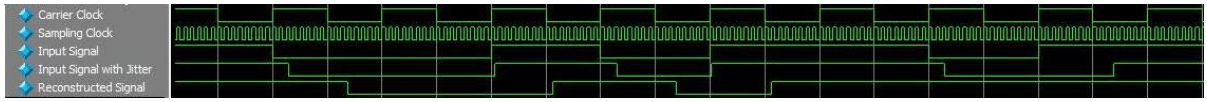
If the jitter at the receiver exceeds $T/2$, the data is no longer valid. Figures 2.13c and 2.13d are two cases that jitter has exceeded $T/2$ and the output of the correction circuit is not useful.



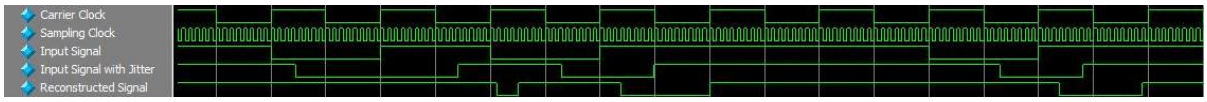
(a) Output Result When Pulse Width Distortion is Less than T/2



(b) Output Result When Pulse Width Distortion is Less than T/2



(c) Output Result When Pulse Width Distortion is Higher than T/2



(d) Output Result When Pulse Width Distortion is Higher than T/2

Figure 2.13: Result of the Correcting Circuit at each Repeater Module

2.6 Timing Consideration in Design of Distributed Controllers

To have better insight about the behavior of the distributed controller, timing parameters of the system must be investigated [60]. Figure 2.14 show the performance of continuous time, discrete time and network controlled systems in different sampling times.

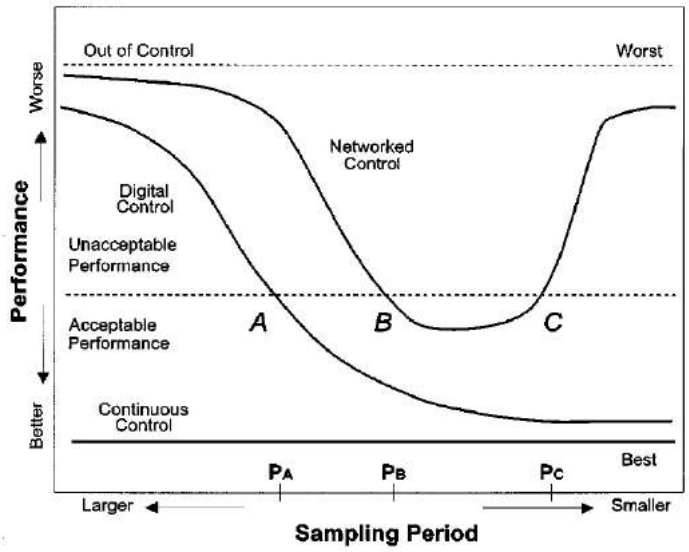


Figure 2.14: Performance of Control Systems versus Sampling Time

In continuous time systems (e.g. analog systems), sampling time has no meaning; therefore the performance is always the same. In digital control system, higher sampling rate will make the system looks more like the continuous system and therefore the quality of performance (QoP) will increase drastically.

In Network Controlled Systems (NCS), the bandwidth of the communication link is limited. Increasing the sampling rate means transferring more amount of data in the network that could decrease the functionality of the system. By having knowledge about the timing in the system,

the maximum safe sampling rate can be selected for the AMMC. The rule of thumb is whenever network bus gets crowded the performance will decrease. In the scheduled NCS, the optimum performance can be gained by using the maximum baud rate of the network link.

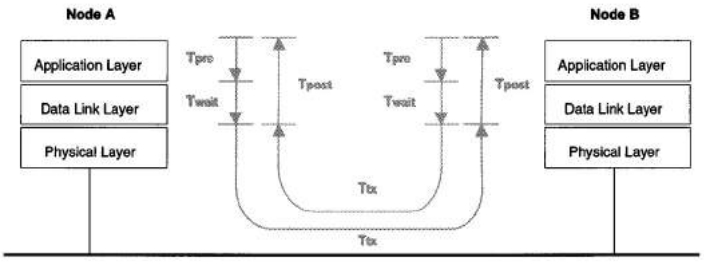


Figure 2.15: Time Diagram Showing the Time Spend to Transfer Data Between Different Node of a Network Control System

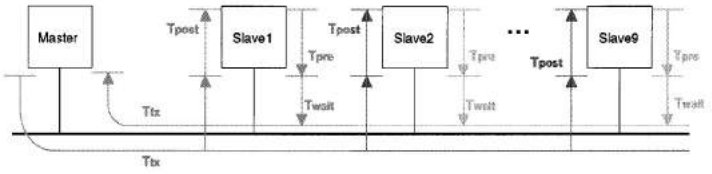


Figure 2.16: Waiting Time Diagram

Figure 2.15 and 2.16 demonstrate the delay time between controllers. The delay in different part of the system can be defined as follow:

1- T_{pre}: The time required to process control signal from the outside world in the controller (analog to digital conversion is most important one).

2- T_{wait}: The time that the imported data shall be buffered in the sender's data frame until it reaches the network link.

3- T_{tx}: The duration of data frame transmission from one node to another node. This delay can be very different in master to slave and slave to master modes. The major part of the T_{delay} is this part because it is out of the controller and in the network link.

4- T_{post}: The delay time after data frame has been received in the destination. It consist of packet parsing loop procedure and the delay until measured signal is used in the main control loop.

Based on the defined delays, the final delay would be:

$T_{delay} = T_{pre} + T_{wait} + T_{tx} + T_{post}$ The total delay from the signal to the control loop can be used in modeling. The difference between single and distributed controller system is in two facts:

1- Network delay: There is always delay between measured signal to the control loop and from the control loop to the plant.

2- Rounding error: Due to speed and latency requirement of the converter, variables should be rounded to decrease the overload on the network.

Therefore, the finalized model to be used in simulation would be like Figure 2.17.

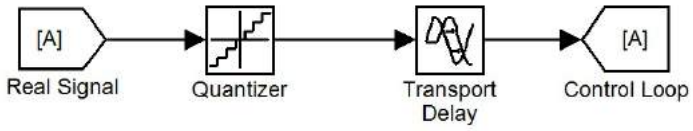


Figure 2.17: Signal to Controller Simulation Model

Chapter 3

Fault-tolerant Controller Architecture for Modular Multi-level Converters

3.1 Introduction

There are different factors that can limit the functionality of a controller. All controllers are made from components and materials that have limited lifetime. External factors and working environment play their role to decrease the expected life of the designed system. Even human errors in software coding can cause failure in some exceptional situations. Although it is not possible to avoid failure in the controller, it is possible to lower the failure rate of control system by changing its architecture. There are several types of fault-tolerant controllers and each one is suitable for specific application and based on the requirements and available resources, the best controller can be chosen. In this chapter, different controller architectures will be investigated and a novel architecture for modular multi-level converters will be proposed. It has distributed architecture and failure in one of the module can be handled by the adjacent controller. Therefore, the failure rate of the final control system would be lowered by a great factor.

3.2 Design Techniques for Fault-tolerant Controllers

Implementation of a fault-tolerant controller must be realized both in hardware and software. In the hardware design, the designer must use the components with low failure rate and long expected life-time. This would increase the mean time to failure (MTTF) of the designed controller board. The other help from designer would be choosing the right controller architecture that can handle the failure conditions. Majority voting redundancy and standby controller are two major techniques that is used to gain fault-tolerant controllers.

3.2.1 Hardware Architectures for Fault-tolerant Controllers

In recent years, the cost of the hardware has decreased a lot and compared to importance of tasks that is done by the controller, it is still logical and economical to use extra hardware in the system design. Therefore, most of the controllers use redundant hardware to achieve fault-tolerancy. There are different ways to arrange the controllers, find the fault in system and rearrange them if a failure happens in the system. The common property between all fault-tolerant controllers is that they try to atleast avoid single point of failure in the system. Some controllers may even be capable to stay functional if more failure happens in the system.

Based on the failure handling method, fault-tolerant controllers can be divided into **static** and **dynamic** categories[94]. Static controllers try to mask the fault using voting methods. There would be several controllers and each one generates output signals based on the input signals it is reading. The voting module (which may consist of different blocks) compare the results and generate the majority signal. In dynamic controllers, fault is detected by the controller itself or the spare controller in the system. Whenever fault happens, the active and passive controllers will switch the roll and the passive controller will take the control of the system. There is also

another architecture (Hybrid) that combines two methods and add extra module in the system for replacement in the case of failure.

Static Redundancy: In this architecture, a masking algorithm is used to detect the fault in the controllers. The minimum of three controller is needed to form a static controller and it is called triple modular redundancy (TMR). The same inputs are given to all three controllers and it is desired that all three generate the same result. The output of the controllers is given to the voting mechanism which compares output result and select the one which has higher vote. In this method, if one controller fails, the other two are still operational and the output result would the result from these controllers.

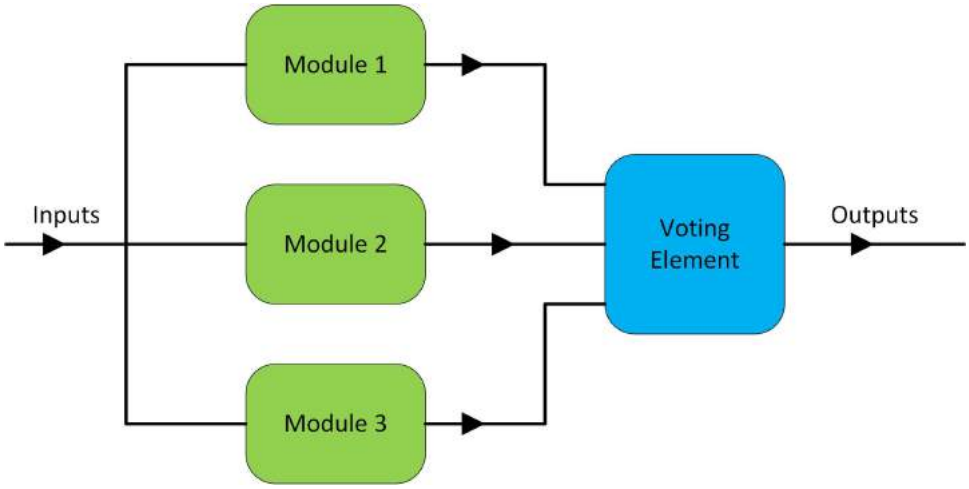


Figure 3.1: Architecture of Triple Modular Redundancy

Figure 3.1 show the image of a basic TMR controller. The main problem in this configuration is that if any fault happens in the voting section or input signals, the whole controller would fail too. This is called single point of failure and to overcome that, it is necessary to duplicate the inputs and voting sections.

The idea of static controllers can be extended to N module redundancy, and the controller is

called M of N modular redundancy controller (NMR). In NMR controllers we have:

$$Maximum\ Number\ of\ Controller\ Failure = \frac{N-1}{2}\ ,\quad N \in \{3,5,7,9,\ldots\} \tag{3.1}$$

Adding more modular controller will increase the fault-tolerancy of the system and increase the cost of total system. It will also complicate the voting module exponentially because of the number of comparison required to find the majority result. In practice, more that 5 modules is not being used for static redundant controllers.

Dynamic Redundancy:In static controllers, masking method is used to detect the faults. This method requires huge amount of redundancy in which minimum of three modules is required for operation. In static systems, instead of masking, the fault detection method is used. In the case of the fault detection, the active system would be changed and a standby module would be replaced instead of the active module. Fault detection can be internal or external using hardware or software resources. As it shown in figure 3.2, this system can operate with the minimum of two modules. The output result of each controller is checked to make sure the active controller is working properly and otherwise the standby controller would be switched over.

The standby module can operate and reconfigured in two method. In **hot** standby, the spare module is operating continuously with the active module. In the case of the failure, it can be replaced immediately with the active module, therefore decreasing the response time to the failure. The disadvantage is the constant power consumption when it is not useful. In the **cold** standby, the spare module is powered down and will be activated in the case of failure detection. In the table 3.1, a comparison between static and dynamic redundancy has been done. Each method has it's own application and the proposed architecture for modular converters, the best architecture would be used.

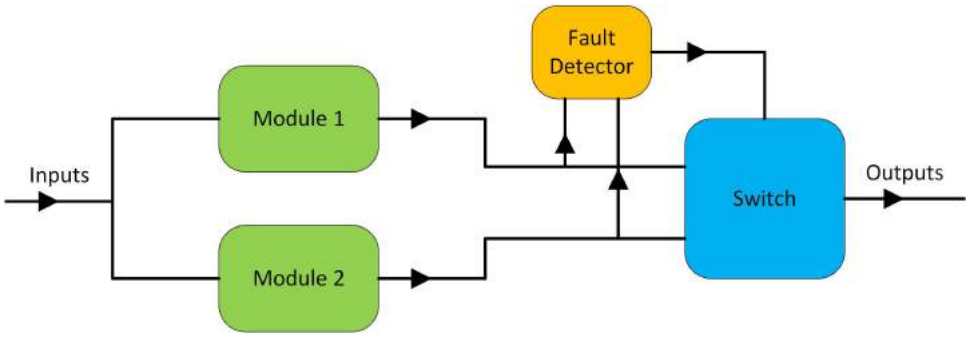


Figure 3.2: Architecture of Standby Redundant System

Table 3.1: Comparison Between Static and Dynamic Redundant Controllers[92]

	Static Redundancy	Dynamic Redundancy
Resource utilization	Resource intensive:Resources are always use, regardless whether faults are present or not. Faults are tolerated by fault masking	Resource friendly:Resources are used on demand in presence of faults by reconfiguration
Timing behavior	Requires no failover time, i.e. the time consumption is low	Additional failover time required
Reliability	Provides highest short term reliability	Provides high long-term reliability

3.3 Re-configurable Controller Arrays and their Application in Power Electronics

The idea of developing a distributed fault-tolerant controller for modular multi-level converters has originated from fault-tolerant array processors. These VLSI chips contain a mesh of interconnected processing elements that can be configured to compute Regular Iterative Algorithms (RIA). These algorithms are useful for high-speed Digital Signal Processing (DSP) of complicated signals (e.g. 2D filtering). In case of failure, the array can be reconfigured to do the same algorithm by bypassing the faulty elements. In the following section, a new controller architecture will be evolved from such arrays to add fault-tolerant capability to the modular multi-level converters.

3.3.1 Fault-tolerant Array Processors for Regular Iterative Algorithms

During the lifetime of a VLSI system, it is probable that the wafer become faulty in some parts. Instead of treating the whole chip as defective, it is possible to reconfigure the architecture of chip to implement the same algorithm as before. An example for such system is processor arrays that consists of a grid of N(N+1) processing elements (PE). Based on the research of T. Kailath and his research team in 1991 [103] [105], it is possible to reconfigure the faulty system into (N+1)N elements and bypass the failed PE. In this case, $N \times N$ of the elements are functional and N modules are faulty. Figure 3.3 demonstrate the array processor proposed by Thomas Kailath and his team. Each element in the grid can be connected to another element by data bus through configuring switches. In the proposed array, there is only one set of data track available between to switches and the reconfiguration algorithm must connect the functional elements using single-track switches [45] [53]. The factory configured array is called *physical*

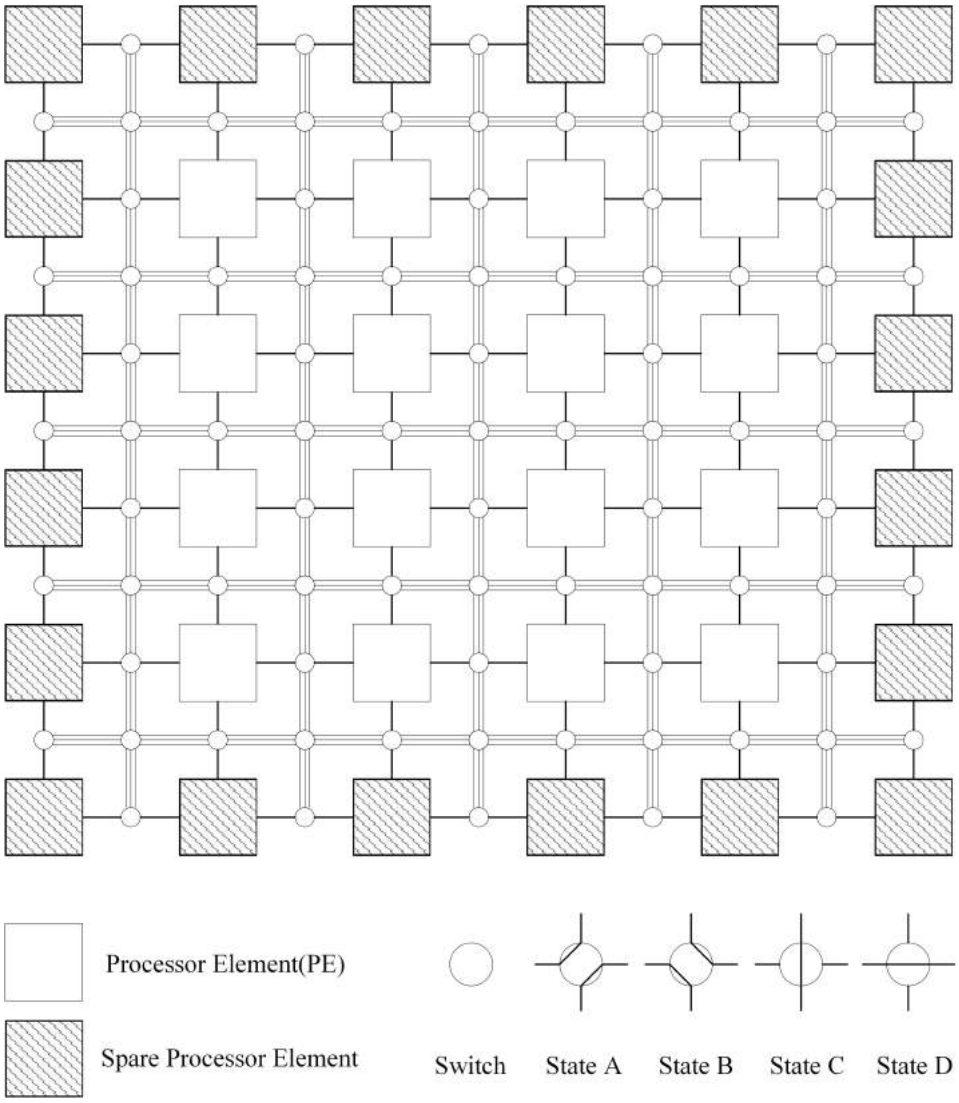


Figure 3.3: Architecture of Array Processors Proposed by Kailath

array and the reconfigured fault-free array is called *logical array*. Both physical and logical arrays are the same at factory and they would be different after fault happens. Based on the reconfigurability theorem, an $(N+2) \times (M+2)$ physical array is configurable to $N \times M$ logical array using one-track routing if and only if there exist compensation paths covering all faulty PEs and they are continuous, straight, non-intersecting and non-overlapping. The proof of the

theorem can be found in [44].

Reconfiguration process includes placement and routing. When placement is fixed (like VLSI

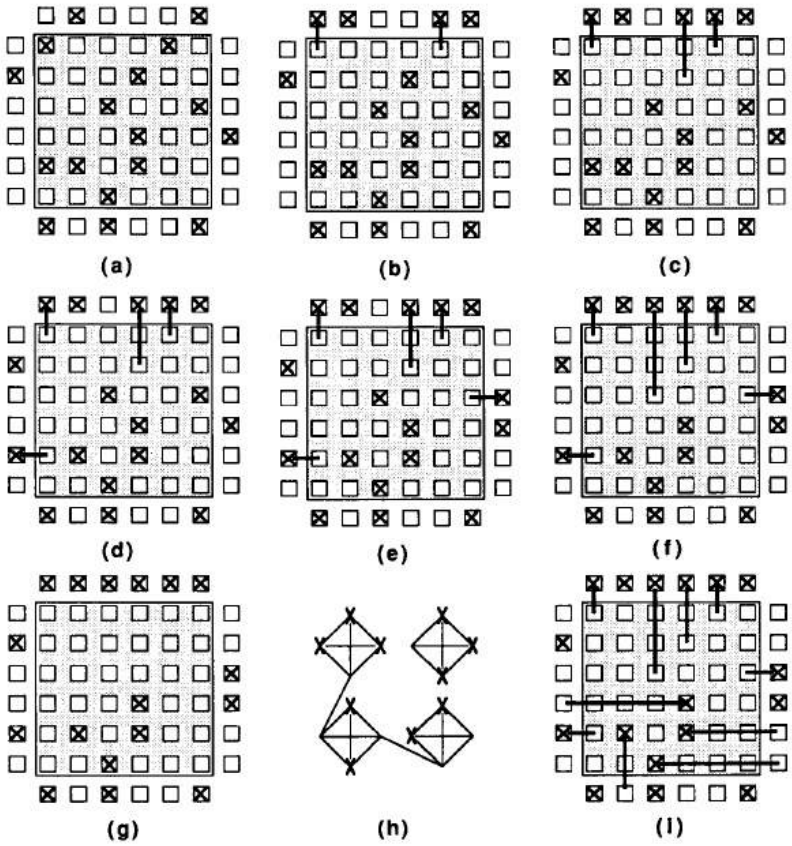


Figure 3.4: (a)-(f)Screening process, (g)Equivalent fault pattern after screening, (h)Contradiction graph for (g), (i)The placement solution

chips), the faulty elements must replaced with spare elements through re-routing. This process consist of two stages[83][91][64]:

- Screening: Determining the obvious choice of the compensation path for as many faults as possible (figure 3.4)

- Contradiction graph: By constructing the contradiction graph (figure 3.5) for the remaining (un-screened) faults and finding the maximum independent set equal to F (number of non-spare failed elements).

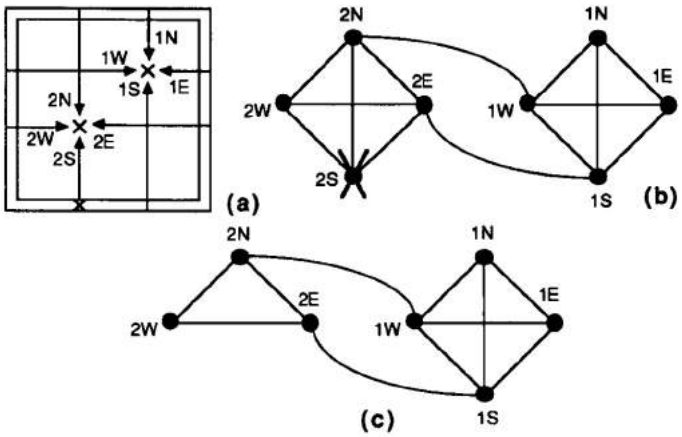


Figure 3.5: (a)Fault pattern, (b)Corresponding contradiction graph with a "crossed" vertex, (c)Reduced equivalent contradiction graph

After screening process and assigning the logical index of failed elements to the spare parts, the equivalent fault pattern with less number of non-spare faulty PEs and more spare faulty PEs will be formed. The contradiction graph for the remaining failed PEs can be formed as steps below:

- *Generation of Vertices*: For each non-spare faulty PE (e.x. k-th faulty PE), four vertices can be assigned (i.e. kN, kS, kE, kW). Therefore four compensation paths are available per failed element. Furthermore, the "crossed" vertices which correspond to non-valid paths must be deleted.

- *Construction of Edges*: For each vertex pair (u,v), an edge may connect them if they are "non-intersecting", "non-overlapping" and do not belong to the same faulty element.

To find the replacement spare PEs for the faulty elements, the **Maximum Independent Set** of the contradiction matrix must be found. That means finding the biggest independent set of vertices that are not connected to each other by an edge. Such problem is classified as NP-hard optimization and it is unlikely that there exist an efficient algorithm for finding the set.

An efficient method for finding the maximal independent set is proposed by Bron and Kerbosch in [15] which is a tree search method. Algorithm consists of *forward step* and *backward step*. In a forward step (e.g. k-th step), an independent vertex set S_k is augmented by another proper vertex to produce new independent set S_{k+1}. In the best condition, the forward step continues until the set become *maximal* independent set. Otherwise, backward step has to be used to achieve the goal. Two independent sets ${Q_k}^-$ and ${Q_k}^+$ are used for this algorithm. For solving the routing problem, only one maximum independent set with the size of F is enough. After finding this set, the rest of the faulty elements can be assigned to the corresponding spare cells.

In the reconfiguration problem, only PEs are considered faulty. Failure in the switches or the connection path is possible. In order to re-configure the array with such errors, PEs status must be re-assigned equivalent to having fault in the connected PEs. Therefore no bypass track can pass through the faulty connection and that part will be left alone.

3.3.2 Evolution of Fault-tolerant Distributed Controlled Power Modules

The same idea of array processors can be applied in power electronics. In multi-level converters, a set of converter modules are connected to each other to form an architecture. Figure 3.6 demonstrate a grid of controllers and power module connected to each other to form a fault-tolerant converter. Each power module has connection to 4 controller (2 main controller and 2 auxiliary controller). If one controller fails, another controller can controller the power module and if a power modules fail, it is possible to re-route the from one power module to another power module.

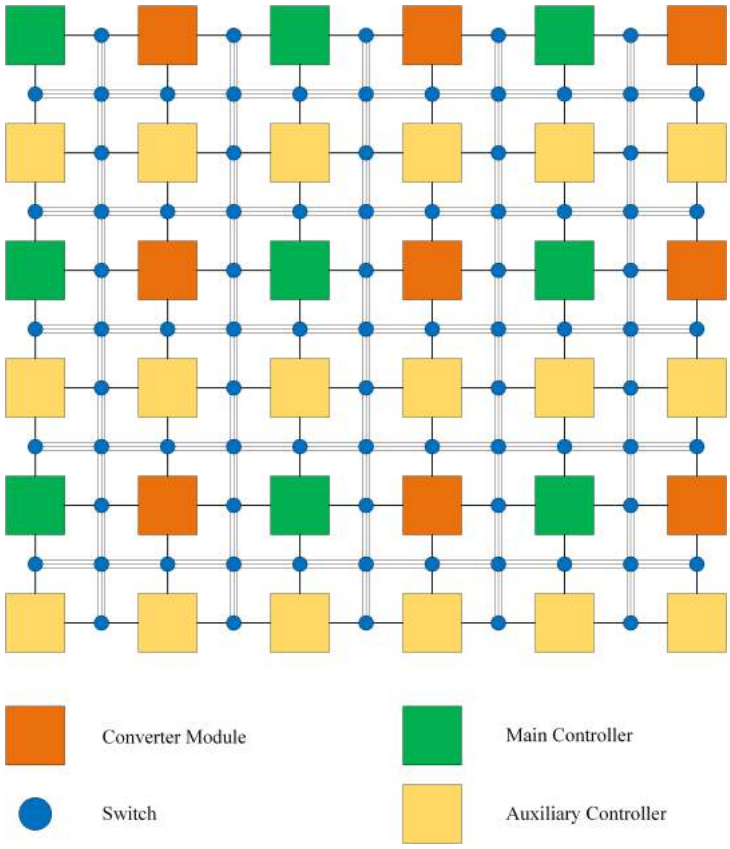


Figure 3.6: Distributed Controller and Converter Module over Meshed Network

In figure 3.6, there are auxiliary controllers in the grid that have no direct access to the power modules. For connecting such controllers to the power modules, 3 switch and 2 data bus must be used. Due to cost and complexity of the data track and the controllers, these controllers may be omitted. The result has been depicted in figure 3.7. Each power module in the grid still has connection to 4 controllers and they can communicate to each other through the data bus.

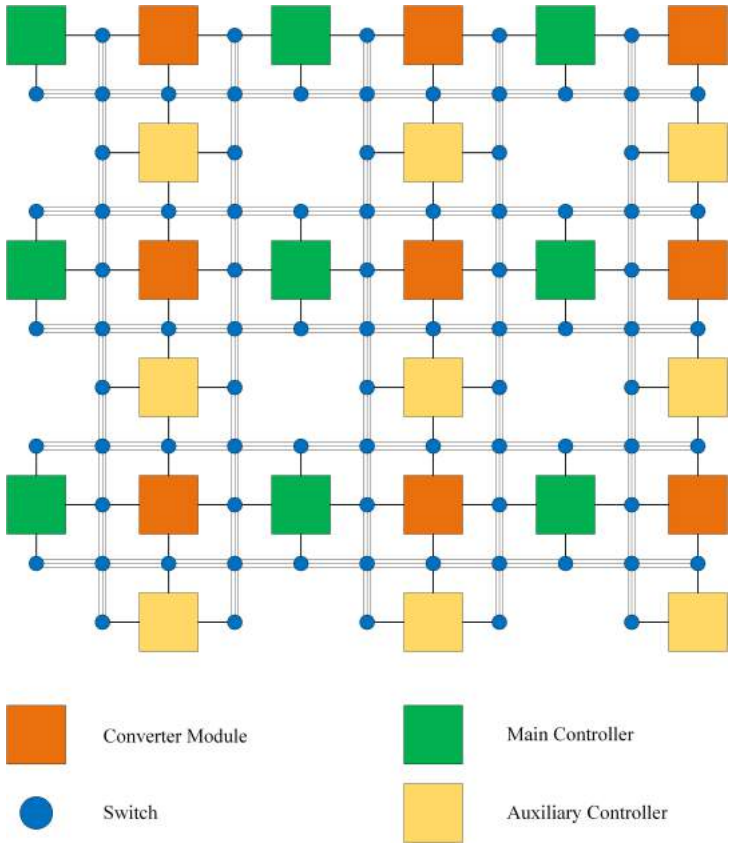


Figure 3.7: Omission of Unusable Auxiliary Controllers from the Grid

The auxiliary controllers can be embedded inside the main controller. This will save resources in routing data bus and switch. As it can be seen in figure 3.8, each controller block has one main controller and two auxiliary controllers for other power modules. Figure 3.9 shows the

grid without spacing between modules.

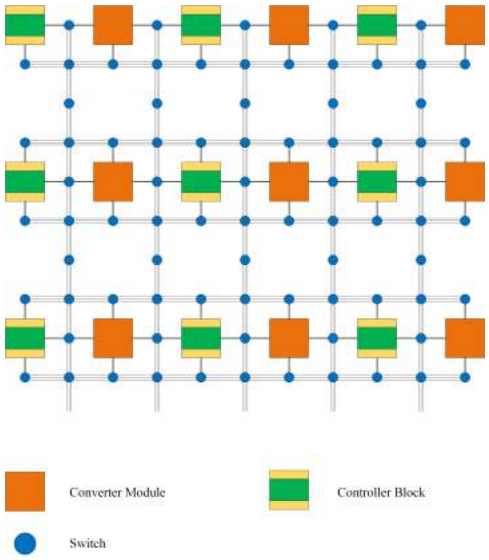


Figure 3.8: Embedding the Auxiliary Controllers Inside the Main Controllers

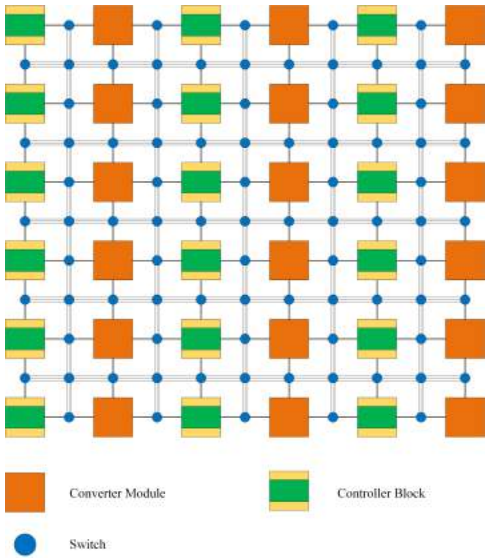


Figure 3.9: Omission of Unused Space and Integrating the Power Modules and Controllers

There are limitations in power electronics that makes integration of controllers and converters different than integration of processing arrays on a chip. These difference can be list as below:

- Due to the high-voltage nature of the power converters and the requirements for isolation, power modules on one converter leg may not be connected to the controllers on another converter leg. They must be separated and there should be a gap between them.
- Controllers must be synchronized to a master controller in order to get shared variables (e.x. grid voltages, currents, phase ...). It is not possible for all of the distributed controllers to read the analog signals to complete the tasks.

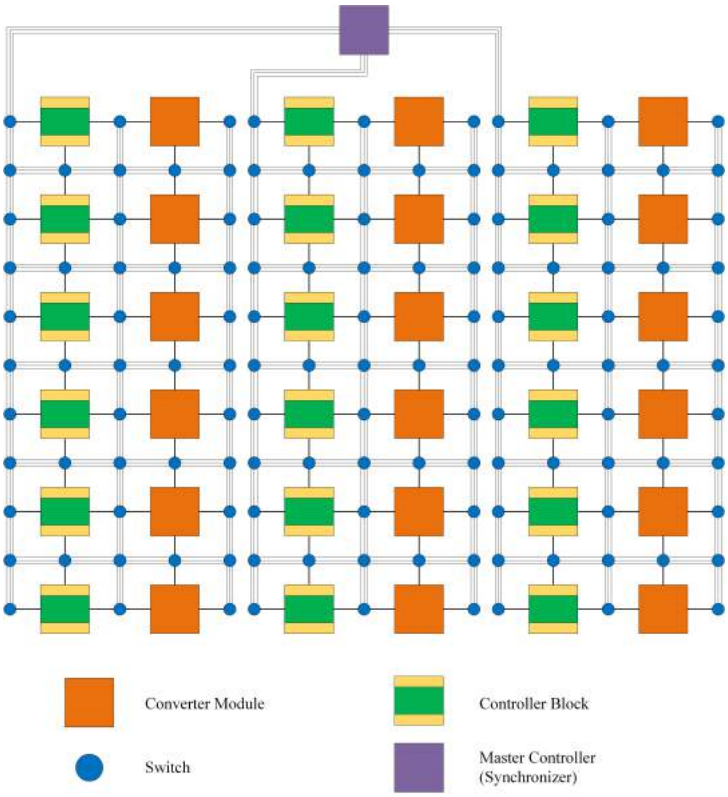


Figure 3.10: Architecture of Customized Array Interconnections for Power Electronics Application

Based on the power electronics requirements, power modules are isolated, controllers are synchronized with a master controller and the revised architecture in figure 3.10 can be achieved.

The last parameter that must be considered is the switch for the power modules and the controller signals. In array processors, the switch is a multi-track data switch that has been implemented by logical gates. In the new architecture, the switch for the power modules is usually a power device which can be turned on to bypass the power. Due to the cost of such switches, it is not economical to complicate the high-power switch.

The switch for controller may still be implemented by logical gates. Since there are three controllers connected to each power module, each controller switch may have 3 states. By default, power module is connected to the closest controller and in the case of failure, power module will be switched over to another controller.

Figure 3.11 demonstrate the final architecture of controllers and power modules in 1^{st} generation of the fault-tolerant controller. In the following sections, the implementation of proposed architecture will be discussed in details.

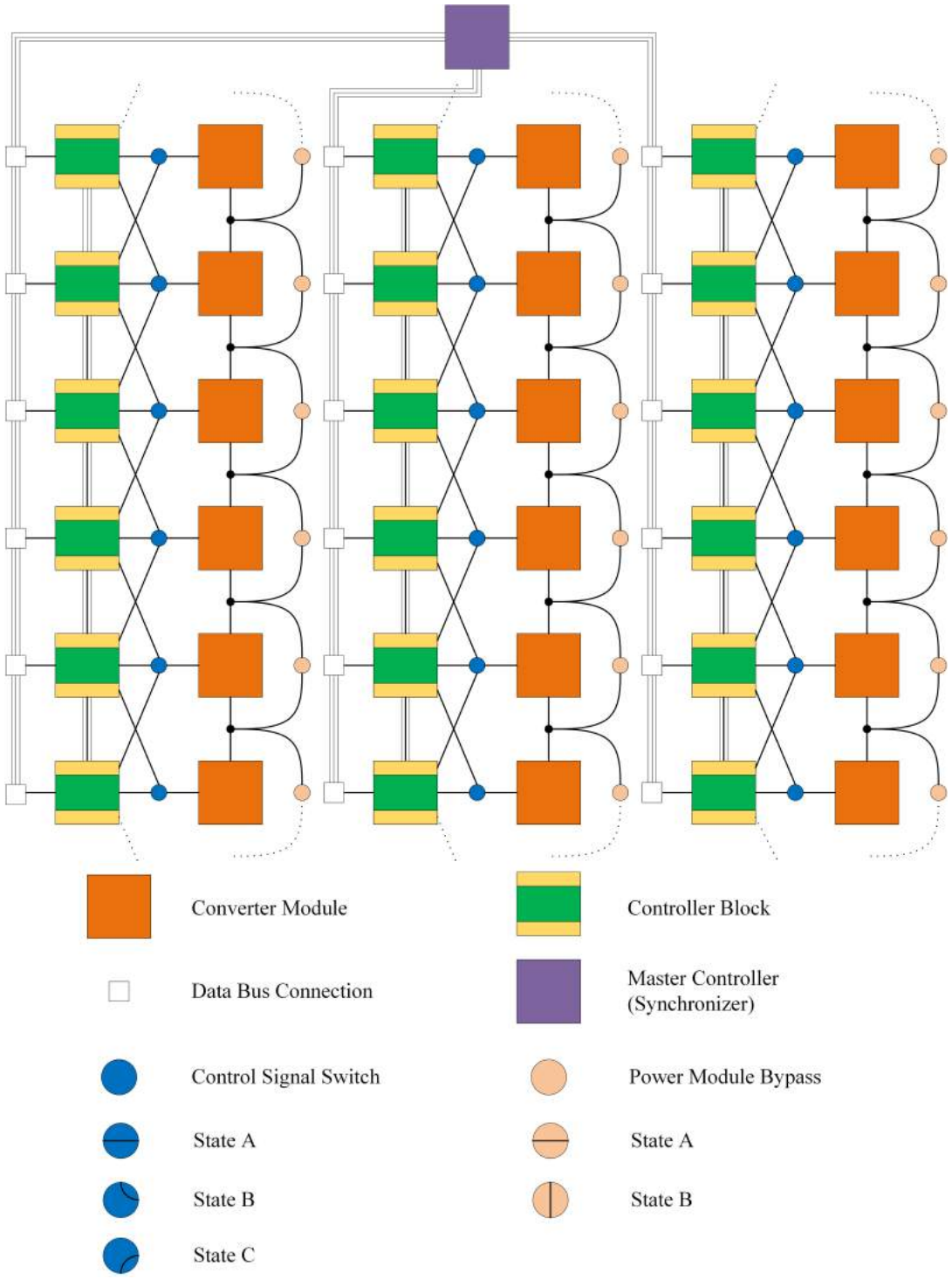


Figure 3.11: Architecture of 1^{st} Generation Fault-tolerant Distributed Controlled Power Modules

3.3.3 Second Generation of Fault-tolerant Distributed Controller

One of the biggest disadvantages of the 1^{st} generation controller is the synchronization method used for the controllers. The global data (i.e. grid voltages, currents, ...) is being acquired by the master controller (synchronizer) and shared between all other controllers through the communication bus. In this case, the master controller must have redundant architecture (static or dynamic) with redundant communication bus to avoid single point of failure. This will make the architecture complicated and increase the required resources for implementation of the system. One of the solutions for this problem is connecting all the controllers in a grid format. In this case, some of the controllers are capable of gathering the global variables and can share them to other controllers through the communication bus. If one of the controllers fails, another one of the controller can take the responsibility and synchronize the whole controllers (figure 3.12). There are two types of controller in this architecture. Some of the controllers have measurement devices for measuring common variables between all the controllers. only one of these controller will act as the master controller to synchronize all other controllers. The controllers that have access to global variables will perform voting algorithms to ensure the correctness of the module that is synchronizing other controllers. If the master controller fails, the next controller will take responsibility of the master controller.

The connection between controllers is point to point and can be realized by fiber optics. This will increase the speed and the performance of the communication link. In this architecture, there is no single point of failure because minimum of two (maximum of four) communication link is available for each controller. The synchronization packet (which contains global variables) will be sent at the start of each control step time. In assigned time for synchronization, the incoming data from the serial bus will be copied to other serial ports. This method will propagate the synchronization packet to all of controllers in the minimum amount of time (like domino tiles).

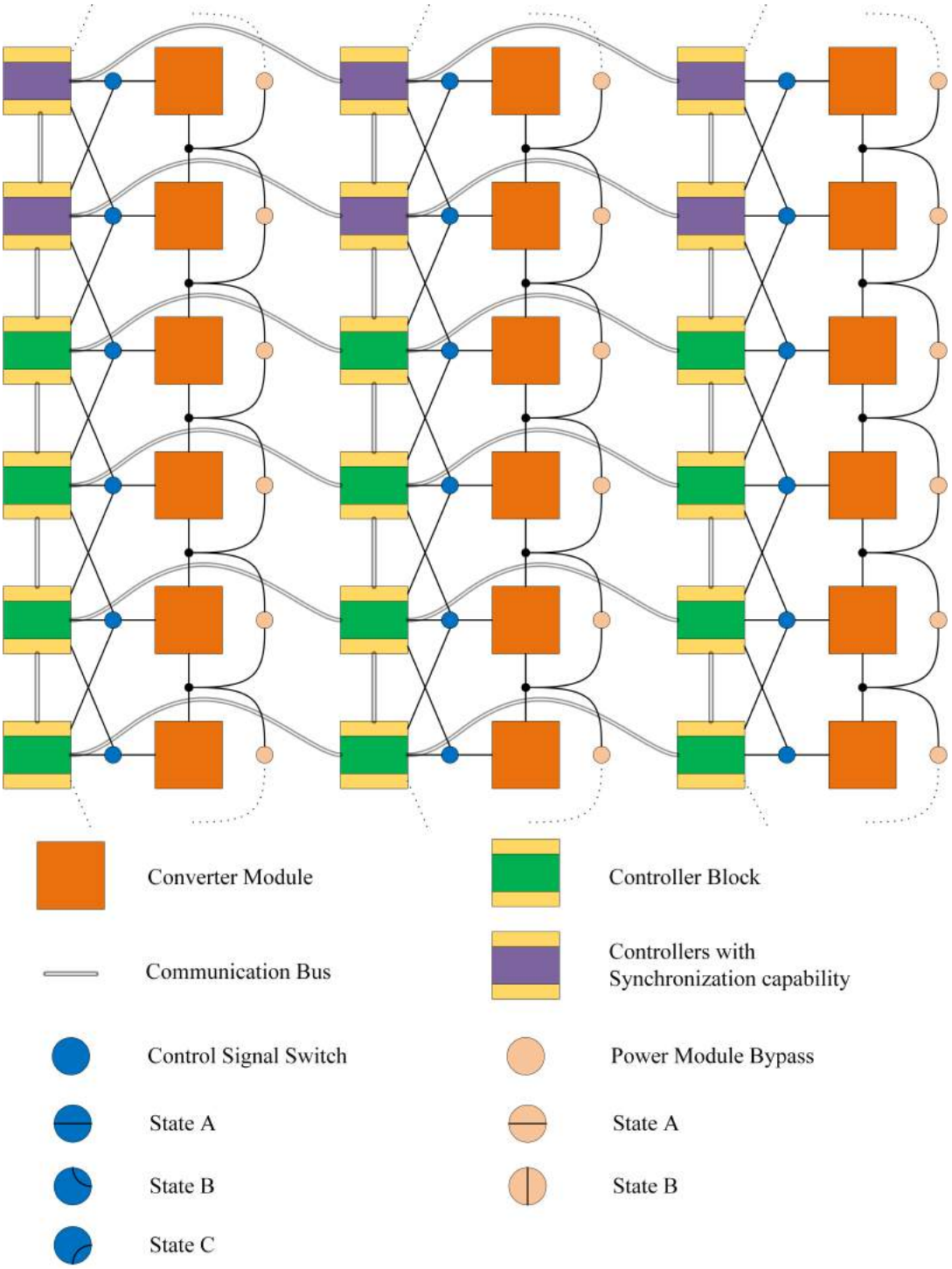


Figure 3.12: Architecture of 2^{nd} Generation Fault-tolerant Distributed Controlled Power Modules

3.4 Proposed Distributed Fault-tolerant Controller for Modular Multi-level Converters

Based on the methodologies for designing fault-tolerant systems, a distributed controller for MMC is proposed. The proposed controller has the capability to reconfigure itself in the case of failure. The restructured converter, can work continuously without interruption by bypassing the faulty modules and dividing the workload over other modules. Different methods have been used to detect the fault in the controller and converter modules and after detection, a signal would be sent to the responsible control block. Based on the failure type in the system, a module might be bypassed completely or the adjacent modules take control of the faulty module (fail-over decision). The reconfigured system will continue its operation and in the meanwhile, the faulty module can be taken out and be replaced.

3.4.1 Architecture of Fault-tolerant Controller for MMC

The proposed fault-tolerant controller is a distributed controller based on dynamic controller architecture (figure 3.20). Dynamic controllers are much simpler in implementation and give better result for the same hardware resource in power electronic system. Since static controllers use masking to find faults, it would be complex to mask the output PWM signals and find the errors. For the minimum static fault-tolerant controller, three controller is required which can tolerate only one failure. In a dynamic controller with three modules, two failure in three adjacent modules is possible. Therefore dynamic fault-tolerant controller is the best option for the proposed controller [75][29][24][87].

In this controller, each module shares the input signals with the adjacent module and they can communicate through data link and flag bits. All of the modules are being synchronized with

the master controller through the shared data link. The data on main communication link is responsible to synchronize modules of a single leg to the reference values of that particular converter leg. There is also a communication link between two adjacent modules, therefore they can transfer flag bits and control variables with each other. The switch block is responsible to

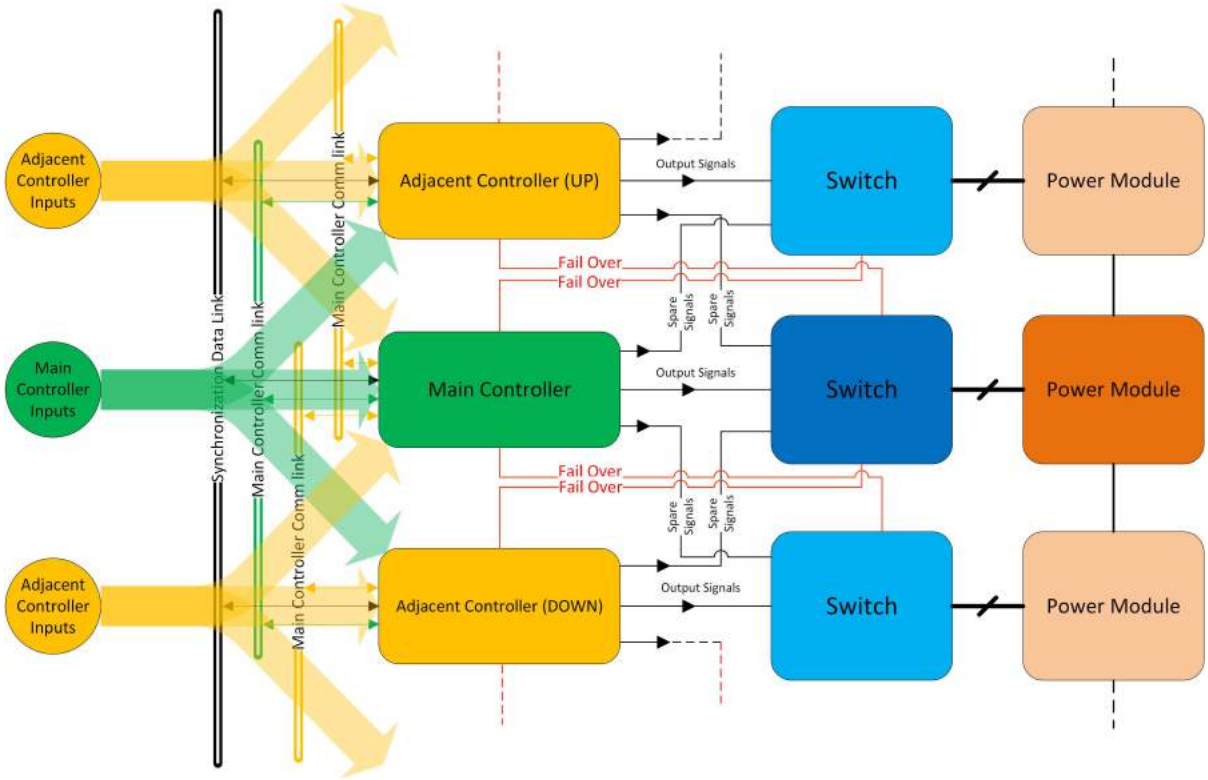


Figure 3.13: Architecture of Proposed Distributed Controller for Modular Converter

select the output of the best functional controller and feed it to the power devices. The selection is based on the fault signals and condition of the main and adjacent controllers. In case of a failure in all of the three available modules, it bypasses the power devices so the converter can continue its operation.

3.4.2 Fault Detection in Converter Modules

There are different techniques to discover fault in the system. The major fault-detection methods can be categorized as **built-in self-test**, **output results comparison** and **watchdog timer** [30][31][6][73][76][32][47].

- **Built-in self-test (BIST)**: In this method, an auxiliary circuit (inside the module) checks the health of the operational components based on the defined behavior of each one[17]. If the measured parametric values are different than the critical threshold, a fault signal can be generated for the component. For the proposed controller architecture, it is necessary to make sure each of the power switches in H-bridge are functional. In this case, IGBT is used as the switching device and whenever gate voltage is higher than a threshold, it must turn on (saturation region). Figure 3.14 show an example for fault detection in IGBT[52]. Base on the gate voltage and collector-emitter voltage, it can detect if IGBT is turning on-off at the desired moments (collector-emitter voltage of IGBT must be near zero when turn on and higher than that when turned off).

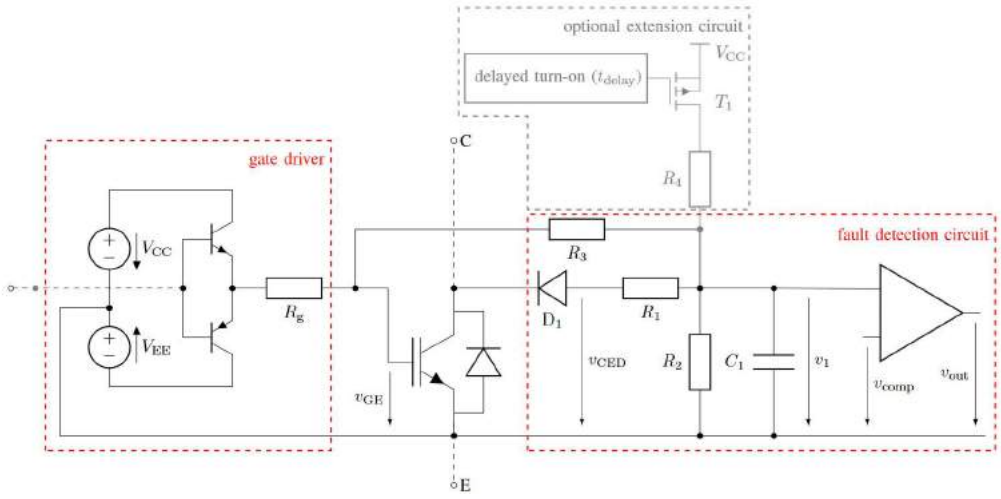


Figure 3.14: IGBT Fault Detection Based on Desaturation Checking

- **Output results comparison**: The functionality of the controller board can be checked by comparing the output result of adjacent modules when feeding the same inputs to all of them (similar to voting mechanism in static controllers). In this case, any failure in the measurement sensor, control block and output signal generation can be detected. The difference between this method and voting mechanism is that here each adjacent module compare the output result with the main controller and there would be 2 fault indication signals coming to each module. Based on the fault signals, the main module can be bypassed or its control can be rolled over to the adjacent modules. Figure 3.15 demonstrate this method in the proposed controller.

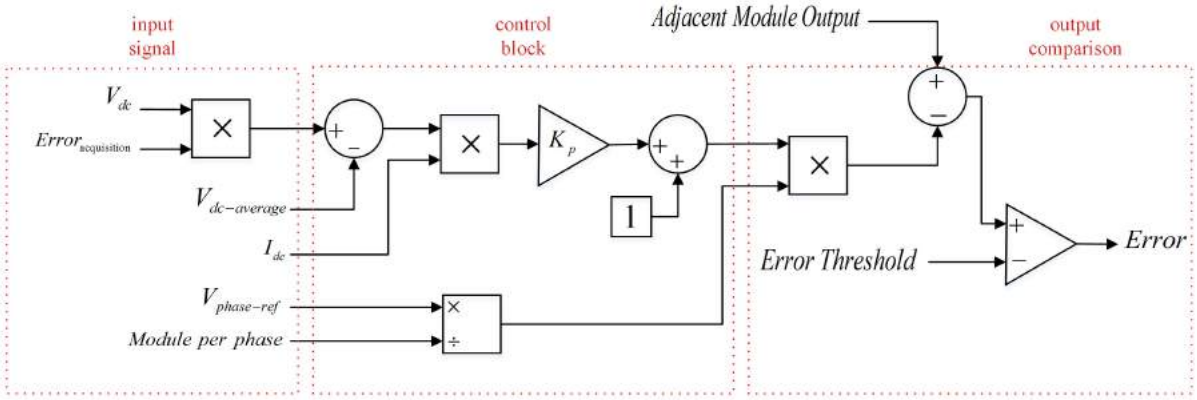


Figure 3.15: Output Comparison Between Adjacent Modules

- **Watchdog timer**: Some control tasks are time critical and delayed response means corresponding control section has been failed. In the proposed controller, a watchdog timer can detect any disconnection between the main controller and adjacent controllers. If there is no update from the adjacent module, it means either the control module or the communication bus has failed. Figure 3.16 depicts the watchdog implementation.

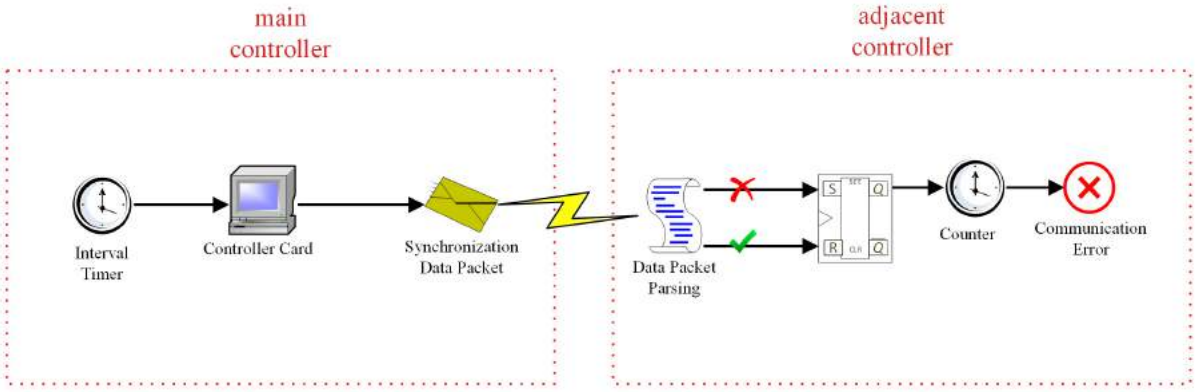


Figure 3.16: Watchdog Timer for Detection of Communication and Controller Failure

3.4.3 Fail-over Strategy

In the event that a failure happens, the output must be switched from the failed module to a standby module (in dynamic controller). This event is called fail-over and decision about how to make this transition is based on the fault signals and condition of the adjacent modules. Fail-over process consists of two steps:

- 1- Fail condition detection: Based on the fault signal check if the fault in the system is fatal and the main controller is no longer operational.
- 2- Switch-over to standby controller: Choosing the right standby controller to take control of the main controller depending on the status of the standby controllers.

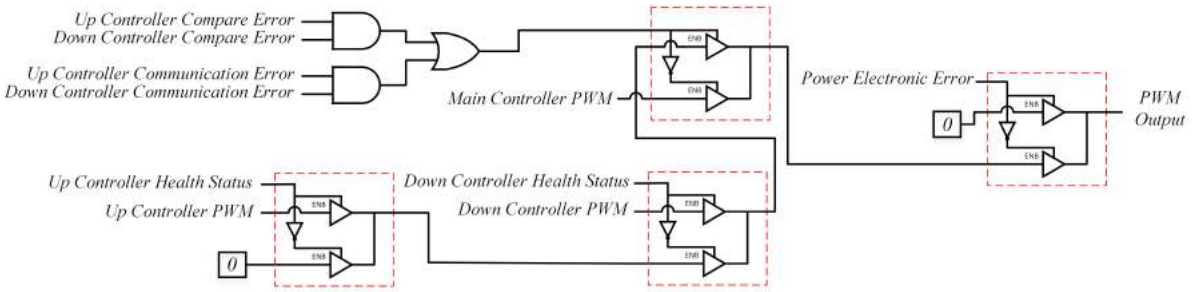


Figure 3.17: Fail-over Strategy Diagram in each Module

Figure 3.17 demonstrates the strategy of the proposed controller. There two situations that means the main controller is not functioning correctly:

- 1- Comparison between output of main controller and adjacent controllers are different. If both comparison are wrong, then it means the main controller is not creating the correct output signal since it is reading the wrong input signal or the microprocessor which is calculating the control block is damaged.

- 2- The synchronization between adjacent modules experience timeout. If both of the modules experience timeout, then it means the main controller has no supply voltage (power failure) or the microprocessor is experiencing a failure (e.x. stack overflow).

Whenever each of these conditions come true, the fail-over signal is turned on and the best controller would be chosen to replace the main controller. Inside each module, a circuit checks the health status of the module and feeds the signal to adjacent modules (figure 3.18). Based on this status signal, a healthy controller module would be selected and if there is no available module to handle the converter, the module would be bypassed.

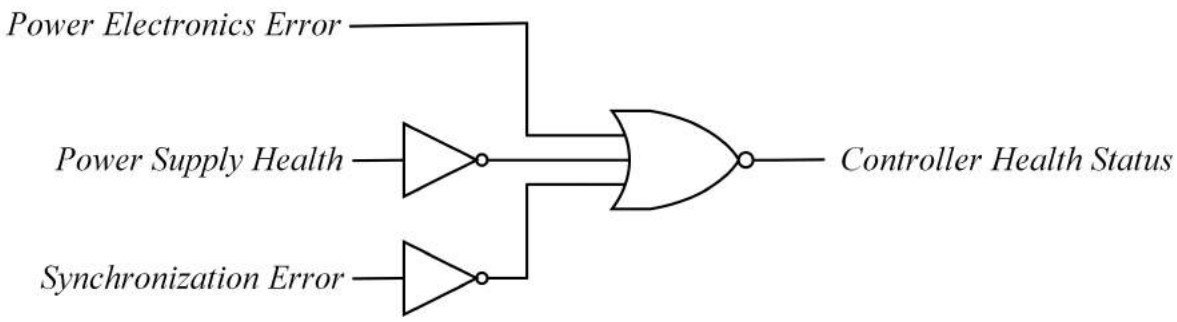


Figure 3.18: Diagram of Controller Health Indicator

3.5 Mathematical Model of the Proposed Fault-tolerant Distributed Controller

Modeling the proposed controller will help to understand the behavior of the system. In the reliability assessment, the mathematical model will help to implement Monte Carlo simulation. Therefore, it is necessary to systematize the controller model for different applications. In the first section, the performance model will be reviewed. This model tells if the controller is still available and functional which will be helpful in Monte Carlo simulation. The other model is re-configuration matrix of the controller which assigns each controller to the corresponding power module based on the faults in the controllers or the power modules. The following nomenclature exist in system model:

- F_{BIST}: $n \times 1$ matrix that contains the failure status for built-in self test in the controllers
- F_{OC}: $n \times 1$ matrix that contains the failure status for output comparison failure detection
- F_{TO}: $n \times 1$ matrix that contains time-out failure status of controllers
- F_{SOFT}: $n \times 1$ matrix that contains soft failures in the controller blocks
- F_{POWER}: $n \times 1$ matrix that contains failure status of power modules
- F_{MODULE}: $n \times 1$ matrix that contains failure status of each module
- U_{PHASE}: number of the completely failed modules per phase (unavailability of each phase)
- U_{MAX}: maximum number of permitted module failure per phase (maximum permitted unavailability)
- CAM: $n \times 1$ control availability matrix (CAM) that contains availability of controller for each power module (0 as unavailable and 1 as available)

- S_P: $n \times 1$ matrix in which each element represent the flag for the power bypass switch
- S_C: $n \times 1$ matrix in which each element represent the routing path for control signals of each power module. 0 means that control signals come from the main controller, +1 for upper and -1 for the lower controller.

3.5.1 Performance Modeling of the Proposed Controller

The proposed controller is designed to handle limited amount of failure and still function continuously. The performance model of the controller will check all the failures in different sections of the system to form the performance matrix. The soft failures in the controllers can be calculated as following:

$$F_{SOFT_{i,1}} = (F_{BIST_{i,1}}) \vee (F_{OC_{i,1}}) \vee (F_{TO_{i,1}}) \tag{3.2}$$

Due to the implemented fault-tolerant controller algorithms, control of each power module may be done by the directly-connected controller or the adjacent controllers. The controller availability matrix (CAM) tells if there is any available controller for the desired power module as given below:

$$CAM_{i,1} = \begin{cases} 0 & \text{if } [(F_{SOFT_{i-1,1}} \wedge F_{SOFT_{i+1,1}} \wedge F_{SOFT_{i,1}}) = 1] \\ 1 & o.w. \end{cases} \tag{3.3}$$

A totally failed module has no available controller or the power module has been failed. Therefore:

$$F_{MODULE} = (\overline{CAM}) \vee (F_{POWER}) \tag{3.4}$$

The number of the unavailable modules per phase will be:

$$U_{PHASE} = \sum_{i=1}^{n} F_{MODULE_{i,1}} \tag{3.5}$$

If $U_{PHASE} \geq U_{MAX}$, then the converter may not function for the desired rating. In this case, the system has failed completely.

The discussed model is helpful in reliability assessment of the controller system and will be investigated more in reliability assessment chapter.

3.5.2 Re-configuration Matrix of the Proposed Controller

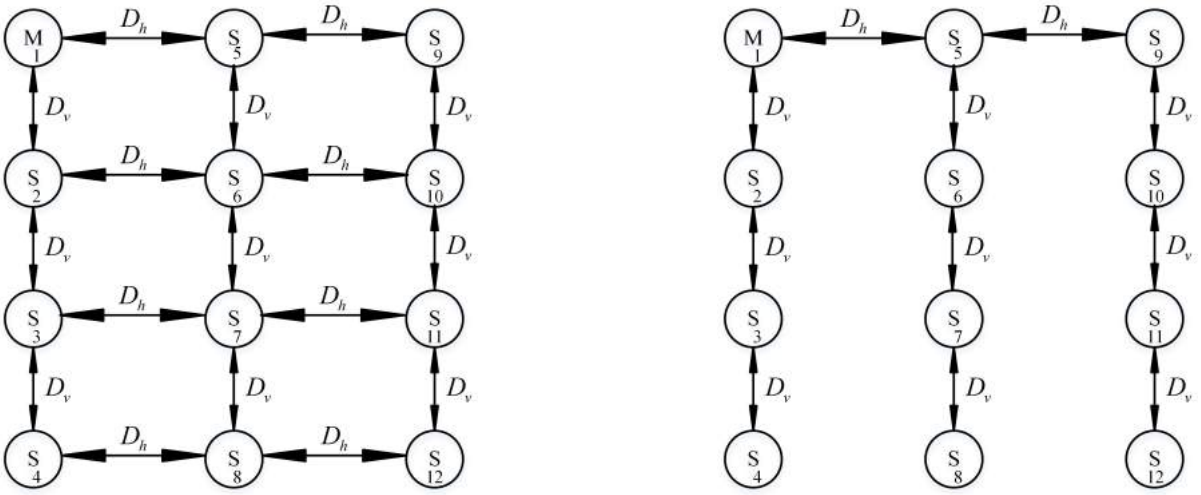
The reconfiguration matrix (S_P and S_C) represents the current connection style for each set of switches in the system. It is important to find the reconfiguration matrix in case of the failure and change the arrangement of the controllers and converters in order to bypass the fault. Whenever a fault happens, the reconfiguration is done locally and data may be reported to master controller to form a matrix for the current status of switches. The reconfiguration matrix of the system can be defined as given below:

$$S_P = F_{POWER} \tag{3.6}$$

$$S_{C_{i,1}} = \begin{cases} 0 & \text{if } (\overline{F_{SOFT_{i,1}}} = 1) \\ +1 & \text{if } [(\overline{F_{SOFT_{i-1,1}}} \wedge F_{SOFT_{i,1}}) = 1] \\ -1 & \text{if } [(\overline{F_{SOFT_{i+1,1}}} \wedge F_{SOFT_{i,1}} \wedge F_{SOFT_{i-1,1}}) = 1] \end{cases} \tag{3.7}$$

3.5.3 Re-configuration Matrix for Second Generation Fault-tolerant Controller

In the second generation controller, a matrix of controllers are connected together. In case of failure in the communication link between controllers, slave controllers or the master (supervisor) controller, the architecture should re-structure (be redoing the synchronization process) to bypass the failure. As discussed in synchronization section (2.5), the controller matrix will be transformed into a tree during the process. This enables instantaneous signal delivery from master controller to the slave controllers(figure **??**). For reliability assessment and controller



(a) Controller Matrix Before Synchronization Mechanism

(b) Controller Tree After Synchronization Mechanism

Figure 3.19: Controller Architecture During Synchronization Mechanism

modeling, it is necessary to formulate the mathematical model of the second generation controller [50]. The complete state of the controllers and communication link in system may be represented by adjacency matrix (M_{adj}). The proposed controller has horizontal (h) and vertical

(v) control elements(C_x), therefore M_{adj} would be a $(h \times v)$ by $(h \times v)$ matrix and each element $(m_{i,j})$ can be defined as following:

$$m_{i,j} \begin{cases} 0 & (\mathrm{C}_i \equiv \text{failed}) \vee (\mathrm{C}_j \equiv \text{failed}) \vee (\mathrm{C}_i \text{ and } \mathrm{C}_j \text{ have no data link}) \\ 1 & \text{otherwise} \end{cases} \tag{3.8}$$

Based on the definition, adjacency matrix of the second generation controller with 4 module per phase(3 phase system) and fully functional components is demonstrated in equation 3.10.

$$M_{adj} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.9}$$

In the controller structure, delay between horizontal(d_h) and vertical(d_v) modules are different. Therefore, adjacency matrix may not be used directly and weighted adjacency matrix(W_{adj})

must be used. Each element($w_{i,j}$) in the weighted adjacency matrix may be defined as following:

$$w_{i,j}\begin{cases} 0 & i=j \\ d_h & \text{Horizontal connection} \\ d_v & \text{Vertical connection} \\ \infty & otherwise \end{cases} \tag{3.10}$$

By definition, weighted adjacency matrix for the fully functional controller with 4 module per phase(3 phase module) would be as following:

$$W_{adj} = \begin{bmatrix} 0 & d_v & \infty & \infty & d_h & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ d_v & 0 & d_v & \infty & \infty & d_h & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & d_v & 0 & d_v & \infty & \infty & d_h & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & d_v & 0 & \infty & \infty & \infty & d_h & \infty & \infty & \infty & \infty \\ d_h & \infty & \infty & \infty & 0 & d_v & \infty & \infty & d_h & \infty & \infty & \infty \\ \infty & d_h & \infty & \infty & d_v & 0 & d_v & \infty & \infty & d_h & \infty & \infty \\ \infty & \infty & d_h & \infty & \infty & d_v & 0 & d_v & \infty & \infty & d_h & \infty \\ \infty & \infty & \infty & d_h & \infty & \infty & d_v & 0 & \infty & \infty & \infty & d_h \\ \infty & \infty & \infty & \infty & d_h & \infty & \infty & \infty & 0 & d_v & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & d_h & \infty & \infty & d_v & 0 & d_v & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & d_h & \infty & \infty & d_v & 0 & d_v \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & d_h & \infty & \infty & d_v & 0 \end{bmatrix} \tag{3.11}$$

By using single-source shortest path algorithm (e.g. Dijkstra), it is possible to find the synchronization path (if it exist) for the controller matrix.

Data: (V, E, A_{sync}, w, s) as inputs
$V_T := \{s\}$;
$A_{sync} := \emptyset$;
for $\forall v \in (V - V_T)$ **do**
    if $\exists$e $\in E(s,v)$ **then**
        $l[v] := w(s,v)$;
    else
        $l[v] := \infty$;
end
while $V_T \neq V$ **do**
    $l[u] := \min\{l[v] | v \in (V - V_T)\}$;
    $v_t = \min\{w(u,v_t) | v_t \in (V - V_T)\}$;
    $V_T := V_T \cup \{u\}$;
    $A_{sync} := A_{sync} \cup E(u,v_t)$;
    for $\forall v \in (V - V_T)$ **do**
        $l[v] := \min\{l[v], l[u] + w(u,v)\}$;
    end
end

Algorithm 3: Pseudo-code for finding the shortest path and associated synchronization delay in the Second Generation Controller

Adjacency matrix of the synchronization path helps identifying the network links which will be used in the synchronization path. However, that is not enough and the associated delay

between master controller and slave controllers is necessary for finding the maximum delay in the system. It is necessary to check if the synchronization can be scheduled or not. Therefore we have:

$$\begin{aligned} &\forall u \in V, max(l[u]) < d_{sync\_\max} \Leftrightarrow \text{system is synchronizable} \\ &T_{sync} = \tfrac{1}{F_{sync}} \\ &T_{sync} > 2.d_{sync} + t_{packet} + t_{par\text{sing}} \\ &\Rightarrow d_{sync\_\max} = \tfrac{T - t_{packet} - t_{par\text{sing}}}{2} \end{aligned} \tag{3.12}$$

In the synchronization process, data has to be sent from the master controller to slave controllers and they must response back with their status(one at the time). Therefore, synchronization delay(d_{sync}) must be calculated twice with packet transfer delay(t_{packet}) and parsing delay($t_{par\text{sing}}$) by the controllers. All of these time delay must fit in the communication period(T_{sync}) or it would interfere with the next synchronization time frame.

3.6 Simulation of Proposed Fault-tolerant Controller

Simulating controller, its software and power converter in the same simulation environment is not easy. In order to investigate the performance of proposed controller, a simulation setup based on Matlab Simulink® is prepared. The design has been partitioned to four sections(figure 3.20). The console is responsible for acquering data from different sections and set the references for control blocks. The other three blocks simulate the master controller, slave controllers and the power converter[108][13].

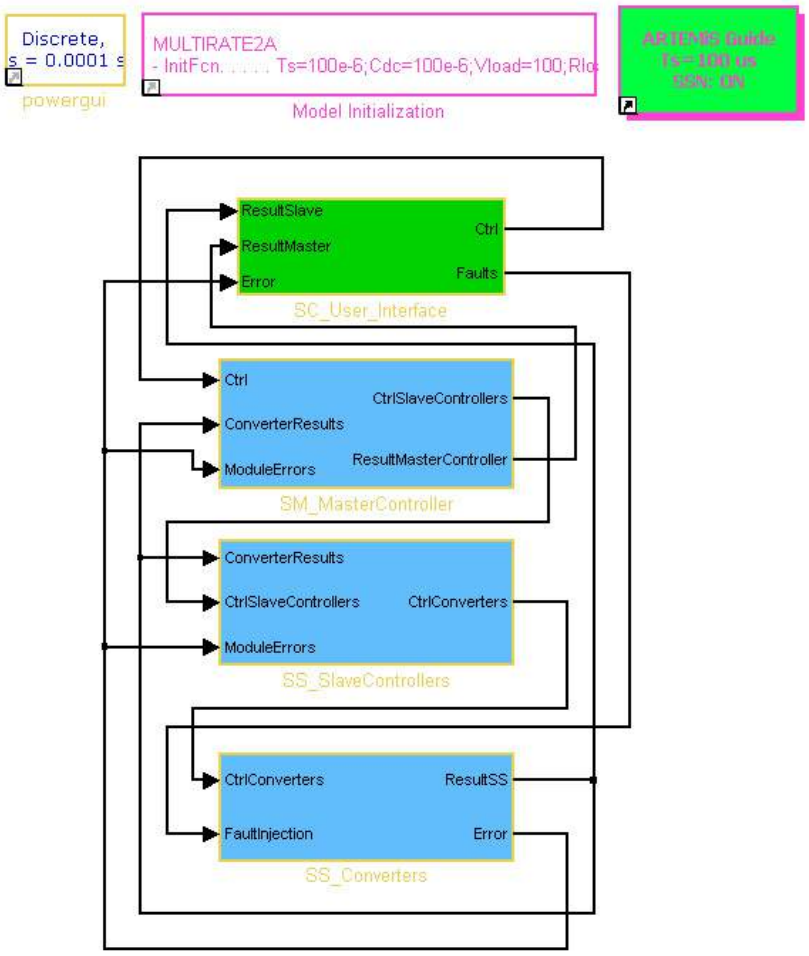


Figure 3.20: Overview of the Block Diagram of the Fault-tolerant Controller Simulation

The master controller is responsible to regulate the DC grid voltage, current control, phase voltage balancing, grid synchronization(phased locked loop) and power flow control. It looks at the multi-level converter as a simple converter and generates the reference points for slave controllers(figure 3.21). Slave controllers are synchronized to the reference points of the master controller and they implement the fault-tolerancy over system(figures 3.22, 3.23, 3.24, 3.25).

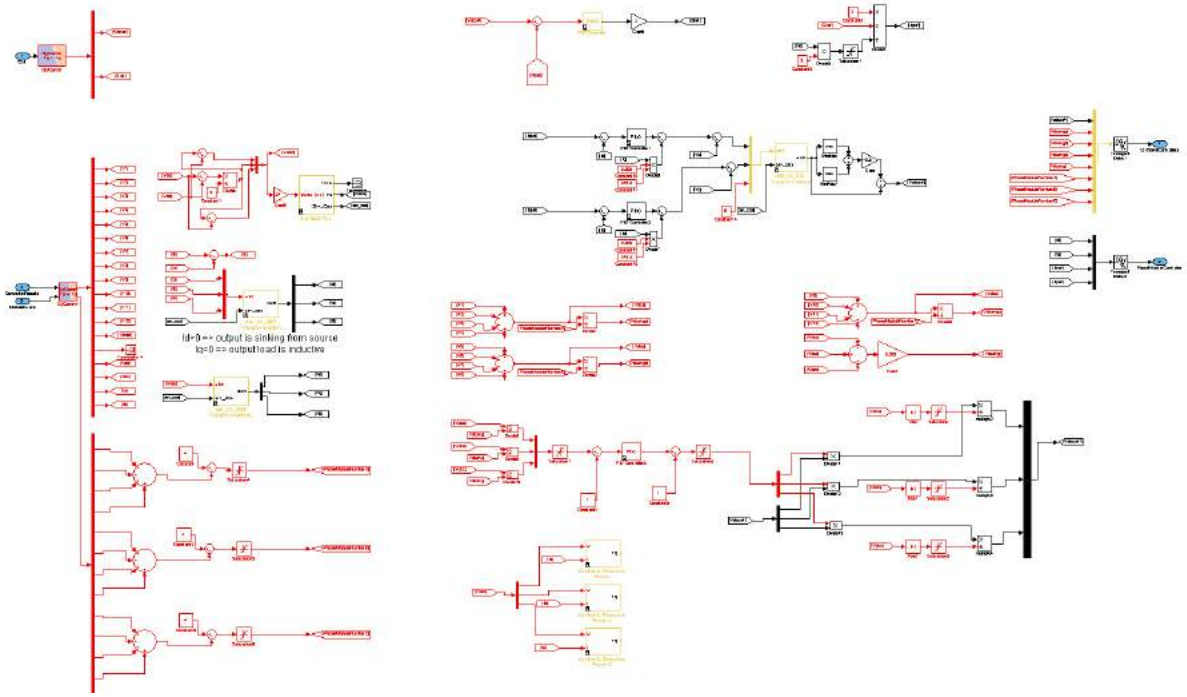


Figure 3.21: Simulation Diagram of Master Controller

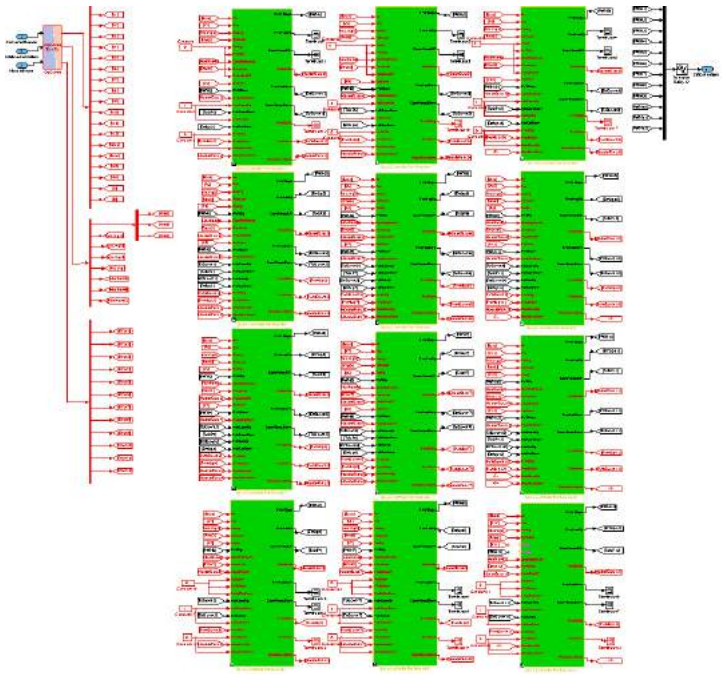


Figure 3.22: Simulation Diagram of Slave Controllers

Figure 3.23: Connection between Slave Controllers

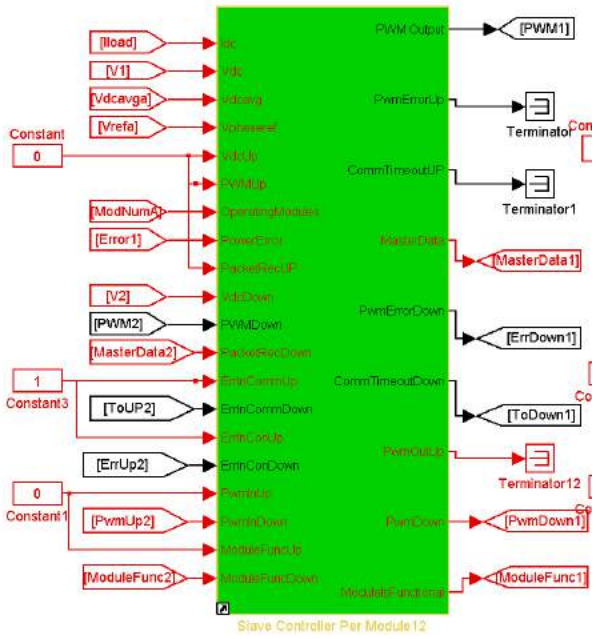


Figure 3.24: Slave Controllers at the Edge of the Converter Leg

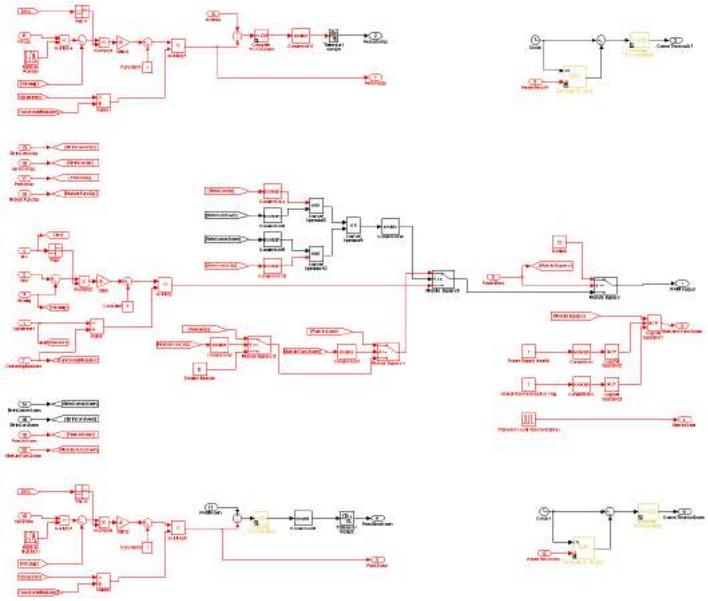


Figure 3.25: Internal Block Diagram of Slave Controllers

The switching frequency of the converter is 10 kHz, therefore it requires simulation time of less than 1μS. This will increase the simulation time by a great factor and development time would be higher than normal. In order to overcome this problem, power electronic components have been chosen for Opal-RT™ RT-Event® library. This library looks at transitions of the PWM signals and can fasten the simulation time by a great factor. By using this library, the simulation step time may increase to 100μS which is the same as period of switching frequency(figure 3.26, 3.27). The DC/DC converter has also been simplified by an average model(voltage-current source). Table 3.2 summarizes the parameters of components used in the simulation.

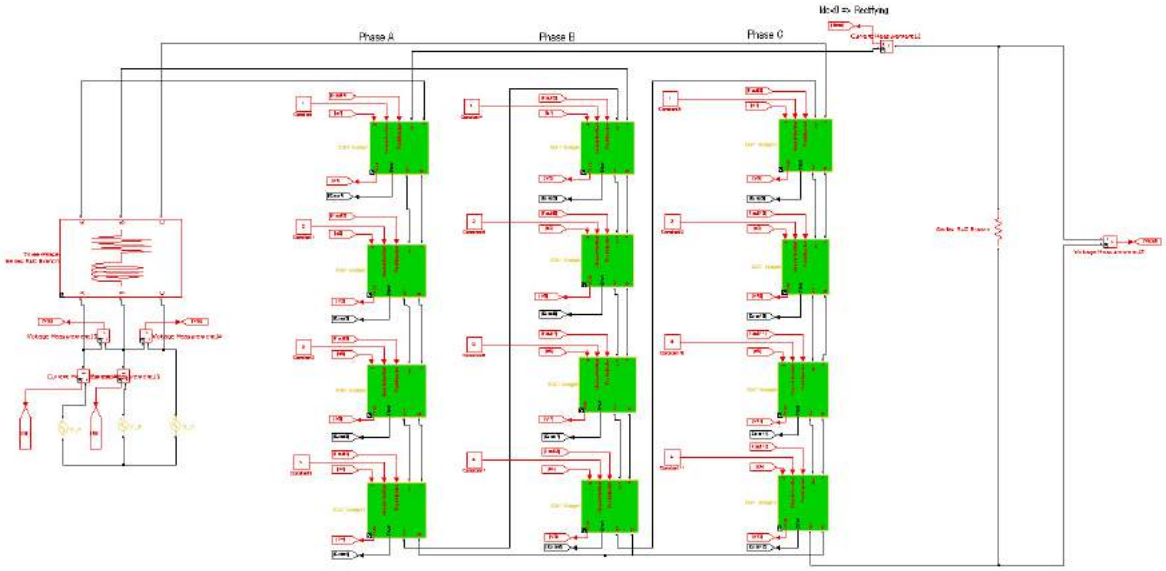


Figure 3.26: Block Diagram of Multi-level Converter

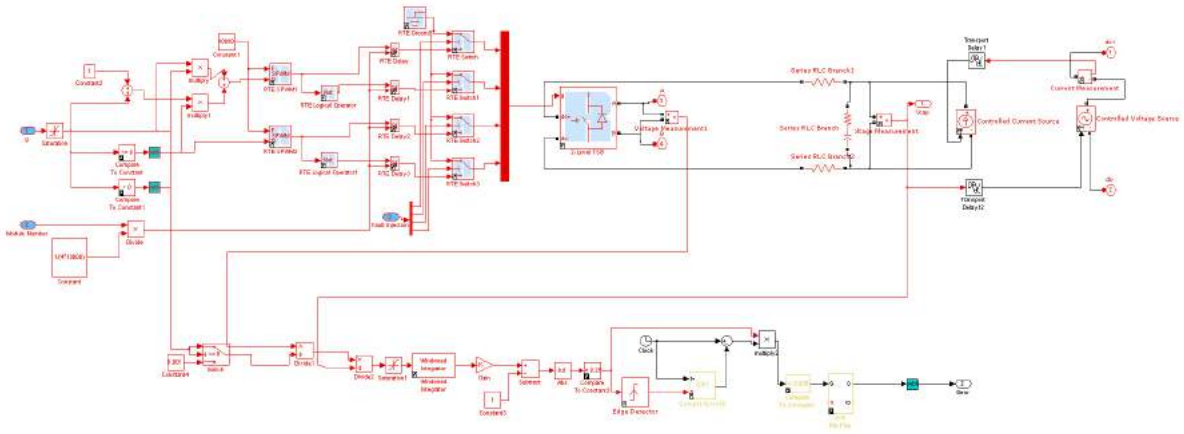


Figure 3.27: Internal Block Diagram of Converter Modules

Table 3.2: Power Parameters for Simulated Fault-tolerant Controller of Cascaded H-bridge

Total module number (per terminal)	4
Module DC capacitor voltage	35 V
Terminal rated power	1 kVA
Switching Frequency	10 kHz
AC input voltage (phase peak)	100 V
AC/DC Capacitor value	6.8 (mF)
DC/DC Capacitor value	N/A(Average model for DC/DC)
Input Filter Inductance	5 (mH)

3.6.1 Converter Output Result Under Normal Operation

In the normal operating mode, the converter act as voltage regulator for DC grid(rectifier mode). In this case, the input current must be sinusoidal and output voltage must reach its reference value at the minimum possible time. The voltage on each phase and their modules must be equal and balanced. Since each leg of converter has the same capacitance, the power flow in each phase must be equal to the other phase. Figure 3.28, 3.29, 3.30 and 3.31 show the simulation result of cascaded H-bridge multi-level converter with distributed controller.

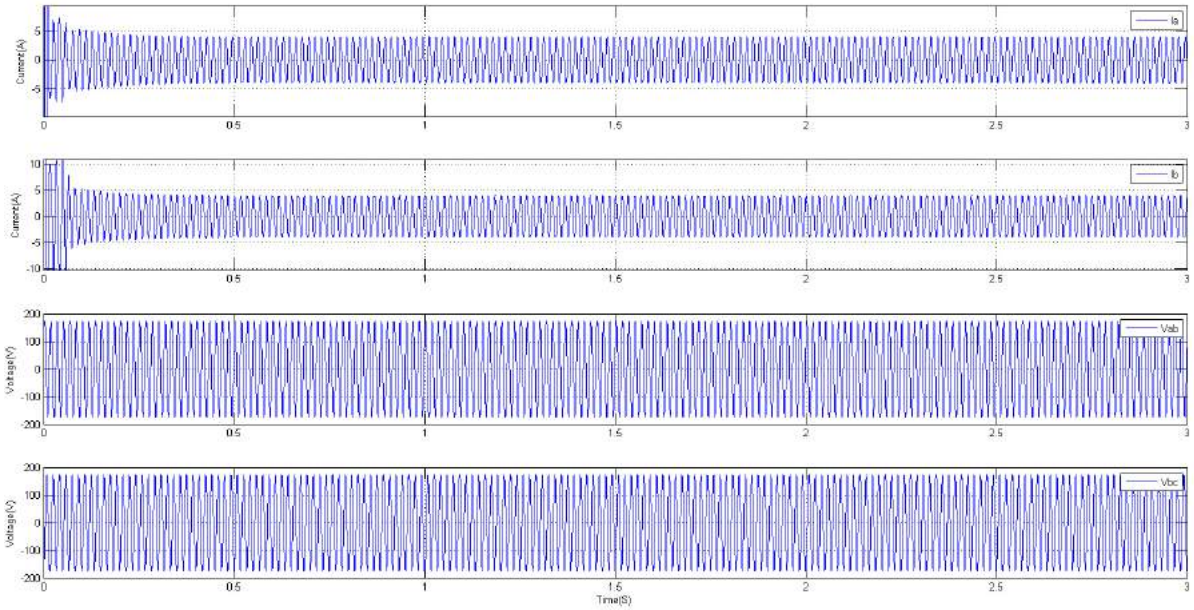


Figure 3.28: Grid Voltage and Current at Normal Operation of Converter

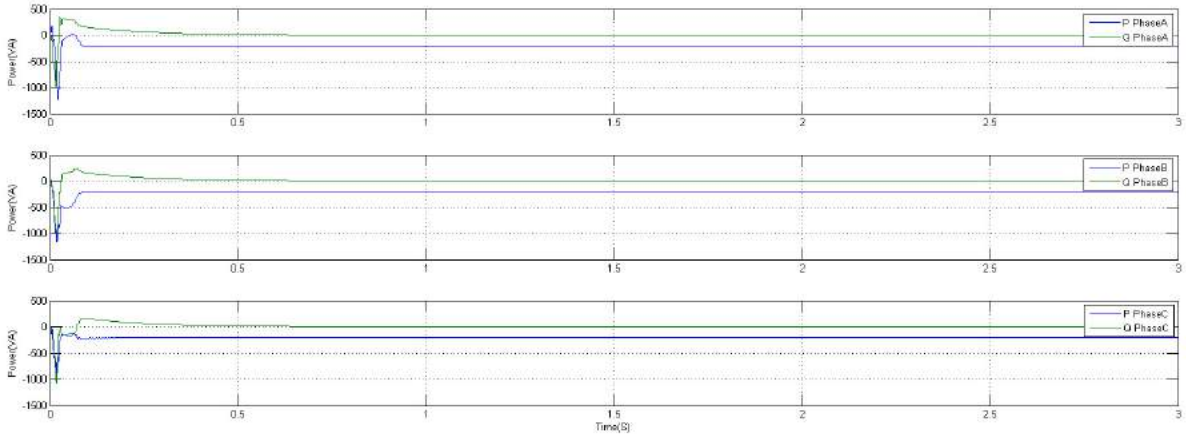


Figure 3.29: Grid Phase Power at Normal Operation of Converter

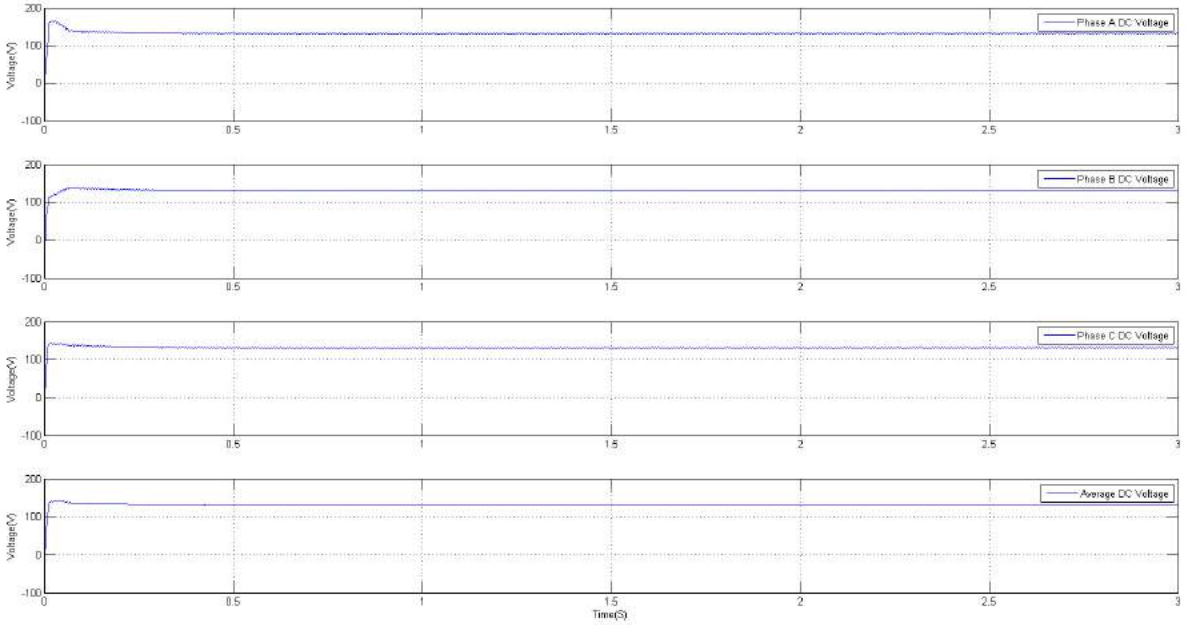


Figure 3.30: DC Voltage of each Phase at Normal Operation of Converter

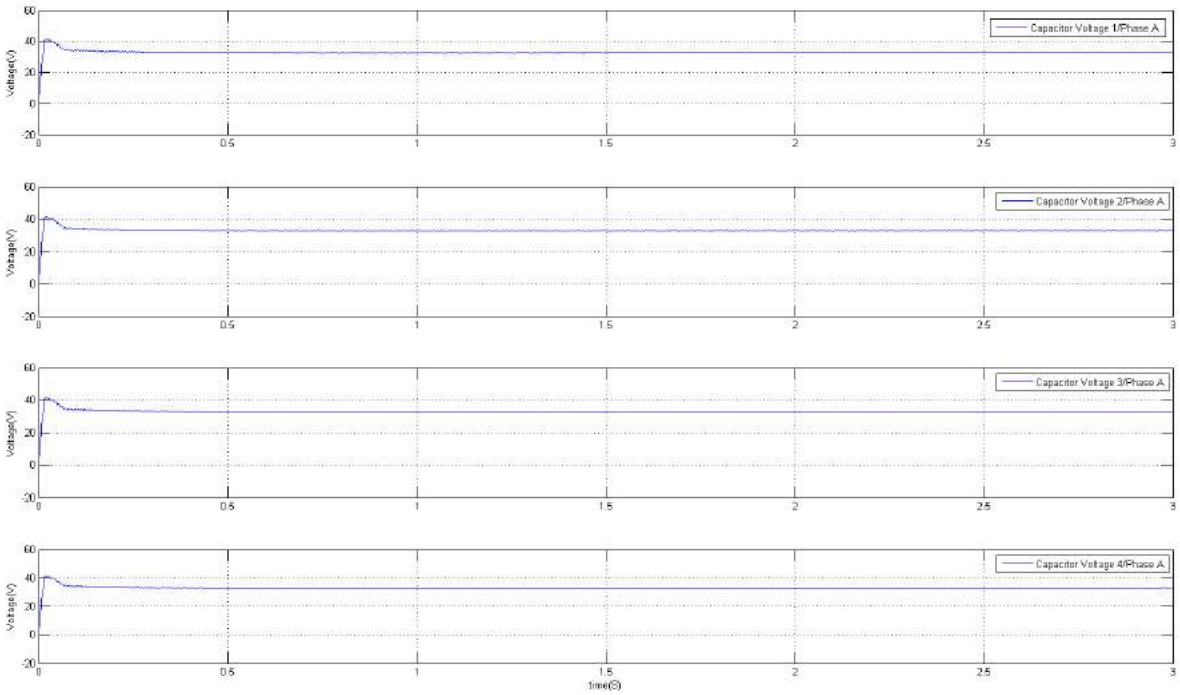


Figure 3.31: Module Capacitor Voltages for Phase A at Normal Operation of Converter

3.6.2 Converter Output Result at Power Electronic Failure(Mode 1)

Whenever a fault happens in the power devices of the converter, it must be identified and the corresponding module must be bypassed. There is a fault detection circuit inside power converters and it continuously checks the health status of all switches. The fault signal will be transmitted to slave module to bypass the whole module. In this case, the power coming to converter leg that holds the failed module would be different and would be higher because the equivalent capacitance of the converter leg is higher and maintaining the same voltage on that leg requires more power.

For simulating the failure in power module, an IGBT inside the H-bridge has been shorted to the trigger the fault detection circuit at t=1. The following result is acquired at failure moment:

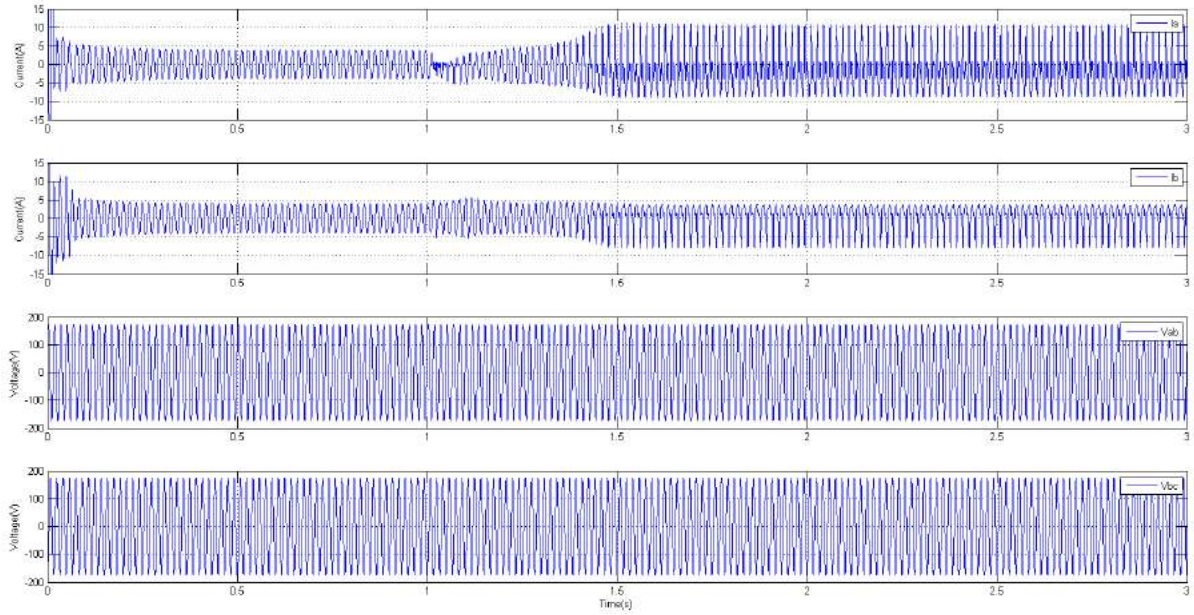


Figure 3.32: Grid Voltage and Current when Failure Happens in Power Module at t=1s

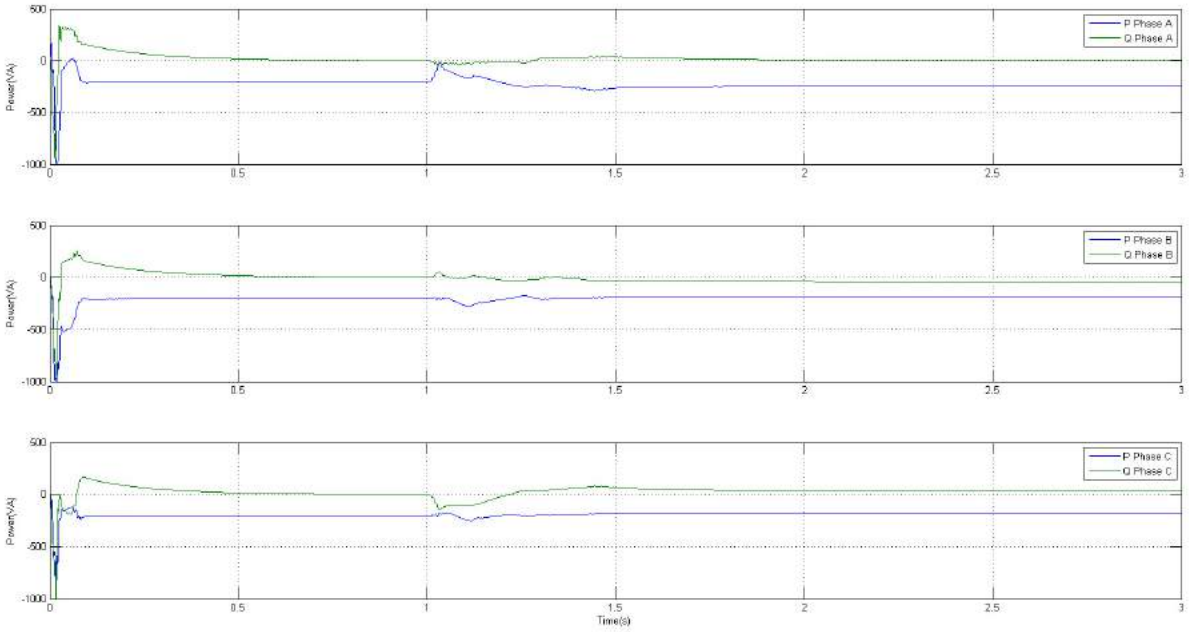


Figure 3.33: Grid Phase Power when Failure Happens in Power Module at t=1s

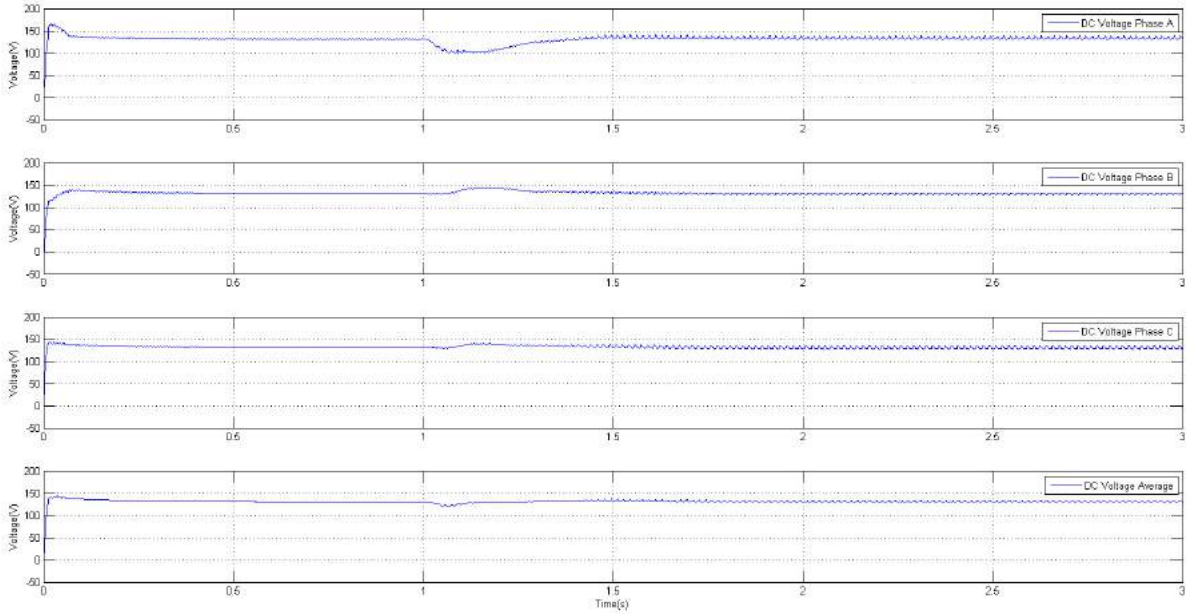


Figure 3.34: DC Voltage of each Phase when Failure Happens in Power Module at t=1s

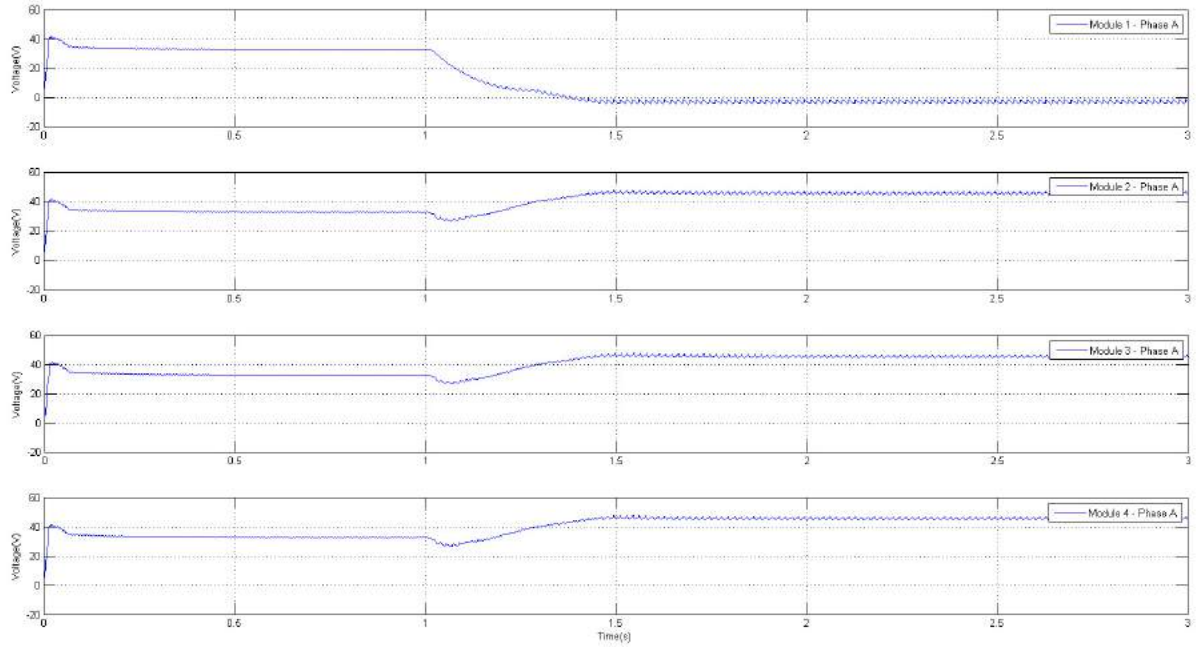


Figure 3.35: Module Capacitor Voltages for Phase A when Failure Happens in Power Module at t=1s

3.6.3 Converter Output Result at Controller Input Failure(Mode 2)

In this test, functionality of the PWM output comparator is investigated. If the input voltage of one module changes because of the error in measurement unit or the sensor, the result of PWM comparator between two adjacent modules would not be the same. In this case, an error flag will be created that indicates the error in main controller and a fail-over will be made to switch the input of power converter to the adjacent PWM generator. In this fault, the capacitor voltages should not feel any disturbance and the result would be the same as the moment before failure. Figure 3.36 and 3.37 demonstrate the result of this test.

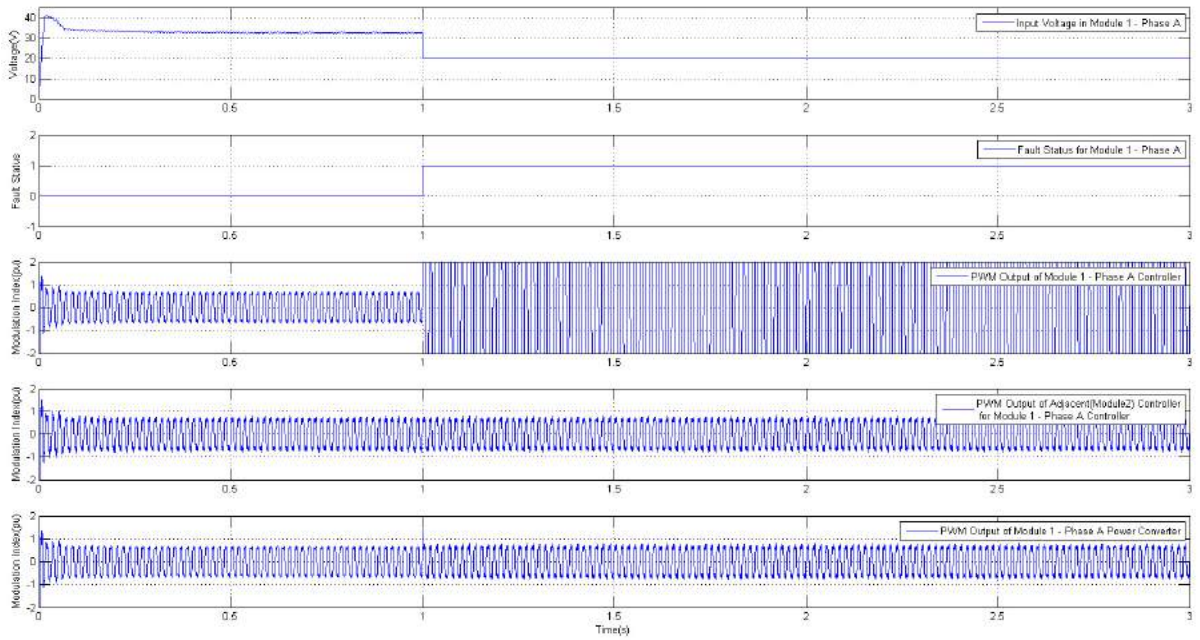


Figure 3.36: PWM Output of Main Controller and Adjacent Controller when Input Sensor Failure Happens in Module 1(Phase A) at t=1s

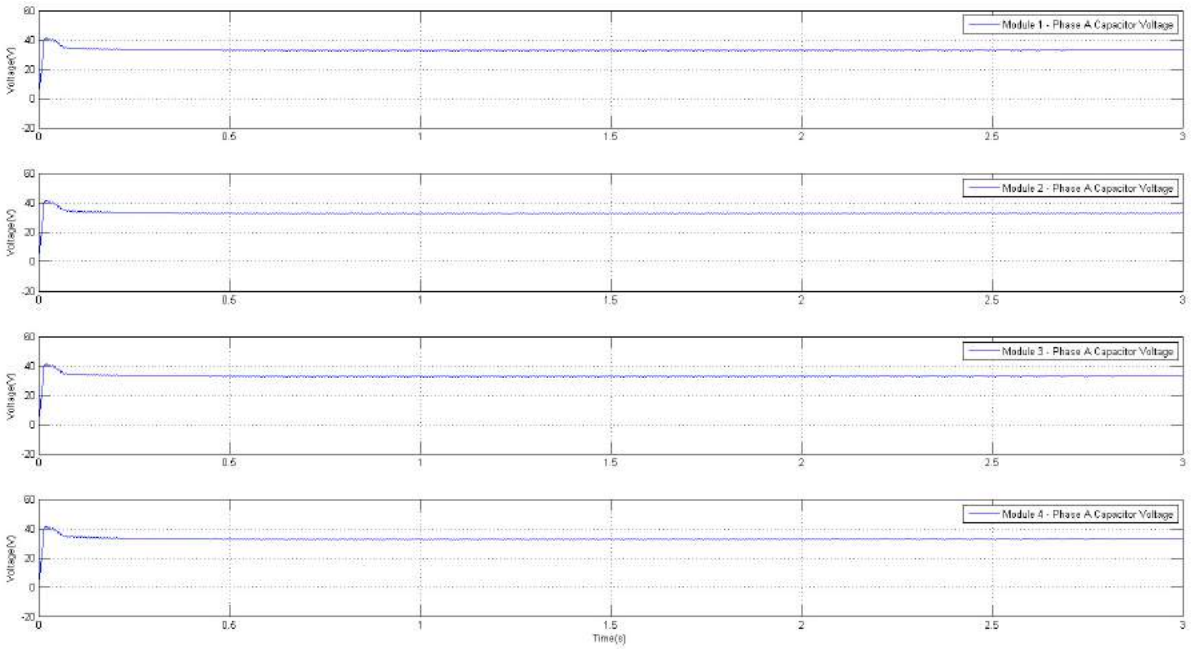


Figure 3.37: Module Capacitor Voltages for Phase A when Input Sensor Failure Happens in Module 1(Phase A) at t=1s

3.6.4 Converter Output Result at Controller Synchronization Failure(Mode 3)

In this test, the failure is located on the data synchronization of the main controller(Module 1). The cause of this failure can be related to the functionality of the microprocessor or any problem with the data link between main controller and adjacent controllers. After failure detection, a fail-over signal will connect the input of power converter to the adjacent controller. Figure 3.38 and 3.39 show the result of this test.

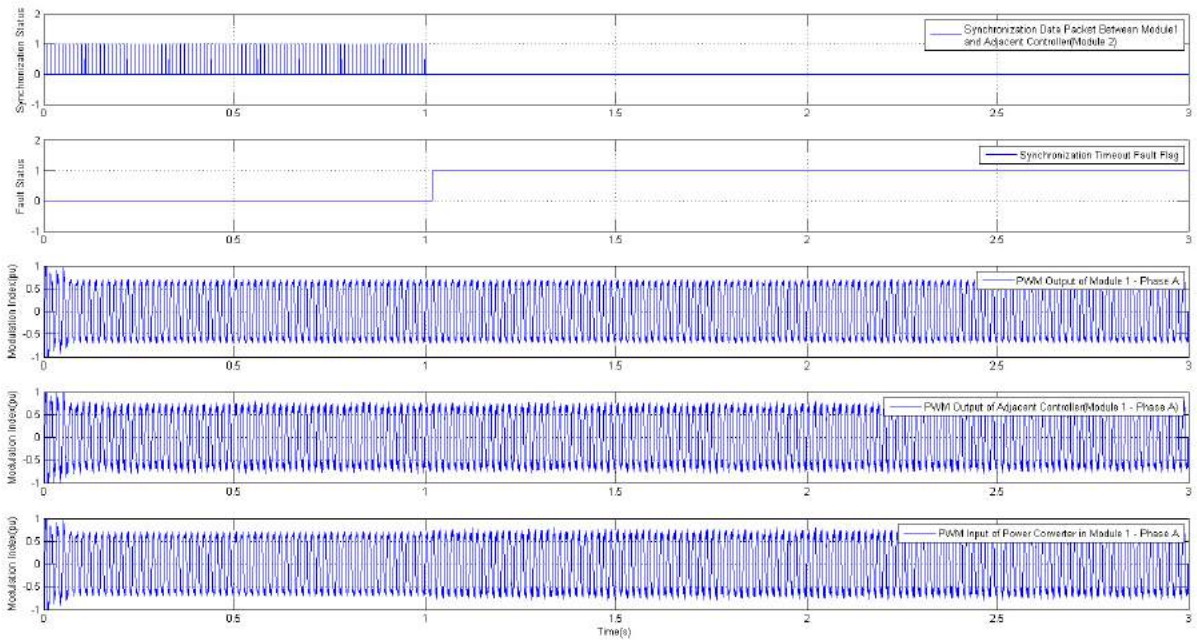


Figure 3.38: PWM Output of Main Controller and Adjacent Controller when Synchronization Failure Happens in Module 1(Phase A) at t=1s

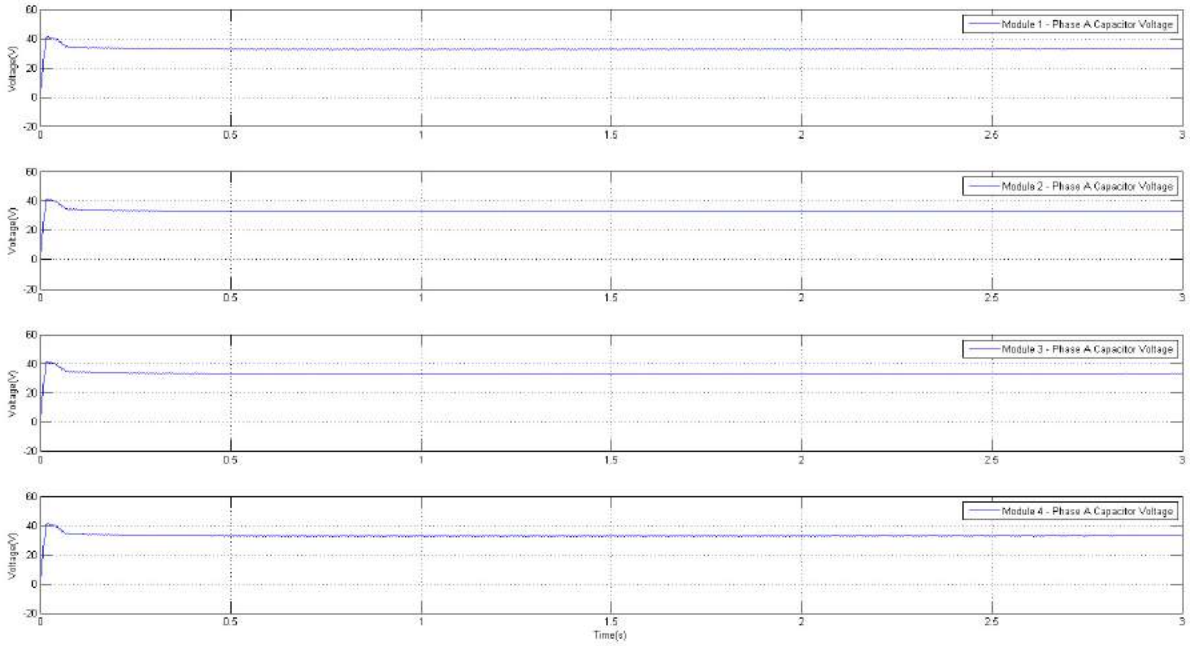


Figure 3.39: Module Capacitor Voltages for Phase A when Input Synchronization Failure Happens in Module 1(Phase A) at t=1s

Chapter 4

Reliability Assessment of Proposed Fault-tolerant Controller Architecture

4.1 Introduction

Reliability and availability analysis is one of the main stages in product development. The final system must go through different tests for grading its performance over the life time. Failure rate of an individual element may be identified using experimental or analytical methods. In experimental method, large sample of a particular component will be tested for a period of time and by using statistical formulas, it is possible to extend the failure rate of that particular component. Since most of the electronic components have complicated functionality, it is not possible to directly use analytical method (which uses physical formulas to guess fatigue) and find the failure rate. The final failure rate depends on the architecture of the system. The proposed controller has its own unique architecture and reliability of the system will be analyzed based on its architecture and components. This chapter contains the result of the analysis done to figuring out the failure rate of the final controller.

4.2 Failure in Controller Systems

Failure in control systems may have internal, external or human factors. Internal causes are happening due to the physics of the devices that has been used to build the controller. Over time, material may loss its properties, change shape and become inapplicable for the intended task it's been designed for. Operating environment and interference can cause instantaneous or permanent failure in the system. It is not possible to control the external factors, but it is possible to design the system in order to endure harsh conditions. Humans play a great role in embedding failure agents when designing the firmware of the control system. These errors may be transparent for a long time and appear on a special occasion. In the following section, some of the important failure causes would be reviewed and techniques in design would be suggested.

4.2.1 Lifetime of Silicon Devices

Although silicon devices are solid and have very stable physics, there are different parameters that can limit the lifespan of these devices [106]. The failure in embedded processors can happen in three stages of life (Figure **??**):

- **Early life**: Failure rate is declining and it is due to early defects (most of them can be eliminated with post-silicon verification)
- **Useful life**: The steady-state life in which the failure rate is the lowest. This is the main failure rate that would be used for analysis
- **Wear-out stage**: In this stage, device failure rate would be high and microprocessor is no longer reliable

Most of the advanced embedded processors are fabricated based on CMOS technology. The main failure cause in CMOS IC is electromigration (EM) in the interconnections. EM is a term

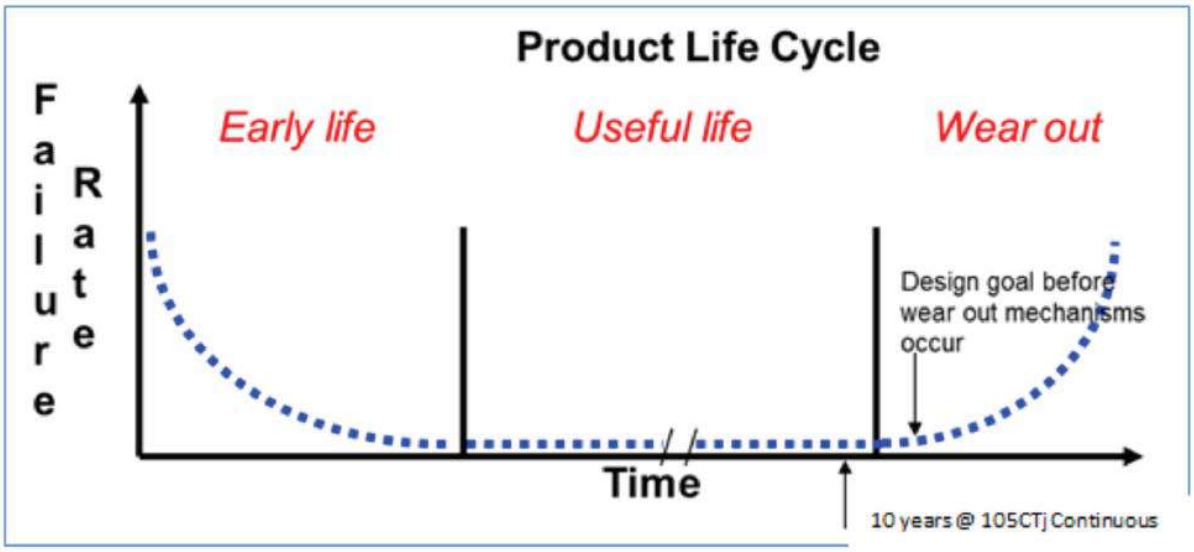


Figure 4.1: Bathtub Curving Showing Different Stage of Reliability

applied to the transport of mass in metals when high current is conducted [14]. The failure rate of a wire follows the black's formula:

$$\lambda_{wire} = \frac{1}{MTF} = AJ^2 \exp(-\frac{\phi}{kT}) \tag{4.1}$$

In equation 4.2, A is a constant based on the cross-sectional area of the interconnect, J is the current density, ϕ is the activation energy (e.g. 0.7 eV for grain boundary diffusion in aluminum), k is the Boltzmann's constant, T is the temperature in Kelvin.

Mean time to failure (MTF) is greatly depended on the material type used for interconnects. Aluminum used to be the main material for interconnects, due to its adherence to the silicon base. It also has low MTF and is has not good property when used purely. Adding 2-4% copper to aluminum can increase resistance to EM by 50 times.

As it can be seen in black's equation, increasing temperature can lower the MTF and increasing the failure rate. In figure 4.7, failure rate of a sample processor has been shown. In higher

temperature, mean of failure rate would be increased. Therefore, it is necessary to have heat transfer mechanism in controller design to dissipate the excess heat.

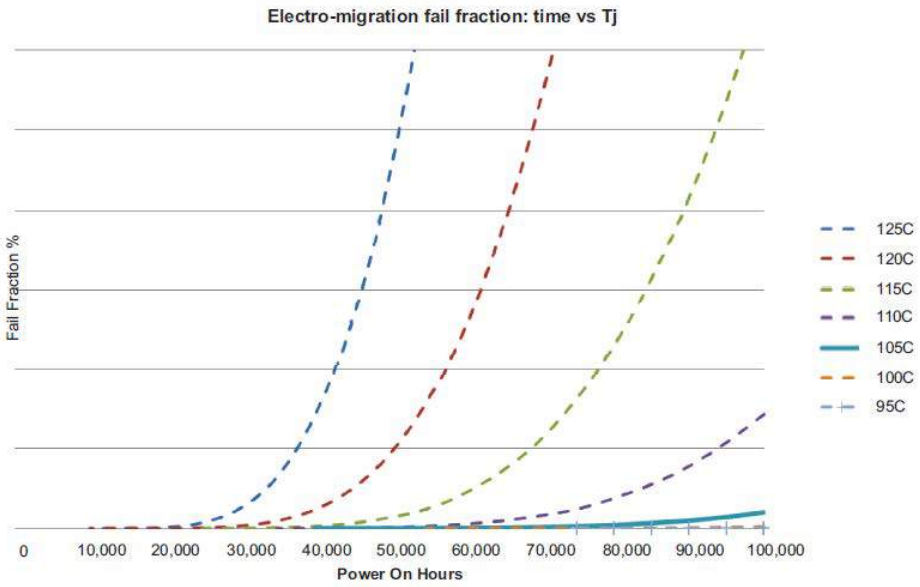


Figure 4.2: Impact of Temperature Increase on Embedded Processors (Texas Instrument)

There are more phenomenons that can change transistor parameters and limit the lifetime of an embedded processor. The other failure mechanism in silicon devices can be listed as following:

- **Time Dependent Dielectric Breakdown**: Refers to the physical process whereby a dielectric stored under a constant electric field, less than the materials breakdown strength, will break down with time. It can be accelerated by operating voltage and temperature[70].
- **Hot Carrier Injection**: When a hole or electron get enough kinetic energy to break the potential barrier. This effect is the basis for NOR flash memory, but can be destructive in other cases. It can change the properties of the transistors and decrease the reliability of

the embedded processor.

- **Negative Bias Temperature Instability**: This phenomenon change the absolute threshold voltage and degrade mobility, drain current and transconductance of the PMOS transistors[84].

4.2.2 High-energy Particles (Ionizing Radiation) Effect on Microprocessors

Exposure of cosmic rays or high energy particles (neutron and alpha) to electronic devices can cause local ionization (Figure 4.8). This would cause a beam of electric current or move of the charge in the material that leads to different consequences.

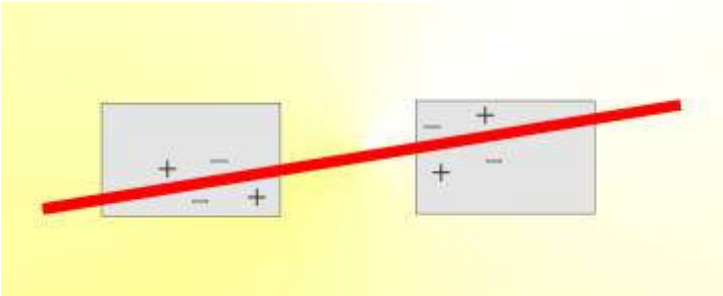


Figure 4.3: Local Ionization by Charged Particles (Space Radiation Associates)

It is impossible to avoid ionization effects. Cosmic rays are everywhere in the space (with greater potential) and on the ground with less concentration. They have enough energy to cause charge movement when passing through the integrated circuit. Decaying nuclear materials (e.g. uranium and thorium) can produce alpha particles and these material may exist in integrated circuit packaging materials[69]. The alpha particle collision with semiconductor body can

induce soft errors in the embedded processor. Figure 4.9 demonstrate effect of such particles on a MOSFET transistor.

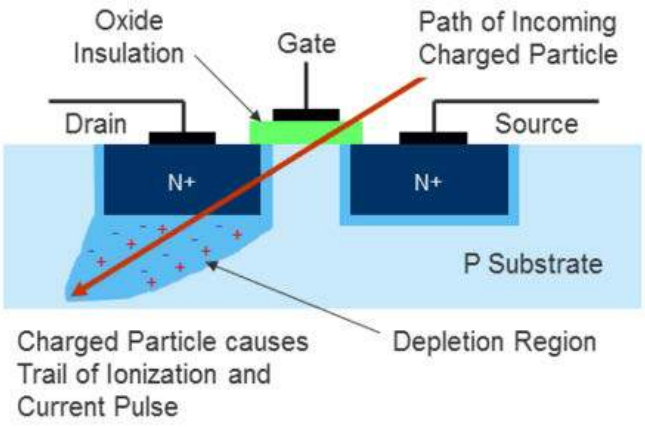


Figure 4.4: Effect of High-energy Particle Collision with MOSFET Transistor

Transistors are building blocks for semiconductor devices like charged-coupled device (CCD), static RAM (SRAM), dynamic RAM (DRAM) and embedded processors. These devices integrate large amount of transistor on a single die, therefore increasing the chance of particle collision on the surface of the die. Single event phenomena can be categorized into three sections:

- **Single event upset**: It is defined as any radiation-induced error that has reversible effect. Single event upset (SEU) are soft errors, non-destructive and can be brought back to normal condition by reset or rewriting to the affected memory cell[74][86]. Since random access memory (RAM) devices are more vulnerable to this effect, in design of fault-tolerant controllers they should be avoided as much as possible. A good example for the SEU effect would be field programmable gate arrays (FPGA) that use memory cells to store the configuration of the logical functions (Figure 4.5).

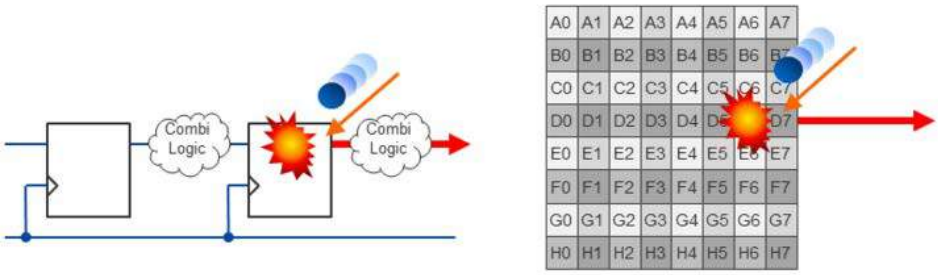


Figure 4.5: SEU Effect on SRAM-based Field Programmable Gate Array (FPGA)

SEU can change the memory bit that defines logical function of the FPGA cell or routing matrix of interconnections. Although this error can be detected by error detection bits (e.g. CRC) and be fixed by re-configuring the device from backup image in the non-volatile memory, it has temporary effect on the hardware behavior of device. To avoid temporary failure in configurable digital circuits, complex programmable logic device (CPLD) or flash-based FPGA that use flash memory cells to store the configuration must be implemented in the control systems.

- **Single event latch-up**: It is a condition that causes loss of device functionality due to the induced current. Latch-up can happen in CMOS ICs if negative or positive spike on input or output pin exceed the rail voltage more than diode drop. In single event latch-up (SEL), latch-up will be triggered by ionizing radiation and the current induced by it. SEL is a hard error that can destroy the device, drag down the bus voltage or damage the power supply.

- **Single event burnout**: It can cause destruction due to the high current in a power transistor. Single event burnout (SEB) is a hard error that include burnout of power MOSFET, gate destruction and pixel failure in CCDs. In the turn-off power MOSFETs, SEB can turn on the device (which is blocking high voltage) and conduct high current in the device which

cause permanent failure.

4.3 Reliability Prediction for Controller Components

In order to predict the reliability for the controller system, individual components must be analyzed. After predicting the individual reliability, it is possible to predict the final reliability based on the series or parallel arrangement of the components. In industry, there are three methods for predicting the reliability of a component:**empirical method, physics of failure** and **life time test**

Empirical method use the data of failure result from previous tests that have been done in the field. In this method, statistical curve fitting is used to predict the failure of the component and by using engineering method, it is possible to match the result for similar components too. There are different standards that defines the failure rate which is mentioned in Table 4.1.

Table 4.1: Prediction Methods for Electronic Components Failure

Prediction Method	Applied Industry	Last Update
MIL-HDBK-217F	Military	1995
Bellcore/Telcordia	Telecom	2006
RDF2000	Telecom	2000
SAE method	Automotive	1987
NTT procedure	Telecom	1985
SN29500	Siemens Products	1999
China 299B	Chinese Military	1998
PRISM	Military/Commercial	2000

Among all these empirical methods, MIL-HDBK-217F [1],[22] is the most internally known standard and it is being used worldwide to predict the component failures. In the handbook,

part count and **part stress** are two of the methods for predicting the component failure. In part count method, it assumes that components are working in normal conditions. Therefore, the total failure rate is summation of all the failure rates of components used in the controller:

$$\lambda_{b,i} = \sum_{i=1}^{n} (\lambda_{ref})_i \tag{4.2}$$

In part stress method, different working parameter is being included that would give better prediction for the failure. In equation 4.3, π_S is the stress factor, π_T is temperature factor, π_E is environment factor, π_Q is quality factor and π_A is adjustment factor. In the handbook, there is a range for each factor and based on the working condition they can be set.

$$\lambda = \sum_{i=1}^{n} (\lambda_{ref,i} \times \pi_S \times \pi_T \times \pi_E \times \pi_Q \times \pi_A) \tag{4.3}$$

The benefit of empirical method is that a wide library of components is available and there are different software available to perform calculations regarding the prediction. On the other hand, these standards are not updating very often and new components or new technologies are not considered in failure rate calculation. Therefore it is good to use other prediction methods to increase the precision.

Another method of failure prediction is using **physics of failure** to predict the rates. *Arrhenius equation* is one of the famous formulas to calculate acceleration for failure in electronic components. In equation 4.4, L(T) is the lifetime characteristic to temperature, A is the scaling factor, Ea is the activation energy, k is Boltzmann constant and T is the temperature.

$$L(T) = A \exp(\frac{E_a}{kT}) \tag{4.4}$$

Arrhenius equation has different forms and it can be used to predict the acceleration factor (AF)

for failure if we have gathered previous data when temperature is changing(equation 4.5).

$$AF = \exp\left(\frac{E_a}{k}\left(\frac{1}{T_{use}} - \frac{1}{T_{stress}}\right)\right) \tag{4.5}$$

For new electronic devices or complicated ICs that failure prediction can't be done by empirical methods, **life time** of device will assessed by testing a large sample of a particular component over time under normal working condition. In this case, statistical methods (Weibull distribution) can be used to fit the curve of failure rate in device. The result can be extended to different temperatures and stresses using Arrhenius equation. Equation 4.6 is one of the statistical prediction methods which can be used to calculate failure rate (λ).

$$\lambda = \sum_{i=1}^{\beta}\left(\frac{x_i}{\left(\sum_{j=1}^{k} TDH_j \times AF_{ij}\right)}\right) \times \frac{M \times 10^9}{\sum_{i=1}^{\beta} x_i} \tag{4.6}$$

λ = failure rate in FITs (Number fails in 10^9 device hours)

β = Number of distinct possible failure mechanisms

k = Number of life tests being combined

x_i= Number of failures for a given failure mechanism i = 1, 2,...β

TDH_j= Total device hours of test time for life test j, j = 1, 2,... k

AF_{ij}= Acceleration factor for appropriate failure mechanism i , j = 1,2,3,...k

where X^2= chi square factor for 2r+2 degrees of freedom

r = total number of failures ($\sum x_i$)

α = risk associated with CL between 0 and 1

4.3.1 Failure Rate Calculation for Controller Card

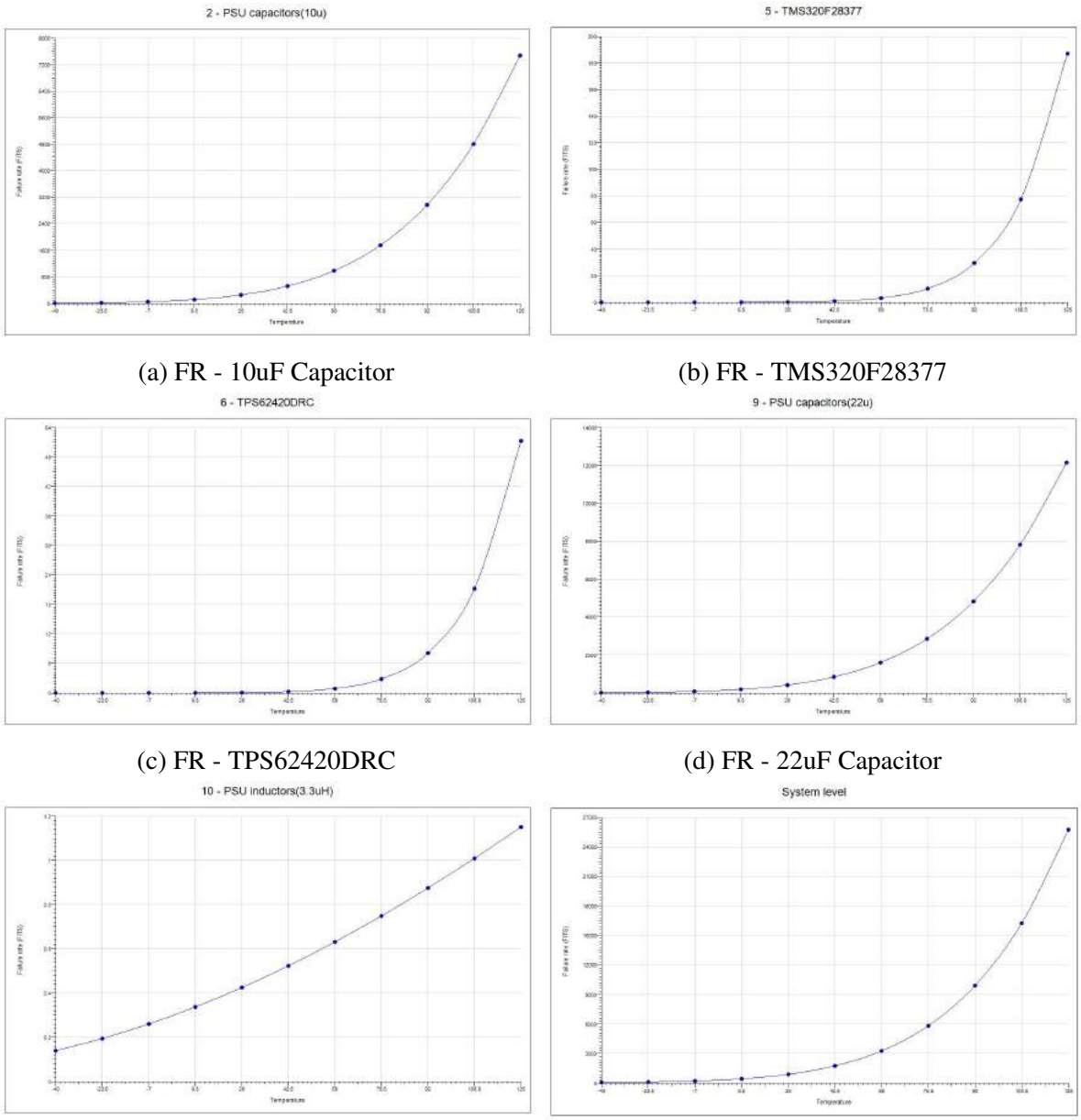
There are different software available that can calculate the failure rates of electronic components using mentioned techniques. Isograph™ Reliability Workbench (RWB) is capable to predict failure rate based on MIL-HDBK-217F handbook. It is possible to define working environment and get the result over wide temperature range. The first stage in prediction is to list the critical component in the control board in which failure of one them can stop the functionality of the whole system. The micro-processor and the power management circuit contain the main critical components (Table 4.2).

Table 4.2: Mean Failure Rate of Critical Components in TI-F28377 Control Card

Component	Failure Rate (FIT)	Description	Failure Prediction Method
10uF Ceramic Capacitor	304.79	PSU Capacitor	Empirical
2.2uF Ceramic Capacitor	186.27	PSU Capacitor	Empirical
12 XTAL Oscilator	77.67	Oscilator	Empirical
TMS320F28377	2.4	Micro Controller	Life test
TPS62420DRC	0.65	Power Management IC	Life test
REF3030	3.16	Voltage Reference	Life test
LMP7709	12.44	Operational Amplifier	Life test
22uF Ceramic Capacitor	495.76	PSU Filter	Empirical
3.3 uH Inductor	0.44	PSU Filter	Empirical

The next stage is to enter all components in the software, therefore it can use physics of failure to extend the operating temperature range and add up all failure rates together. For common components like capacitors, inductors, oscilators,... it is possible to use one of the empirical method (e.g. MIL-HDBK-217F) to predict the failure rate and for custom components like microprocessors, power management IC (PMIC),... the failure test data from the vendor is

usable (usually good products come with failure test results). Figure **??** demonstrate the result of prediction for TI-F28377 control card.



(a) FR - 10uF Capacitor

(b) FR - TMS320F28377

(c) FR - TPS62420DRC

(d) FR - 22uF Capacitor

(e) FR - 3.3uH Inductor

(f) FR - Control Card

Figure 4.6: Failure Rate (FR) Prediction for TI-F28377 Control Card over Temperature Range

It is possible to find some important design techniques from figure **??**. Capacitors are responsible for a great amount of failure rate in the controller card while ICs have very small impact. That is because of the nature of capacitor and usage of material to store electric energy. Even in ceramic capacitors, the chance of cracking make them vulnerable in long time applications. Therefore it is a good idea to use fewer capacitors in the design or use high grade capacitors in power supply unit. Meanwhile silicon chips have evolved greatly with failure and thanks to the research and advancement in fabrication, they have small portion of static failure (Although they are responsible for transient faults). Figure 4.7 demonstrates portion of each component in failure rate of the controller card.

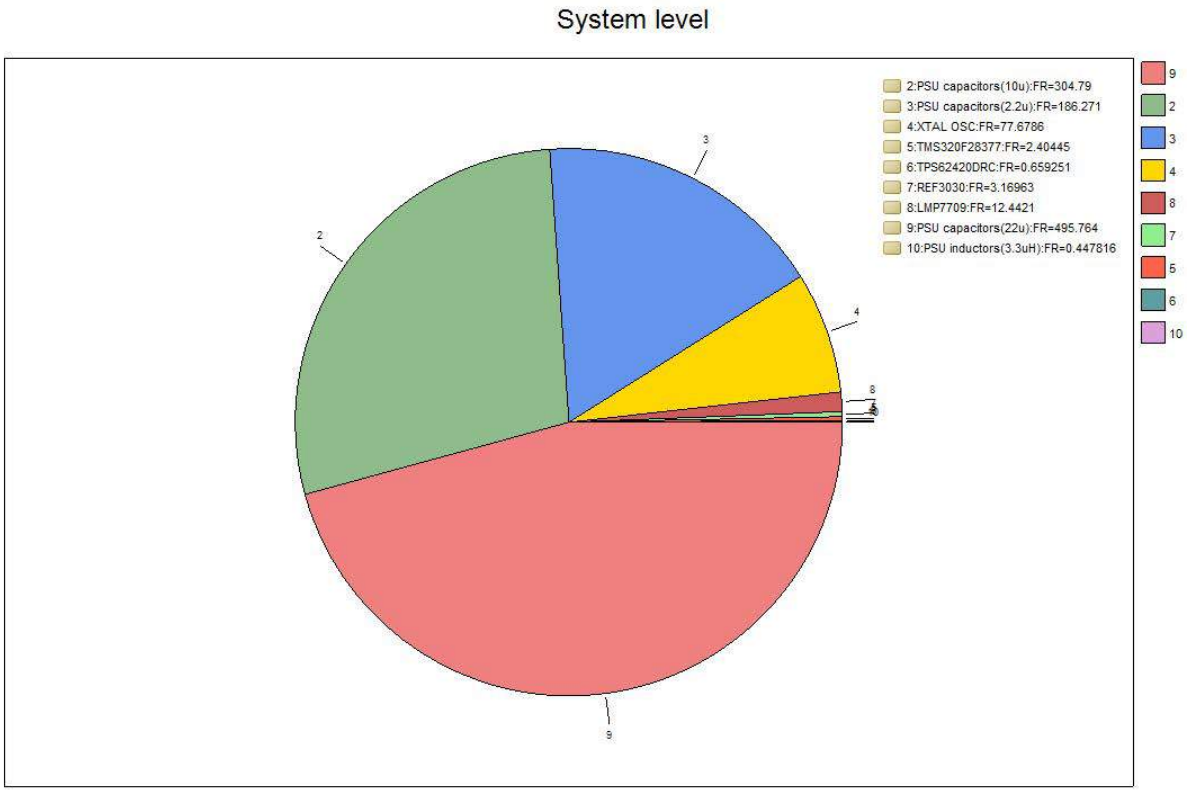


Figure 4.7: Portion of Failure Rate for each Component in Controller Card

The other parameter in predicting failure rate is **stress**. Stress can be in the form of voltage, temperature, environment,... Each of these stresses has its own variable and final stress is the multiplication of all of them. Figure 4.8 is the result of simulation for failure rate with different stresses.

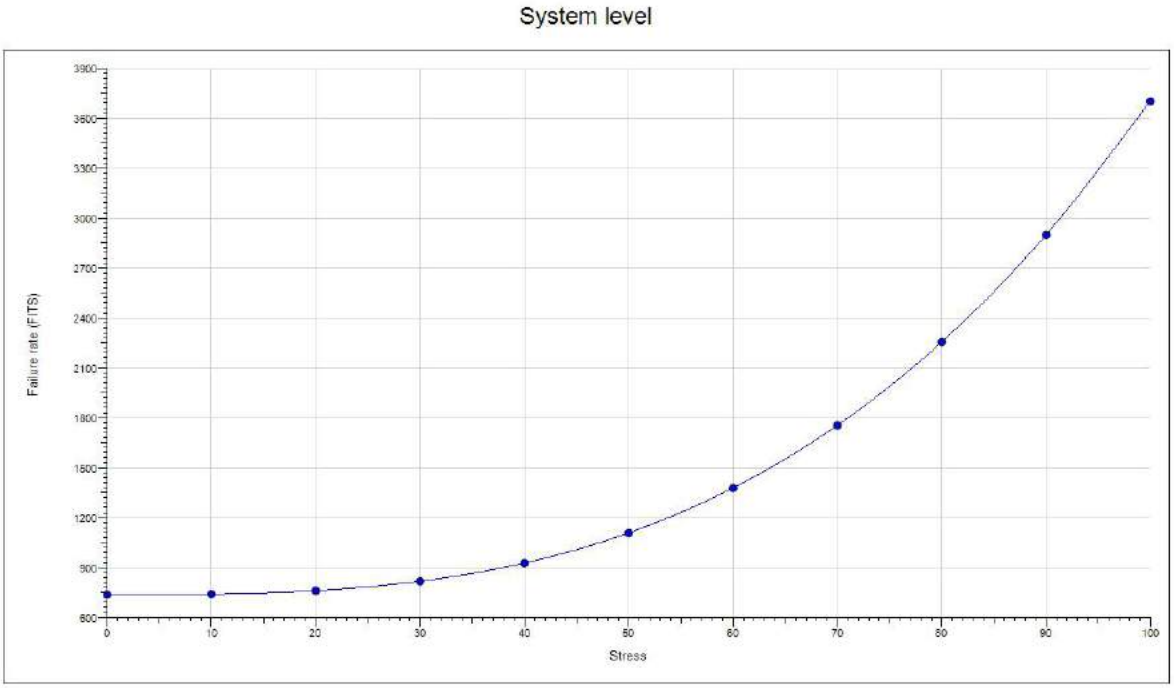


Figure 4.8: Failure Rate of Controller Card versus Stress

In reliability analysis, the average time a system functions before it fails is called mean time to failure (MTTF). This parameter will help the user to know when the system life is over and is the time to renew the whole plant. Figure 4.9 & 4.10 demonstrate the MTTF versus temperature and stress variation.

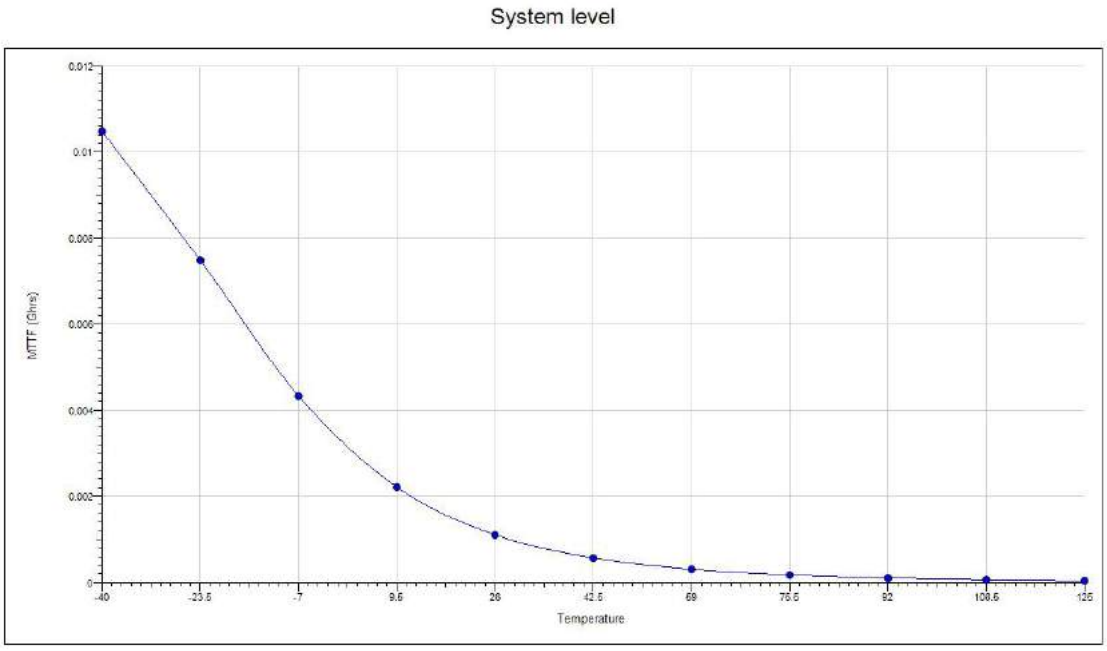


Figure 4.9: Mean Time to Failure of Controller Card versus Temperature

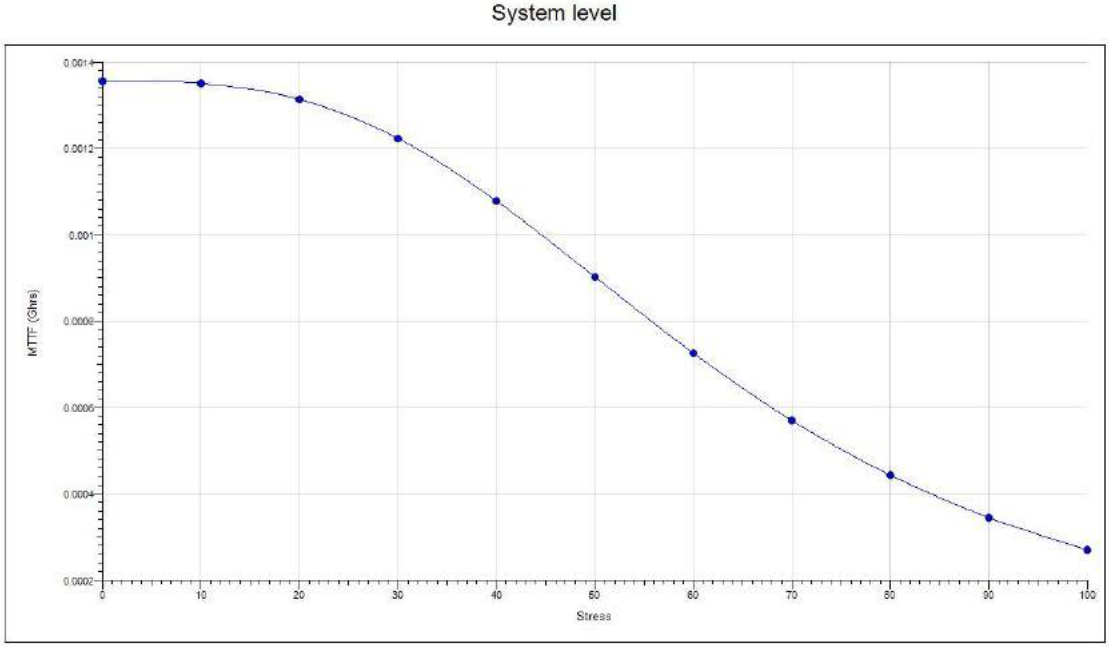


Figure 4.10: Mean Time to Failure of Controller Card versus Stress

4.4 Monte Carlo Simulation for Reliability Estimation of the Proposed Controller

One of the numerical methods for estimation of a mathematical model is Monte Carlo simulation[9, 46, 96]. This method is used for solving mathematical and physical problems using statistical tools. Each simulation consists of four steps:

- 1- Defining domain for input
- 2- Generate results from random variables in the defined domain
- 3- Sorting the results in to different categories
- 4- Counting the occurrence number of specific states among total events

The convergence rate of the Monte Carlo method is slow and it requires huge amount of samples for estimation of the desired value with low error. Thanks to the advancement in computer simulation, it is possible to perform the simulation much easier. Therefore, it is possible to use it as tool for reliability estimation.
The reliability of a component is governed by the following formula:

$$\frac{dR(t)}{dt} = -\lambda R(t) \tag{4.7}$$

with initial condition ($R_{t=0} = 1$), reliability becomes:

$$R(t) = exp(-\lambda t) \tag{4.8}$$

Therefore, the reliability at any given time is equal to:

$$R(t+\Delta t) = R(t)exp(-\lambda \Delta t) \tag{4.9}$$

Using the mentioned reliability formula and the performance matrix of the proposed controller, estimation process for reliability can be summarize as following steps:

- 1- Choosing exponential probability distribution function (pdf) for the reliability of components where $R(t) \in \Re$
- 2- Generating uniform random variable ($u \in U(0,\lambda)$) and compare them to $R(t)$ where $t \in (0, lifetime)$
- 3- Count the number of failure cases base on the performance matrix model

  $u > R(t_i) \rightarrow$ component i is failed

  $u < R(t_i) \rightarrow$ component i is functioning
- 4- Calculate the availability of the controller by calculating the $\frac{Total\ trials - Failure\ cases}{Total\ trials}$

The above procedure has been simplified in the algorithm 4. In order to perform simulation, 1000000 sample for each point has been gathered and simulation has been applied to (figure 4.11). In order to find the failure case, binary decision based on performance matrix (3.5) can be used[61].

```
Result: Reliability estimation of the proposed controller
NumberOfTrials = 1000000;
TimeStep = 100;
lambda = 1,count = 1,LifeTime = 1;
LifeTimeStep = LifeTime/TimeStep;
for (i=0;i<LifeTime;i+=Step) do
    Time = i;
    FailedControllers = 0;
    for (j=1;j<NumberOfTrials;j++) do
        ReliabilityThreshold = lambda * exp(-i);
        fail-count = 0;
        for (k=1;k<=4;k++) do
            if (random('unif',0,lambda) > ReliabilityThreshold) then
                ControllerAvailability[k] = 1;
            else
                ControllerAvailability[k] = 0;
        end
        if (CheckForFailure(ControllerAvailability[])) then
            FailedControllers ++;
        Availability(count) = (NumberOfTrials - FailedControllers)/NumberOfTrials;
        Availability-t(count) = i;
        count ++;
    end
end
```

Algorithm 4: Pseudo-code of Monte Carlo Simulation for Reliability Estimation

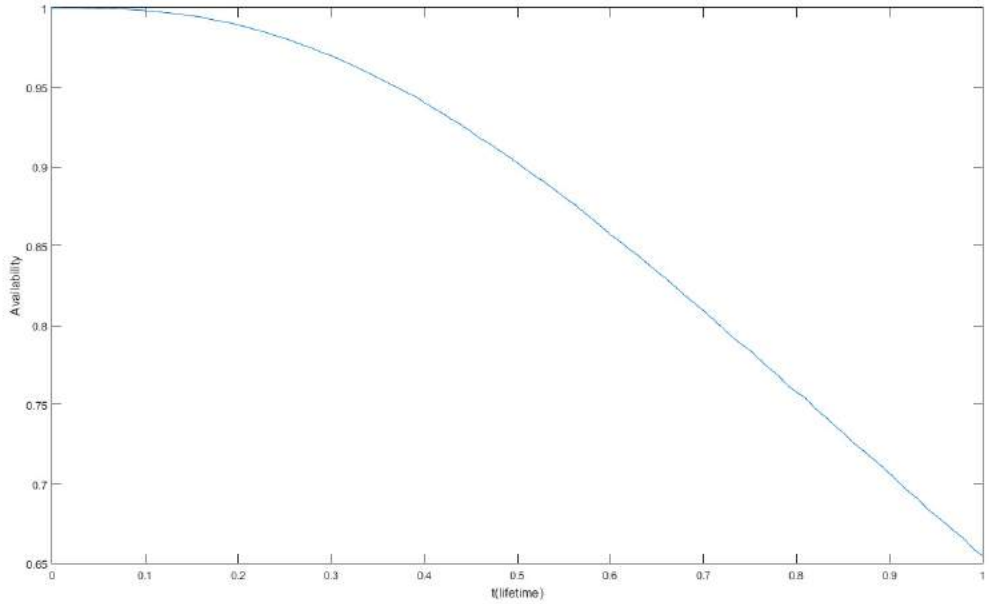


Figure 4.11: Monte Carlo Simulation Result of the Proposed Controller with 4 Module per Leg and Failure Acceptance of 1 (n=1) with 100 steps and 1000000 iteration per step

4.5 Markov Process and Reliability Assessment

Fault-tolerant controllers appear in different architectures (e.g. static,dynamic,...) and each architecture has its own advantage and failure rate. Single controller has failure rate (λ) which was calculated in previous section. In fault-tolerant controllers, multiple controllers have been arranged to function in a specific way. Therefore at a specific time, each controller is at defined state (functional or failed). Transition between states and the rates can be defined with the Markov model of the controller[78]. State and time in Markov modeling can be continuous or discrete (4.3). In reliability analysis of controller boards, state which defines the controllers status are defined discrete and the time is chosen to be continuous.

Table 4.3: Markov Model Types

Type	Acronym	State Space	Time Space
1	DSMC	Discrete	Discrete
2	DSMP	Discrete	Continuous
3	CSMC	Continuous	Discrete
4	CSMP	Continuous	Continuous

The important tool in reliability analysis is Poisson process. This process can be used to model systems in which number of occurrence per time interval is being evaluated. Let's assume a non-negative interval constant which has dimension of ($time^{-1}$). This is the rate of occurrence or transition rate. If $K_{t,t+h}$ denotes the number of events occurring in time interval *(t,t+h)*, where h is a very small time interval, then we can assume:

$$P(K_{t,t+h} = 0) = 1 - \lambda h + o(h) \tag{4.10}$$

$$P(K_{t,t+h} = 1) = \lambda h + o(h) \tag{4.11}$$

and it follows that:

$$P(K_{t,t+h} > 1) = o(h) \tag{4.12}$$

$o(h)$ denotes function with powers of h greater than first. Under these conditions, $o(h)$ will approach zero as $h \to 0$. Assume a Poisson process with rate λ, the mathematical model can be represented with a set of differential equations. By assuming that system can exist in one of N state from 1,2,...N then the probability that system is in state j is given by $P_j(t)$. On a system level, the state probabilities are given by the state probability vector $P(t)$ as following:

$$P(t) \equiv [P_1(t) \;\; P_2(t) \;\; ... \;\; P_N(t)] \tag{4.13}$$

The state space defining the system behavior is given by:

$$\frac{d}{dt}P(t) = P(t) \times A \tag{4.14}$$

Where A is a $N \times N$ state transition rate matrix for the system.

The element of the A matrix is defined by:

1- Off-diagonal elements($i \neq j$)

$$a_{ij} = \lim_{dt \to 0} \left\{ \frac{\text{Prob}[StateChange(i \to j) during (t, t+dt)]}{dt} \right\} \tag{4.15}$$

2- Diagonal elements:

$$a_{ii} = -\sum_{j \neq i} a_{ij} \tag{4.16}$$

To solve the differential equation, we also need initial conditions, that is the probability vector at $t = 0$, given by $P(0)$. Then the state probabilities cab be evaluated from:

$$P(t) = P(0) \times \exp(-At) \tag{4.17}$$

The set of differential equations representing system can also be shown by state-space equations. By doing that, it is possible to use computer programs to solve the equations and find numerical value for answer. Figure 4.12 shows the block diagram for state-space representation of a system and equations 4.18 and 4.19 show the mathematical representation.

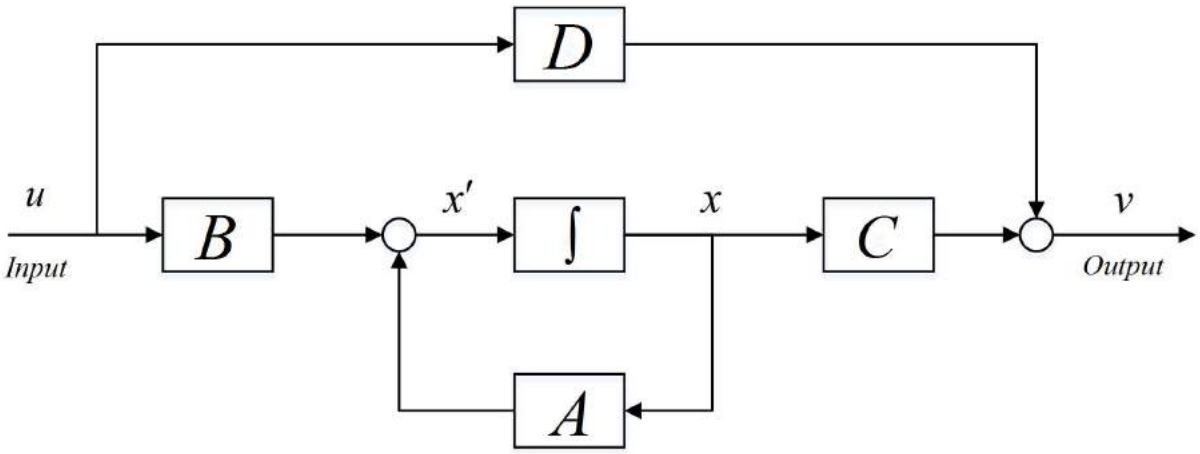


Figure 4.12: Block Diagram Representation of State-space Equations

$$x' = Ax + Bu \tag{4.18}$$

$$v = Cx + Du \tag{4.19}$$

In this diagram, A-matrix contains the transition rates of the states. B-matrix represents

the input signals and since the inputs are represented through initial states, this matrix is zero. C-matrix represents the output of the system and it contains elements for availability and unavailability of the system. D-matrix is related to transmission coefficients. This matrix can be used if a failed component has been replaced, otherwise this can also be zero.

4.5.1 Markov Model of Common Fault-tolerant Controllers

Instead of mathematical representation of the differential equations, it is possible to represent them using graphical method. In graphical representation, each state of the controller can be represented by a circle and if there is any possible transition between states, it can be shown be a arrow line. Figure 4.13 show a single controller (simplest controller architecture) which has 2

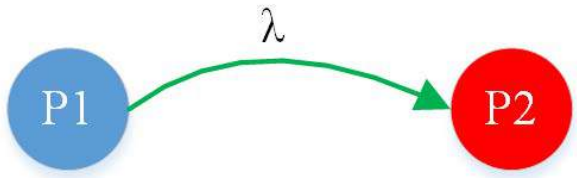


Figure 4.13: Markov Model of a Single Controller

state of operating. At P1, the system is functional and it is desired to stay at this state. However, a fault in the controller change the state of the system to P2. In this case, the system has failed and it is no longer functioning. Since the probability of transition between states are assumed exponential (based on Poisson process result), only the rate of transitions will be shown. In a controller system, there might be several available (green) and unavailable (red) states in the

system. In calculating each type of states, the probability of presence in each type of state will be calculated based on the transition and failure rate matrices. Based on the model discussed in the previous section we have:

$$P_1(t+h) = P_1(t)[1-\lambda h + o(h)] \tag{4.20}$$

$$\frac{P_1(t+h)-P_1(t)}{h} = -\frac{\lambda h P_1(t)}{h} + \frac{o(h)}{h} \tag{4.21}$$

Taking the limit as $h \to 0$, we obtain:

$$\lim_{h\to 0} \frac{P_1(t+h)-P_1(t)}{h} = \frac{d}{dt}P_1(t) = P_1'(t) \tag{4.22}$$

by noting that $\lim_{h\to 0} \frac{o(h)}{h} = 0$, then:

$$P_1{}'(t) = -\lambda P_1(t) \tag{4.23}$$

The same method can be used for P_2 to get the following result:

$$P_2{}'(t) = \lambda P_1(t) \tag{4.24}$$

Therefore, we have the following equation set with initial conditions:

$$\begin{cases} P'_1(t) = -\lambda P_1(t) \quad P_1(0) = 1 \\ P'_2(t) = \quad \lambda P_1(t) \quad P_2(0) = 0 \end{cases} \tag{4.25}$$

The solution to these equations is as below:

$$\begin{aligned} P_1(t) &= e^{-\lambda t} \\ P_2(t) &= 1 - e^{-\lambda t} \end{aligned} \tag{4.26}$$

It is desired that system stays in P1 and avoid P2. The probability that system is in P1 is called reliability (availability) and the probability of system in state P2 is called unavailability.

$$R(t) = P_1(t) = e^{-\lambda t} \tag{4.27}$$

In this case, λ is the summation of all failure rates for components in the controller card.Therefore we have:

$$\lambda = \lambda_1 + \lambda_2 + \lambda_3 + \cdots + \lambda_n \tag{4.28}$$

In some systems, the failure would not follow exponential model. It is possible to have non-exponential transition probability (semi-Markov modeling), but it makes calculation more complicated[18]. In repairable systems (systems with recovery blocks and transient failure), back transition from failed state is also possible (repair rate known as ε). In the current failure rate calculation, static failure is being modeled therefore repair rate is not included in the calculation. In fault tolerant controllers (figure **??**), Markov model can be used to calculate the reliability of the system. In static fault-tolerant controller (TMR) the approximate formula for

(a) Markov Model for Static Redundancy (b) Markov Model for Dynamic Redundancy

Figure 4.14: Markov Model for Fault-tolerant Controllers

probability of failure is given by:

$$P(t) \approx 3(\lambda t)^2 \qquad (\lambda t \ll 1) \tag{4.29}$$

and the exact formula for reliability is:

$$R(t) = P_1(t) = e^{-3\lambda t} + 3e^{-2\lambda t}(1 - e^{-\lambda t}) \tag{4.30}$$

For dynamic controller with 2 controller (two-component parallel system), the approximate formula for probability of failure is given by:

$$P(t) \approx (\lambda t)^2 \qquad (\lambda t \ll 1) \tag{4.31}$$

The exact formula for reliability of a two-component parallel system is given by:

$$R(t) = P_1(t) = 1 - (1 - e^{-\lambda t})^2 \tag{4.32}$$

In the proposed controller, for each converter module, three parallel controller are available (figure 4.15). This controller architecture has better reliability in comparison with triple module redundancy (TMR) and for the same amount of hardware, more reliability can be achieved. For

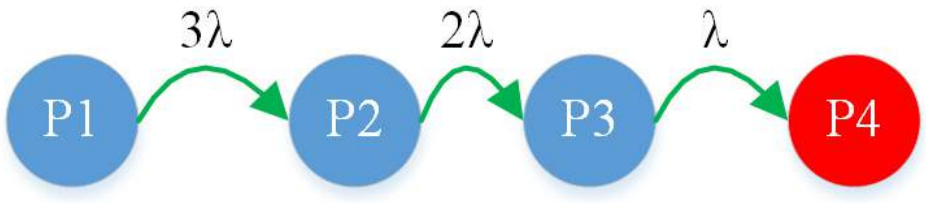


Figure 4.15: Markov Model of Three-component Parallel Controller

three-component parallel redundancy, the approximate probability of failure is given by:

$$P(t) \approx (\lambda t)^3 \tag{4.33}$$

Therefore, the exact formula for the reliability of a three-component parallel system is:

$$R(t) = 1 - (1 - e^{-\lambda t})^3 \tag{4.34}$$

In order to choose the best controller method, reliability of different controller architecture has been calculated (figure 4.16). For the same amount of time-failure rate (λt), three-component parallel fault-tolerant controller shows better reliability than other architectures. Therefore this architecture could be the best option for the proposed controller in MMC.

The analytical method discussed here can be used for simple systems. In complicated controllers

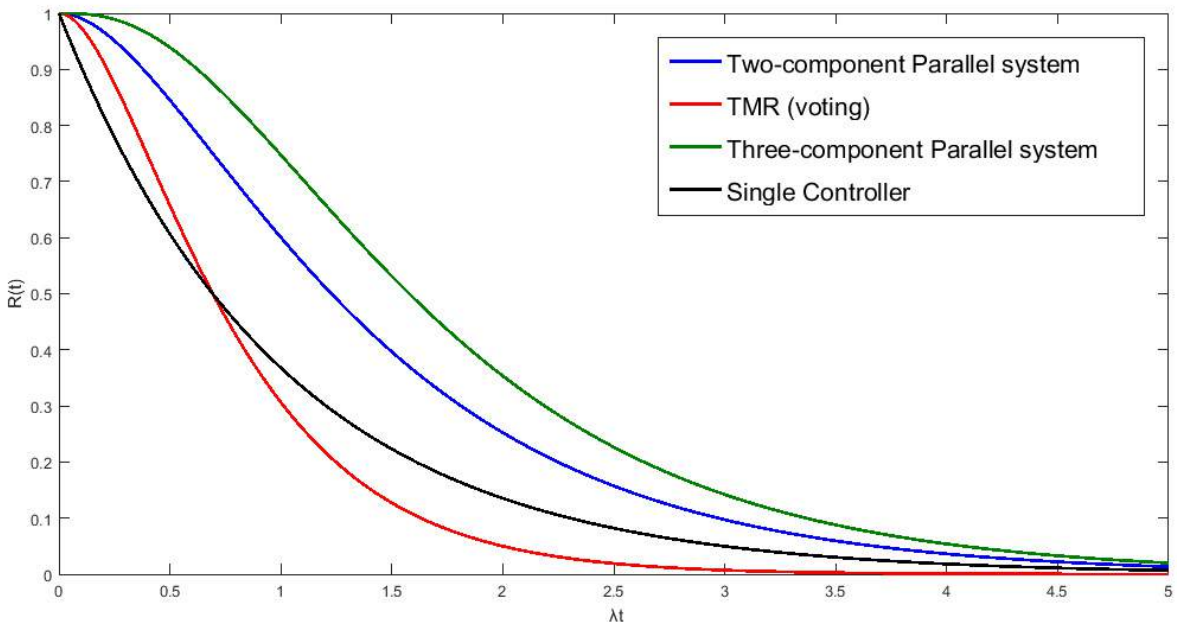


Figure 4.16: Reliability Assessment of Different Controller Architectures

(e.g. proposed controller), numerical method is used to solve the differential equations and find the solution for reliability [18][78]. In the following section, reliability of the proposed controller will be investigated by using computer analysis.

4.6 Computer Aided Reliability Assessment of the Markov Chain Model

The process of reliability assessment starts with defining the cases in which system would fail. Therefore, it is possible to complete Markov chain, assign failure rates, extract system matrix representation and solve the equations. A computer program will compute all the statistical values necessary for evaluation of fault-tolerant system based on the mathematical model. Markov model of basic controller for modular multi-level controller (MMC) with no bypass

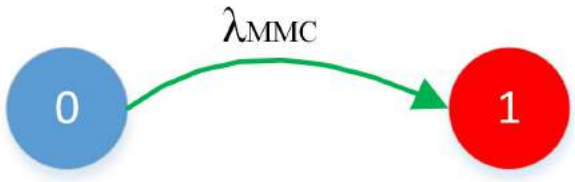


Figure 4.17: Markov Model of MMC with No Bypass Capability

capability is depicted in figure 4.17. In this mode failure of one module leads to failure of whole converter, therefore we have:

$$\lambda_{MMC} = Number\ of\ Modules \times \lambda_{\text{Single } module} + \lambda_{Controller} \tag{4.35}$$

To benefit from modularity of MMC, it is necessary to bypass modules in case of a failure. If open circuit happens in one module, current can't pass through that converter leg and MMC would not function correctly. Short circuit can cause wrong functionality in the system and it

is not desired too. After detection of failure that module must be bypassed and a fault signal should be sent to master controller to compensate the voltage of other modules. Markov model of a MMC with bypass capability (in which it becomes unavailable with *n* module failure) is depicted in figure 4.18.

Centralized controller for MMC is extremely vulnerable to fault. Since there is only one

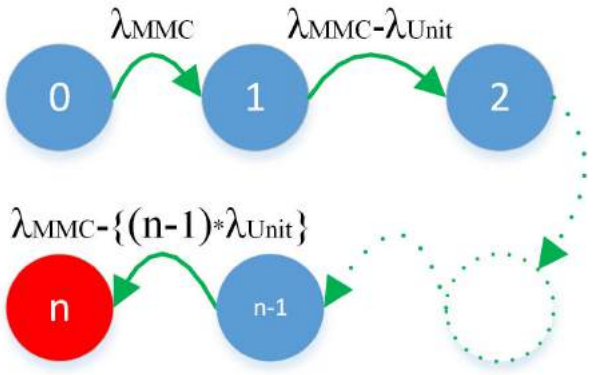


Figure 4.18: Markov Model of MMC with Bypass Capability

controller in the system, failure in the controller stops the functionality of the entire system. By using distributed controllers in converter and fault-tolerant control architectures, it is possible to increase availability of the converter. In the proposed controller architecture, the adjacent controllers act as standby controller for each other. Failure in one of the controllers, triggers the neighbor controller to handle the task of the failed controller. A module is called failed, only if there is no adjacent controller to handle its tasks. In this case, the module and its controller will be bypassed completely since there is no way to control it. Based on the voltage rating of the power semiconductor in each module, each leg can handle up to a maximum of failed modules (*n* modules).

Figure 4.19 shows the controller for a MMC with 4 modules per leg (the same converter for

simulation and experimental result). Each state must be reviewed one by one to check if system is available or not. In this case, there are three unavailable states (0111,1110 and 1111). Failure rate from each state to the other one is equal to the failure of one controller (λ_{Unit}). As it can

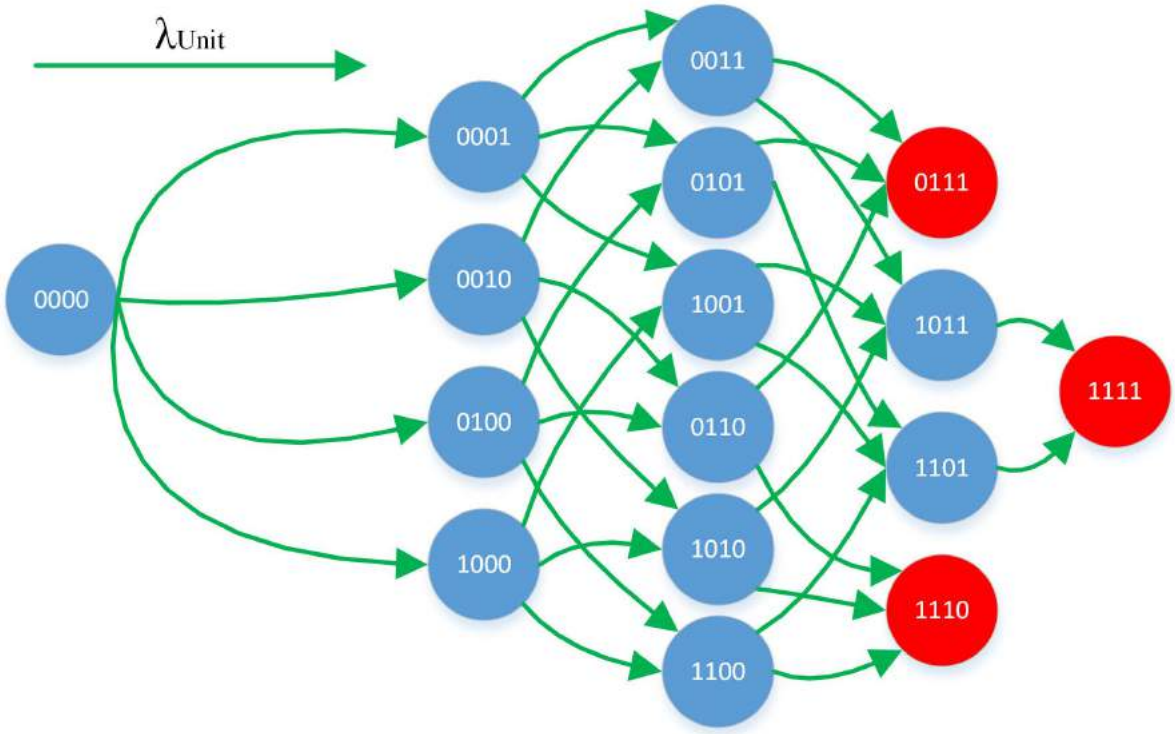


Figure 4.19: Markov Model of MMC Controller (One Leg) with 4 Module per Leg and Failure Acceptance of 1 (n=1)

be seen, modeling of such controller can be complicated because each failure case is one state and the total number of states is 2^n. Equations 4.36 to 4.41 represent state-space equations for Markov model of proposed controller (one leg). In order to evaluate larger systems (e.g. 100 modules per leg), a computer program must be written to define all the possible states, extract the state-space equations and solve the equations to find statistical data of the model.

$$A = \begin{bmatrix}
-4\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\lambda & -3\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\lambda & \lambda & -3\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\lambda & \lambda & 0 & -3\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\lambda & \lambda & 0 & 0 & -3\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \lambda & 0 & 0 & -2\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \lambda & 0 & 0 & -2\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \lambda & 0 & 0 & -2\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \lambda & \lambda & 0 & 0 & 0 & 0 & -2\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \lambda & 0 & \lambda & 0 & 0 & 0 & 0 & -2\lambda & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \lambda & \lambda & 0 & 0 & 0 & 0 & 0 & -2\lambda & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \lambda & \lambda & 0 & \lambda & 0 & 0 & -\lambda & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \lambda & 0 & \lambda & 0 & \lambda & 0 & 0 & -\lambda & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \lambda & \lambda & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & \lambda & \lambda & 0 & 0 & 0 & -\lambda & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda & \lambda & \lambda & \lambda & \lambda & 0
\end{bmatrix} \tag{4.36}$$

$$B = [\emptyset] \tag{4.37}$$

$$C = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{bmatrix} \tag{4.38}$$

$$D = [\emptyset] \tag{4.39}$$

$$x_{t=0} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.40}$$

$$\begin{cases} x' = Ax + Bu \\ v = Cx + Du \end{cases} \tag{4.41}$$

One of the software for evaluating the proposed Markov model is Isograph™ Reliability Workbench (RWB). This software is capable to calculate reliability parameters for Markov model of a system based on state-space method. Figure 4.20 shows the Markov model for one leg of the converter. Failure rate for this model has been calculated in the previous section ($\lambda = 9.04 \times 10^{-7} per\ hour @ 25°C$) and unavailability mode is set to 0111, 1110 and 1111 cases. The result of availability for precise modeling has been shown in figure 4.21. Part a shows

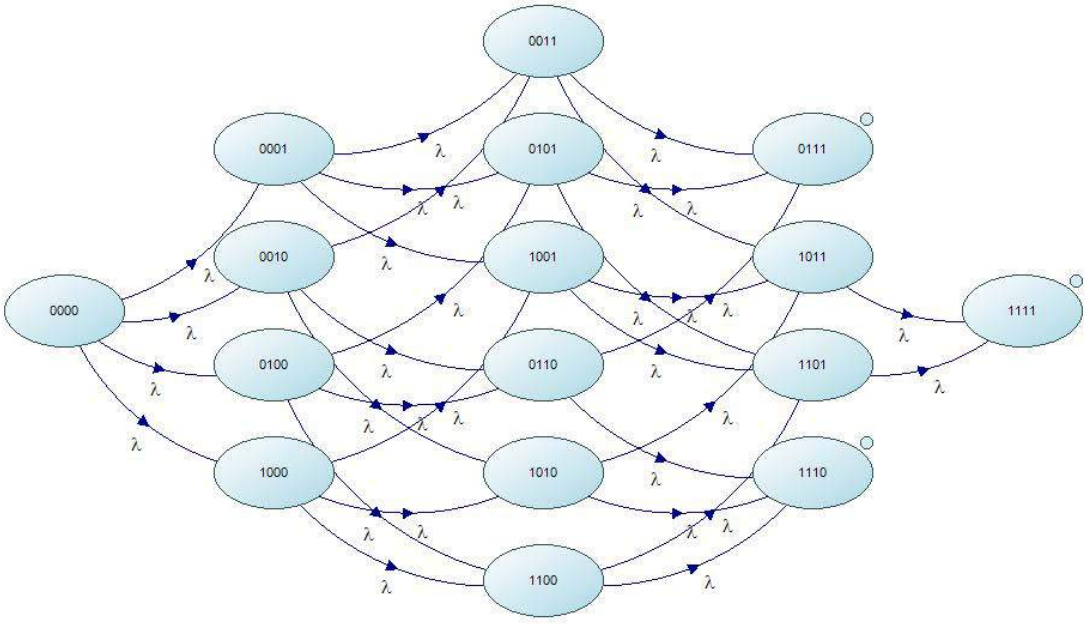


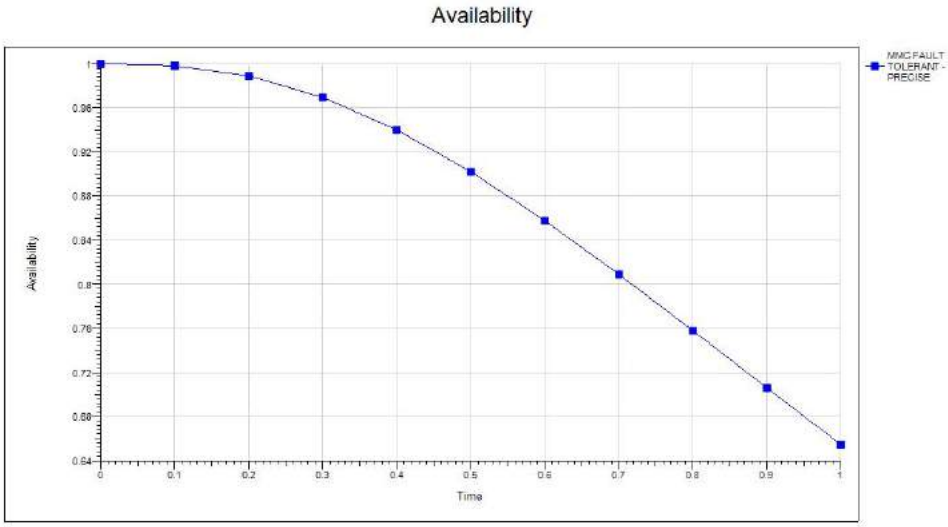
Figure 4.20: Precise Markov Model of MMC Controller in RWB

the availability for $\lambda = 1$ which is the normalized form and part b shows the result for $\lambda = 9.04 \times 10^{-7}$ which is the failure calculated from previous section.

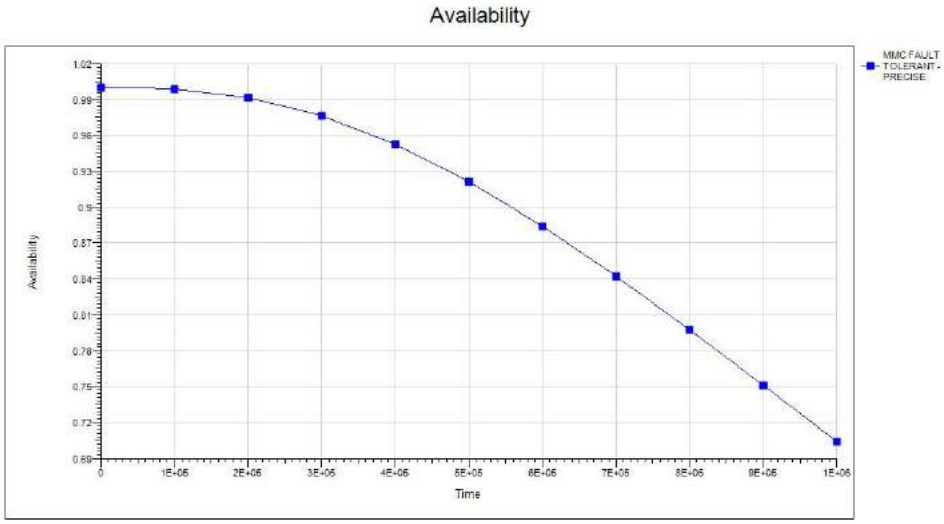
Unavailability of a system equals $1 - R(t)$ and is another reliability property of a system. Figure **??** demonstrate the unavailability of the proposed controller.

To summarize of the reliability analysis of the proposed controller, the result of reliability

Figure 4.21: Availability for Precise Model of MMC Controller (One Leg) with 4 Module per Leg and Failure Acceptance of 1 (n=1)

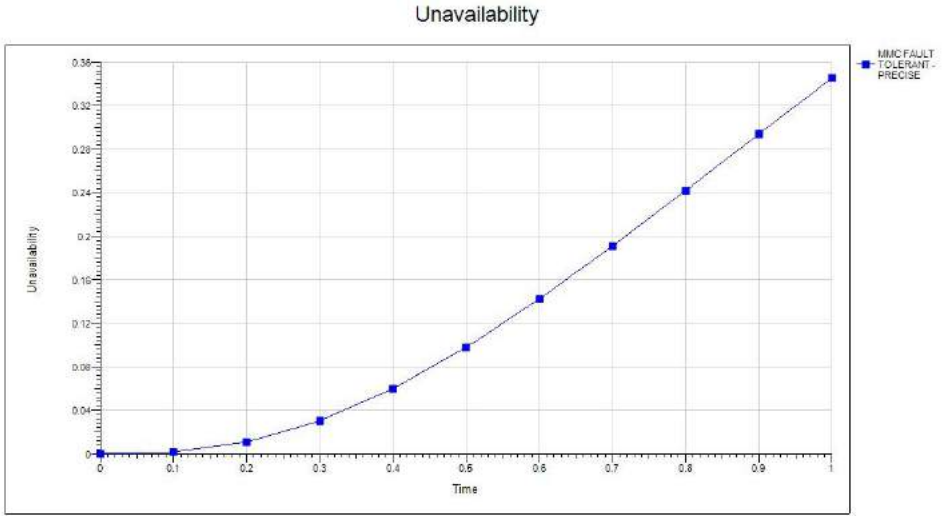


(a) with $\lambda = 1$

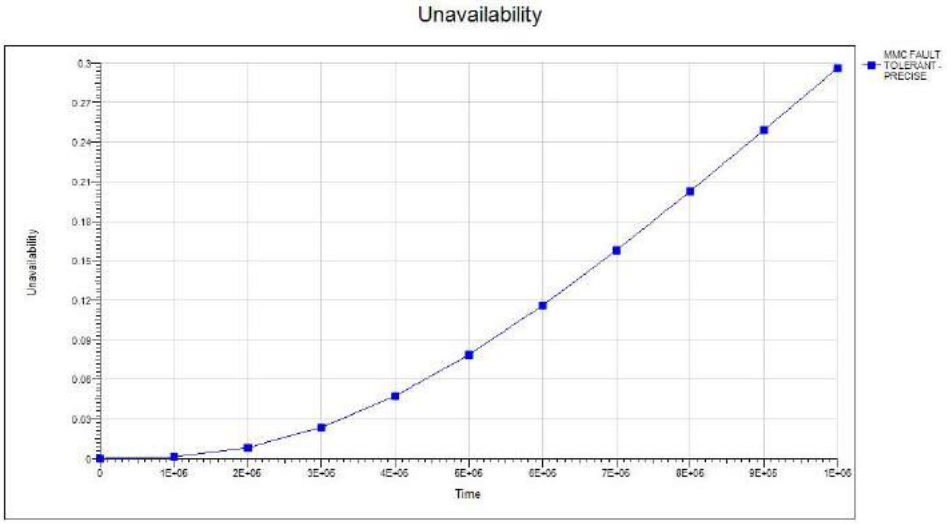


(b) $\lambda = 9.04 \times 10^{-7}$

assessment has been included in table 4.4.



(a) with $\lambda = 1$



(b) $\lambda = 9.04 \times 10^{-7}$

Figure 4.22: Unavailability for Precise Model of MMC Controller (One Leg) with 4 Module per Leg and Failure Acceptance of 1 (n=1)

Table 4.4: Summary of Reliability Analysis for Proposed Controller and Other Controllers

	Single Controller	Distributed Controller (with Bypass)	Proposed Controller	Proposed Controller
	$\lambda = 1$	$\lambda = 1$	$\lambda = 1$	$\lambda = 9.04 \times 10^{-7}$
Lifetime	1	1	1	1000000
Unavailability	0.9817	0.8558	0.3454	0.2960
Availability	0.0183	0.1442	0.6545	0.7039
Unreliability	0.9817	0.8558	0.3454	0.2960
Reliability	0.0183	0.1442	0.6545	0.7039
Failure Frequency	0.0732	0.3777	0.5102	0.0000
Repair Frequency	0	0	0	0
Conditional Failure Intensity	4	2.619	0.7796	0.0000
Conditional Repair Intensity	0	0	0	0
No of Expected Failures	0.9825	0.8551	0.3454	0.2960
No of Expected Repair	0	0	0	0
Total Downtime	0.7544	0.4693	0.1238	102915.093
Total Uptime	0.2456	0.5307	0.8761	898084.907
	Mean Values			
Mean Unavailability	0.7544	0.4693	0.1238	0.1029
Mean Availability	0.2456	0.5307	0.8761	0.8970

Table 4.4 shows that the performance of the proposed controller is superior compared to single controller and distributed controller only with bypass capability. The mean availability is 0.24 for single controller, 0.53 for distributed controller and 0.87 for proposed controller. In this reliability assessment, the repair rate is not considered for the controllers otherwise the proposed controller could have higher availability rate.

Chapter 5

Fault-tolerant Firmware Design for Proposed Controller

5.1 Introduction

In this chapter, methods of designing fault-tolerant firmware will be investigated. The proposed methods are helpful to insure that software has the minimum amount of fault and in the case of failure due to the firmware design, it would be handled efficiently. First, methods for fault avoidance and fault-tolerant firmware will be investigated. Later in this chapter, special topics in design of the second-generation fault-tolerant controller like Agreement on the master controller and grouping of the slave controllers will be presented.

5.2 Software Fault Categorization

Software is not a physical entity and unlike hardware, it can't get affected by environmental or aging effects. Therefore, some studies suggest that software faults are permanent (unlike

transient hardware faults)[41] and the code is already broken. Gray[35] has classified software bugs into **Bohrbugs** and **Heisenbugs**(Figure 5.1). Bohrbugs are permanent design fault and can be revealed in the first design stages. They can be detected and removed in the development stages of the software through debugging tools. Although some bohrbugs may exist in the final version of software, they can easily be debugged. Heisenbugs on the other hand have complicated structure. They can only be detected on certain collision of events and rarely reproducible. Therefore, heisenbugs are transient failures and may not happen again if software is restarted. Studies have shown that 70% of the bugs in Tandem computer are cause by transient faults mainly by race conditions and timing errors[57]. The third category of fault is **aging**

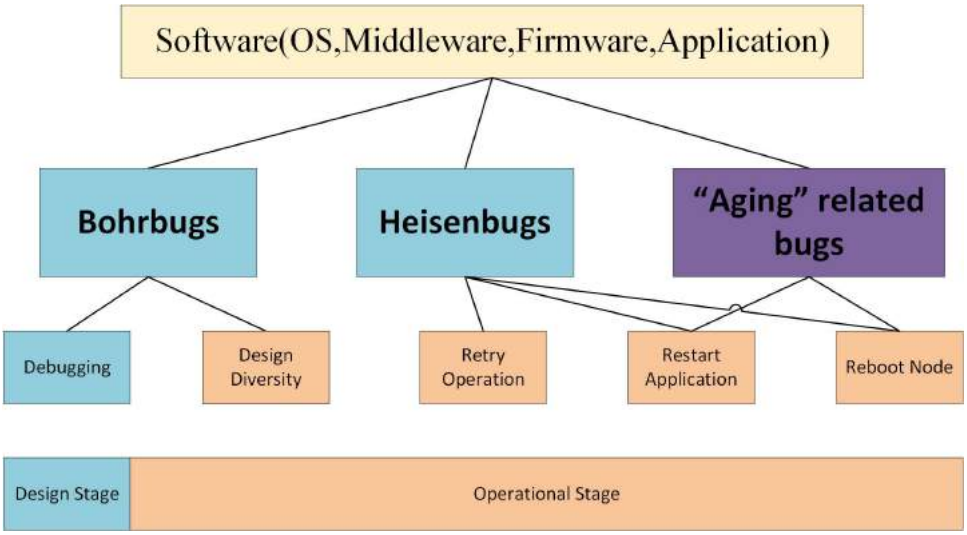


Figure 5.1: Classification of Software Fault Based on Gray Model

related bugs. These faults can accumulate during run-time of the software and may lead to performance degradation or transient errors[101]. Most cause of the errors are memory related including data corruption, unreleased file locks,... Rejuvenation process is necessary for fault management and is based on restarting the application occasionally.

5.3 Fault Avoidance Techniques in Firmware Design

It is always good idea to avoid any fault in the firmware before deploying fault-tolerant methods to handle these faults. In development of the firmware, different techniques, tools and methods may be used which decrease the chance of failure in the firmware and speed up software validation in the testings. In the following, these methods will be investigated [81].

5.3.1 MISRA-C Coding Style and Application in Fault-tolerant Firmware Design

Automotive industry is one of the areas in which robustness of firmware design is extremely important. Each car has several modules like engine control unit (ECU), anti-lock braking system (ABS), ... that protect the life of the passengers. These modules must operate in the best possible way and that requires good firmware design too. C language is the leading tool for developing firmware and it has several benefits that makes it preferable to other programming languages. The common ISO C gives a lot of freedom to developers such as memory access using pointers and coding styles that is not desired for robust firmware design. This freedom may also lead to failure in some cases and it is better to avoid using features that can cause it. UK's Motor Industry Software Reliability Association (MISRA) has came up with a set of 127 rules for C language programming that helps code developers avoid unwanted results [72]. The standard is called MISRA-C and out of proposed rules, 93 are required and 34 are advisory. These rules may be used in development of fault-tolerant firmware design to avoid Bohrbugs and Heisenbugs in early stages. Some compilers have the capability to check for the MISRA-C rules and message if any offense is detected. The following are some of the example rules:

- **Rule 17.5 (advisory):** *The function argument corresponding to a parameter declared to*

have an array type shall have an appropriate number of elements.

Passing an array with different number of elements to another array parameter is possible in C language. This may result in unexpected result.

- **Rule 7.3 (required):** *The lowercase character "l" shall not be used in a literal suffix.*

  Using the upper case "L" removes the ambiguity between "1" (logic one) and "l" (letter el) when declaring literals.

- **Rule 9.3 (required):** *Arrays shall not be partially initialized.*

  Providing explicit initialization for each element makes it clear that every element has been considered.

- **Rule 14.8 (required):** *The statement forming the body of an "if", "else if", "else", "while", "do ... while", or "for" statement shall always be enclosed in braces.*

  This rule prevents unintended code design, because all the code for one condition sits in the braces of that condition. It is not possible to have one sentence code following the condition.

- **Rule 12.3 (required):** *The comma operator should not be used.*

  The comma operator may be harmful to the readability of the code.

- **Rule 17.4 (mandatory):** *All exit paths from a function with non-void return type shall have an explicit return statement with an expression.*

  If a non-void function does not return a value but the calling uses the returned value, the behaviour is undefined. Therefore all control path must be checked to return a value.

5.3.2 Software Partitioning

In hardware design, parts of circuit that do different tasks are isolated from each other by using different power sources. The reason may be power, frequency or signal ratings, but what they all have in common is that they can't share the same resources and this can affect their performance. In high-reliability software that contains multiple tasks, the same technique may be used to separate software tasks from each other and separate resources (memory and time) that is being used by each module [104, 27, 49, 28]. Software partitioning will help the extendibility of the software design in the future without affecting current software architecture.
In correspondence with federal aviation regulations/joint aviation requirements 25, software safety level may be categorized as in table 5.1.

Table 5.1: Software Safety Level and Its Impact Factor

Software Safety Level	Failure Impact	Failure Rate
Level A	Catastrophic	$10^{-9}/h$
Level B	Hazardous	$10^{-7}/h$
Level C	Major	$10^{-5}/h$
Level D	Minor	$10^{-3}/h$
Level E	No effect	n/a

In each level, consequence of software failure has been specified and based on that, developers may separate software partitions. In this architecture, data and control registers of each level may only accessed by the same or higher software level. This will prevent unintentional data access which leads to memory corruption in the critical software sections.
Figure 5.2 [49] demonstrates the architecture of memory (spatial) partitioning mechanism

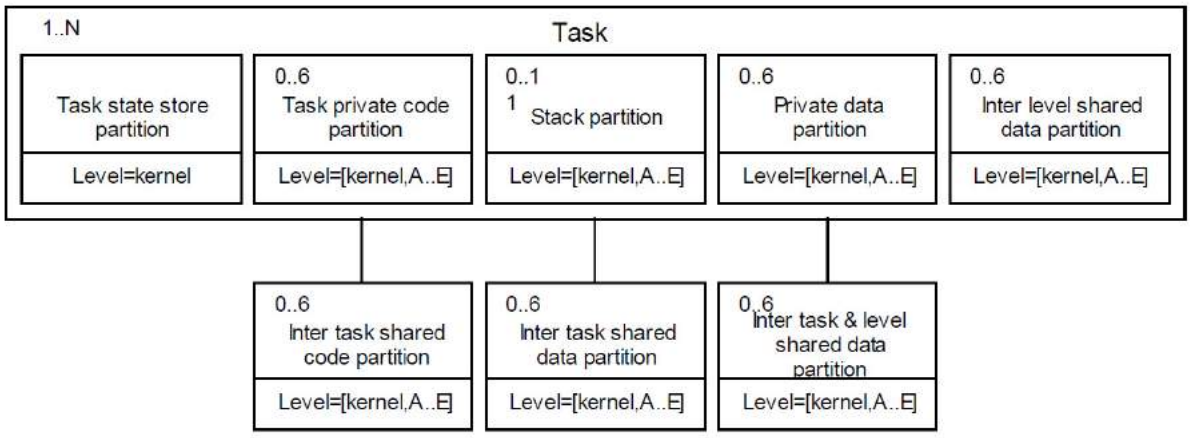


Figure 5.2: Architecture of Software Partitioning Based on Safety Levels

based on the safety level of avionics systems. In this mechanism, each software level use separate memory pages, stack, private data and inter-level shared data. Kernel is responsible for controlling the interactions between these levels and schedule tasks based on the safety levels. Memory management unit (MMU) is responsible to separate memory usage of tasks and allocate each task with separate memory space. Whenever each task is being executed, MMU will switch to the memory page of the running task and context switch will happen as fast as possible with isolating content of each task from other tasks. Inter-layer data between tasks can be passed through shared memory and mutual exclusion (mutex) mechanism. Based on the software level, having a backup of the available data might be necessary. In this case, a copy of the variables would be stored and restored in case of the fault detection. Multiple copies of data for catastrophic tasks are necessary. After completion of each operation, all available copies would be compared and the majority value would be chosen as the correct value.

In single thread firmware design, there is no kernel or MMU available to handle memory partitioning. In this case, memory access can be checked to make sure it is in the allowed memory range. In pointer memory access, the range may be defined for each task and each task may only

access an specified range of memory.

The other resource that must be partitioned is time (temporal partitioning) [27]. Based on the scheduling method, each task will be given a time slot to complete its procedures. The scheduler must make sure each task is not taking processing time of other tasks and it is limited to its predefined time slot. In case of failure in satisfying this criteria, that task must be isolated and its effect should not propagate to other software modules. Figure 5.3 demonstrate an example of such scheduling where each task has specific time slot and it would be preempted by scheduler if timing criteria is not met.

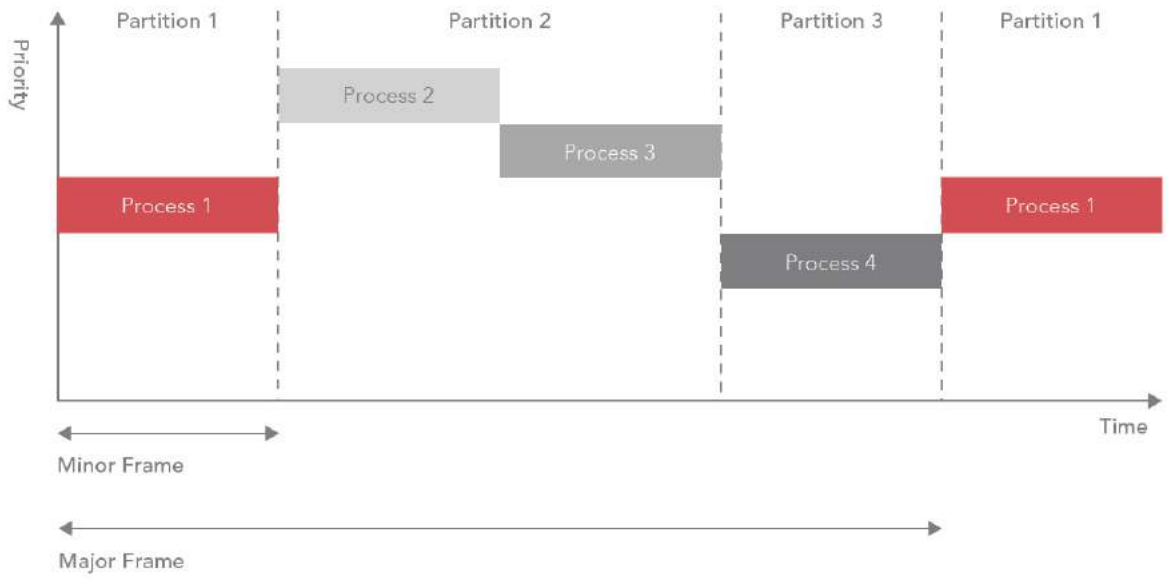


Figure 5.3: Temporal Partitioning in Reliable Systems

5.3.3 Scheduling, Timing and Interactions

In hard real-time systems, each task must meet the required deadline, otherwise it may be accounted as a failed task [16, 82]. In order to reach required deadline for each task, a task scheduler must be used to manage task handling based on task execution time (C_i), priority, period (T_i) and waiting time (R_i). Based on the application, task scheduling might be static or dynamic. In static scheduling (round-robin loop), task follow a static sequence and each task is run in order. In this case, worst case response time for high priority task is sum of all task times. In dynamic scheduling, tasks are run based on priority, execution time and the specified rate for each individual task. Tasks don't follow an order and may preempt each other if the priority of the new task is higher.

Dynamic scheduling is more complicated than static scheduling and usually a real-time operating system (RTOS) is used for handling it. In the proposed controller, no RTOS has been used for scheduling task and operations, therefore static scheduling has been used to schedule tasks. In this scheduler, each task runs once in each loop cycle and only interrupt routines may interrupt the running task (task run until compilation). In static scheduler, it is important to measure execution time for each task and the period that each task run during the round robin loop. The necessary condition for static scheduling is as following:

$$\sum_{1}^{i} \frac{C_i}{T_i} < 1 \tag{5.1}$$

In equation 5.1, T_i refers to the period of each task and C_i is the execution time for each task. Usually, interrupts may happen more than once in each loop and worst cast must be considered when designing the scheduler.

Table 5.2: Execution Time for Main Tasks in Master Controller

Task	Execution Time (C_i) (μS)	Period (T_i) (μS)
Master_Control_Loop()	37	200 (called @ 150)
Update_Slave_Controller()	42	200 (called @ 10)
RX_Data_Parsing()	10	200 (called @ 100)
RX_Interrupt_Handling	0.5	3
Round_Robin_Loop (total)	122.3	200

Table 5.3: Execution Time for Main Tasks in Slave Controller

Task	Execution Time (C_i) (μS)	Period (T_i) (μS)
Slave_Control_Loop()	6	200 (called @ 150)
Update_Master_Controller()	2	200 (called @ 10)
Master_RX_Data_Parsing()	9	200 (called @ 100)
Update_Upper_Controller()	5	200 (called @ 195)
Update_Lower_Controller()	5	200 (called @ 195)
Parsing_Data_From_Upper_Controller()	3.5	200 (called @ 0)
Parsing_Data_From_Lower_Controller()	3.5	200 (called @ 0)
RX_Interrupt_Handling	0.5	3
Round_Robin_Loop (total)	67.3	200

In table 5.2 and 5.3, timing characteristics of tasks in master and slave controllers have been represented. Total utilization of processor in master controller is higher than slave controllers (due to complicated control loop in master controller). Processor utilization in slave controllers will increase as the number of power modules which are being controlled by each slave controller increase (represented result is for one power module per controller). Since the total execution time for each round robin loop in controllers is lower than the control period, controller structure is realizable and it is working in the safe region.

Interactions between master and slave controllers have been visualized in figure 5.4. Master controller is responsible for synchronizing all slave controllers in the system, therefore slave controllers will follow scheduling mechanism of the master controller through synchronization process. Whenever control timer interrupt is called, master controller compute the supervisory control and sends out the global data to all slave controllers. In each transmission, a slave controller would be requested to sends back the local data and status of the controller. In the remaining time in the control step time (which is 200 μS in the implemented controller), slave controllers check the status of the adjacent controllers to makes sure everything is as desired.

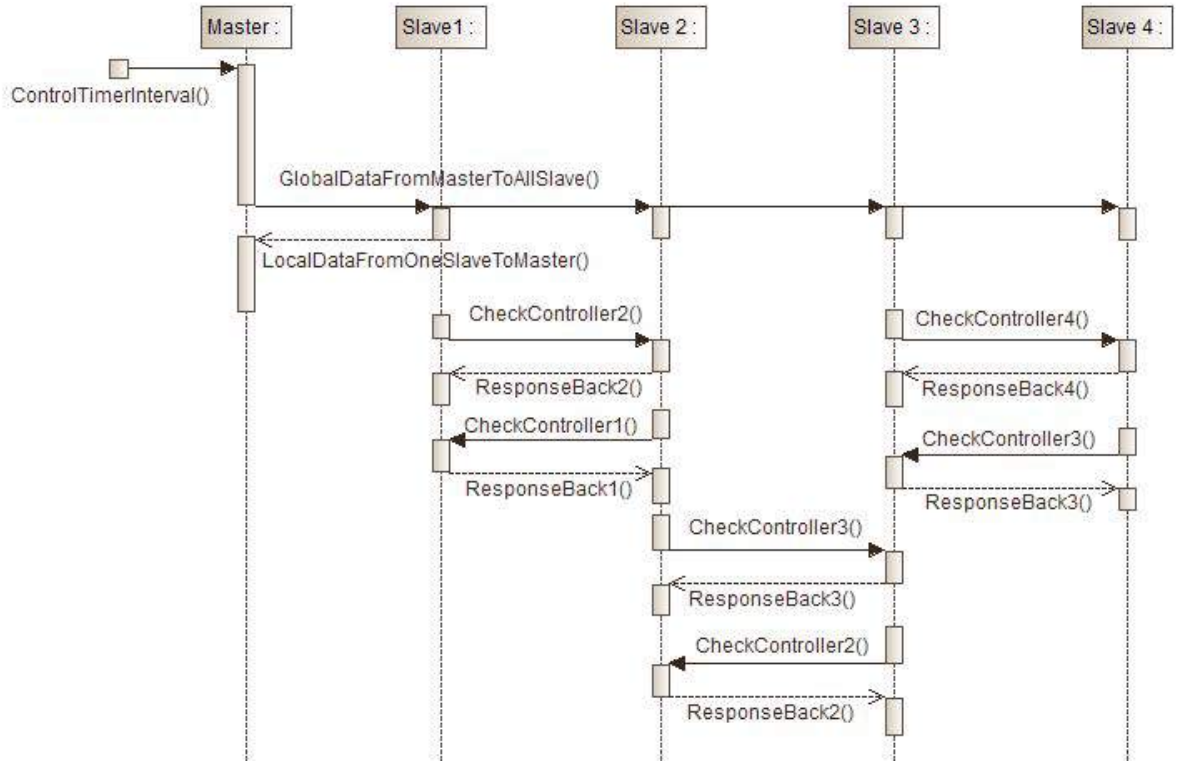


Figure 5.4: Sequence Diagram of Controller Interactions in First Generation Fault-tolerant Controller

Figure 5.5 demonstrates the sequence diagram of tasks in each controller. Each task is scheduled by the scheduler function to happen in the right time. Since all controllers are synchronized and they share data bus with each other, each task must happen in the right moment or it cause failure in the system. As discussed before, the scheduler is a round robin loop which is timed by an interval timer and only high priority interrupt like serial communication may preempt the scheduled task.

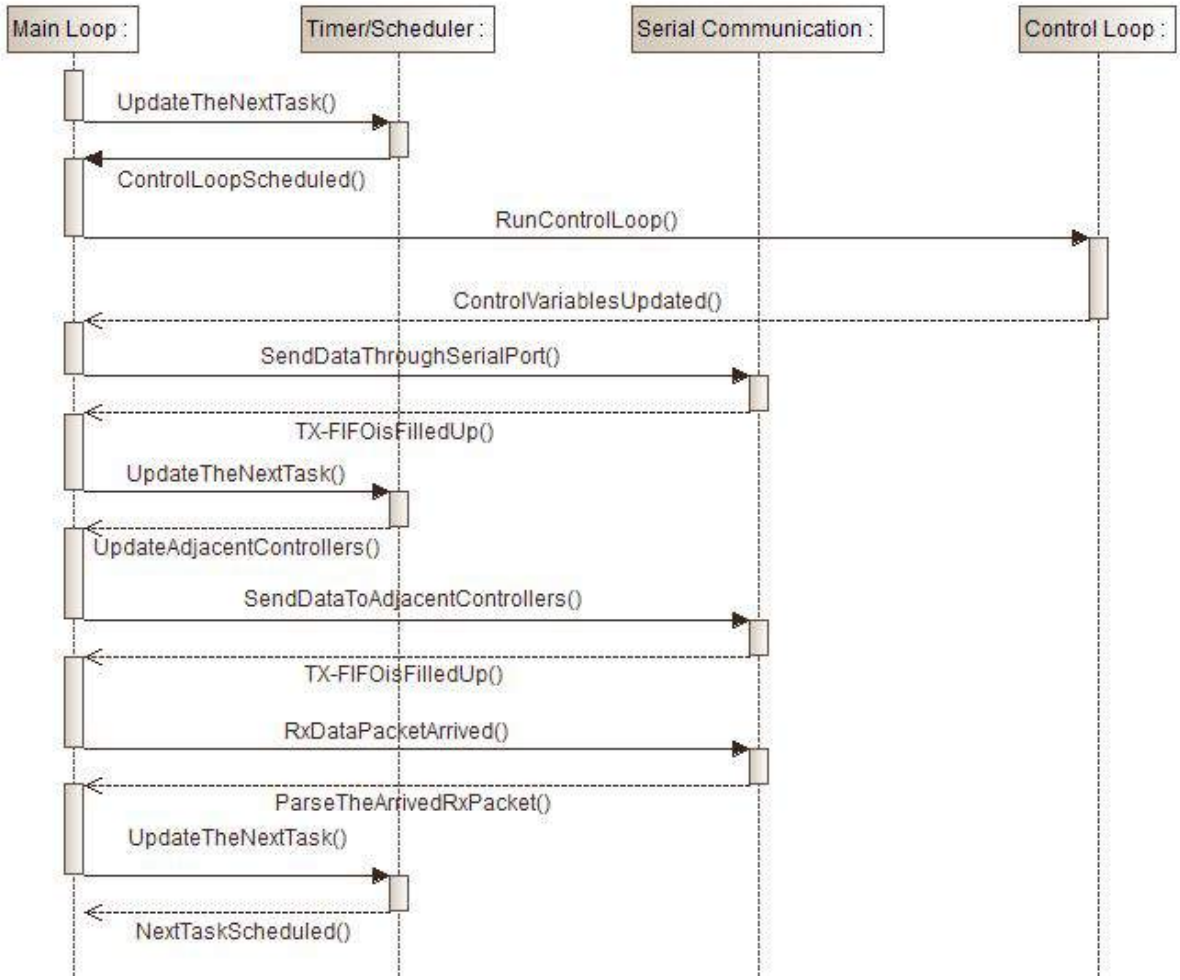


Figure 5.5: Sequence Diagram of Task Interactions in First Generation Fault-tolerant Controller

5.3.4 Software Rejuvenation

As discussed in the first section of the chapter, some types of software bugs are age related. Therefore as time passes, failure rate increases due to this type of software fault [100, 36, 42, 110, 102]. There are different reasons that aging related bugs may develop in the system and cause failure. Table 5.4 summarizes several reasons for aging related bugs in software. Age related bugs are very common in computers with operating systems. Whenever each task is created, fix amount of memory resource will be assigned to it. If these resources are not depleted after the task is done, that resource will be leaked and may not be available to the operating system (OS). As time passes, the leakage would be so much that there is no more memory available.

Table 5.4: Classes of Aging Related Bugs in Software

Class	**Extension**	**Examples**
Resource Leakage	(1) OS-specific (2) App-specific	-Unreleased memory (1,2) File handlers (1) sockets (1) -Unterminated processes (1) Threads (1,2)
Fragmentation	(1) OS-specific (2) App-specific	-Physical memory (1) -File system (1) -Database files (2)
Numerical error accrual	(1) OS-specific (2) App-specific	-Round-off (1,2)
Data corruption accrual	(1) OS-specific (2) App-specific	-File system (1) -Database files (2)

Fragmentation in memory has the same effect and will decrease the available memory to the OS. Garbage collector may decrease the amount of leaked or fragmented memory from tasks which are not running anymore, but it takes processing power and may not be effective all the times.

Numerical error in data variables is another type of age related bug. In floating point data, the resolution of variable decreases as the stored value gets bigger. This might happen in control blocks that store variables in floating point or where fix point data is converted to floating point and converted back to fixed point again. This rounding error is not OS related and might happen in all processor based systems. If the variable that has been stored get corrupted (that might happen due to high energy radiation), it might cause the same failure in the system.

Aging effects can be classified into *volatile* or *non-volatile* effects. volatile effects may be removed by fresh start of the computer. All of the mentioned bugs are volatile unless the bug is stored in a non-volatile memory or some type of memory which is not cleared by system restart. In the rejuvenation process, system would have a fresh start and all of its states must return to the initial values. Therefore, software rejuvenation is a periodic preemptive roll-back to the initial checkpoint to prevent failure in a continuously running application [42]. In the proposed controller, rejuvenation process may be scheduled whenever adjacent controllers are functional and there is no problem for the controller to go in rejuvenation process and become temporarily inactive. Rejuvenation model for the proposed controller is represented in figure 5.6 [110]. In

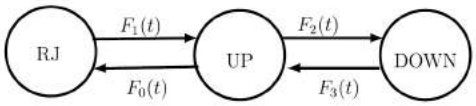


Figure 5.6: Rejuvenation Model for Fault-tolerant Controller

state 0, system is Up and is functioning correctly. State 1 is the rejuvenation state (RJ) and state 2 is the Down state. Based on the model, general distribution functions $F_1(t)$ to $F_4(t)$ may be defined. In the model, t_0 is the rejuvenation trigger interval and the mean time to carry out rejuvenation and reactive repair from Down state are respectively t_1 and t_2. Two-parameter Weibull pdf for the TTF is:

$$\begin{aligned} & f(t) = \frac{\beta}{\eta}(\frac{t}{\eta})^{\beta-1} e^{-(\frac{t}{\eta})^{\beta}} \\ & where: \ f(t) \geq 0, t \geq 0, \beta \geq 0, \eta \geq 0 \\ & \eta = scale \ parameter \\ & \beta = shape \ parameter \ (or \ slope) \end{aligned} \tag{5.2}$$

The CDF of this Weibull distribution is:

$$F(t) = 1 - e^{-(\frac{t}{\eta})^{\beta}} \tag{5.3}$$

The sojourn time in Up state is then given by:

$$h_0 = \int_0^{t_0} (1 - F(t))dt = \frac{\eta}{\beta}\Gamma(\frac{1}{\beta})G(\frac{1}{\eta^{\beta}}t_0^{\beta}, \frac{1}{\beta}) \tag{5.4}$$

where $G(x,\beta) = \frac{1}{\Gamma(\beta)} \int_0^x e^{-u}u^{\beta-1}du$ is the incomplete gamma function. Therefore steady state availability is:

$$A_{weib} = \frac{h_0}{h_0 + (1 - F(t_0))t_1 + F(t_0)t_2} \tag{5.5}$$

In the equations above, A_{weib} and $\frac{d(A_{weib})}{dt_0} = 0$ system of equations may be solved to find the optimum value of t_0. Controller may be scheduled for rejuvenation whenever interval time

arrives and the adjacent controller are Up and there is no side effect regarding this process.

In embedded systems, rejuvenation process may be applied to software drivers and initialization programs of peripherals in the system. Device registers may get corrupted and settings of the peripherals might not be as desired. Scheduler may set a free time slot to rejuvenate these peripheral and make sure they are functioning correctly.

Watchdog timer is a helpful tool in rejuvenation process. Not only it does rejuvenate the software in case of failure in the critical tasks, but it may be used as a force to rejuvenate the software based on the schedule. In this case, the timer would not be reset in the critical tasks and by over flowing it, the system would have a fresh start whenever it resets.

5.4 Fault-tolerant Firmware Design Methods

In the previous section, different design techniques have been used to avoid the possibility of failure in the firmware design and increase the reliability of the software. These techniques are necessary to decrease the cost of verification in the final stages. Although these techniques are necessary, but they are not enough. Fault-tolerant firmware design techniques help handling exceptions and faults in case it happens. In the following sections, these techniques are investigated.

5.4.1 N-version Programming

In N-version programming, different versions ($N \geq 2$) of software are implemented by different developers and these implementations are designed to give the same result [62, 19]. There are two different approaches in arranging the software module in functional behavior. In one method, all of the implementations will run simultaneously on the same processor or on different processors. The output of the software modules are compared and the majority in the output result is chosen as the correct value. This diversity in the software can remove the systematic errors in development stage just like using multiple hardware in the system. Common faults between different versions can't be detected and they still may cause error in the system. The cost of this method gets higher than N times the cost of single version. This software method is equivalent to the static redundancy in hardware implementation.

In the second approach, there are multiple versions of software for one particular functionality. Not all of the software versions are running in the system and only in case of fault detection, that specific version would be invoked to take over the failed software module. Figure 5.7 demonstrate the state transition in this approach. The methodology of the approach is similar to dynamic redundancy in hardware design. In power electronics control, fault-tolerant software

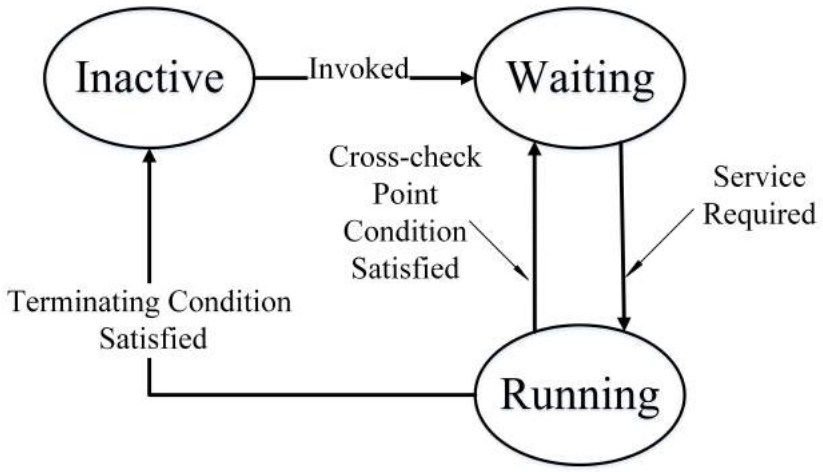


Figure 5.7: State Transition for Dynamic N-version Programming

techniques may be used in different sections. Grid synchronization in power system is one of the categories that requires fault-tolerancy. There are different methods to extract phase value from the grid voltages [43] and each one has its own benefits.

Figure 5.8 presents N-version programming implementation in grid synchronization. Out of available methods, zero crossing (ZC) method and phase locked loop (PLL) grid synchronization methods have been chosen to extract the phase value of the grid voltages. By default, ZC method is being used to for synchronization due to its fast settling time. In this method, zero crossing points of the measured voltages will be measured and according to that, frequency of the grid will be calculated. Those points also identify the reference phases of the grid (i.e. 60, 120,...). This method may easily get affected by unusual variations in the measured voltage and may give wrong output result.

In case of failure in this method (variation of frequency more that allowed threshold) due to harmonics or high frequency noise in the measured voltages, controller will switch to the PLL synchronization method. PLL method requires more processing power but it is more immune to harmonics and high frequency noises in the system. Controller continue using PLL method for

synchronization and it may goes back to the ZC method when the failure has been removed.

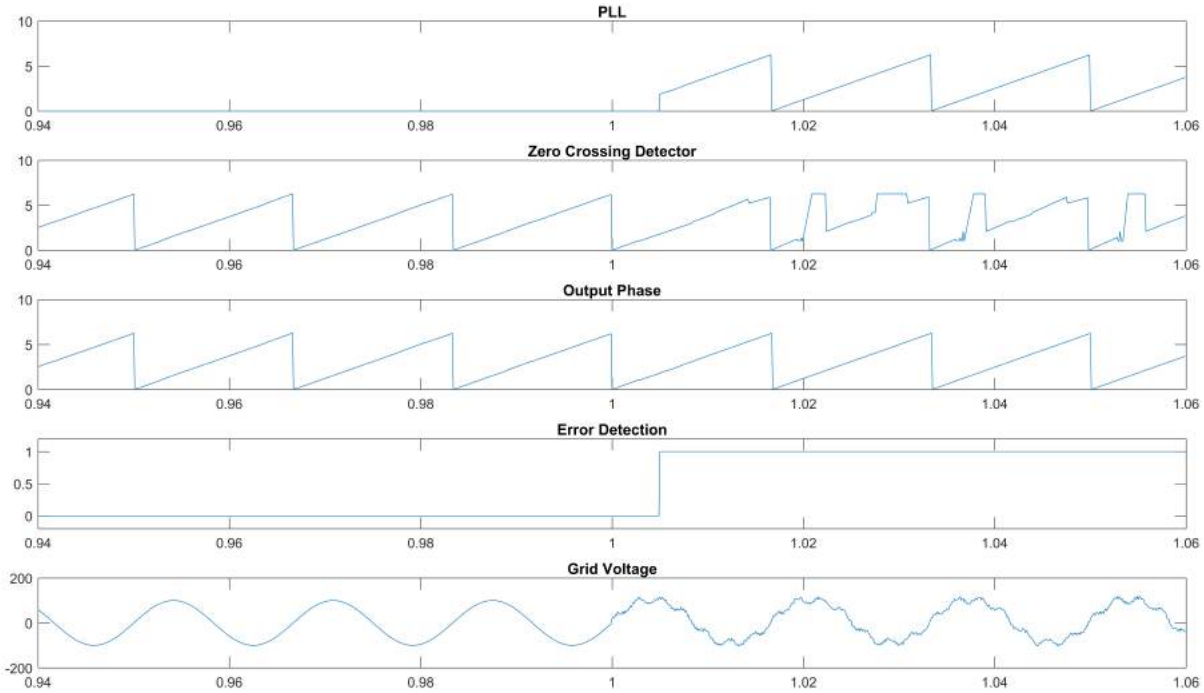


Figure 5.8: N-version Programming Implementation in Grid Synchronization

5.4.2 Recovery Blocks

Design of recovery blocks is based on **acceptance tests**. These tests may include checking for run-time errors (e.g. divide by zero), reason-ability, execution time and mathematical errors. Test for each parameter has its procedure, for example mathematical errors can be checked by inverting the mathematical formula and check if the same result is obtained. Systems using recovery blocks should duplicate the software into primary and secondary blocks. In the first module, the main code exist and at the end, its output would be tested for correctness. If the result is correct, the program would be ended; otherwise the second module (backup module) would be run. The final software code should looks like the following:

primary module

acceptance test

secondary module

acceptance test

Running the first module might corrupt the initial data memory, state of system and make it unusable for running the second module. Therefore, it is better to backup the data at the start of the first module and in the case of failure, recover the initial data. Therefore the final software style would be as following:

backup the data memory

primary module

acceptance test

recover the initial state

secondary module

acceptance test

5.5 Consensus on Selection of the Master Controller in Second Generation of Fault-tolerant Controller

In the second generation fault-tolerant controller, there are several controllers which can synchronize other slave controllers (vice-masters). Only one of these vice-master controllers may handle the synchronization task and other controllers should function as slave. The process of electing the best vice-master as the master controller is based on the transaction between vice-master controllers that will lead to general agreement on the master controller [79, 93, 7]. The process is happening repeatedly to ensure that the master controller is functioning correctly and in case of failure, it would be replaced by another controller in the least amount of time.

In the consensus process, several assumptions must be made before reaching the final agreement. All of controllers either respond to a message before timeout or never give respond to it (packets that arrive after timeout period are considered as dropped). Any modification of messages by the controllers is not acceptable and will change the nature of the problem. Therefore, problem is not cathegorized as a byzantine generals problem [55]. In other words, all of the faults in the system are assumed symmetric (single-value) whose behavior is perceived identical by non-faulty controllers.

The second assumption is that the system is synchronous. In synchronous systems, there are finite bounds on processing and communication delays between non-faulty controllers. These bounds are known to the controllers and they can make decision based on that. In asynchronous systems, these bounds are not known, therefore it is not possible to design a deterministic consensus protocol in an asynchronous system.

The goal is to form a voting algorithm which reaches the final agreement in a single step. The decision of choosing the master out of the vice-masters is based on the functionality and health of the controller. The chosen master controller would have the highest ranking among other

controllers.

The first step of decision making starts with broadcasting a set of mutually-measured variables by each vice-master controller to other vice-master controllers which has been acquired by measuring physical variables or as a result of dedicated algorithms. Therefore, each vice-master will have the data set from other vice-master controllers. After categorization of received variables in the same data set and forming a multiset ($\mathbf{V}$) of all variables, we would have:

$$\begin{aligned} V &= \{V_1, V_2, ..., V_i\} \\ V_j &= \langle v_1, ..., v_k \rangle \quad \forall j \in \{1, ..., i\} \\ v_k &\leq v_{k+1} \quad \forall k \in \{1, ..., k\} \end{aligned} \tag{5.6}$$

There are two properties of multiset for reaching the final agreement. Its *range* $\rho(V)$ and its *diameter* $\delta(V)$ which is defined as following:

$$\begin{aligned} &\rho(V_i) = [v_1 \quad v_k] : \textit{real interval spanned by } V_i \\ &\delta(V_i) = (v_k - v_1) : \textit{arithematic difference between maximum and minimum} \end{aligned} \tag{5.7}$$

The goal of proposed agreement algorithm is to achieve convergence in a single step. The voting algorithm $F(V)$ is single-step convergence if two convergence conditions are met:

- Validity: For each non-faulty process i, the voted value is within the range of correct values generated by functioning controllers, i.e. $F(V_i) \in \rho(U_{all})$
- Convergence: For each pair of non-faulty process, i and j, the difference between their voted value is smaller than the diameter of the correct values received. i.e. $|F(V_i) - F(V_j)| \leq C\delta(U_{all})$ where $0 \leq C < 1$.

Parameter C is the *convergence rate* which tells the performance of the voting algorithm. Smaller C means faster convergence rate for the voting algorithm.

Based on the V_i that has been gathered, it is possible to get the estimated correct variables (U_i) for each set. There are different voting algorithms based on statistical methods to find the best value for each set. The method which has been used is median-subsequent-reduce (MSR) and it calculates the median of a selected sub-sequence of the set. Therefore we have:

$$\begin{aligned} F(V_i) &= median[Sel_\sigma(Red^T(V_i))] \\ Red^T(V_i) &= \left\langle v_{(1+\tau)}, ..., v_{(V-\tau)} \right\rangle \end{aligned} \tag{5.8}$$

In the equation 5.8, $Red^T(V)$ omits the τ smallest and τ largest elements from the multiset. The selection function Sel_σ applies a subsequent function to select a submultiset of σ from the reduced multiset. The final voted value is the median of the selected multiset.

The next stage is to rank all the vice-master controllers based on the data they have provided and their specification. The ranking process is happening inside each vice-master and the chosen master controller will be announced to all vice-masters. For the i'th vice-master controller, $W_{i,j}$ is the rank that it has given to the j'th vice-master controller as it has given below:

$$\begin{aligned} W_j &= \sum_{1}^{i} C_i . |F(V_i) - v_i| + D_j \\ M &= \{ j | W_j \; is \; minimum \} \end{aligned} \tag{5.9}$$

The ranking is based on the difference between the value vice-controllers have provided and the voted value. The higher the difference, the less chance it would be chosen. Other factors like position of the vice-master in the controller array also make difference in the ranking (D_j).

The chosen master controller by each vice-master controller (M_i) will be broadcast to other

vice-master controllers. Therefore, each controller knows which vice-master is the final master controller. The chosen master will perform synchronization process and controllers that have chosen a vice-master other than the selected master controller will be bypassed by other controllers.

Consensus algorithm has been presented in pseudo-code 5. This agreement algorithm is happening continuously to make sure master controller is functioning correctly. If there is any error, the faulty master controller would be bypassed and another vice-master would become the master controller.

This algorithm has been simulated and the result is demonstrated in figure 5.9 and 5.10. In figure 5.9, horizontal delay is 3, vertical delay is 2 and the maximum permitted delay is 20 time units. In the proposed algorithm, each vice-master reads a series of global data and broadcast its reading. The voting algorithm finds the estimated value for the reading and based on the difference, it selects the best master controller (controller 1). The proposed master controller is broadcasted to other controllers, so they can know which controller is giving the right decision (if the decision is different than the majority, they would be isolated). In this test, delays is less than the maximum permitted and controllers are functioning correctly, the outcome of every controller is the same.

In the next test (figure 5.10), horizontal delay is 6, vertical delay is 5 and the maximum permitted delay is 20 time units. Due to the transmission delay between controllers, some controller (C0, C8, C10 and C11) can't make the right decision. These controllers would be bypassed by the neighboring controllers to isolate any possible failure in the system.

Result: Agreement on the master controller in the second generation fault-tolerant controller

Acquire-global-variables();

Broadcast-acquired-global-variables();

Wait-until-deadline();

for *1 to i* **do**

$F(V_i) = median[Sel_\sigma(Red^T(V_i))]$;

end

$W_M = \infty$;

for *1 to j* **do**

for *1 to i* **do**

$W_j = C_i.|F(V_i) - v_i|$;

end

$W_j = W_j + D_j$;

if $W_j < W_M$ **then**

$M = j$;

$W_M = W_j$;

end

Broadcast-selected-master-controller(j);

Gather-other-selected-masters();

Bypass-controllers-with-different-selected-master-controller();

Algorithm 5: Pseudo-code of Consensus Procedure on Selection of Master Controller for the Second Generation Fault-tolerant Controller

	C 0	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9	C 10	C 11	Master Selected	
C 0	-	3	6	9	2	5	8	11	4	7	10	13	1	
C 1	3	-	3	6	9	2	5	8	11	4	7	10	1	
C 2	6	3	-	3	6	9	2	5	8	11	4	7	1	
C 3	9	6	3	-	3	6	9	2	5	8	11	4	1	
C 4	2	9	6	3	-	3	6	9	2	5	8	11	1	
C 5	5	2	9	6	3	-	3	6	9	2	5	8	1	
C 6	8	5	2	9	6	3	-	3	6	9	2	5	1	
C 7	11	8	5	2	9	6	3	-	3	6	9	2	1	
C 8	4	11	8	5	2	9	6	3	-	3	6	9	1	
C 9	7	4	11	8	5	2	9	6	3	-	3	6	1	
C 10	10	7	4	11	8	5	2	9	6	3	-	3	1	
C 11	13	10	7	4	11	8	5	2	9	6	3	-	1	
Final Agreed Master	1													

Figure 5.9: Simulation Result for General Agreement on the Master Controller (4 module per phase, horizontal delay of 3, vertical delay of 2, maximum permitted delay of 20)

	C 0	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9	C 10	C 11	Master Selected	
C 0	-	6	12	18	5	11	17	TimeOut	10	16	TimeOut	TimeOut	9	
C 1	6	-	6	12	18	5	11	17	TimeOut	10	16	TimeOut	1	
C 2	12	6	-	6	12	18	5	11	17	TimeOut	10	16	1	
C 3	18	12	6	-	6	12	18	5	11	17	TimeOut	10	1	
C 4	5	18	12	6	-	6	12	18	5	11	17	TimeOut	1	
C 5	11	5	18	12	6	-	6	12	18	5	11	17	1	
C 6	17	11	5	18	12	6	-	6	12	18	5	11	1	
C 7	TimeOut	17	11	5	18	12	6	-	6	12	18	5	1	
C 8	10	TimeOut	17	11	5	18	12	6	-	6	12	18	5	
C 9	16	10	TimeOut	17	11	5	18	12	6	-	6	12	1	
C 10	TimeOut	16	10	TimeOut	17	11	5	18	12	6	-	6	8	
C 11	TimeOut	TimeOut	16	10	TimeOut	17	11	5	18	12	6	-	9	
Final Agreed Master	1													

Figure 5.10: Simulation Result for General Agreement on the Master Controller (4 module per phase, horizontal delay of 6, vertical delay of 5, maximum permitted delay of 20)

Chapter 6

Implementation and Experimental Verification of Proposed Fault-tolerant Controller for Modular Multi-level Converters

This chapter will represent the experimental result of the implemented hardware for verifying the feasibility of the proposed controller. Implementation of the controller system consist of two main stages. In the first stage, the functionality and correctness of the control software will be assessed. In order to do so, the controller will interact with an emulator through digital and analog I/O. The emulation and interaction between controller and the emulator may be in real-time (hardware in the loop) or in offline mode (processor in the loop). HIL emulation requires advanced hardware but it gives output result faster, therefore it is the test method which has been used for the controller firmware evaluation. The final test has been done on real hardware setup in which scaled down model of the power converter is tested.

6.1 Implementation of Proposed Methods on the Fault-tolerant Controller Test-bed

A hardware test-bed has been designed (Figure 6.1) to demonstrate and test the behavior of the proposed controller in connection with converter hardware or the hardware in the loop (HIL) simulation. The test-bed consist of 13 texas instrument F28379D controller card (12 slave controllers and one master controller). The output signals from controllers are fed into three separate FPGA cards (ACM-204-40C8). These FPGAs help implementing the fault-detection, fail-over, output comparison and other necessary circuits for the system. The analog signals from external resources are leveled to match the range of the micro controllers. More detail about the internal circuits may be found in the third appendix. Figure 6.2 demonstrates the block

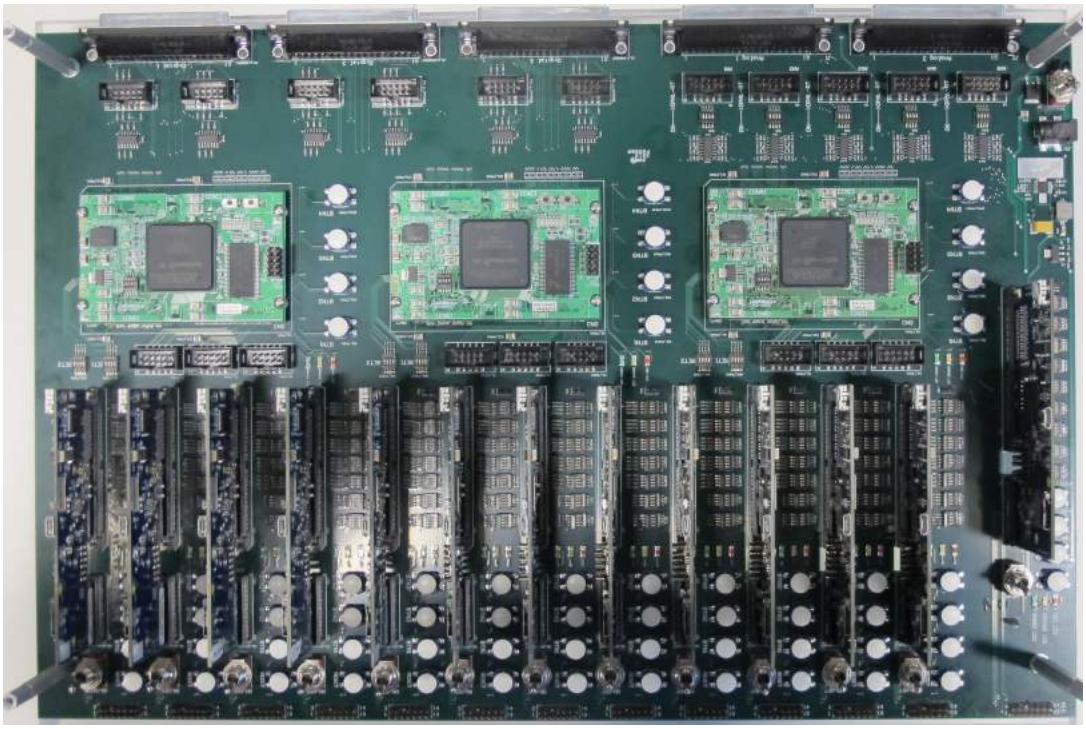


Figure 6.1: Fault-tolerant Controller Test-bed

diagram of the test-bed. Fault handling circuits have been implemented in the FPGAs. Incoming analog signals are leveled to match controller specification and have been shared between all controllers. It is possible to select the associated analog inputs based on the architecture. The

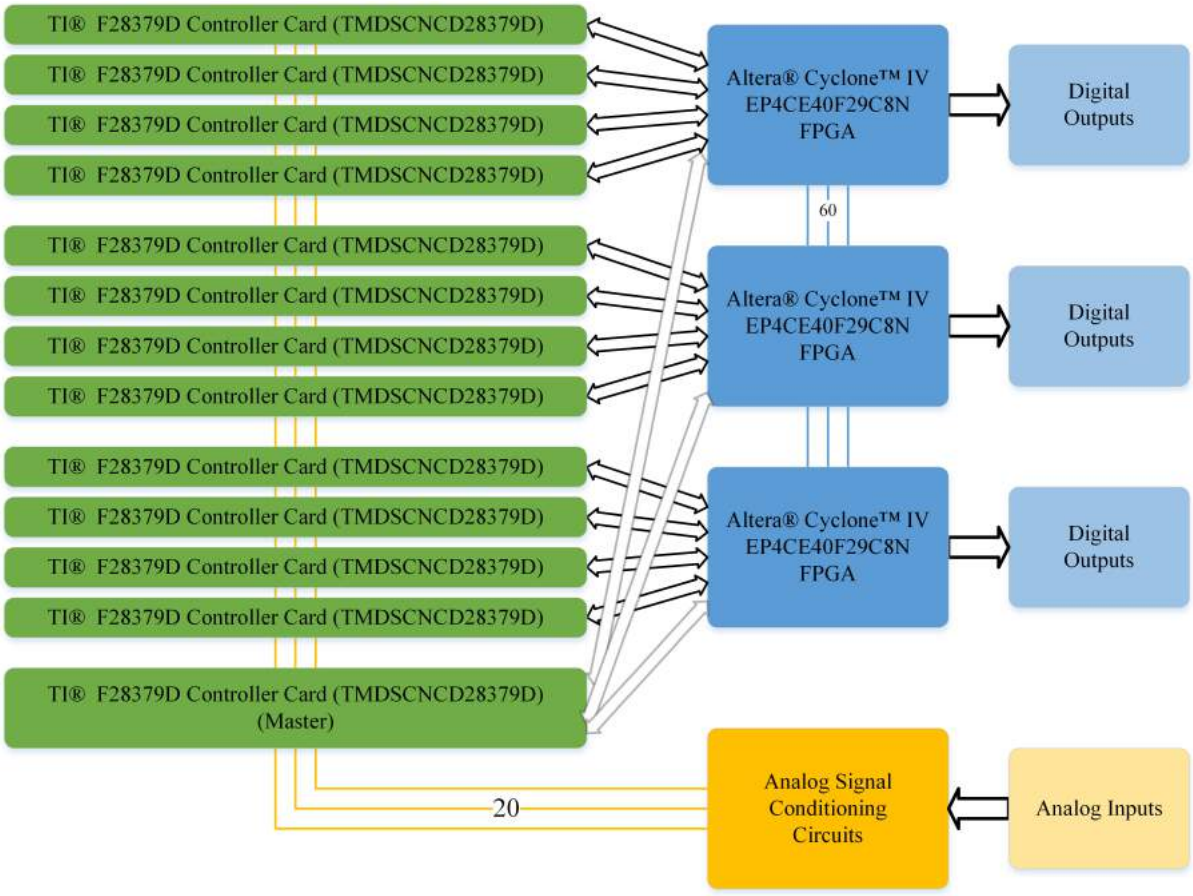


Figure 6.2: Block Diagram of Fault-tolerant Controller Test-bed

final architecture of the fault-tolerant controller (after programming and configuring the FPGA) has been demonstrated in figure 6.3. In each phase, four controllers are connected to the master controller through the shared communication bus. Master controller will run the supervisory control (high level control) and update the slave controller at each control step. Each slave

controller does the control procedures for itself and the neighbor controllers. The output result of the slave controllers are compared in the fault detector block, which detects errors if outputs are not similar for the same module. Fault detection signal, external error from micro controller (for signaling communication error) and the health status of the controller are given to the fail over circuit. Fail over circuit decides which controller should control each converter block based on the functionality status of the main controller and the adjacent controllers (discussed in detail in chapter 3). As it can be seen in figure 6.3, all of the blocks which are related to fault handling for module 2 have been high lighted. All of the fault injection results in the following sections (both CHB and MMC configuration) are for module 2 and the nomenclature follows signals in the picture.

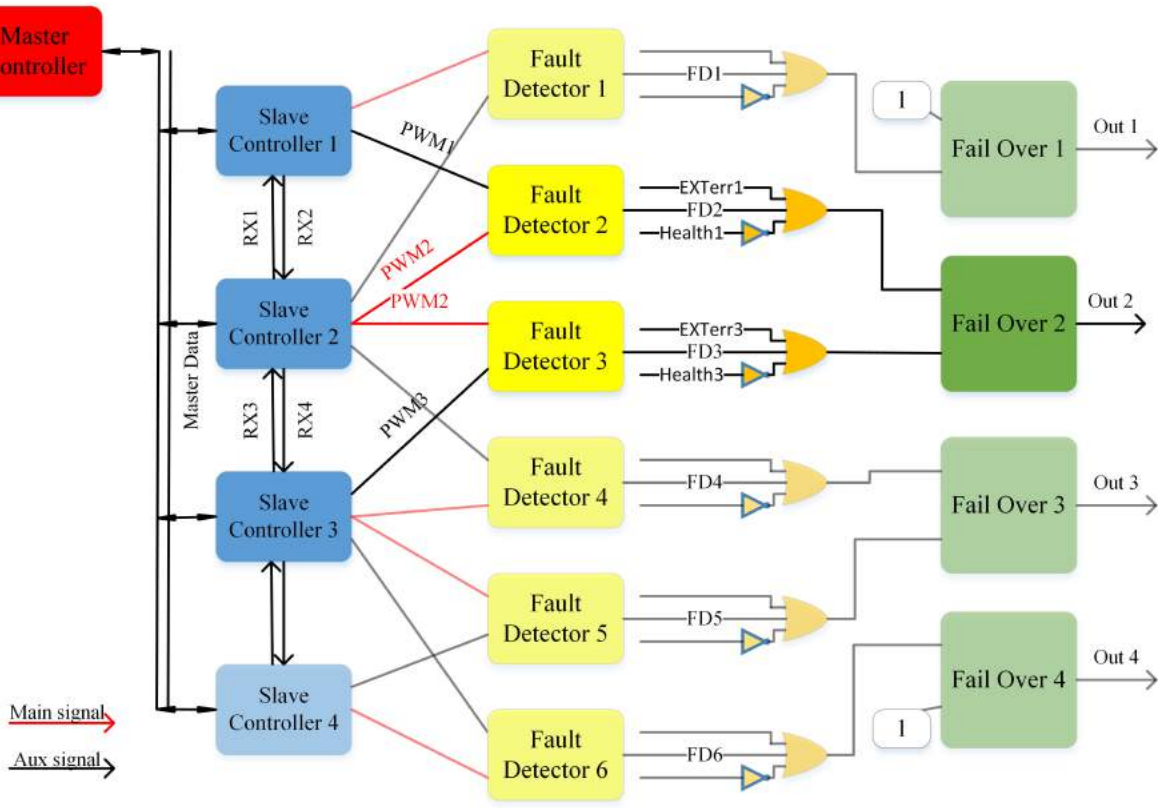


Figure 6.3: Fault Handling for each Phase (4 module per phase) of Converter

6.2 Theory of Operation for Hardware in the Loop (HIL) Simulation

In most power electronics circuits, the exact power semiconductor model is required for few cycles in which transient values are changing or exact calculation (e.g. loss) is required. After this period, there is no need for exact model of the switch and simplified models may solve the problem. In HIL emulation, it is not possible to use the exact model and run the model in real-time, therefore the semiconductor switch (which is a non-linear component) must be simplified[77].

There are different ways of simplifying the semiconductor switch to get the best result. By assuming ideal model (a switch is either short circuit or open circuit) for n switch in a circuit, it is possible to have 2^n different states. Each state of the circuit can be represented by state-space equations and it can be solved to find the new values in each simulation step. Drawbacks of this method is the efforts needed for extraction of the state-space model and possibility of state-variable discontinuity and/or changes in the number of states at switching transitions[77, 68, 65, 21].

Another method is putting numerical integration to element equation rather than network

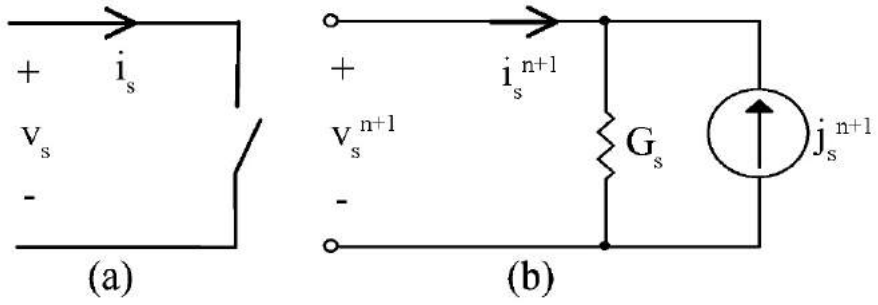


Figure 6.4: Simplified Model of Switch (a)Ideal Switch (b)Discrete-time Switch Model

state-space equations. This method is being used in general simulators like Spice. A small R_{on} is used when transistor is on and a large R_{off} represent the switch when it is turned off. The chosen value for resistors must be in range that convergence is possible. Even in this method, simulation times might be long since more iterative step must be taken for solving the equations specially in the switch transients.

Figure 6.4(a) demonstrate two simplified model for the power switch. In the ideal switch model, the switch is either turn off or on. Therefore we have:

$$\begin{aligned} s = 0 \Leftrightarrow off \Leftrightarrow i_s = 0 \\ s = 1 \Leftrightarrow on \Leftrightarrow v_s = 0 \end{aligned} \tag{6.1}$$

The ideal model is not the best model for real-time simulation. The other switch model that is suitable for discrete-time simulation is represented in figure 6.4(b). In common simulators like SPICE, this model represent a switch with low R_{on} when it is on, large R_{off} when it is off and smooth transition rate from each state to another one. It is desired that G_s remains constant and j_s varies, therefore only one matrix inversion is necessary for the entire simulation. This model allows fixed nodal admittance matrix during the whole simulation, otherwise one matrix for each state of switches was necessary. Based on this and assuming that $T = t^{n+1} - t^n$, the switch model equations are:

$$j_s^{n+1} = \begin{cases} -i_s^n & if \quad s^{n+1} = 1 \\ G_s v_s^n & if \quad s^{n+1} = 0 \end{cases} \tag{6.2}$$

Therefore introduced errors in this method (voltage when switch is on and current when it is off) are:

$$\begin{cases} v_s^{n+1}(s=1) = \frac{1}{G_s}(i_s^{n+1} - i_s^n) \\ i_s^{n+1}(s=0) = G_s(v_s^{n+1} - v_s^n) \end{cases} \tag{6.3}$$

The proposed mathematical model is not memory-less and it depends on values of the previous n simulation steps. The representation of a capacitor using backward Euler algorithm is as below:

$$\begin{aligned} v^{n+1} &= v_c^n + \frac{1}{C_s}\int_{t^n}^{t^{n+1}} i_c(t)dt \\ v^{n+1} &\approx v_c^n + \frac{T}{C_s} i_c^{n+1} \\ i_c^{n+1} &\approx \frac{C_s}{T} v_c^{n+1} - \frac{C_s}{T} v_c^n = G_c v_c^{n+1} - j_c^{n+1} \end{aligned} \tag{6.4}$$

In the same way, representation of an inductor may be achieved:

$$i_l^{n+1} \approx \frac{T}{L_s} v_l^{n+1} + i_l^n = G_l v_l^{n+1} - j_l^{n+1} \tag{6.5}$$

By comparing equation 6.2 with equation 6.4 and 6.5, it can be realized that the switch model is discrete-time model of capacitor (or inductor). Moreover, it implies that switch is inductor L_s when on and capacitance C_s when off (figure 6.5). In order to keep the conductance G_s constant, we should have:

$$G_s = G_c = G_l = \frac{C_s}{T} = \frac{T}{L_s} \tag{6.6}$$

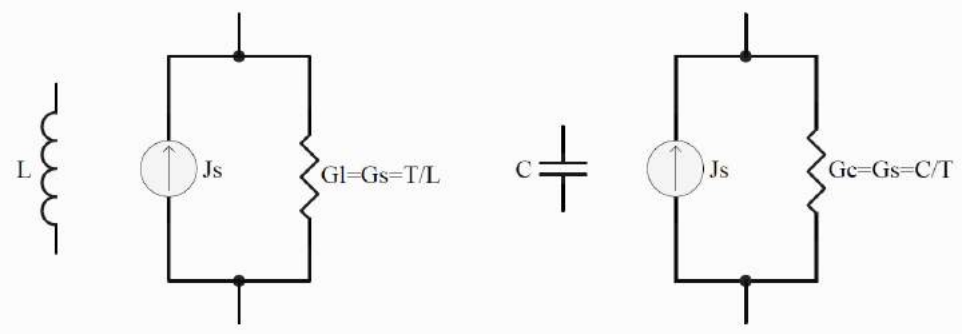


Figure 6.5: Representation of the Pejovic Switch in On (inductor) and Off (capacitor) Mode

6.3 Hardware in the Loop Simulation for the Proposed Fault-tolerant Controller

The first stage of system test and development involves connecting the controller to the hardware in the loop (HIL) simulator (figure 6.6). This type of simulation helps speeding up the simulation with highest accuracy and no possibility of component loss. The first step is to draw the converter

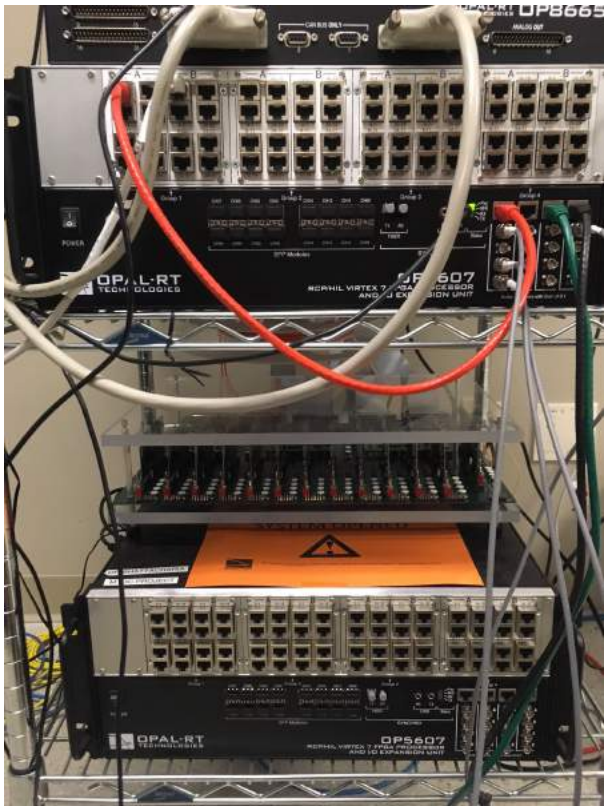


Figure 6.6: Fault-tolerant Controller in Connection with Opal-rt System (front view)

circuit and extract the netlist using the circuit compiler. The netlist defines the circuit which must be simulated in real-time using FPGA simulator. After loading the netlist on the simulator,

it will run in a loop and will update the outputs based on the input signals and the current state of the system. Therefore, controller must interact with the simulator in which it acts as the converter circuit emulator (figure 6.7).

In the following sections, cascaded H-bridge converter and modular multi-level converter have

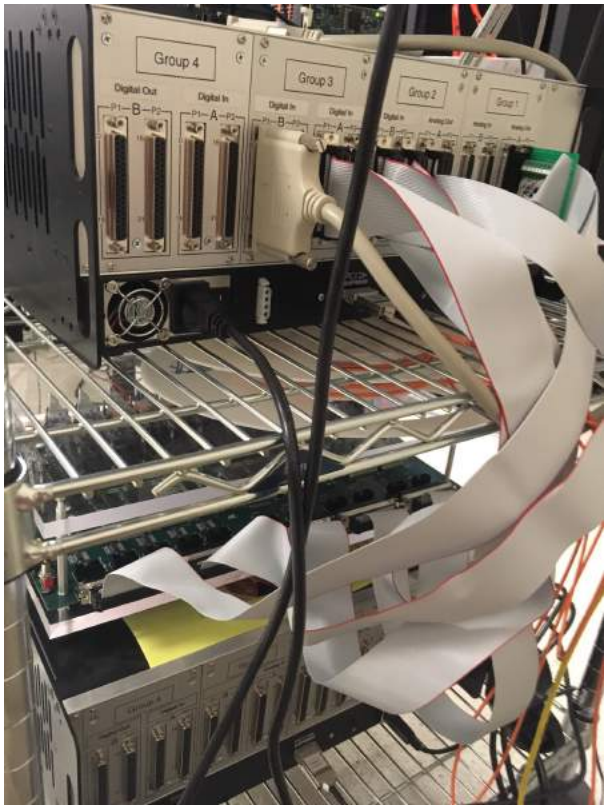


Figure 6.7: Fault-tolerant Controller in Connection with Opal-rt System (back view)

been emulated on the HIL simulator, different failure strategy have been tested and the output result are acquired.

6.4 Hardware in the Loop Simulation Result for Cascaded H-bridge Converter Using Fault-tolerant Controller

Principle of operation for cascaded H-bridge converter (CHB) has been represented in appendix A. The same converter has been modeled in Opal-rt HIL simulator to verify the feasibility of the proposed fault-tolerant converter (figure 6.8). The converter system consist of 4 module per

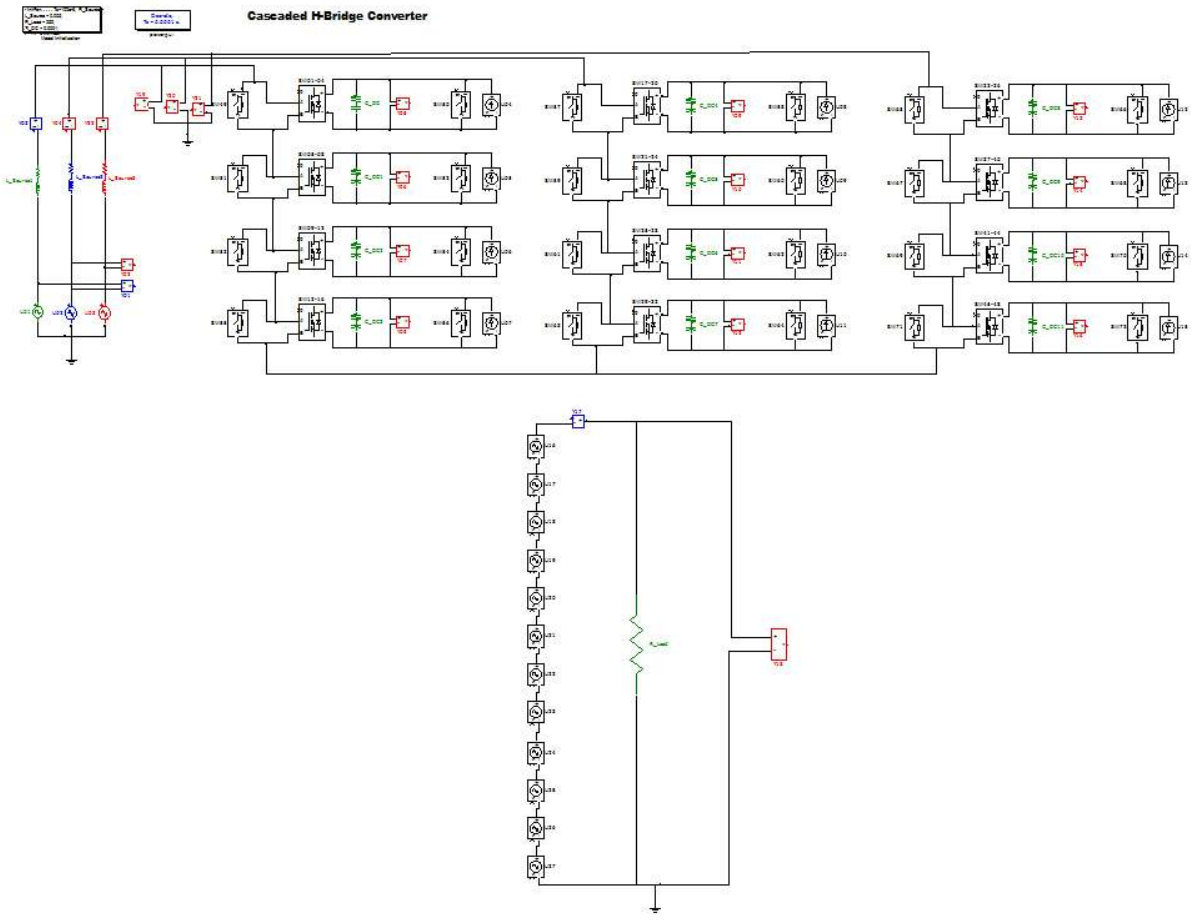


Figure 6.8: Implemented Cascaded H-bridge Converter with Isolated DC/DC Output Converter in Opal-rt HIL simulator

Table 6.1: Setup Configuration for CHB HIL Simulation

Design Parameter	Symbol	Value
Grid line-line voltage	E	120 (Vrms)
Line frequency	f	60 (Hz)
Series inductance	L_{series}	5 (mH)
Switching frequency	$f_{switching}$	10 (kHz)
Converter module per phase	N	4
Nominal module capacitor voltage	V_{module}	35 (V)
Converter module capacitance	C_{module}	6800 (μF)
Nominal converter current	$I_{converter}$	10 (A)
Load Resistance	R_{load}	250 (Ω)

phase. Each module is a cascaded h-bridge converter in which the AC sides are in series and the DC sides are connected to dc capacitor. The capacitor is connected to a DC/DC converter which has been simplified by a voltage/current source to decrease the required processing resource. Detail of the converter setup has been presented in table 6.1. The converter regulates high voltage dc side at 400 (V), which means each module would have dc voltage around 35 (V). The load is a 250 ohm resistor which is connected in parallel with dc capacitor.

There are several measurement sensor in the converter setup which has been connected to analog outputs. Analog scaling is done in a way that the output voltage never goes into saturation region. The gating signal for the power switches comes directly from the digital inputs (which provided by controller). Different failure case has been tested on this converter setup and result is presented in the following sections.

6.4.1 Cascaded H-bridge Converter under Normal Operation

This section demonstrate the experimental result in normal operation of the converter system. Figure 6.9 shows the grid voltage (line to line) and the grid currents. The measured grid voltage is fed in the synchronous reference frame PLL and the grid phase is shown in figure 6.10. Figure 6.11 shows the capacitor voltages in phase A of the converter. The goal is to regulate all capacitor voltages at 35 (V). Figure 6.12 shows the active and reactive current from the grid. Since converter is operating in rectifier mode, active current (I_d) is negative. Converter is not compensating reactive power of the grid and no reactive current (I_q) is being injected to the grid. Figure 6.13 shows the reference PWM which has been generated. The reference signal contains third harmonics to increase the voltage range of the converter.

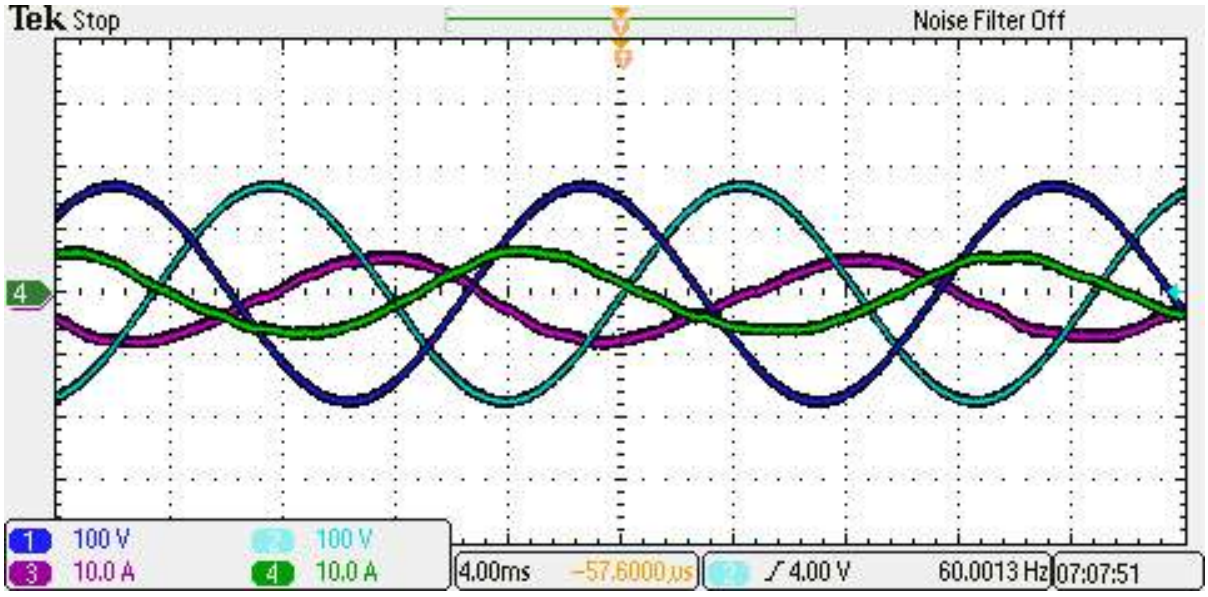


Figure 6.9: Grid Voltage and Current in Normal Operation of CHB (1:Vab 2:Vbc 3:Ia 4:Ib)

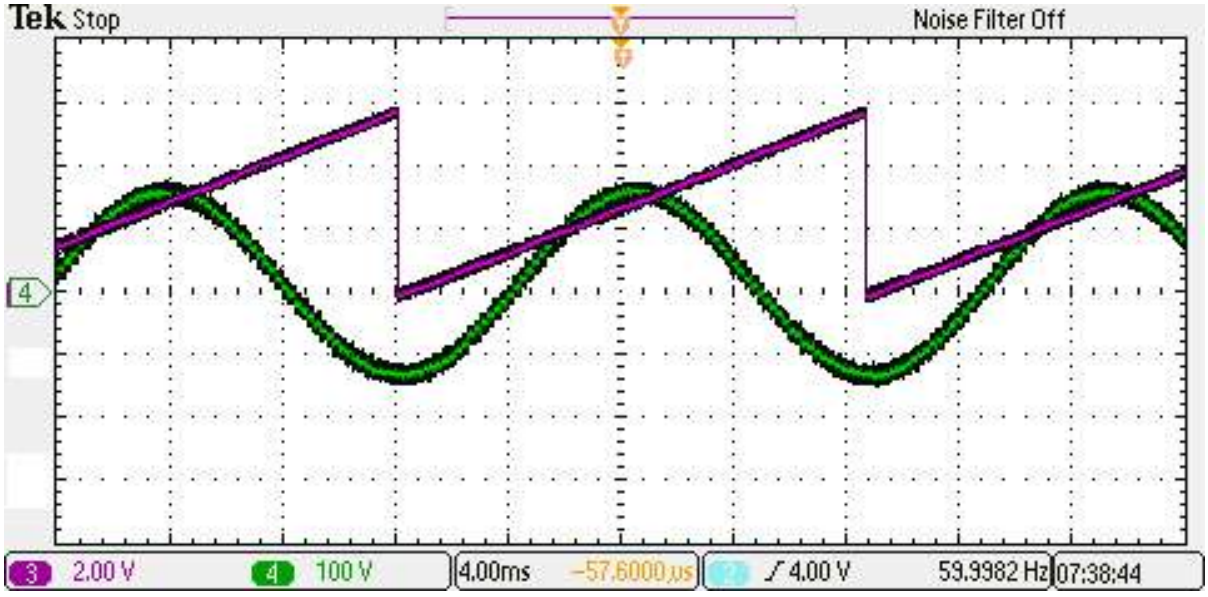


Figure 6.10: Synchronous Reference Frame PLL and Phase A Voltage (3:Phase(radian) 4:Phase A Voltage

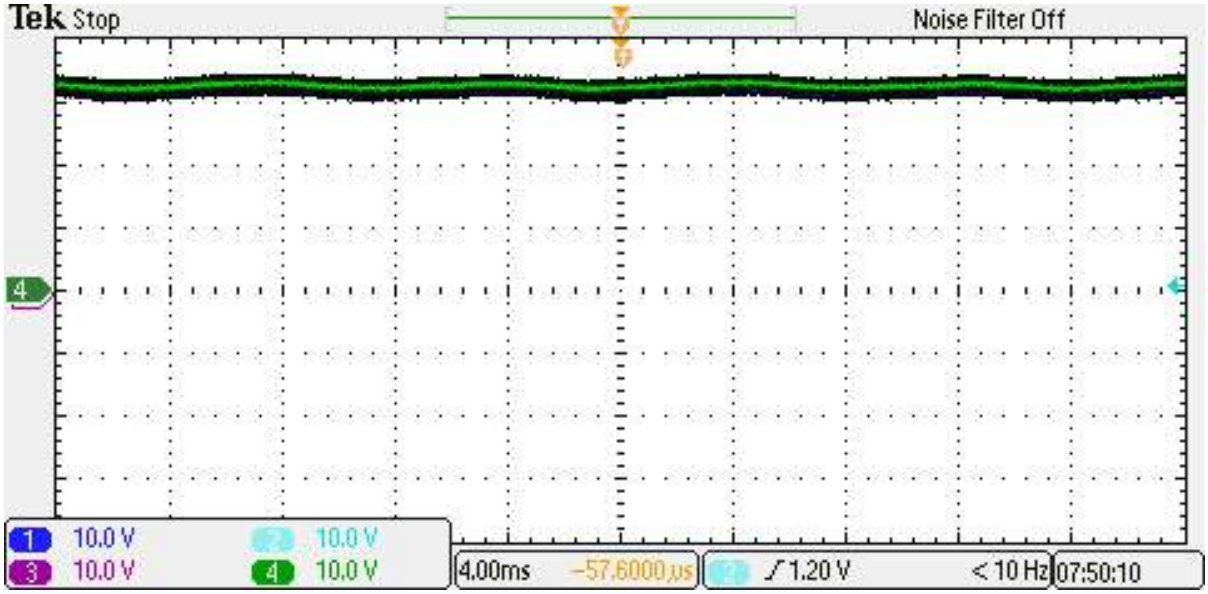


Figure 6.11: Capacitor Voltage of each Module in Normal Operation (1:Module 1 2:Module 2 3:Module 3 4:Module 4)

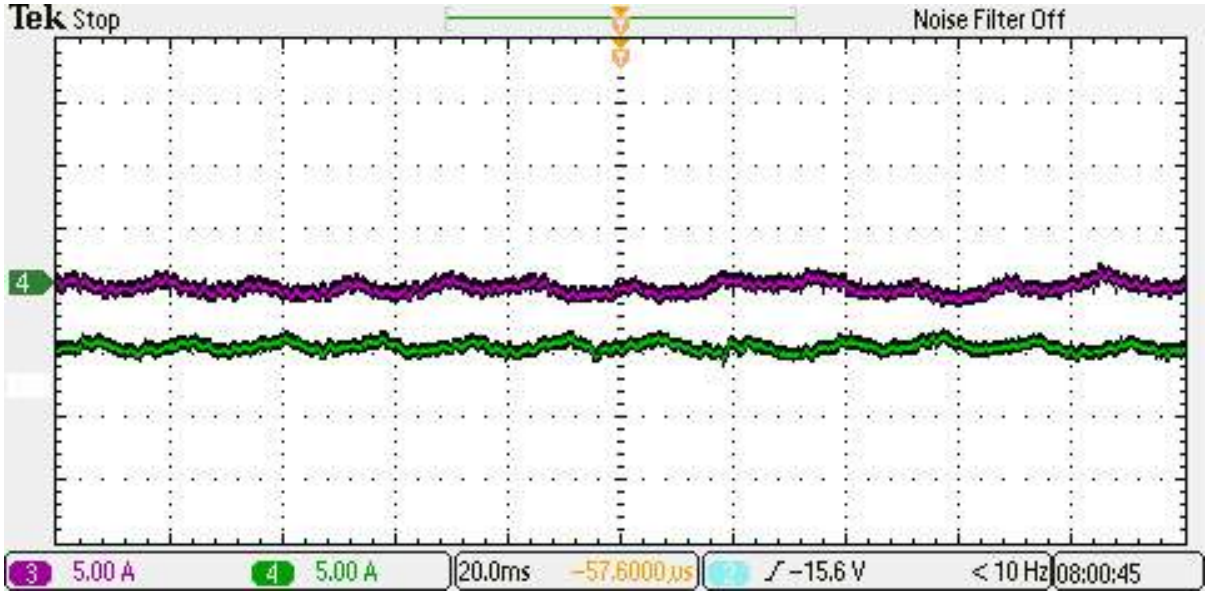


Figure 6.12: Active and Reactive Current in Normal Operation (3:Reactive Current 4:Active Current)

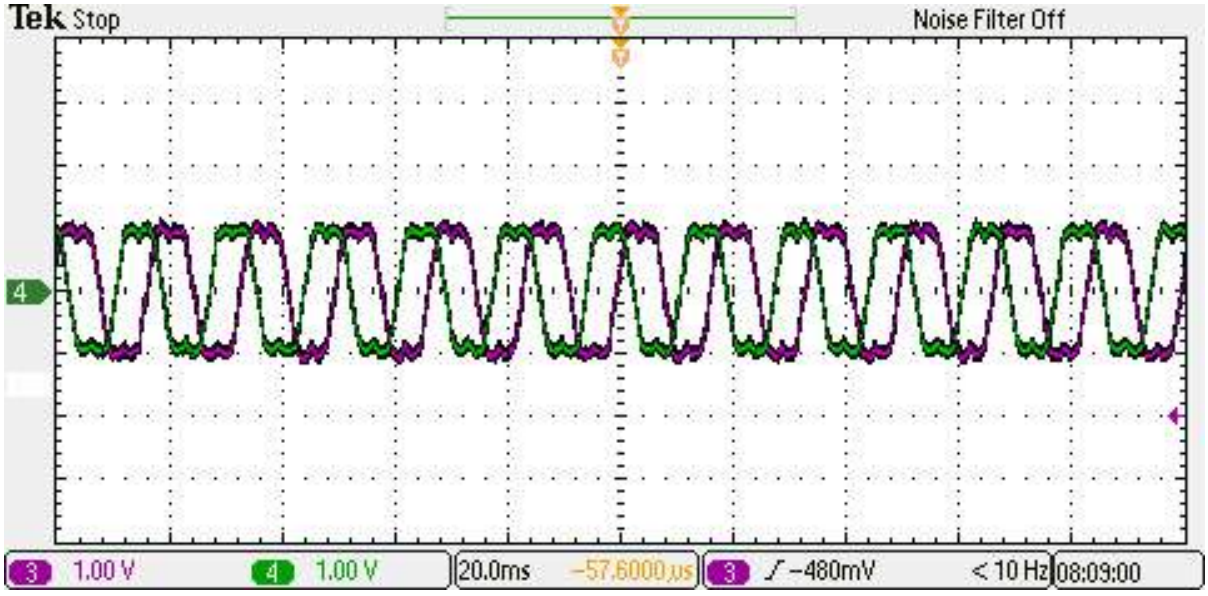


Figure 6.13: PWM Reference Signals for Phase A and B in Normal Operation (3:Phase A 4:Phase B)

6.4.2 Fault Injection Result In Controller Architecture for Cascaded H-bridge Converter

In this section, functionality of the controller circuit has been investigated under different failure modes. In each mode, different failure has been injected to the controller cards or the associated control circuitry for each controller to emulate single point of failure in the system. In all of the results, focus is on controller number 2 and it has been tried to show the experimental result related to this controller (i.e. effect of failure in adjacent controllers on this controller). For understanding definition of each signal, figure 6.3 may be used which has block diagram of the fault handling circuit and the related signal names. All of the results have been captured by oscilloscope or logic analyzer from the experimental test bed in connection with the converter system.

Figure 6.14 shows the case in which communication link from controller 1 to controller 2 has been failed. Therefore controller 2 may not receive any data and error flag for controller 1 will turn on. Since it is not common failure, controller 2 may not get affected at all.

Figure 6.15 shows the case in which communication link from controller 2 to other controllers has failed (this may happen as a result of failure in controller 2). Therefore controller 1 and 3 may not receive any data and error flag from controller 1 and 3 will turn on. Since it is common failure, controller 2 will be bypassed.

Figure 6.16 shows the case in which built in self-test circuit (which turns on when internal failure happens) in controller 1 has turned on and its effect of controller 2 has been demonstrated. This failure will turn on the fault detection for controller 1 (FD1) and turn off the health status for controller 1. Since it is not a common failure, controller 2 will continue its operation without interruption.

Figure 6.17 demonstrates the case in which built in self-test circuit (which turns on when

internal failure happens) in controller 2 has turned on and its effect of controller 2 has been shown. This failure will turn on the fault detection related to controller 2 (FD2 and FD3) and turn on external error indicators from controller 1 and 3. Since it is a common failure, controller 2 will be bypassed and controller 1 will take charge of controlling the second power module.

Figure 6.18 demonstrates the case in which power supply of the controller 1 has failed (by turning off the controller card). This failure will turn on the fault detection related to controller 1 (FD1 and FD2) and turn on external error indicators from controller 1. Since it is not a common failure, controller 2 will continue its operation without interruption.

Figure 6.19 demonstrates the case in which power supply of the controller 1 and 3 have failed simultaneously (by turning off the controller card). This failure will turn on the fault detection related to controller 1 and 3 (FD1 to FD4), turn on external error indicators from controller 1 and 3, and turn off health indicators related to controller 1 and 3. Since both health indicators of the failed controller have been turned off, controller 2 will continue its operation without interruption (i.e. controller 2 will switch its control circuitry when health indicators are still on).

Figure 6.20 shows the case in which power supply of the controller 2 has failed (by turning off the controller card). This failure will turn on the fault detection related to controller 2 (FD1 to FD4) and turn on external error indicators from controller 1 and 3. Since it is a common failure, controller 2 will be bypassed by the first controller.

Figure 6.21 shows the case in which controller card 1 has been restarted (by external reset pin). External restart will make it unavailable for a period of time and during this time, it may not function correctly. This will turn on the external error for controller 1 and clears the health indicator for this controller. Since it is not a common mode failure, controller 2 will continue its operation without interruption.

Figure 6.22 shows the case in which controller card 2 has been restarted (by external reset pin). External restart will make it unavailable for a period of time and during this time, it may

not function correctly. This will turn on the external error for controller 1 and 3 and clears the health indicator for these controllers too. Since it is a common mode failure, controller 2 will be bypass by the first controller.

Figure 6.23 demonstrates the case in which voltage measurement sensor in module 1 fails. This action has been emulated by a step change (40% decrease in the value) in the measured capacitor voltage. The change in the measured value will trigger the external error signal in module 1 but since it is not a common error, it will not bypass the controller module 2.

Figure 6.24 demonstrates the case in which voltage measurement sensor in module 3 fails while module 1 has already been failed. This action has been emulated by a step change (40% decrease in the value) in the measured capacitor voltage of module 3 while module 1 has been turned off. The change in the measured value will trigger the external error signal in module 1 and 3 but since it is a common error, it will bypass the controller module 2 to its adjacent controller (controller module 3).

Figure 6.25 demonstrates the case in which voltage measurement sensor in module 2 fails. This action has been emulated by a step change (40% decrease in the value) in the measured capacitor voltage of module 2. The change in the measured value will trigger the external error signal in module 1 and 3 but since it is a common error, it will bypass the controller module 2 to its adjacent controller (controller module 1).

Figure 6.26 shows the case in which power module has been failed (i.e. one of IGBTs has been failed) and it has been bypassed. Due to this, the capacitor voltage would reach zero and phase voltage must be compensated by other power modules. As it can be seen, phase voltage has been compensated by slight increase in other module's capacitor voltage.

Figure 6.27 shows the phase voltages in case of power module failure. Failure will cause fluctuation in phase voltages as well as the total capacitor voltage but it will settle down after some time.

Figure 6.28 shows grid currents and voltages in case of power module failure. Failure will cause unbalanced currents but it will be settle down after some time.

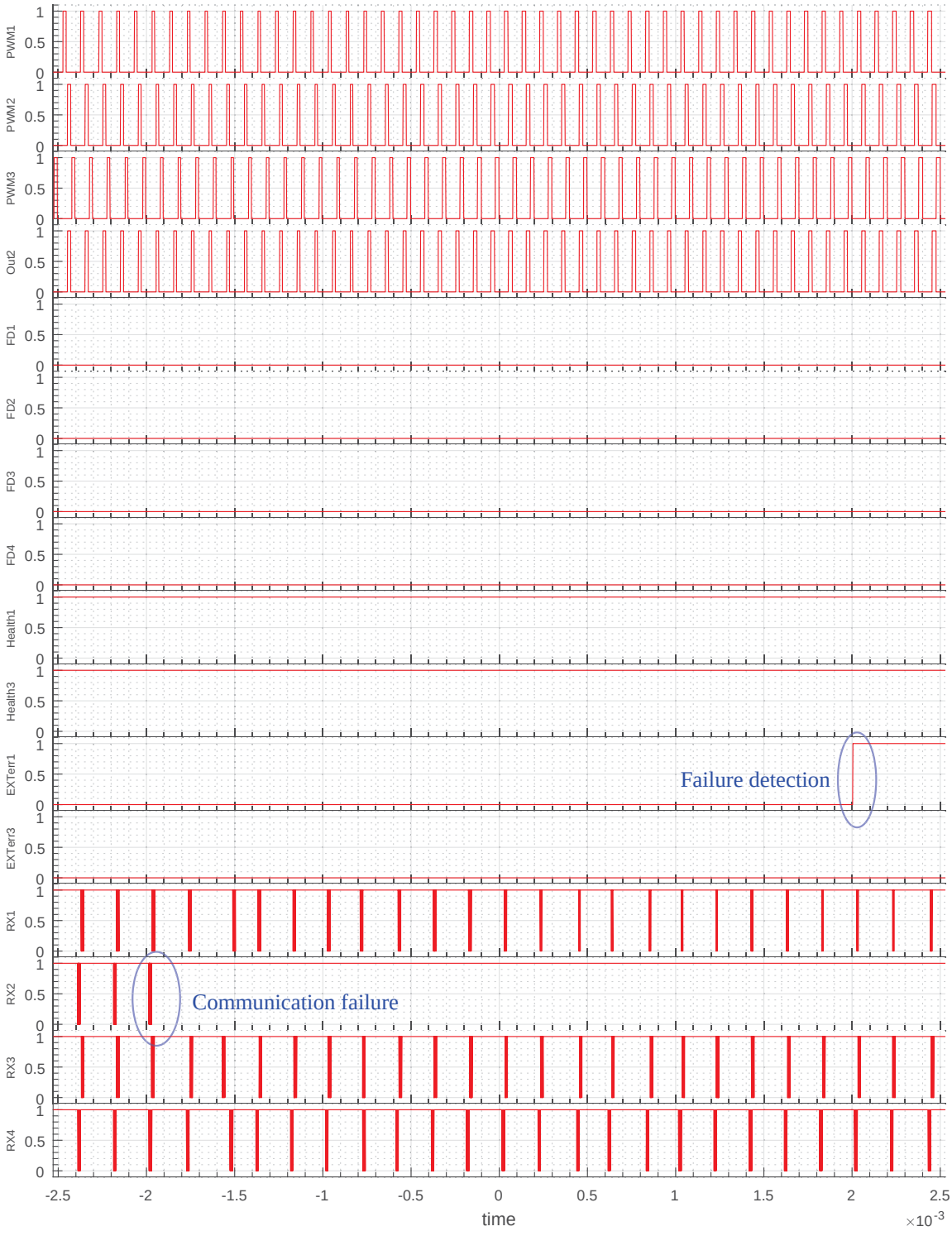


Figure 6.14: Communication Failure in Module 1 and Its Effect on Module 2 in CHB

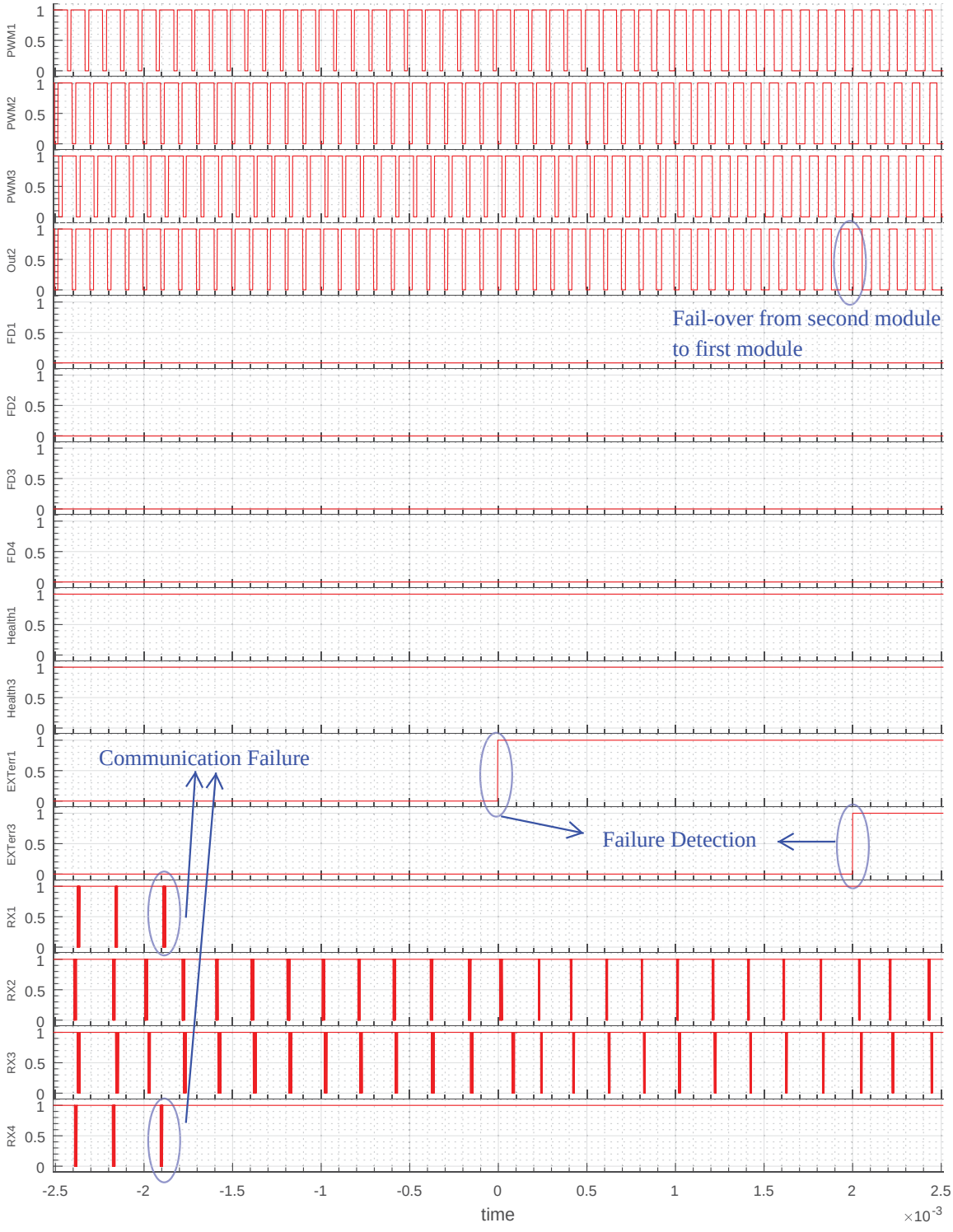


Figure 6.15: Communication Failure in Module 2 and Its Effect on Module 2 in CHB

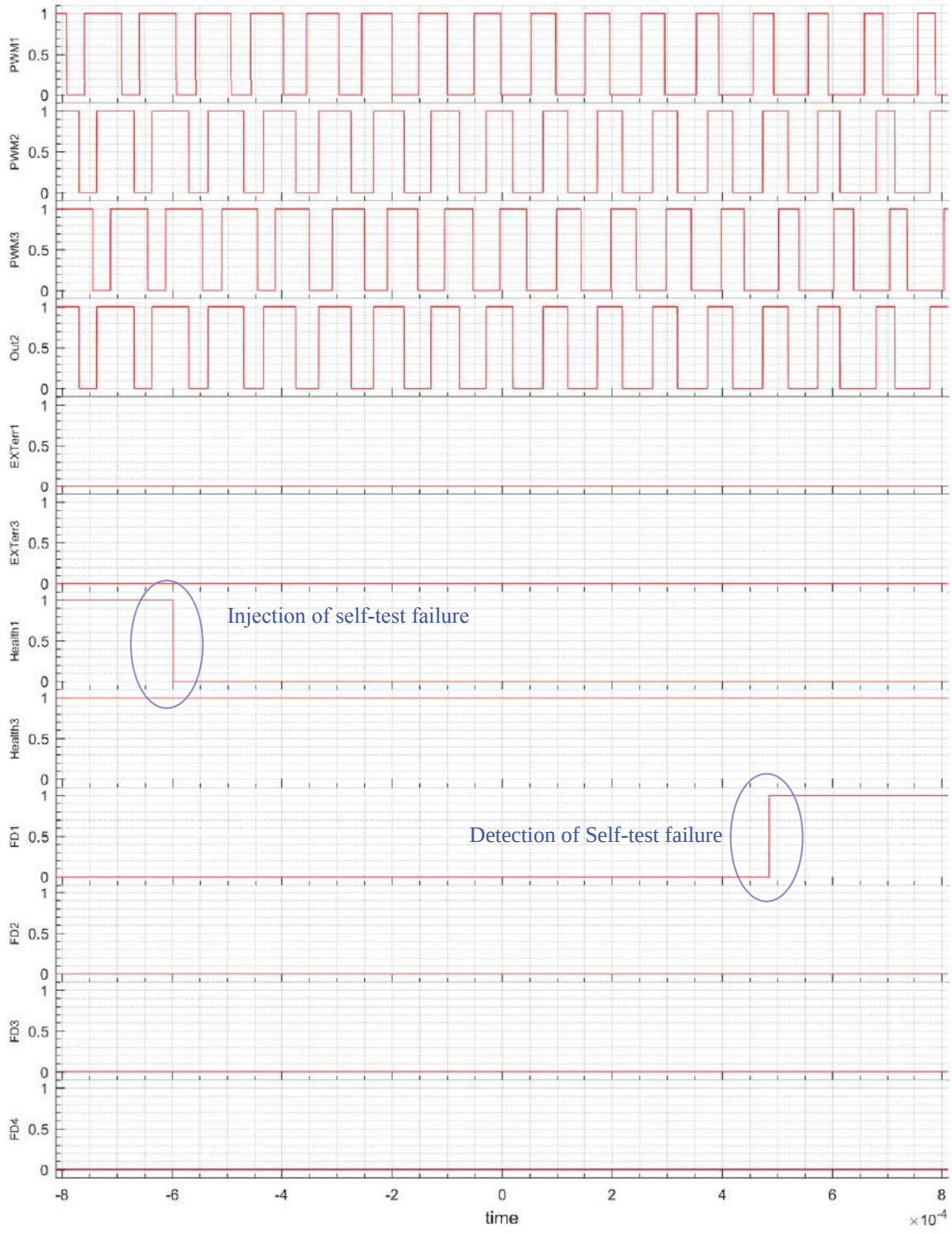


Figure 6.16: Built In Self-test (BIST) Failure in Module 1 and Its Effect on Module 2 in CHB

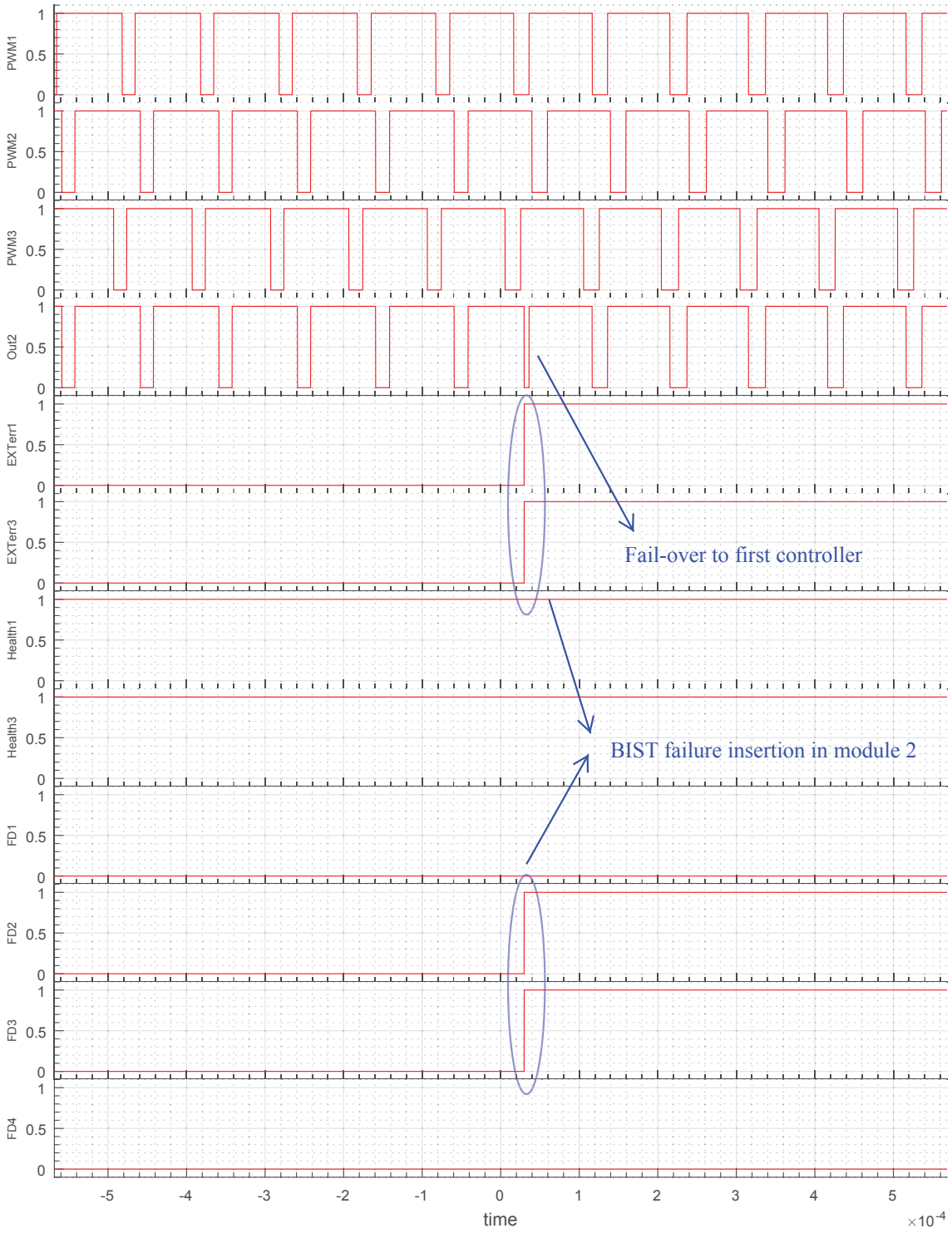


Figure 6.17: Built In Self-test (BIST) Failure in Module 2 and Its Effect on Module 2 in CHB

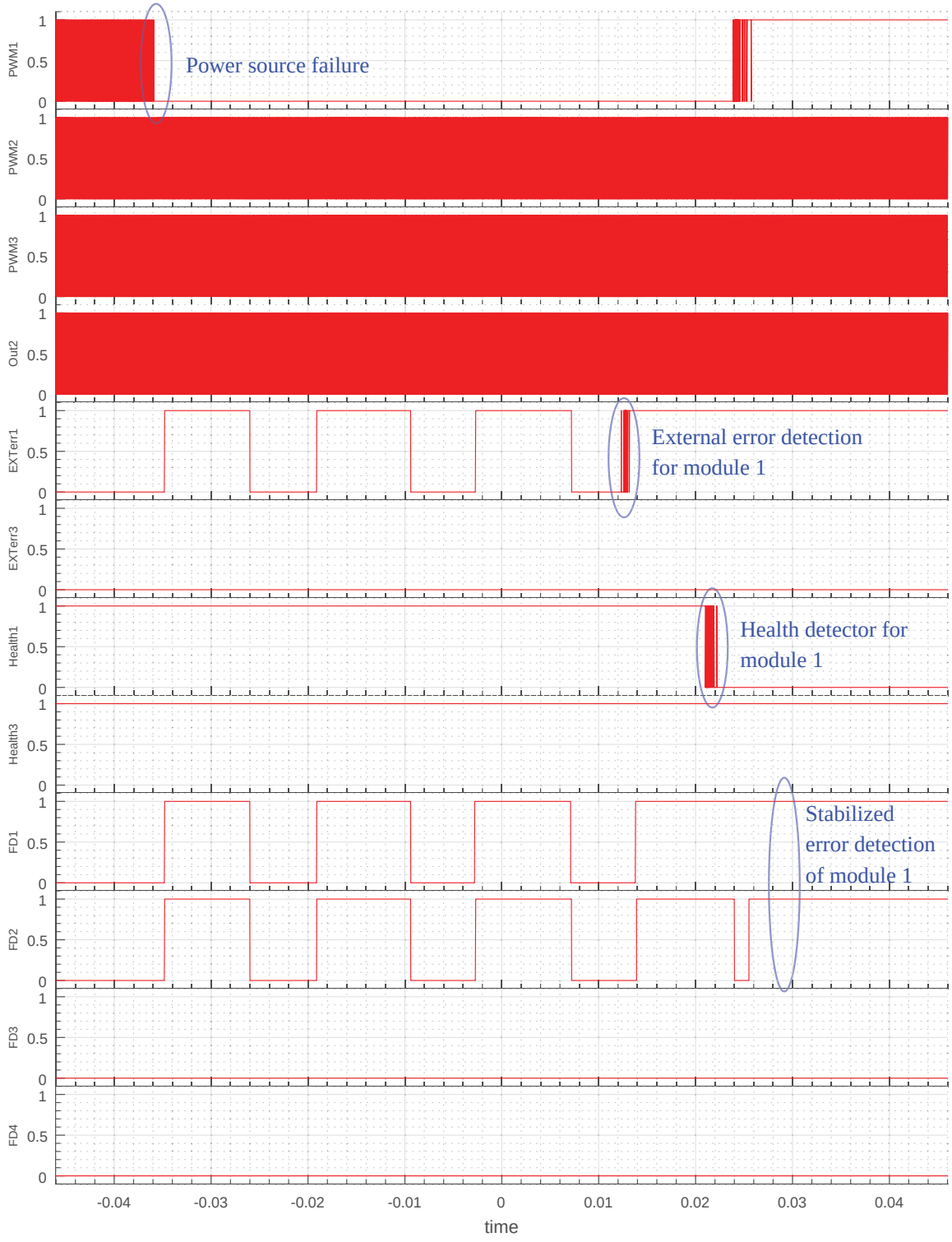


Figure 6.18: Power Failure in Module 1 and Its effect on Module 2 in CHB

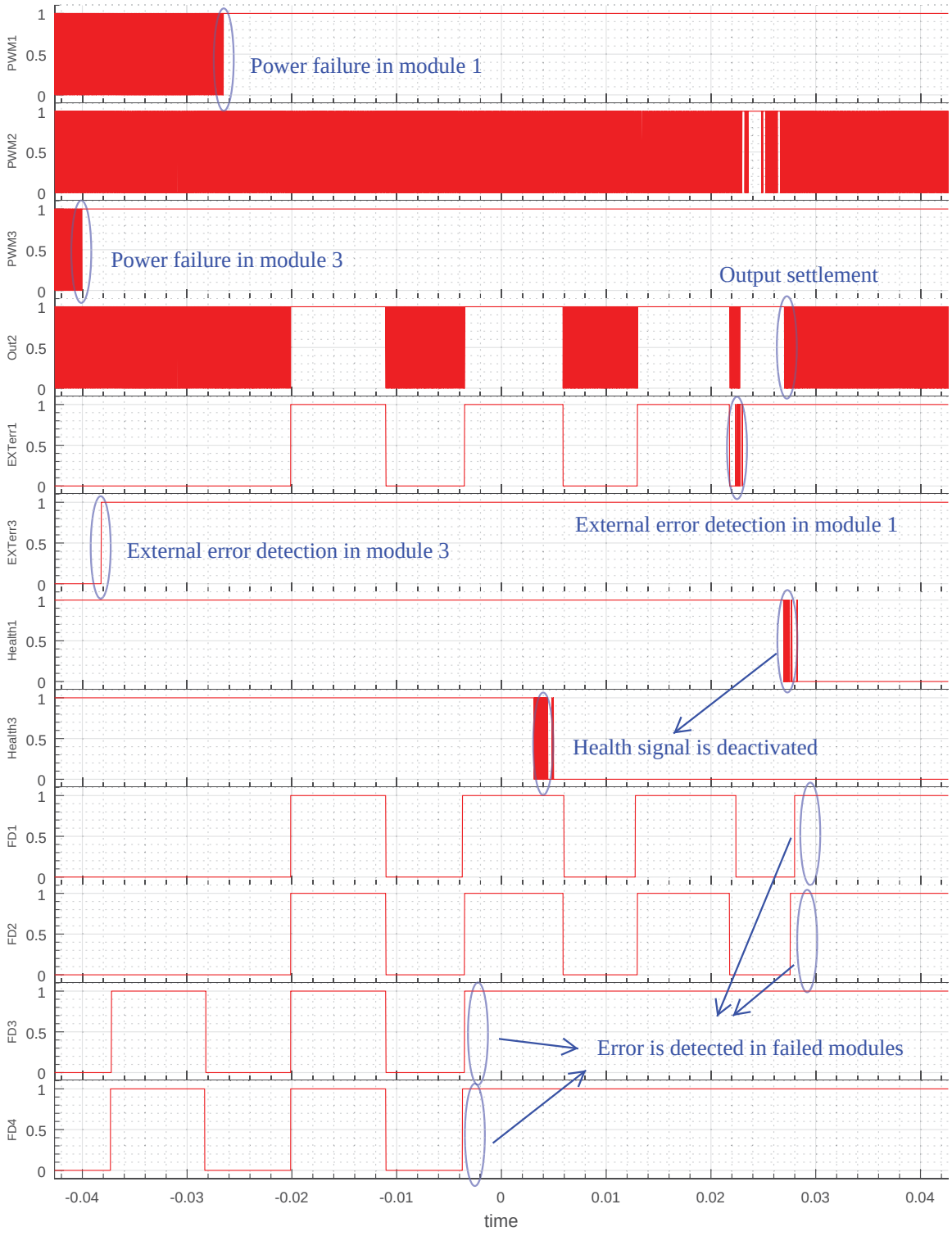


Figure 6.19: Power Failure in Module 1 and Module 3 and Its effect on Module 2 in CHB

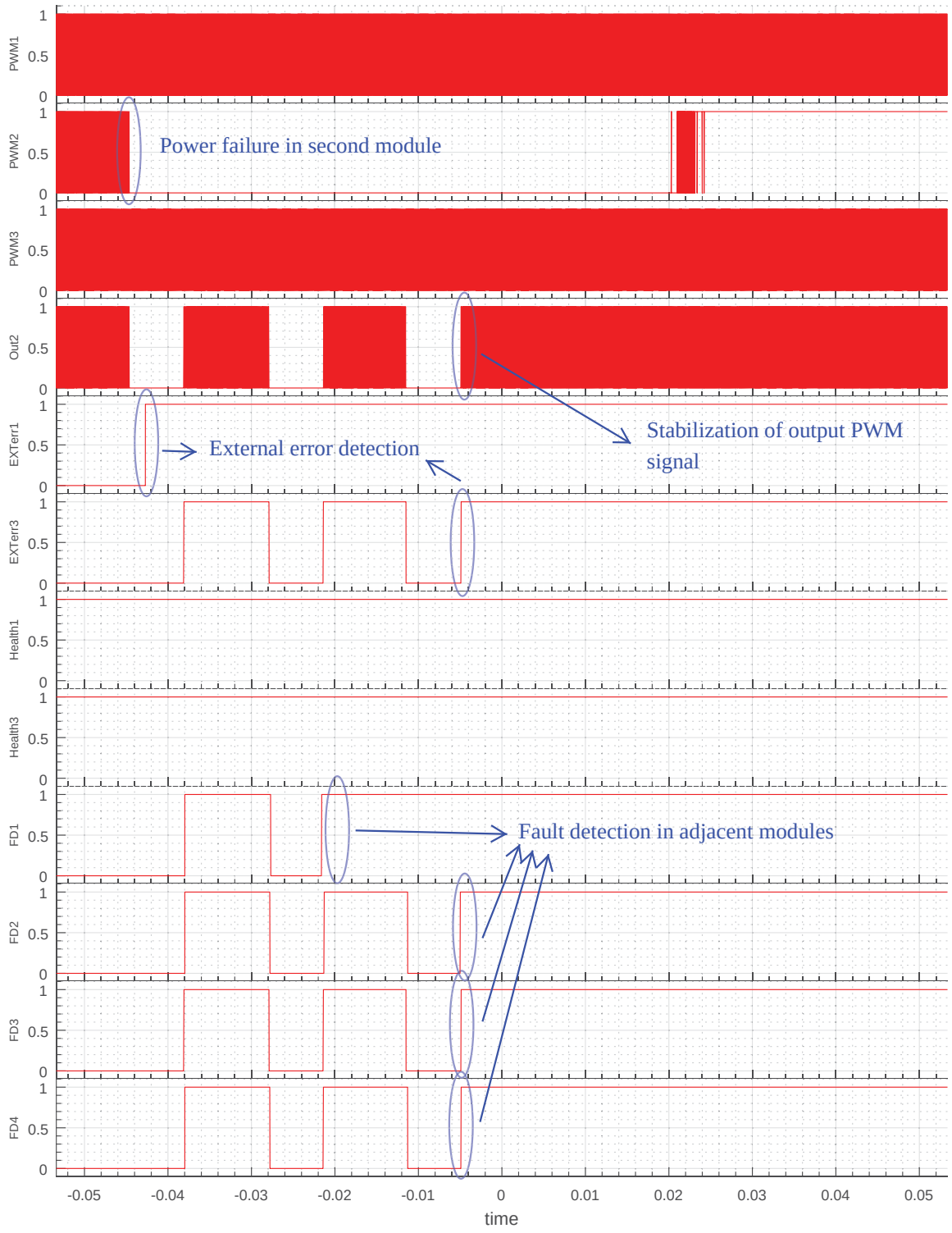


Figure 6.20: Power Failure in Module 2 and Its effect on Module 2 in CHB

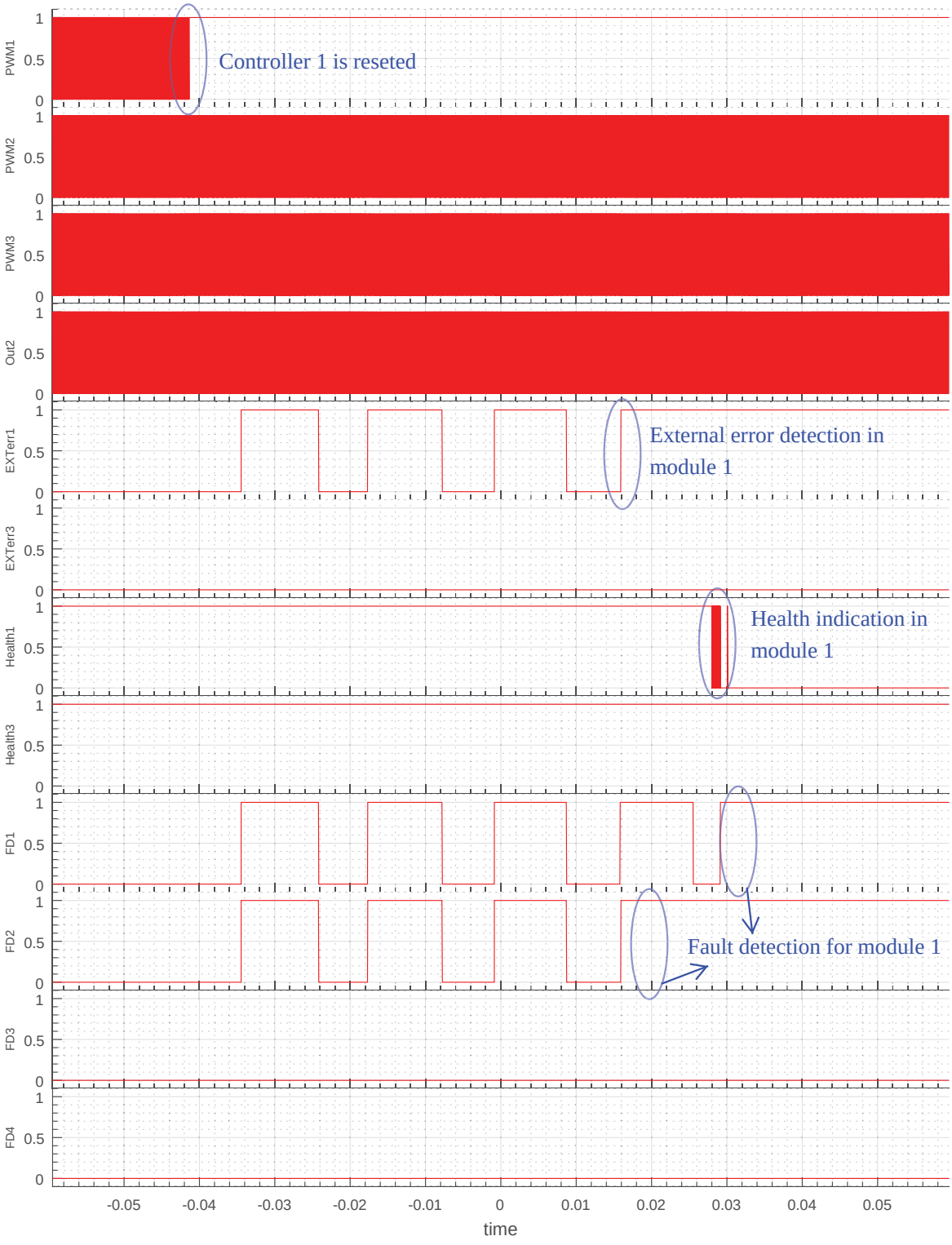


Figure 6.21: Micro-controller Reset in Module 1 and Its effect on Module 2 in CHB

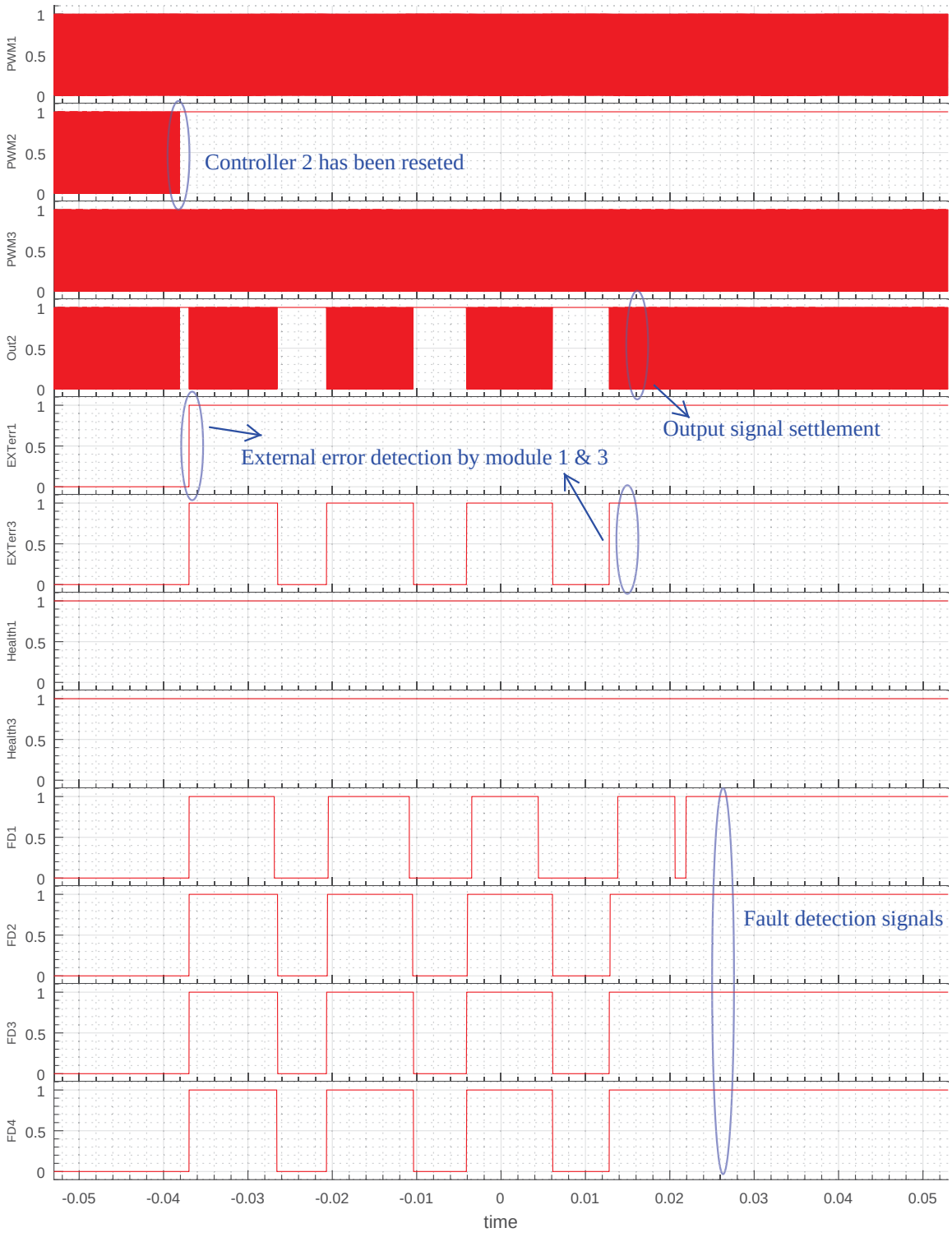


Figure 6.22: Micro-controller Reset in Module 2 and Its effect on Module 2 in CHB

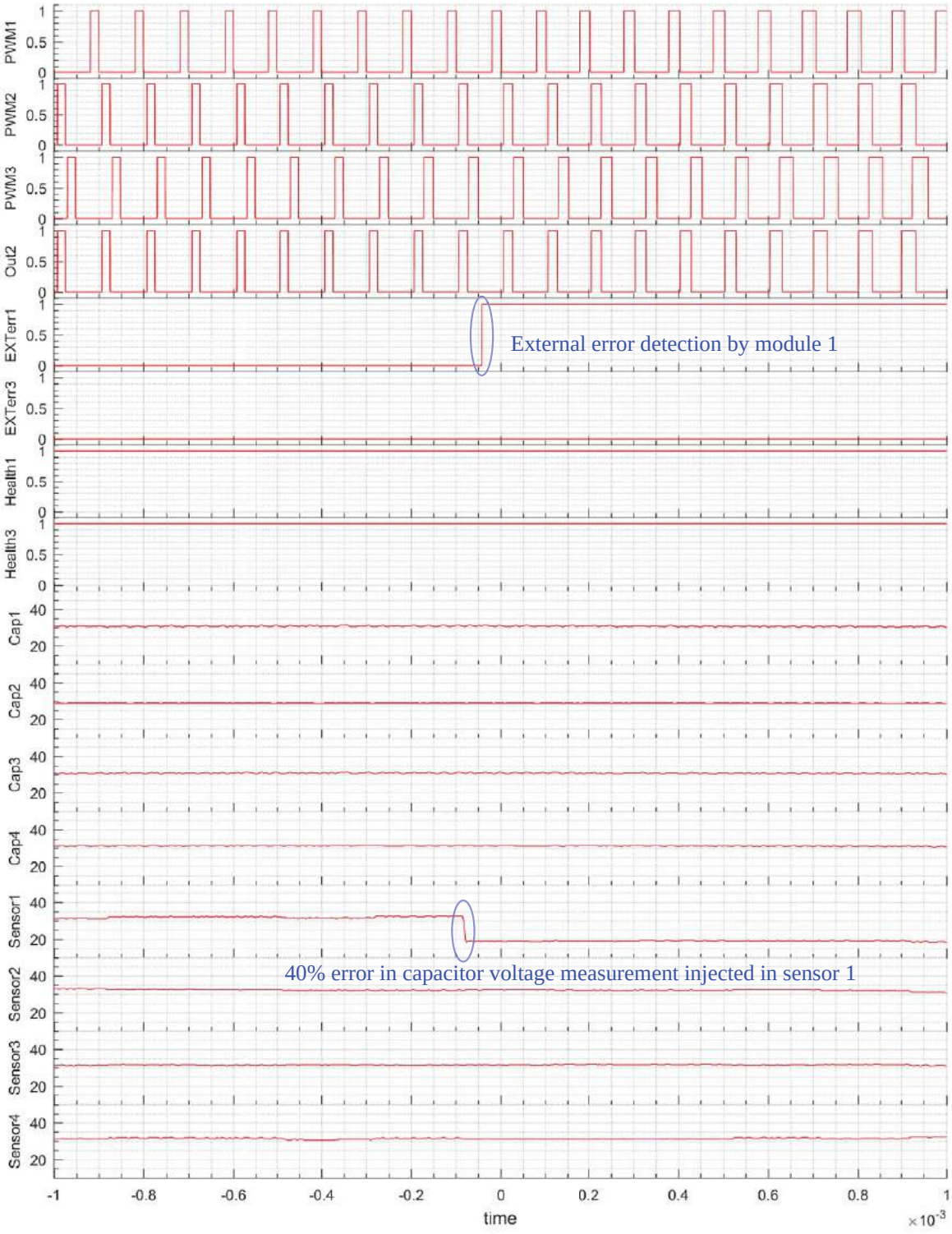


Figure 6.23: Voltage Sensor Failure in Module 1 and Its effect on Module 2 in CHB

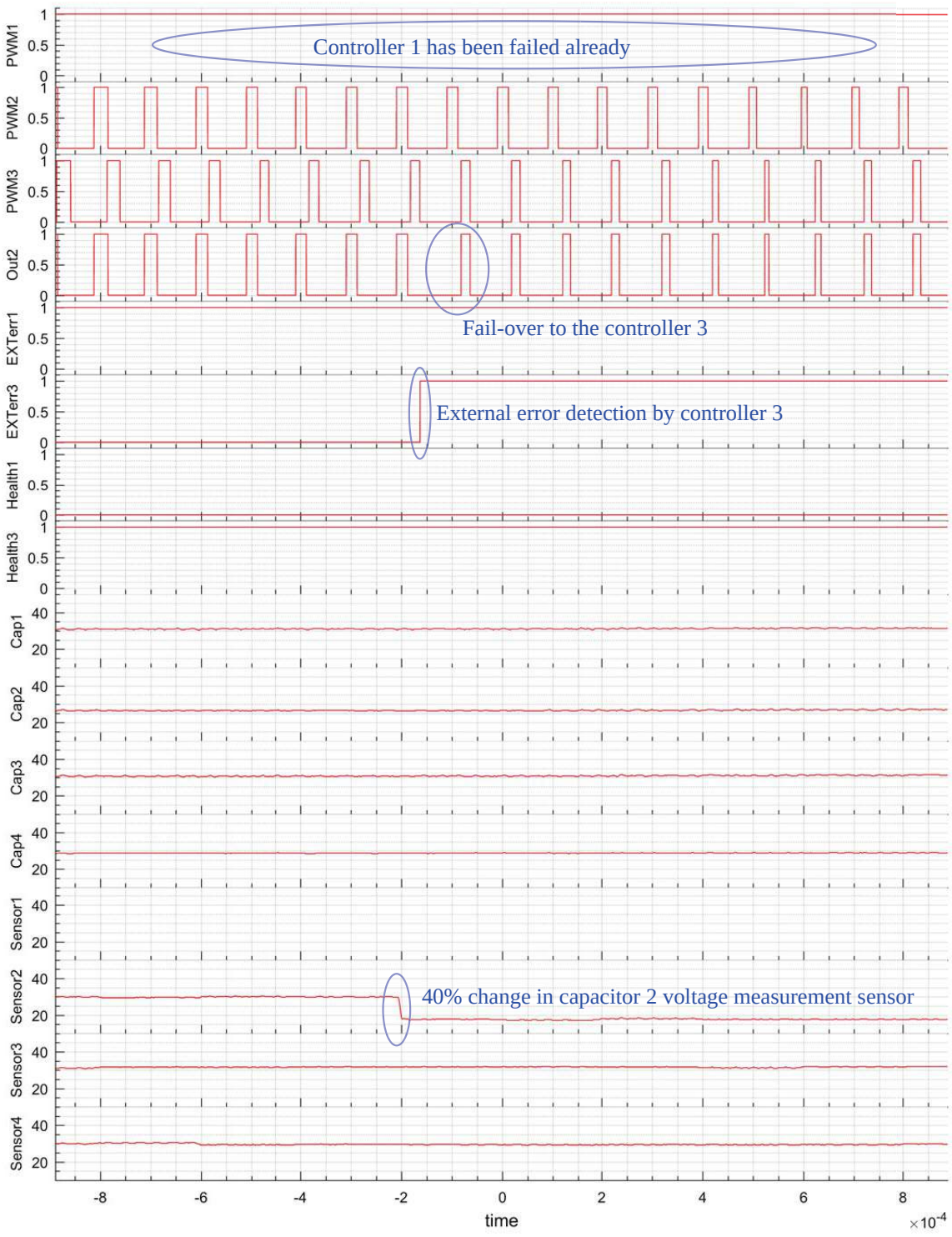


Figure 6.24: Voltage Sensor Failure in Module 3 While Module 1 has failed and Its effect on Module 2 in CHB

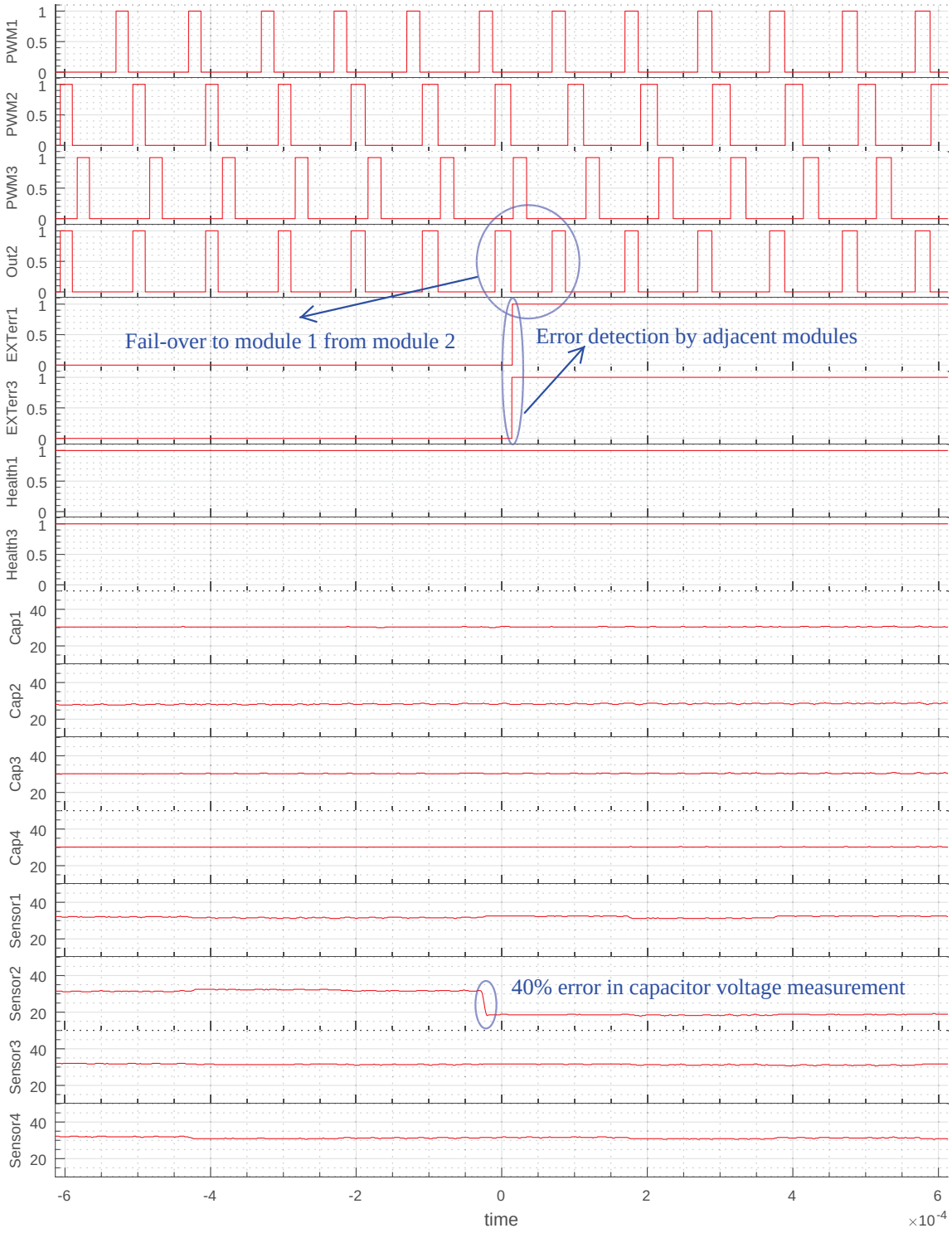


Figure 6.25: Voltage Sensor Failure in Module 2 and Its effect on Module 2 in CHB

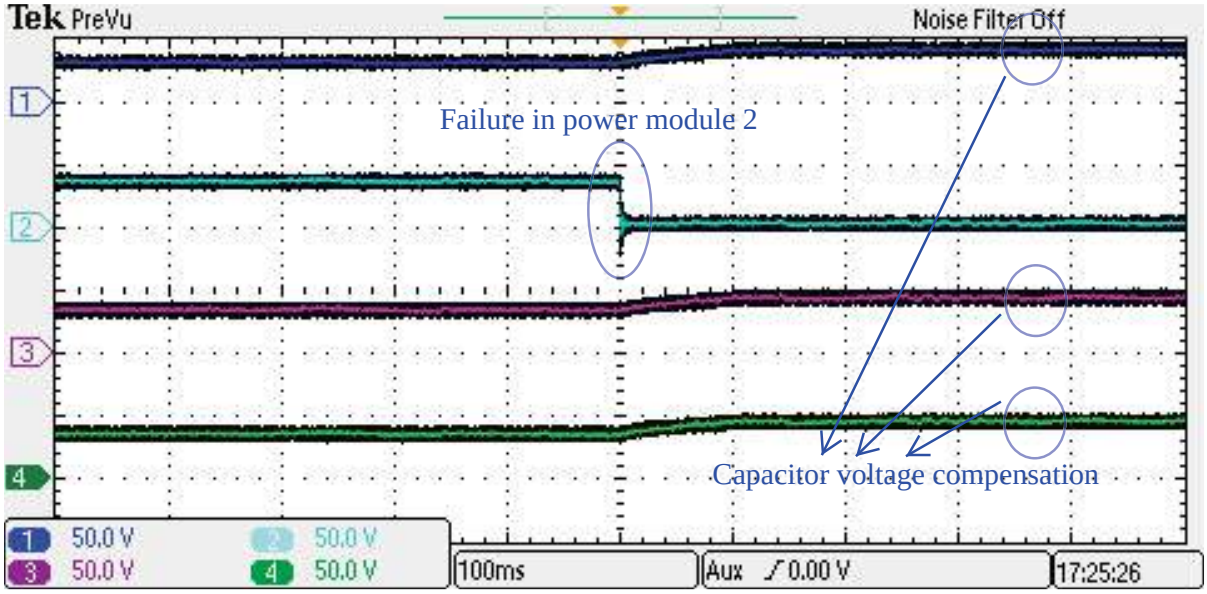


Figure 6.26: Capacitor Voltage in each Module at the Time of Failure in Power Electronic Circuit (Power Module Bypassed)

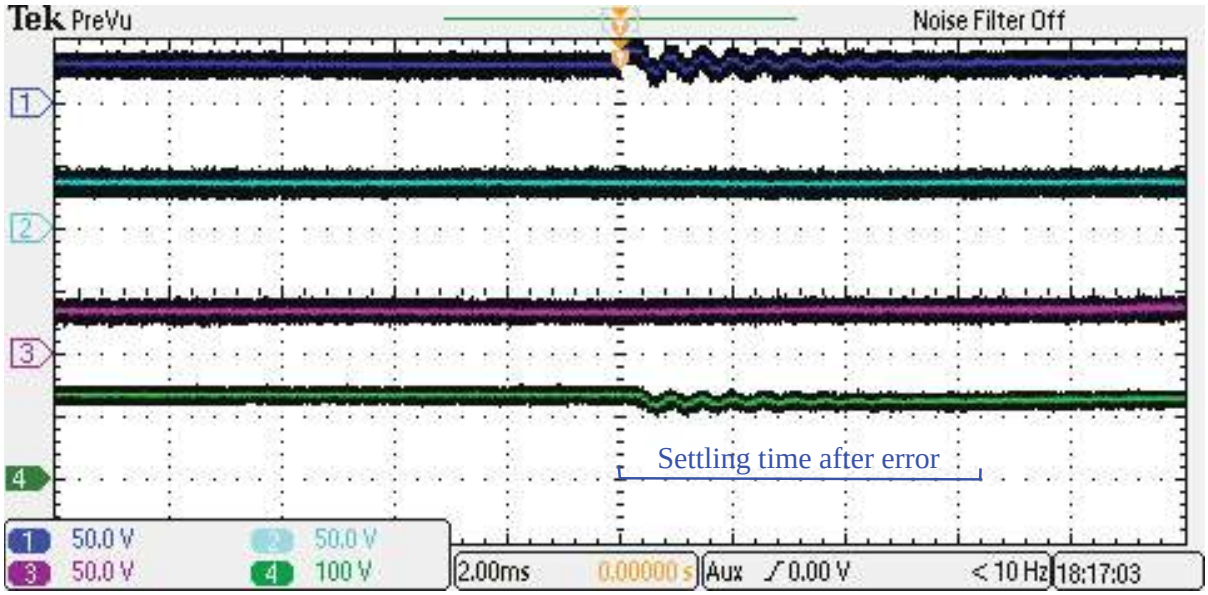


Figure 6.27: Average Capacitor Voltages in each Phase of the Converter at the Time of Failure in Power Electronic Circuit (Power Module Bypassed)

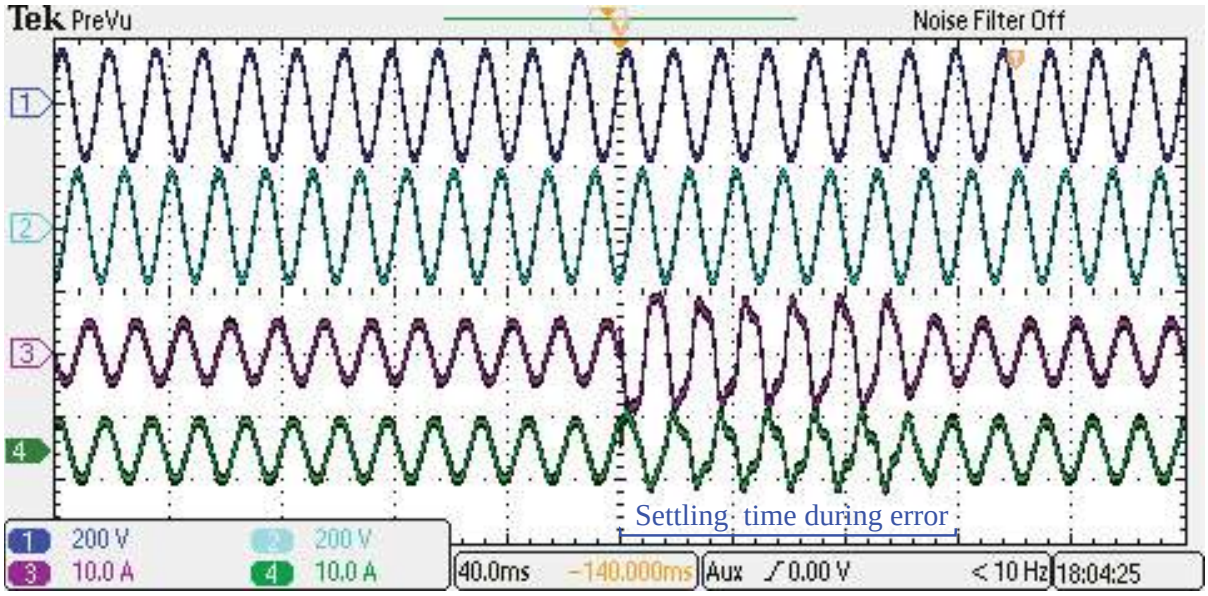


Figure 6.28: Grid Voltages and Currents at the Time of Failure in Power Electronic Circuit (Power Module Bypassed)

6.5 Hardware in the Loop Simulation Result for Modular Mult-level Converter Using Fault-tolerant Controller

Principle of operation for modular multi-level converter (MMC) has been represented in appendix B. The same converter has been modeled in Opal-rt HIL simulator to verify the feasibility of the proposed fault-tolerant converter (figure 6.29). The converter system consist of 4 module

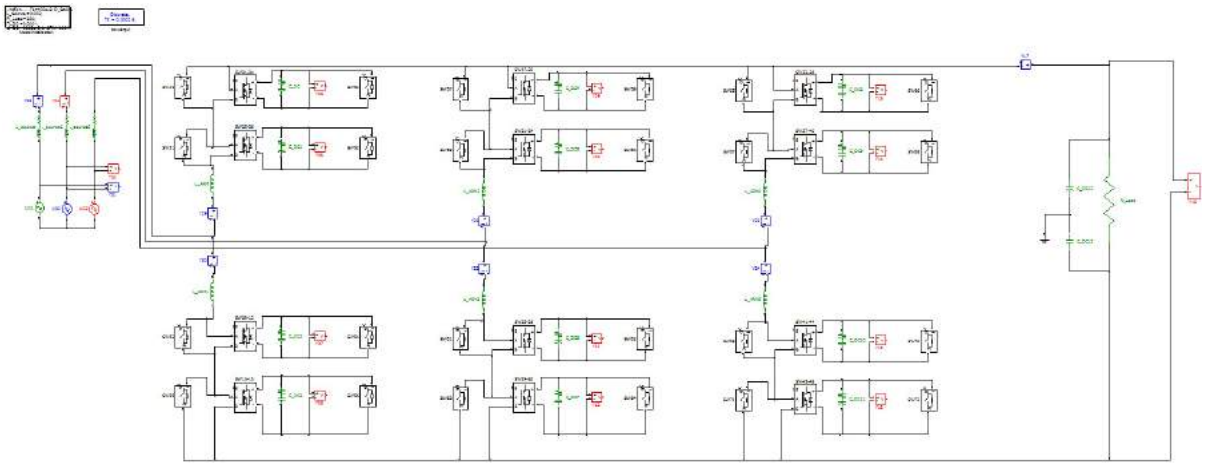


Figure 6.29: Implemented Modular Multi-level Converter (MMC) in Opal-rt HIL simulator

per phase. Each module is a cascaded h-bridge converter in which the AC sides are in series and the DC sides are connected to dc capacitor. Detail of the converter setup has been presented in table 6.2. The converter regulates high voltage dc side at 150 (V), which means each module would have dc voltage around 75 (V). The load is a 250 ohm resistor which is connected in parallel with dc capacitor.

There are several measurement sensor in the converter setup which has been connected to analog outputs. Analog scaling is done in a way that the output voltage never goes into saturation region. The gating signal for the power switches comes directly from the digital inputs (which

Table 6.2: Setup Configuration for MMC HIL Simulation

Design Parameter	Symbol	Value
Grid line-line voltage	E	120 (Vrms)
Line frequency	f	60 (Hz)
Series inductance	L_{series}	5 (mH)
Switching frequency	$f_{switching}$	10 (kHz)
Converter module per phase	N	4
Nominal module capacitor voltage	V_{module}	75 (V)
Converter module capacitance	C_{module}	6800 (μF)
Nominal converter current	$I_{converter}$	10 (A)
Load Resistance	R_{load}	250 (Ω)
Converter Arm Inductance	L_{arm}	1 (mH)

provided by controller). Different failure case has been tested on this converter setup and result is presented in the following sections.

6.5.1 Modular Multi-level Converter under Normal Operation

This section demonstrate the experimental result in normal operation of the MMC converter system. Figure 6.30 shows the grid voltage (line to line) and the grid currents. The measured grid voltage is fed in the synchronous reference frame PLL and the grid phase is shown in figure 6.31. Figure 6.32 shows the capacitor voltages in phase A of the converter. The goal is to regulate all capacitor voltages at 75 (V). Figure 6.33 shows the active and reactive current from the grid. Since converter is operating in rectifier mode, active current (I_d) is negative. Converter is not compensating reactive power of the grid and no reactive current (I_q) is being injected to the grid.

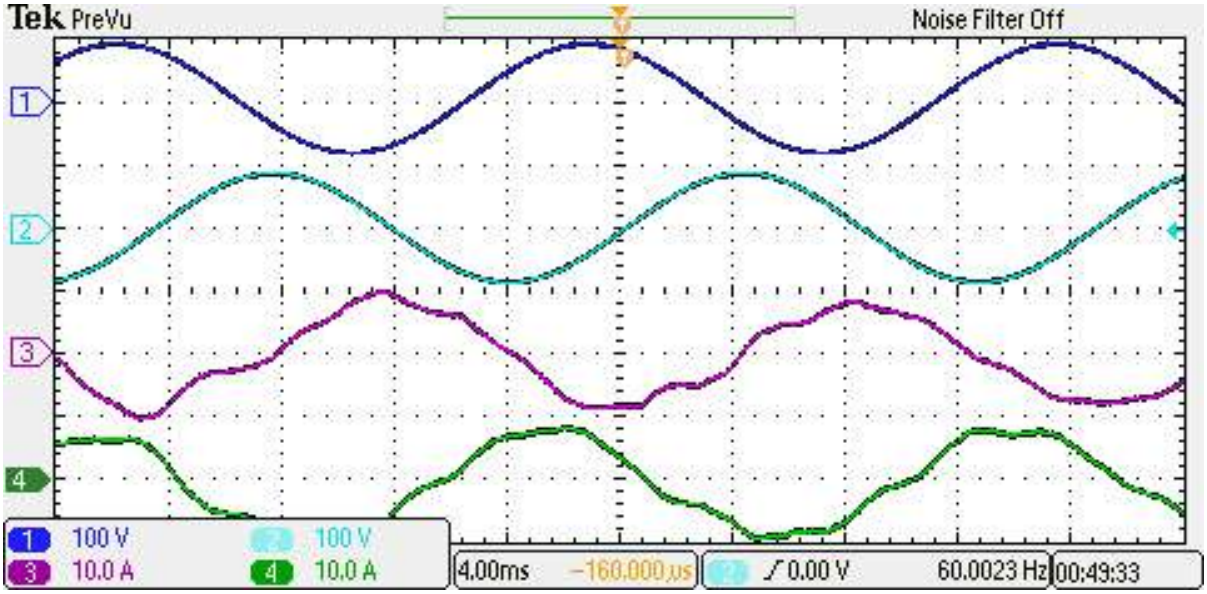


Figure 6.30: Grid Voltage and Current in Normal Operation of MMC (1:Vab 2:Vbc 3:Ia 4:Ib)

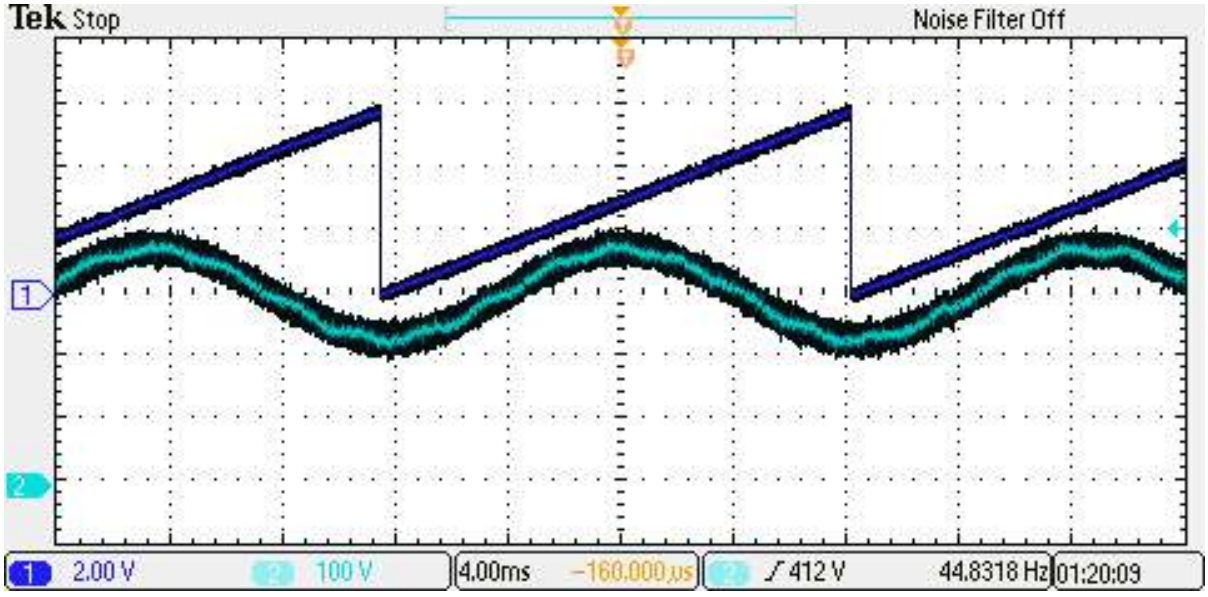


Figure 6.31: Synchronous Reference Frame PLL and Phase A Voltage (1:Phase (radian) 2:Phase A Voltage

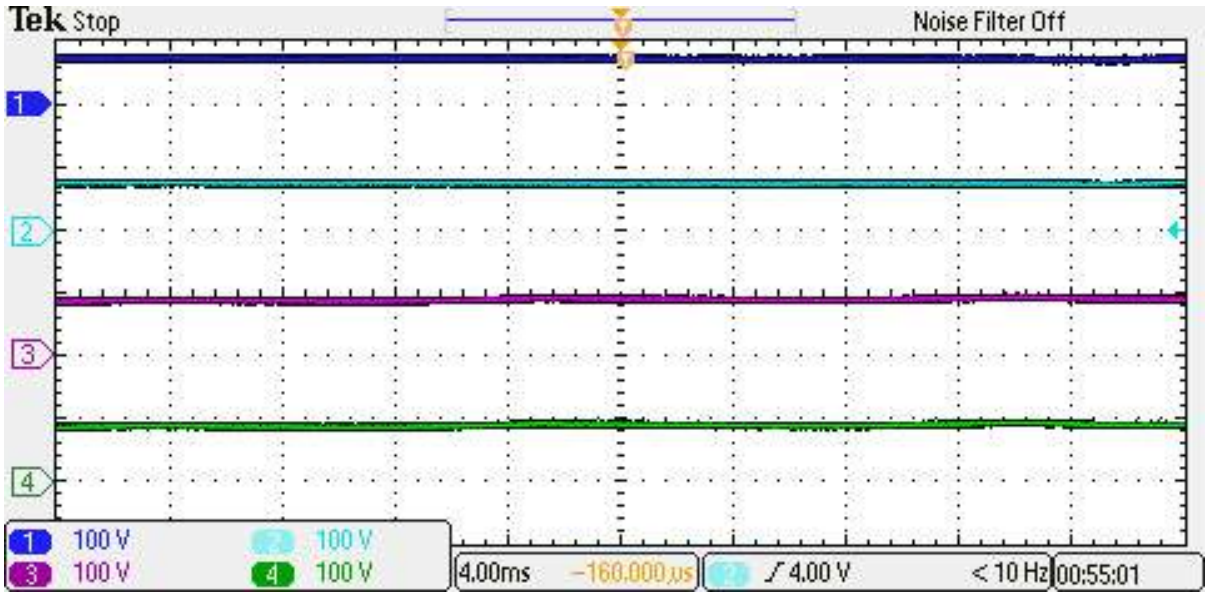


Figure 6.32: Capacitor Voltage of each Module in Normal Operation (1:Module 1 2:Module 2 3:Module 3 4:Module 4)

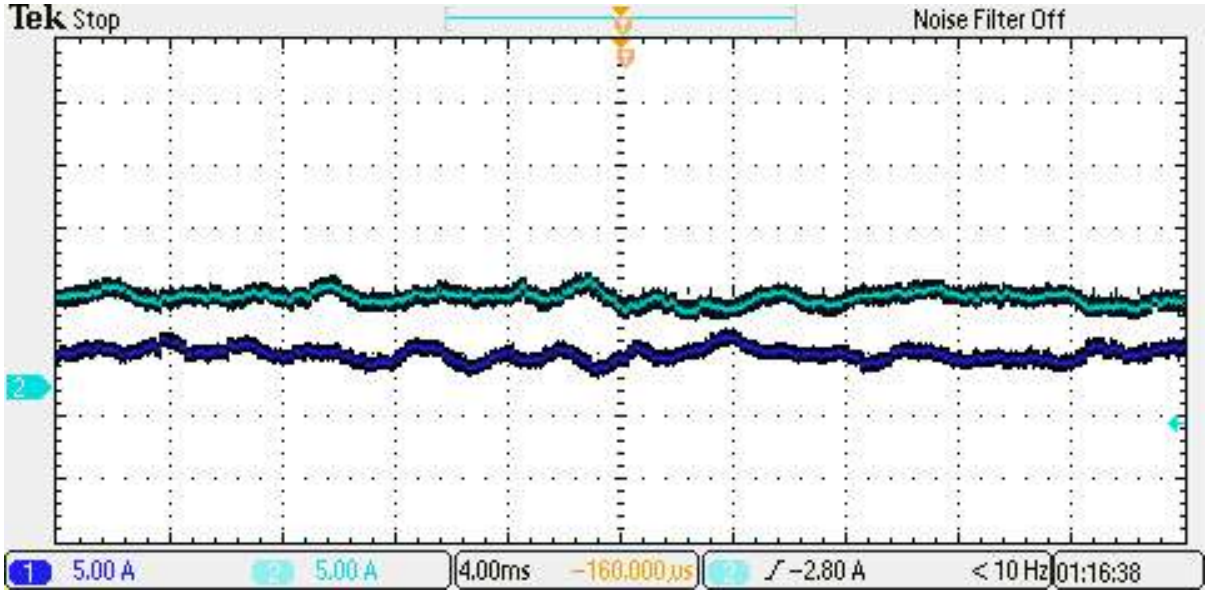


Figure 6.33: Active and Reactive Current in Normal Operation (1:Active Current 2:Reactive Current)

6.5.2 Fault Injection Result In Controller Architecture for Modular Multi-level Converter

In this section, functionality of the controller circuit has been investigated under different failure modes. In each mode, different failure has been injected to the controller cards or the associated control circuitry for each controller to emulate single point of failure in the system. In all of the results, focus is on controller number 2 and it has been tried to show the experimental result related to this controller (i.e. effect of failure in adjacent controllers on this controller). For understanding definition of each signal, figure 6.3 may be used which has block diagram of the fault handling circuit and the related signal names. All of the results have been captured by oscilloscope or logic analyzer from the experimental test bed in connection with the converter system. Result in this section might be very similar to the fault injection result for CHB converter. The goal is to show that fault-tolerant controller is not converter dependent and It may be used for different type of power converters.

Figure 6.34 shows the case in which communication link from controller 1 to controller 2 has been failed. Therefore controller 2 may not receive any data and error flag for controller 1 will turn on. Since it is not common failure, controller 2 may not get affected at all.

Figure 6.35 shows the case in which communication link from controller 2 to other controllers has failed (this may happen as a result of failure in controller 2). Therefore controller 1 and 3 may not receive any data and error flag from controller 1 and 3 will turn on. Since it is common failure, controller 2 will be bypassed.

Figure 6.36 shows the case in which built in self-test circuit (which turns on when internal failure happens) in controller 1 has turned on and its effect of controller 2 has been demonstrated. This failure will turn on the fault detection for controller 1 (FD1) and turn off the health status for controller 1. Since it is not a common failure, controller 2 will continue its operation without

interruption.

Figure 6.37 demonstrates the case in which built in self-test circuit (which turns on when internal failure happens) in controller 2 has turned on and its effect of controller 2 has been shown. This failure will turn on the fault detection related to controller 2 (FD2 and FD3) and turn on external error indicators from controller 1 and 3. Since it is a common failure, controller 2 will be bypassed and controller 1 will take charge of controlling the second power module.

Figure 6.38 demonstrates the case in which power supply of the controller 1 has failed (by turning off the controller card). This failure will turn on the fault detection related to controller 1 (FD1 and FD2) and turn on external error indicators from controller 1. Since it is not a common failure, controller 2 will continue its operation without interruption.

Figure 6.39 demonstrates the case in which power supply of the controller 1 and 3 have failed simultaneously (by turning off the controller card). This failure will turn on the fault detection related to controller 1 and 3 (FD1 to FD4), turn on external error indicators from controller 1 and 3, and turn off health indicators related to controller 1 and 3. Since both health indicators of the failed controller have been turned off, controller 2 will continue its operation without interruption (i.e. controller 2 will switch its control circuitry when health indicators are still on).

Figure 6.40 shows the case in which power supply of the controller 2 has failed (by turning off the controller card). This failure will turn on the fault detection related to controller 2 (FD1 to FD4) and turn on external error indicators from controller 1 and 3. Since it is a common failure, controller 2 will be bypassed by the first controller.

Figure 6.41 shows the case in which controller card 1 has been restarted (by external reset pin). External restart will make it unavailable for a period of time and during this time, it may not function correctly. This will turn on the external error for controller 1 and clears the health indicator for this controller. Since it is not a common mode failure, controller 2 will continue its operation without interruption.

Figure 6.42 shows the case in which controller card 2 has been restarted (by external reset pin). External restart will make it unavailable for a period of time and during this time, it may not function correctly. This will turn on the external error for controller 1 and 3 and clears the health indicator for these controllers too. Since it is a common mode failure, controller 2 will be bypass by the first controller.

Figure 6.43 demonstrates the case in which voltage measurement sensor in module 1 fails. This action has been emulated by a step change (40% decrease in the value) in the measured capacitor voltage. The change in the measured value will trigger the external error signal in module 1 but since it is not a common error, it will not bypass the controller module 2.

Figure 6.44 demonstrates the case in which voltage measurement sensor in module 3 fails while module 1 has already been failed. This action has been emulated by a step change (40% decrease in the value) in the measured capacitor voltage of module 3 while module 1 has been turned off. The change in the measured value will trigger the external error signal in module 1 and 3 but since it is a common error, it will bypass the controller module 2 to its adjacent controller (controller module 3).

Figure 6.45 demonstrates the case in which voltage measurement sensor in module 2 fails. This action has been emulated by a step change (40% decrease in the value) in the measured capacitor voltage of module 2. The change in the measured value will trigger the external error signal in module 1 and 3 but since it is a common error, it will bypass the controller module 2 to its adjacent controller (controller module 1).

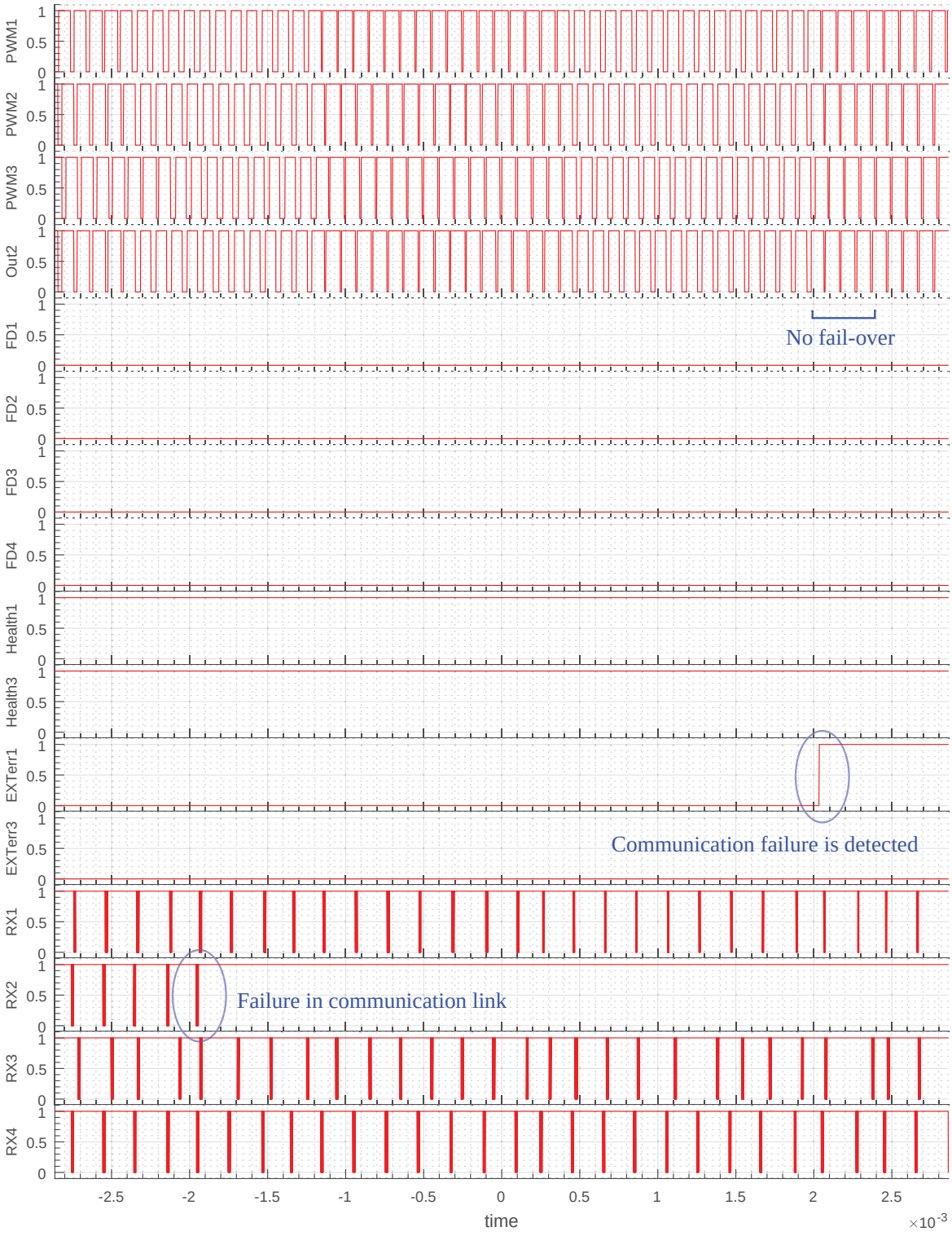


Figure 6.34: Communication Failure in Module 1 and Its Effect on Module 2 in MMC

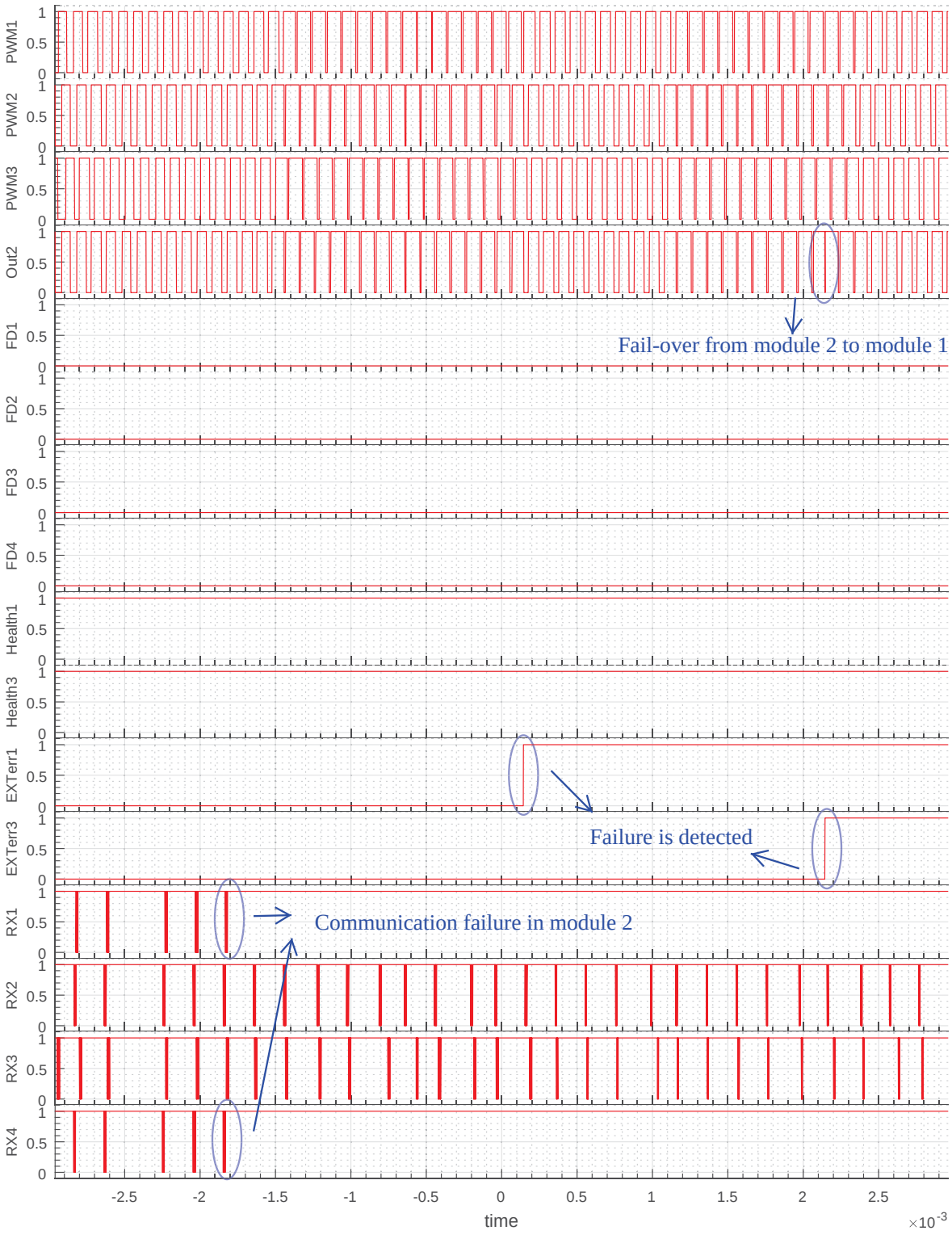


Figure 6.35: Communication Failure in Module 2 and Its Effect on Module 2 in MMC

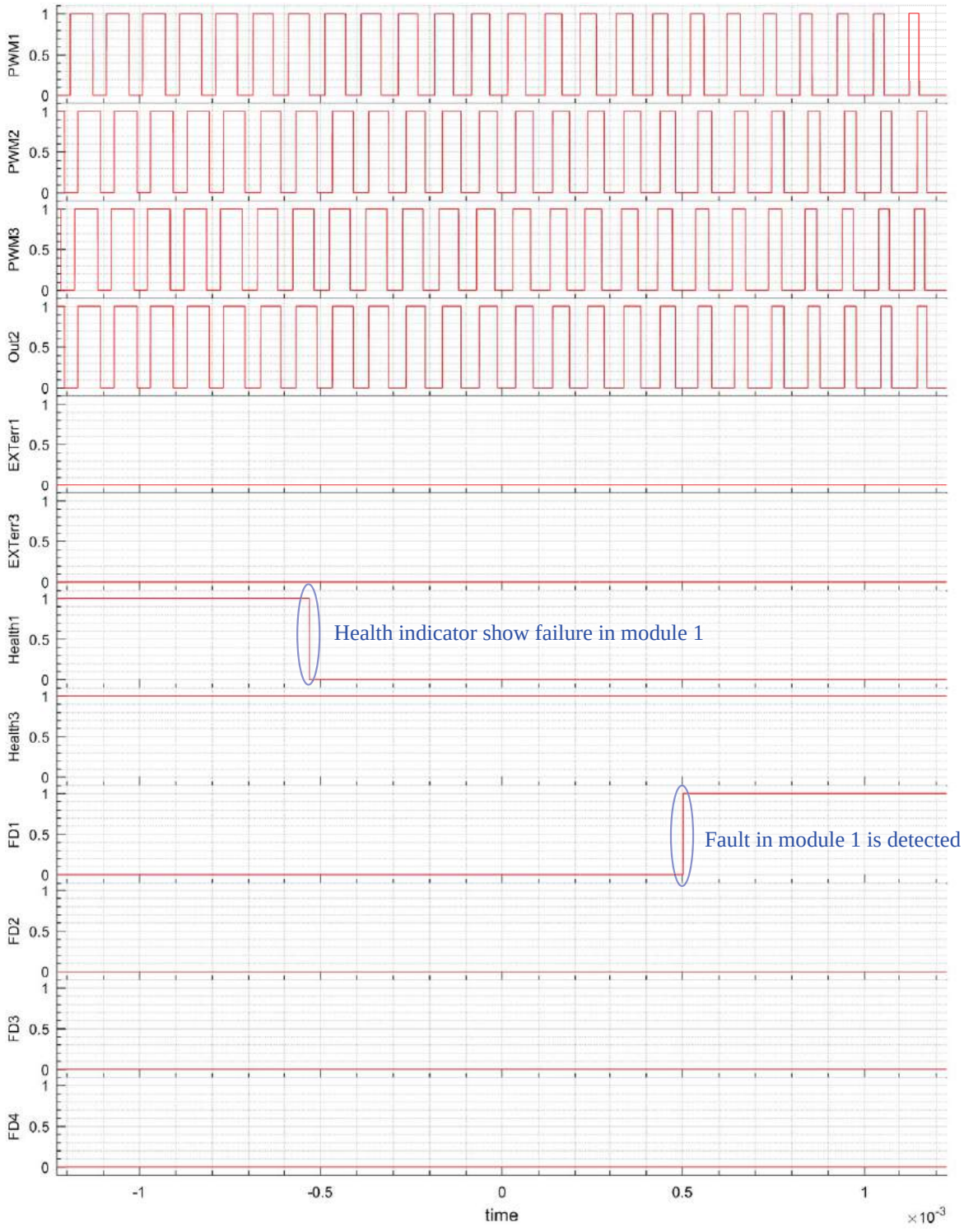


Figure 6.36: Built In Self-test (BIST) Failure in Module 1 and Its Effect on Module 2 in MMC

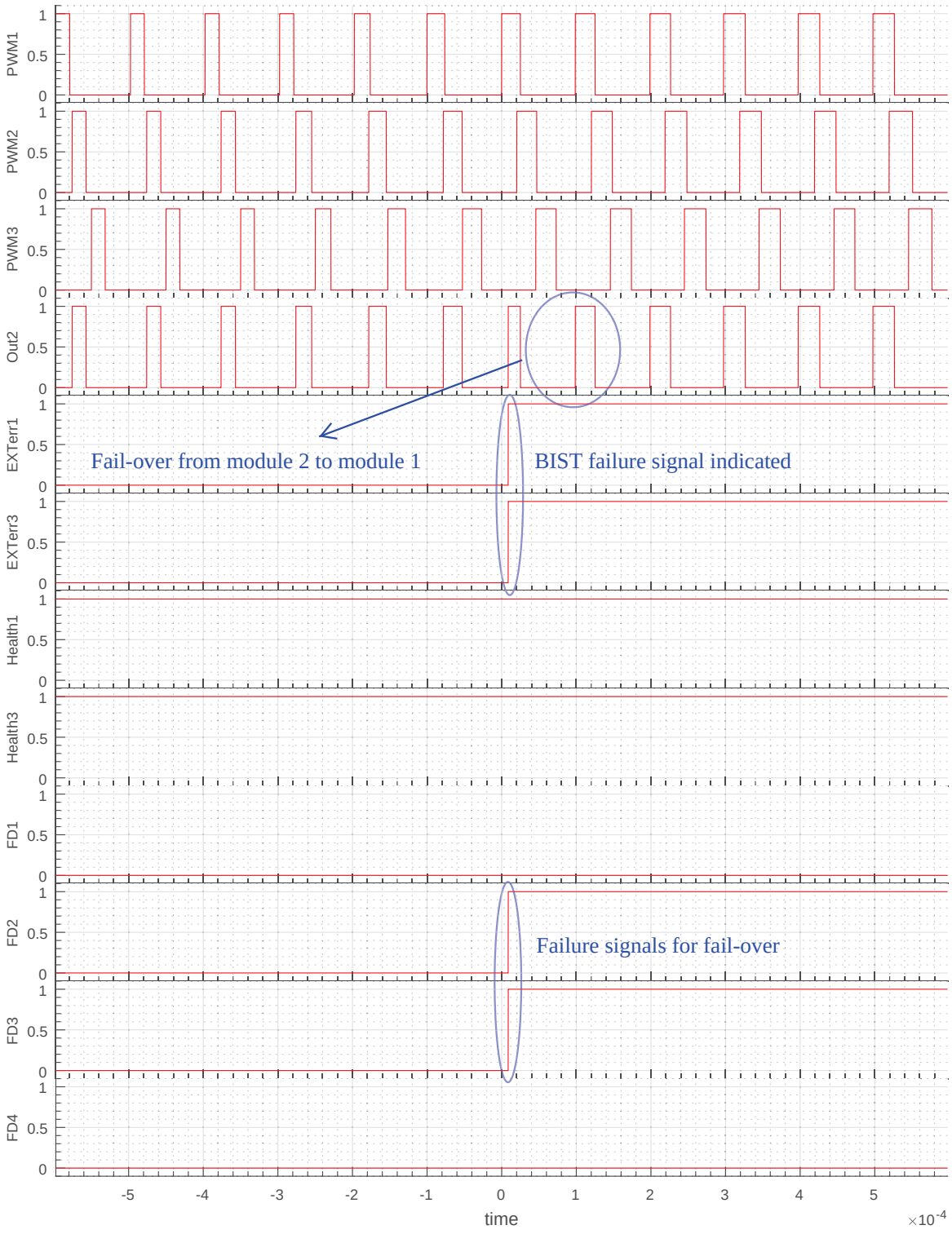


Figure 6.37: Built In Self-test (BIST) Failure in Module 2 and Its Effect on Module 2 in MMC

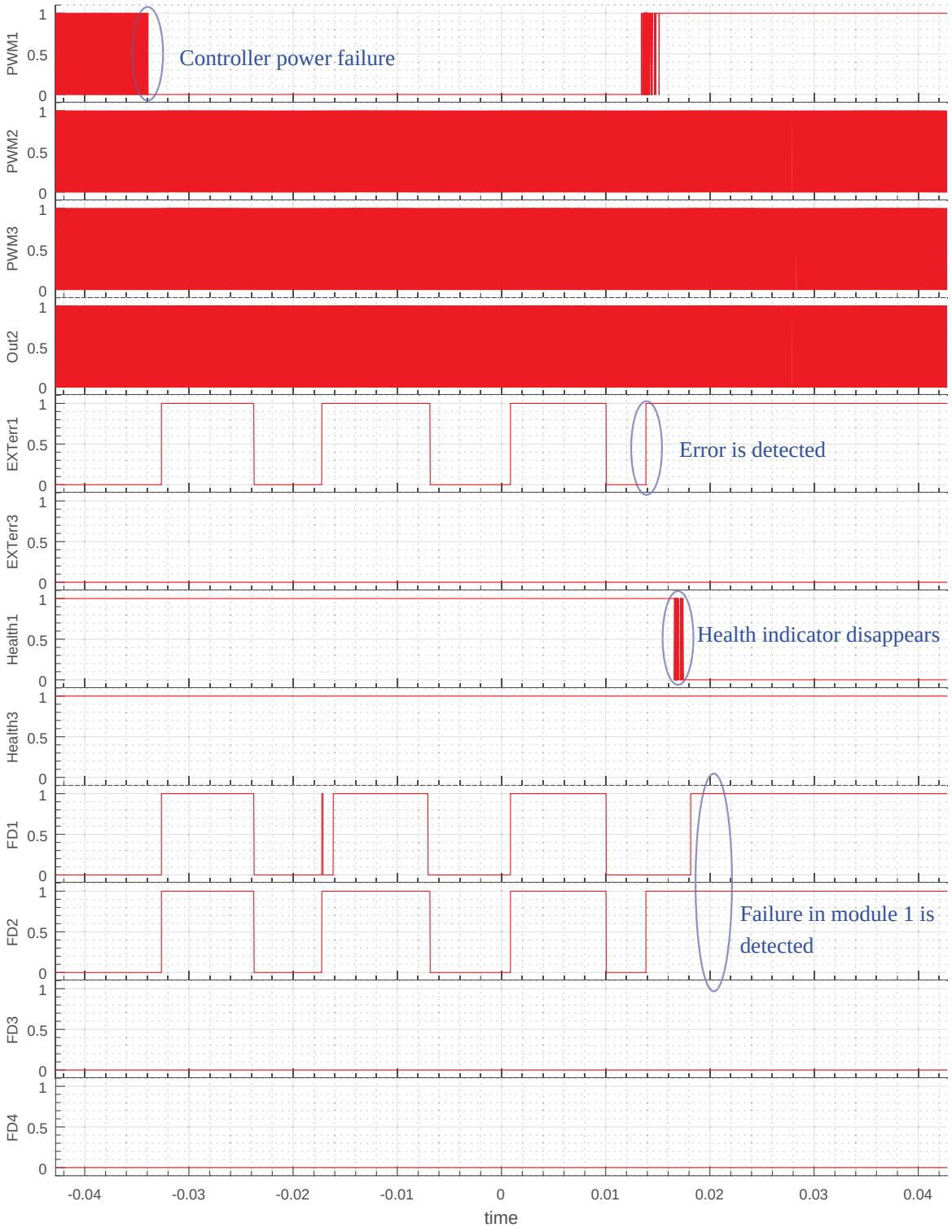


Figure 6.38: Power Failure in Module 1 and Its effect on Module 2 in MMC

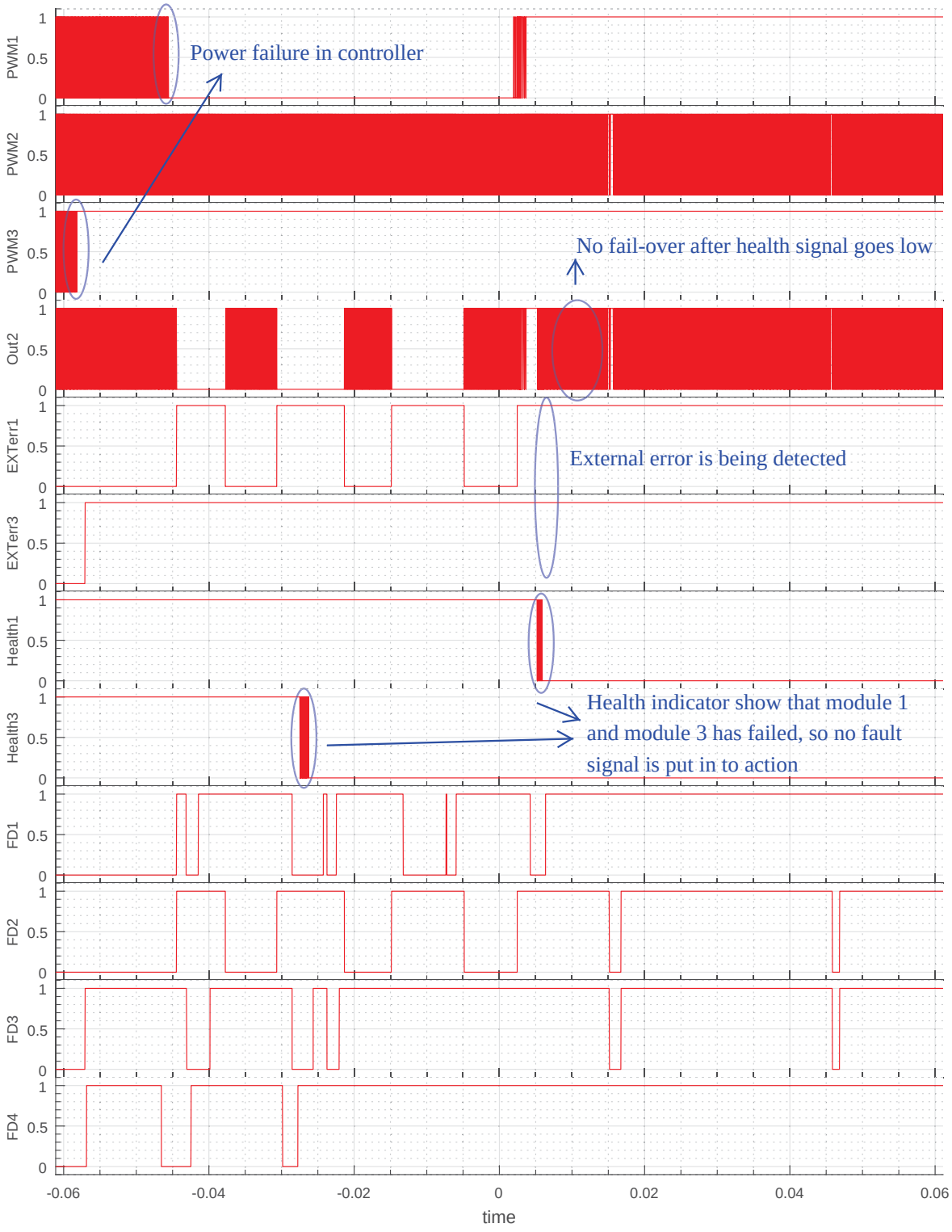


Figure 6.39: Power Failure in Module 1 and Module 3 and Its effect on Module 2 in MMC

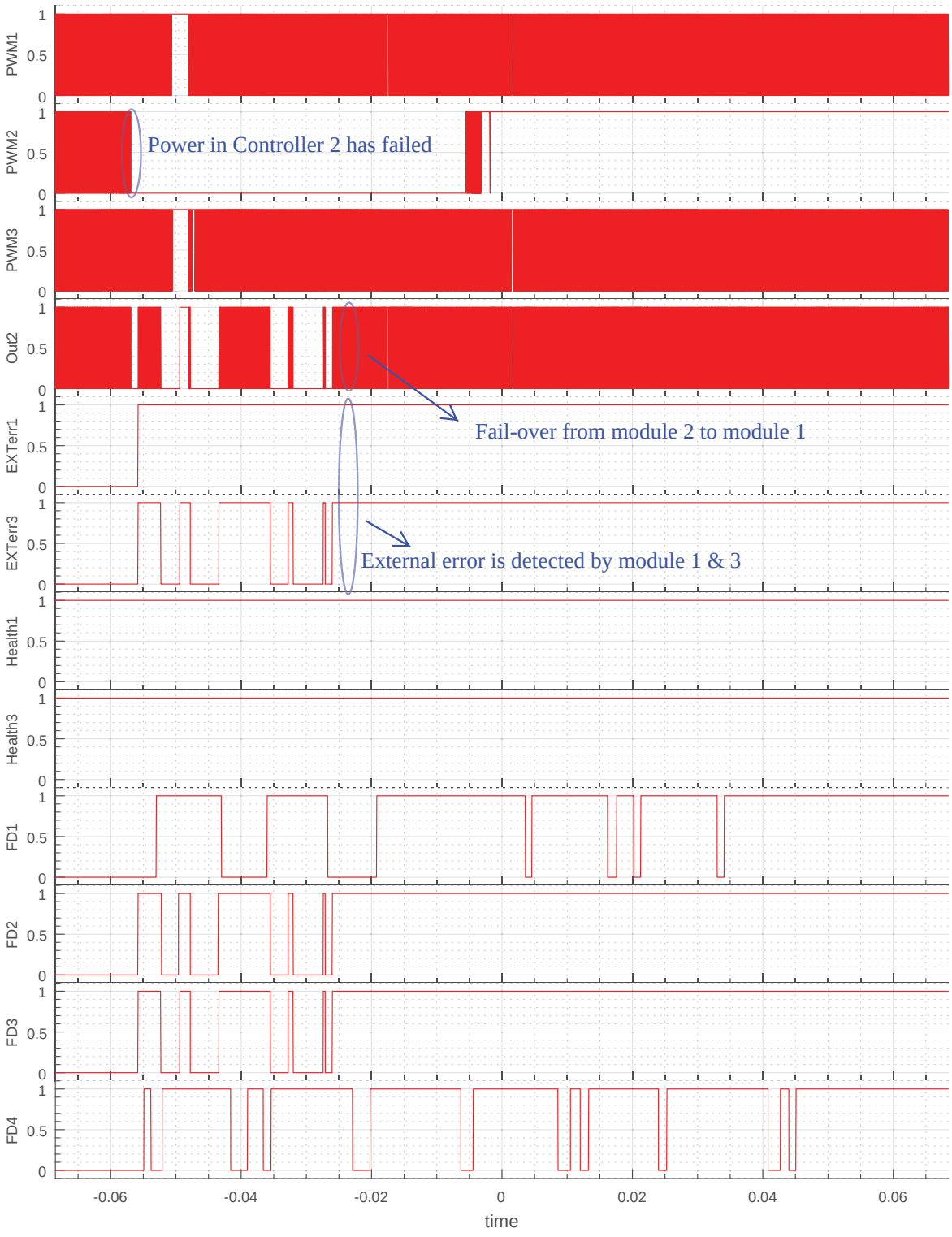


Figure 6.40: Power Failure in Module 2 and Its effect on Module 2 in MMC

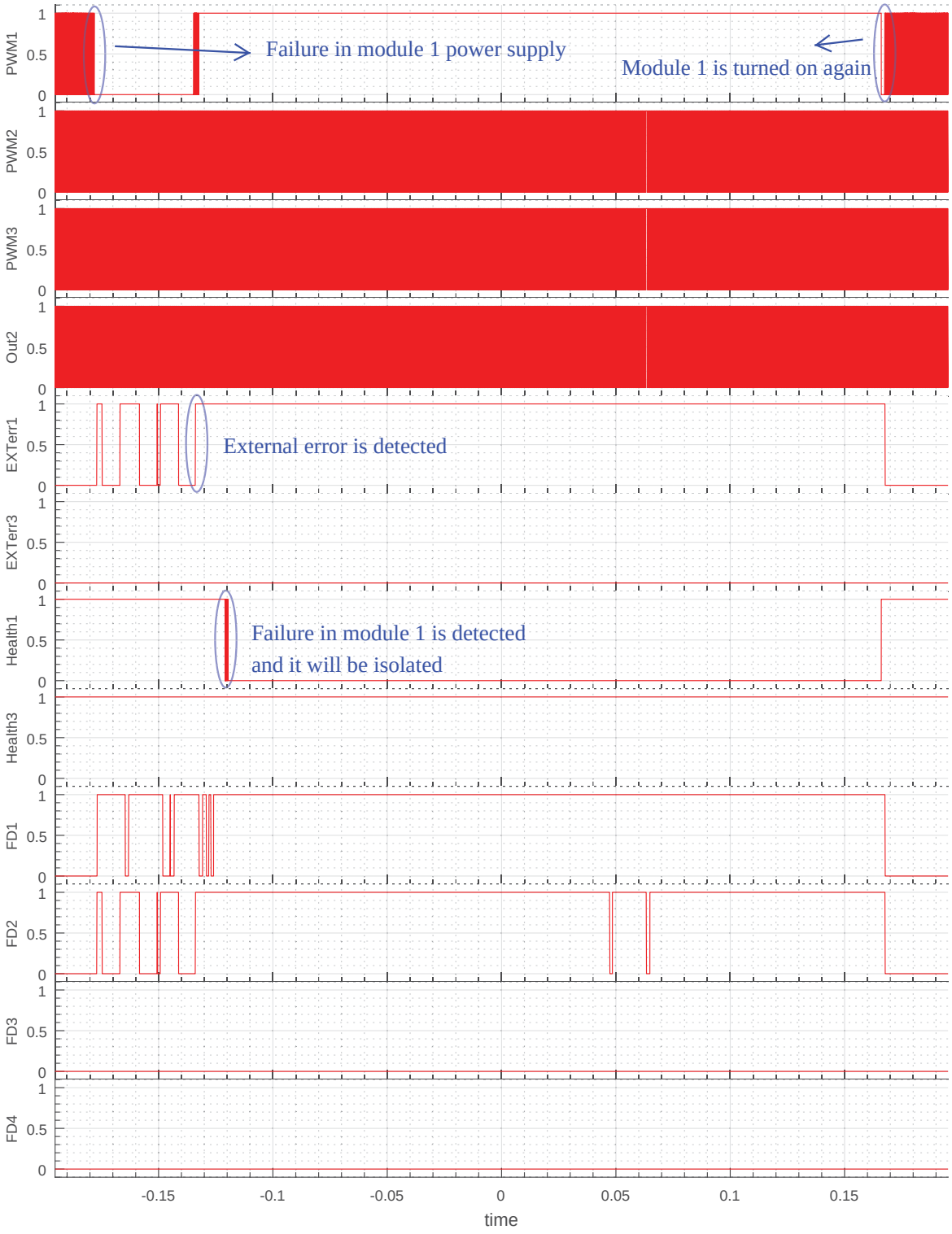


Figure 6.41: Micro-controller Reset in Module 1 and Its effect on Module 2 in MMC

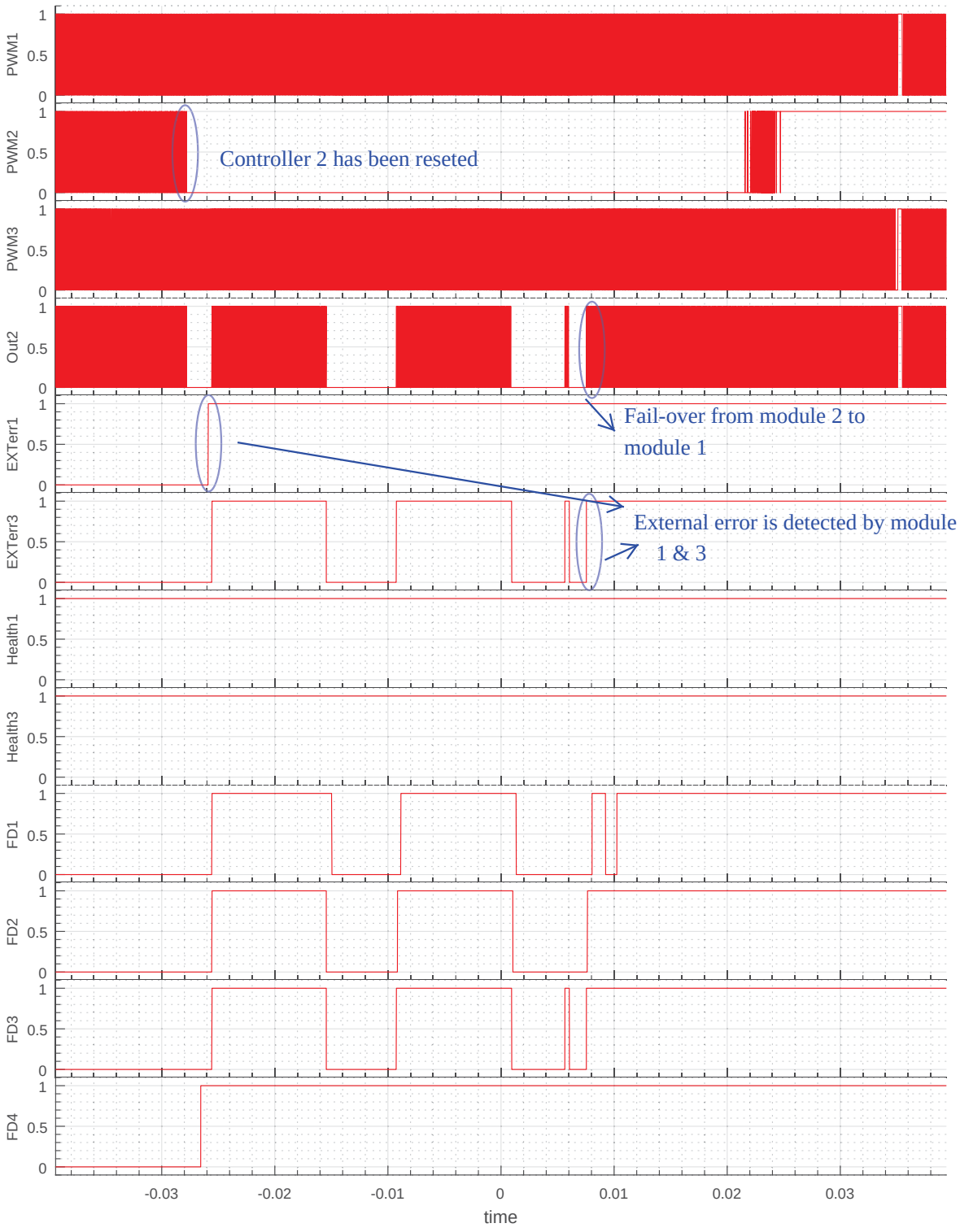


Figure 6.42: Micro-controller Reset in Module 2 and Its effect on Module 2 in MMC

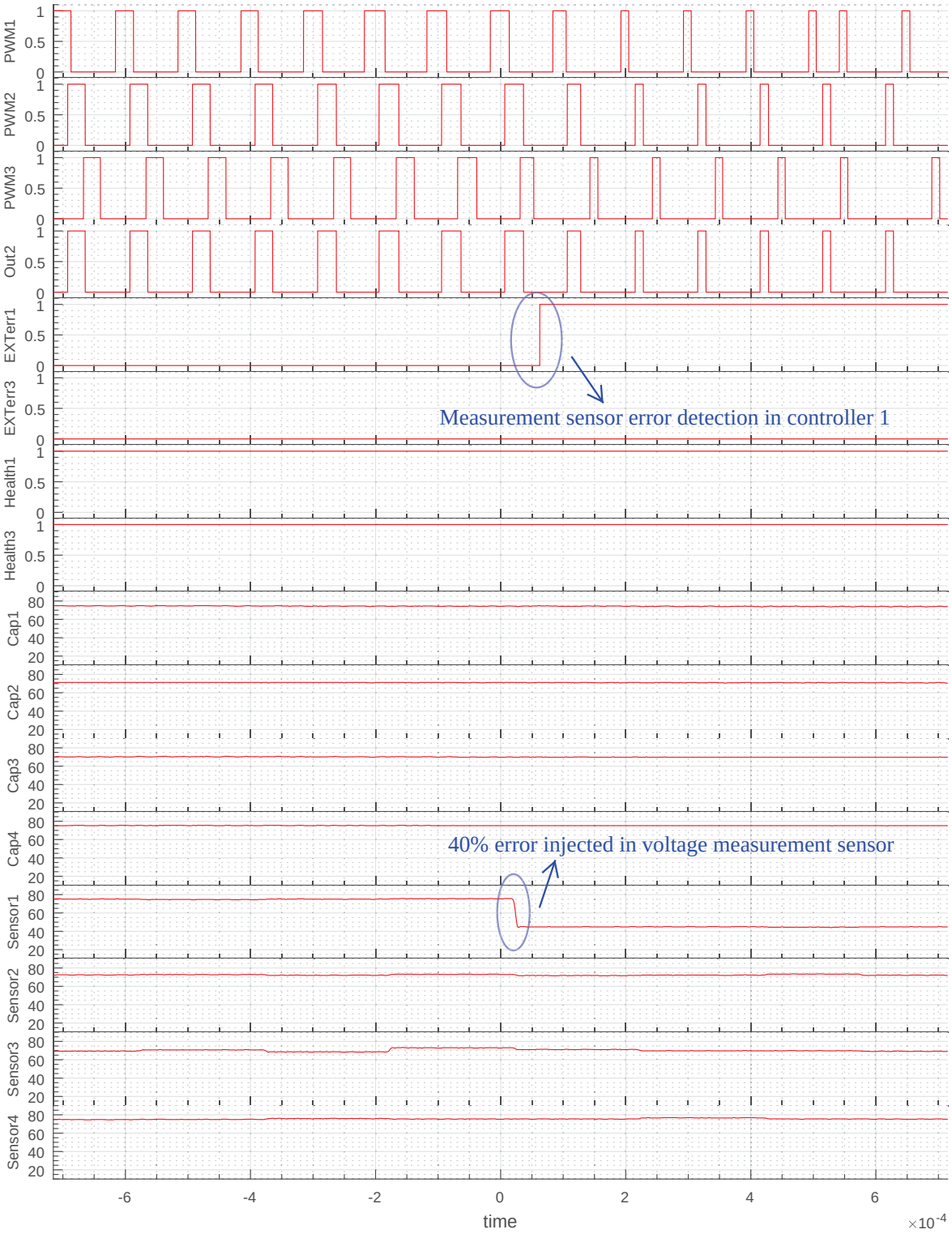


Figure 6.43: Voltage Sensor Failure in Module 1 and Its effect on Module 2 in MMC

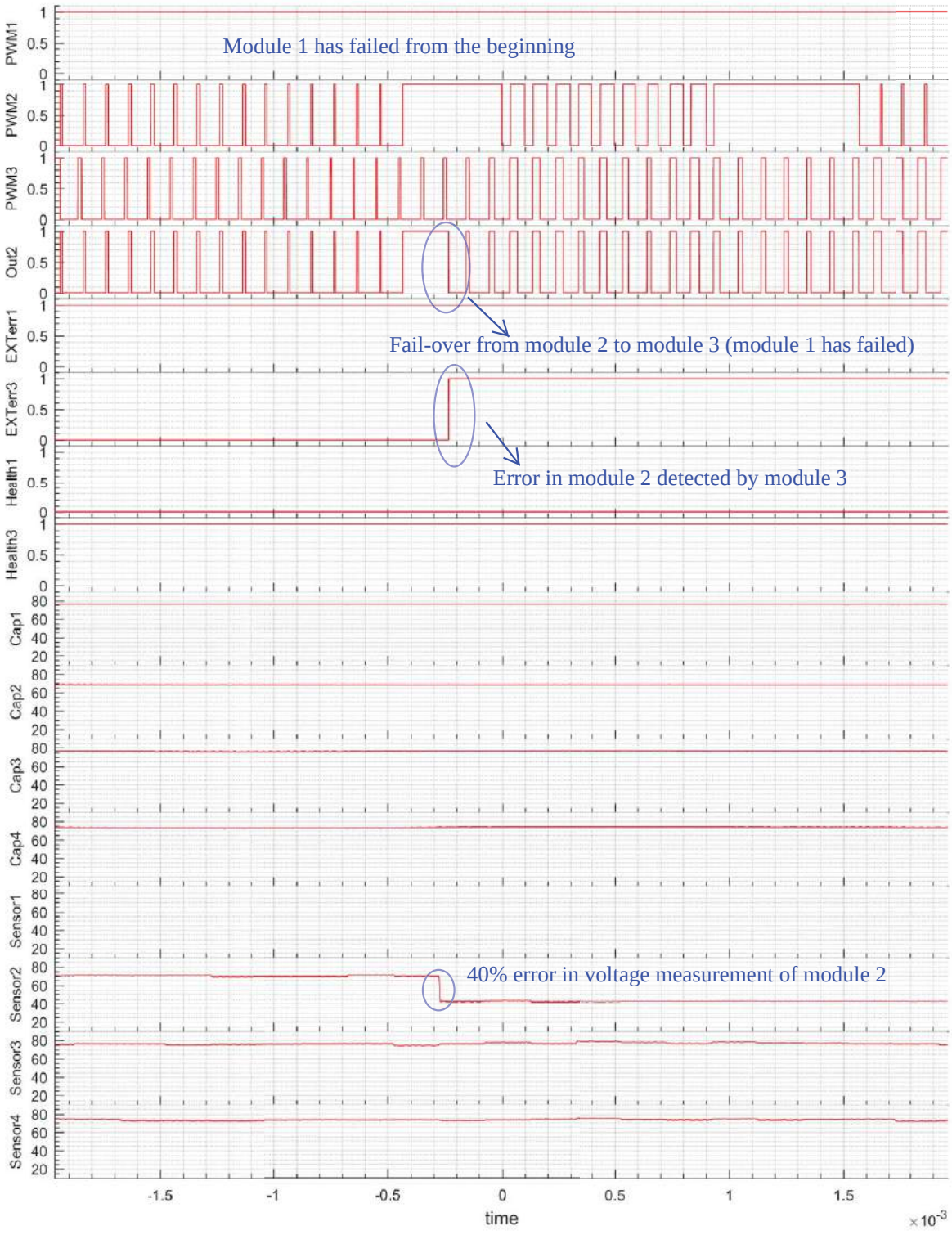


Figure 6.44: Voltage Sensor Failure in Module 3 While Module 1 has failed and Its effect on Module 2 in MMC

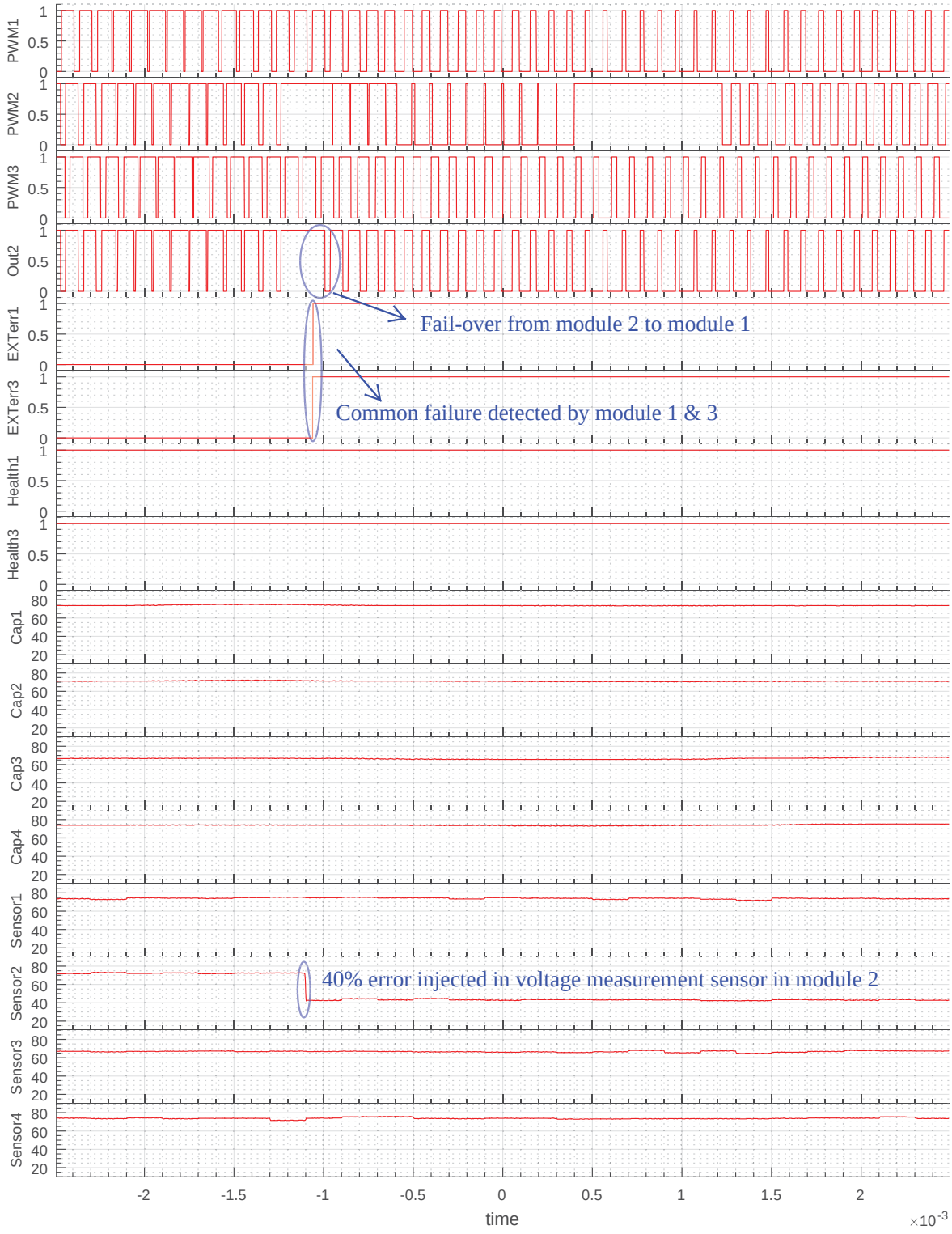


Figure 6.45: Voltage Sensor Failure in Module 2 and Its effect on Module 2 in MMC

6.6 Real Hardware Experimental Result for Cascaded H-bridge Converter Using Fault-tolerant Converter

This section demonstrates experimental results related to the hardware setup of cascaded H-bridge converter (CHB). Figure 6.46 shows the image of the experimental setup. Hardware architecture of the converter is as same as configuration of the CHB in the opal-rt HIL simulation. Fundamental of the controlling CHB has been represented in Appendix A.

Figure 6.46: Experimental Setup of Cascaded H-bridge Converter (CHB) in Connection with Fault-tolerant Controller

Figure 6.47: Side View of Cascaded H-bridge Converter (CHB)

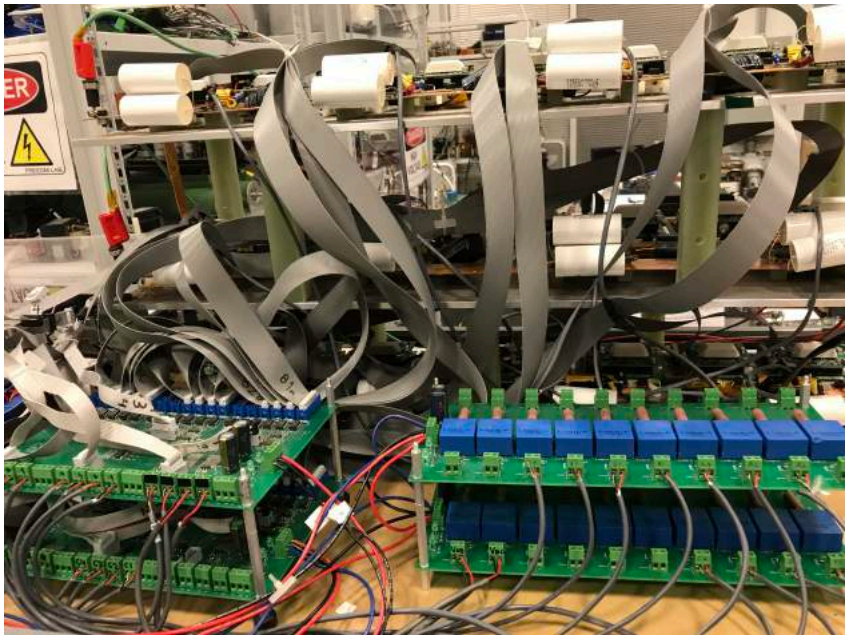


Figure 6.48: Interconnection Between Interface Card, Measurement Board and Cascaded H-bridge Converter (CHB)

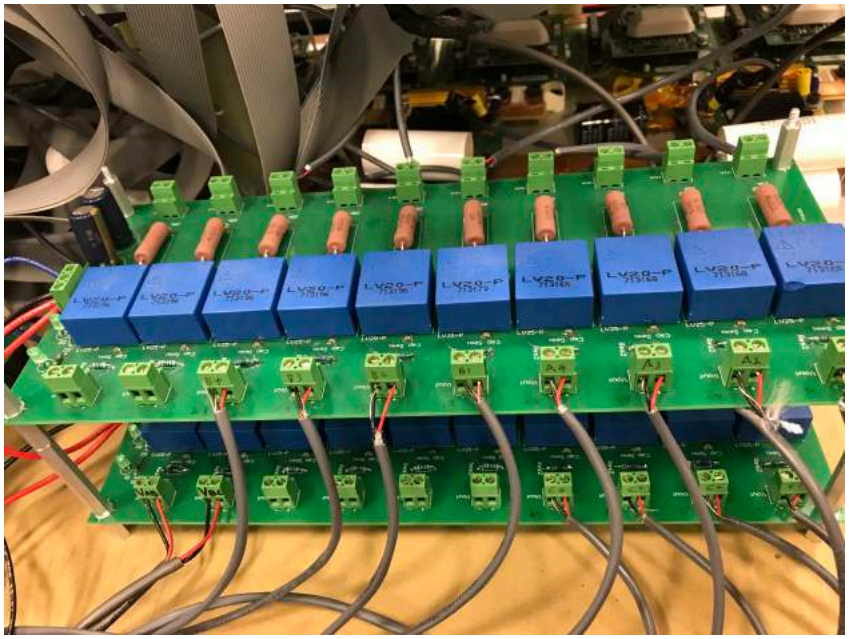


Figure 6.49: Voltage Measurement Board in CHB Setup

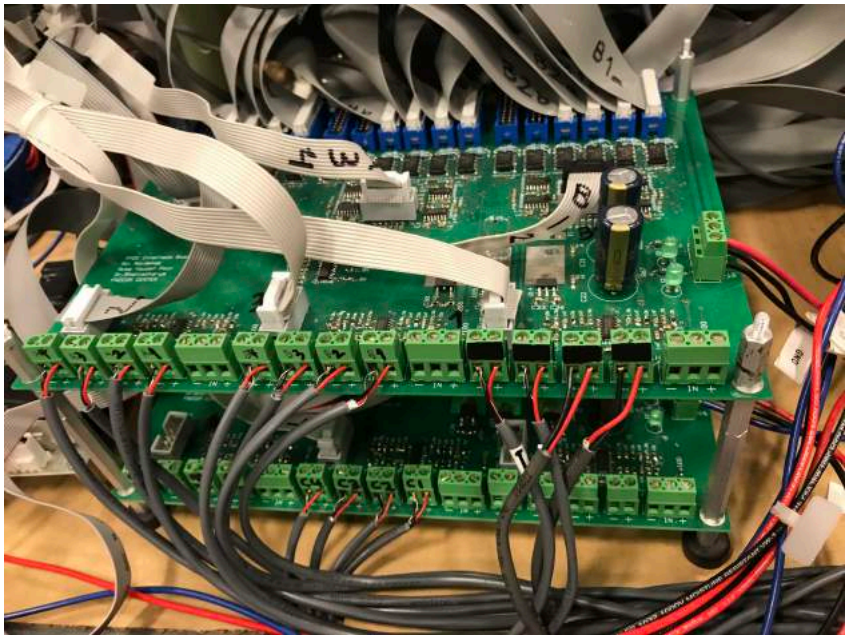


Figure 6.50: Interface Board in CHB Setup

Table 6.3: Setup Configuration for CHB Hardware Experimental Setup

Design Parameter	Symbol	Value
Grid line-line voltage	E	120 (Vrms)
Line frequency	f	60 (Hz)
Series inductance	L_{series}	5 (mH)
Switching frequency	$f_{switching}$	10 (kHz)
Converter module per phase	N	4
Nominal module capacitor voltage	V_{module}	27 (V)
Converter module capacitance	C_{module}	6800 (μF)
Nominal converter current	$I_{converter}$	10 (A)
Load Resistance	R_{load}	420 (Ω)

Table 6.3 demonstrates the specification of the CHB converter setup. It is pretty much the same as the HIL simulation and the same controller firmware (with minor modifications) has been used for achieving the results. Since all the tests have been done in the HIL simulation, the transition process from simulation to real hardware accomplished quickly. All of the results that has been presented are related to the second controller in phase C of the converter system. Results show different case of failures and their effects on the fault handling system in the converter.

Figure 6.51 demonstrates the normal operation of the converter when there is no failure in the system. The first two row is the grid line-to-line voltages, the 3rd and 4th row is the grid currents (Ia and Ib), the 5th row is the DC bus voltage, 6th row is the active current which is drawn from grid and row 7 to 10 are the capacitor voltages in the third phase of the converter.

Figure 6.52 demonstrates converter operation in normal condition. All the results are the same as the previous figure, however row 5 is the PLL measured phase and row 6 is the voltage of the phase A.

Figure 6.53 demonstrates failure in communication link of the controller 2. In this case, the failure is common between adjacent controllers and it cause fail-over from the failed controller (second controller).

Figure 6.54 demonstrates failure in communication link of the controller 3. In this case, the failure is not common and it cause to fail-over in the system.

Figure 6.55 shows the power supply failure in the first controller. Since the failure is not happening in second controller and it is not a common failure, no fail-over happens.

Figure 6.56 shows the power supply failure in controller 2. In this case, failure is a common fault and controller 2 is bypassed by the adjacent controller.

Figure 6.57 shows the case in which controller 3 has been reset. This case emulates the case that software is being rejuvenated. Since it is not a common failure in the system, no fail-over happens in the system.

Figure 6.58 shows the case in which controller 2 has been reset. This case emulates the case that software is being rejuvenated. It is a common failure in the system and it would be bypassed by the adjacent controllers.

Figure 6.59 presents the case that voltage measurement sensor in the first controller has been failed. Failure has been emulated with 40 percent difference in the measured signal. Since it is not a common failure, controller 2 is not bypassed.

Figure 6.60 presents the case that voltage measurement sensor in the controller 2 has been failed. Failure has been emulated with 40 percent difference in the measured signal. It is a common failure in the system and cause bypassing controller 2 and fail-over to the first controller.

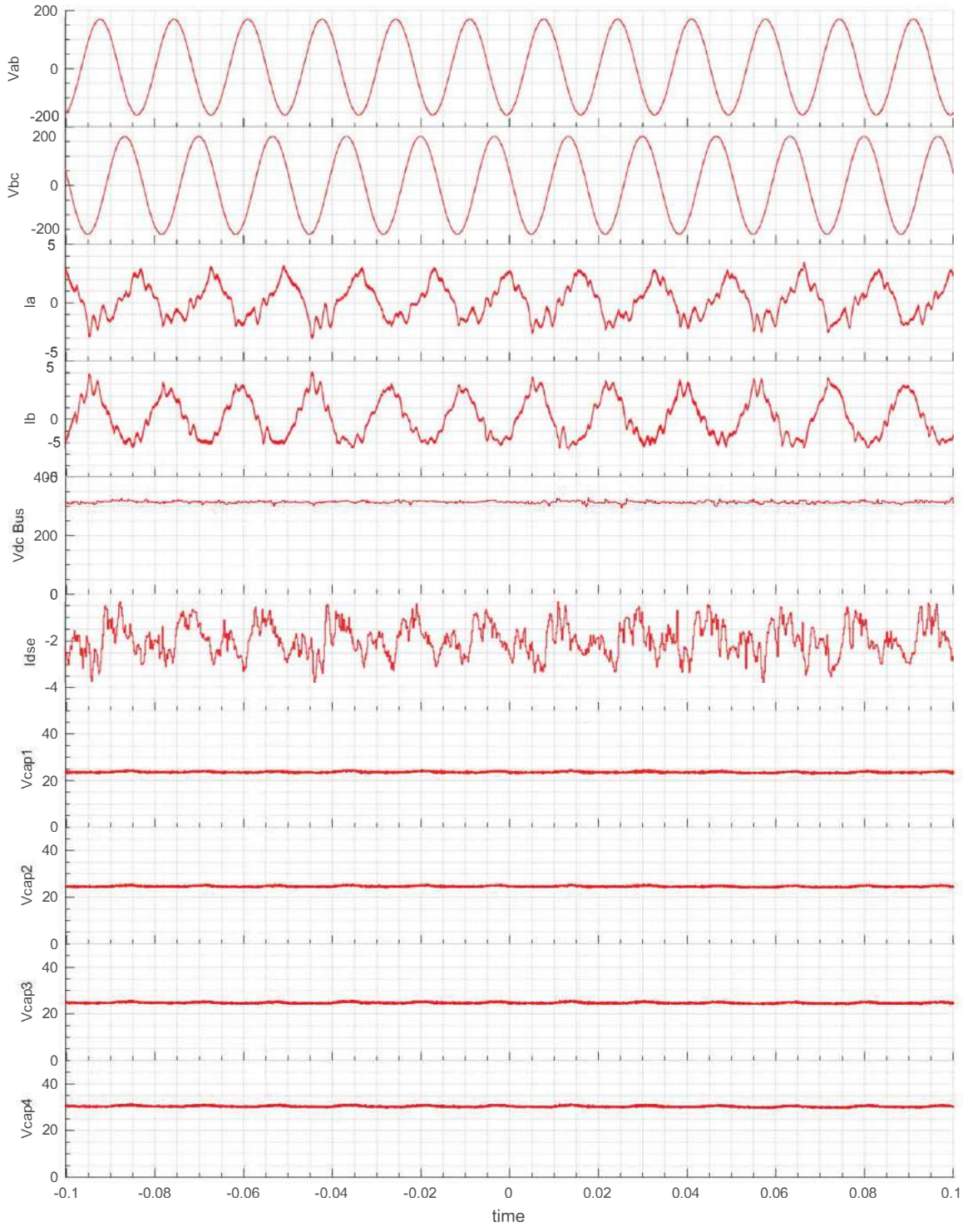


Figure 6.51: CHB in Normal Operation (DC Bus Voltage and Reactive Current are Shown)

Figure 6.52: CHB in Normal Operation (PLL Output is Shown)

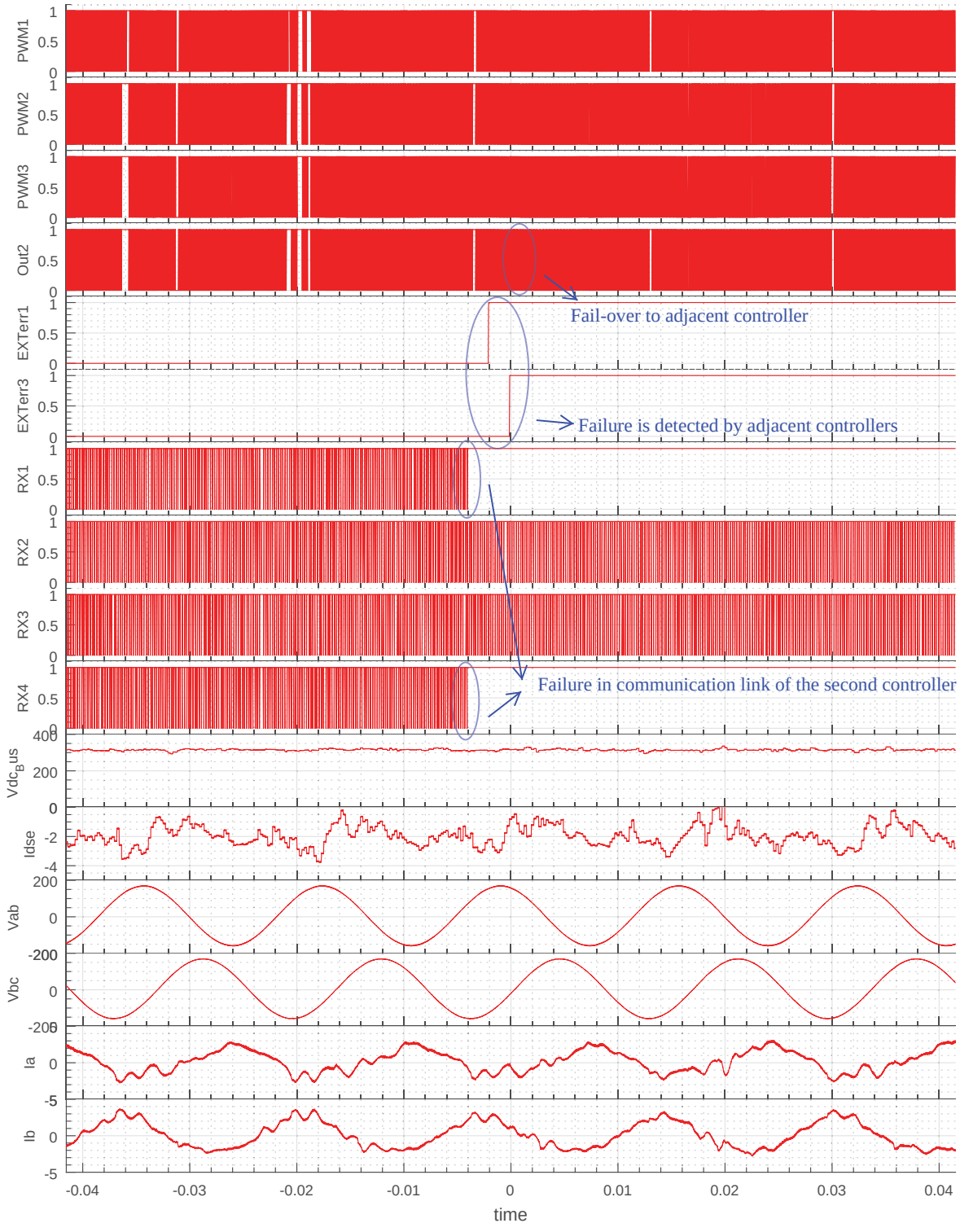


Figure 6.53: Experimental Result of Communication Failure in Controller 2 and its Effect on the System

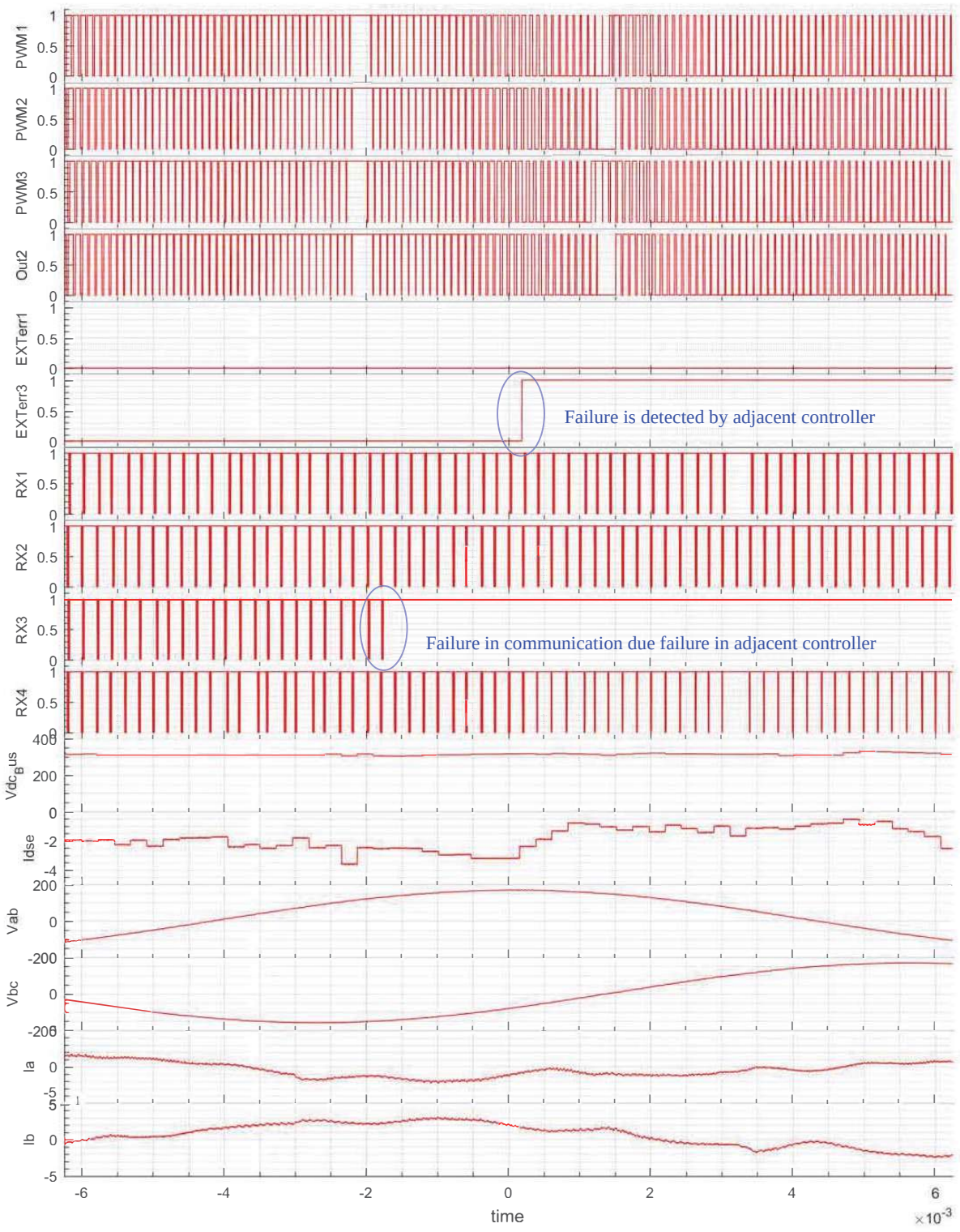


Figure 6.54: Experimental Result of Communication Failure in Controller 3 and its Effect on the System

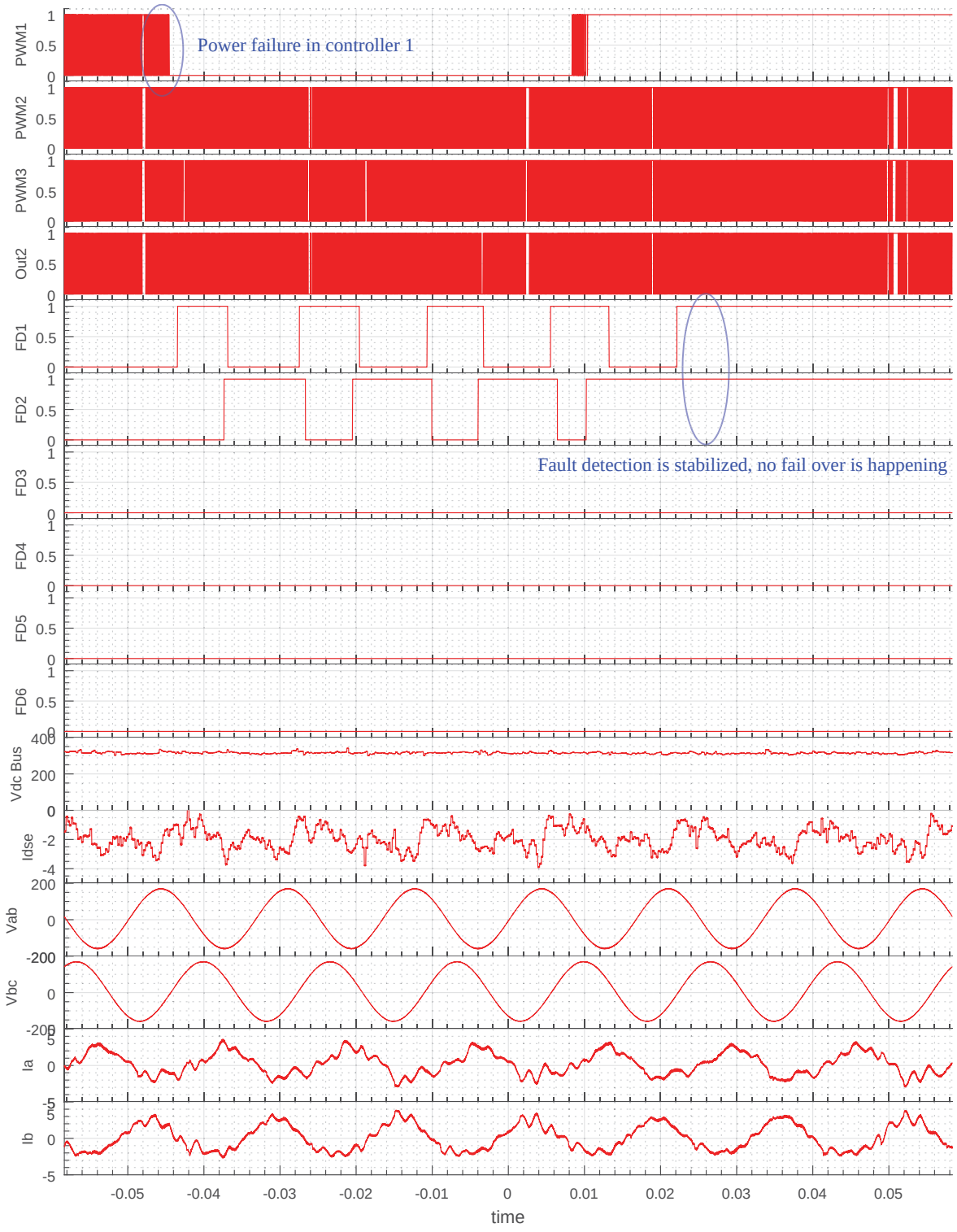


Figure 6.55: Experimental Result of Power Supply Failure in Controller 1 and its Effect on the System

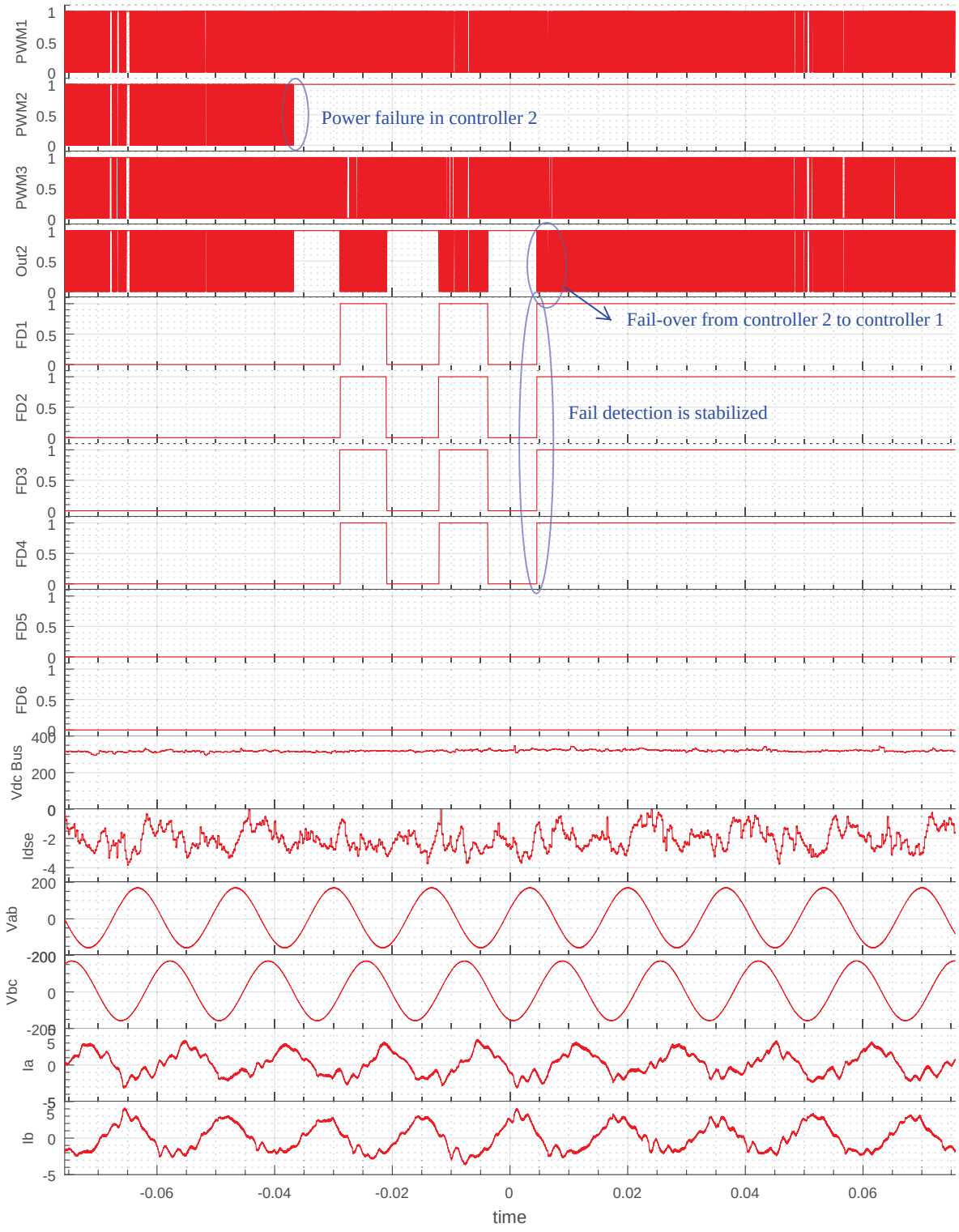


Figure 6.56: Experimental Result of Power Supply Failure in Controller 2 and its Effect on the System

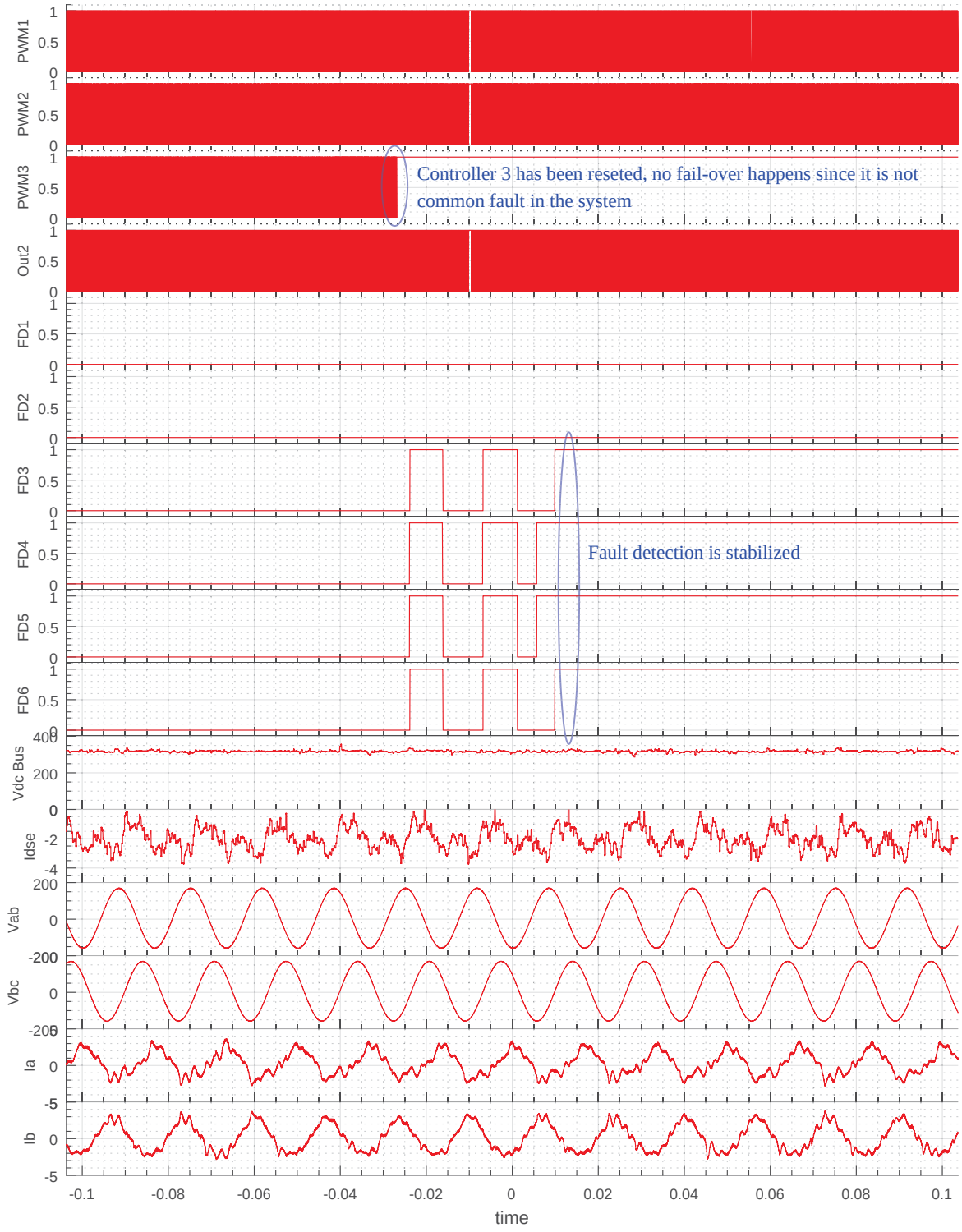


Figure 6.57: Experimental Result of Microprocessor Reset in Controller 3 and its Effect on the System

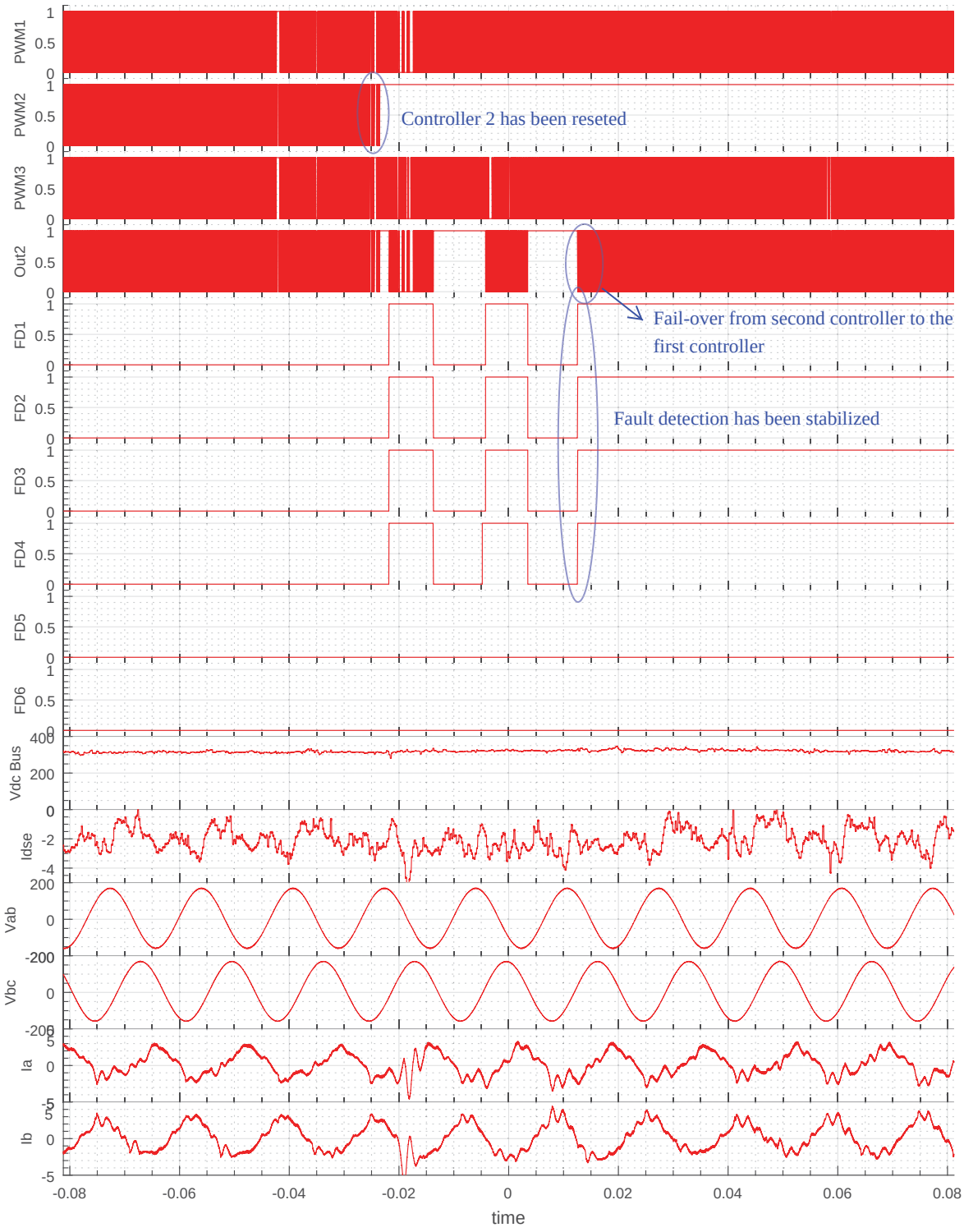


Figure 6.58: Experimental Result of Microprocessor Reset in Controller 2 and its Effect on the System

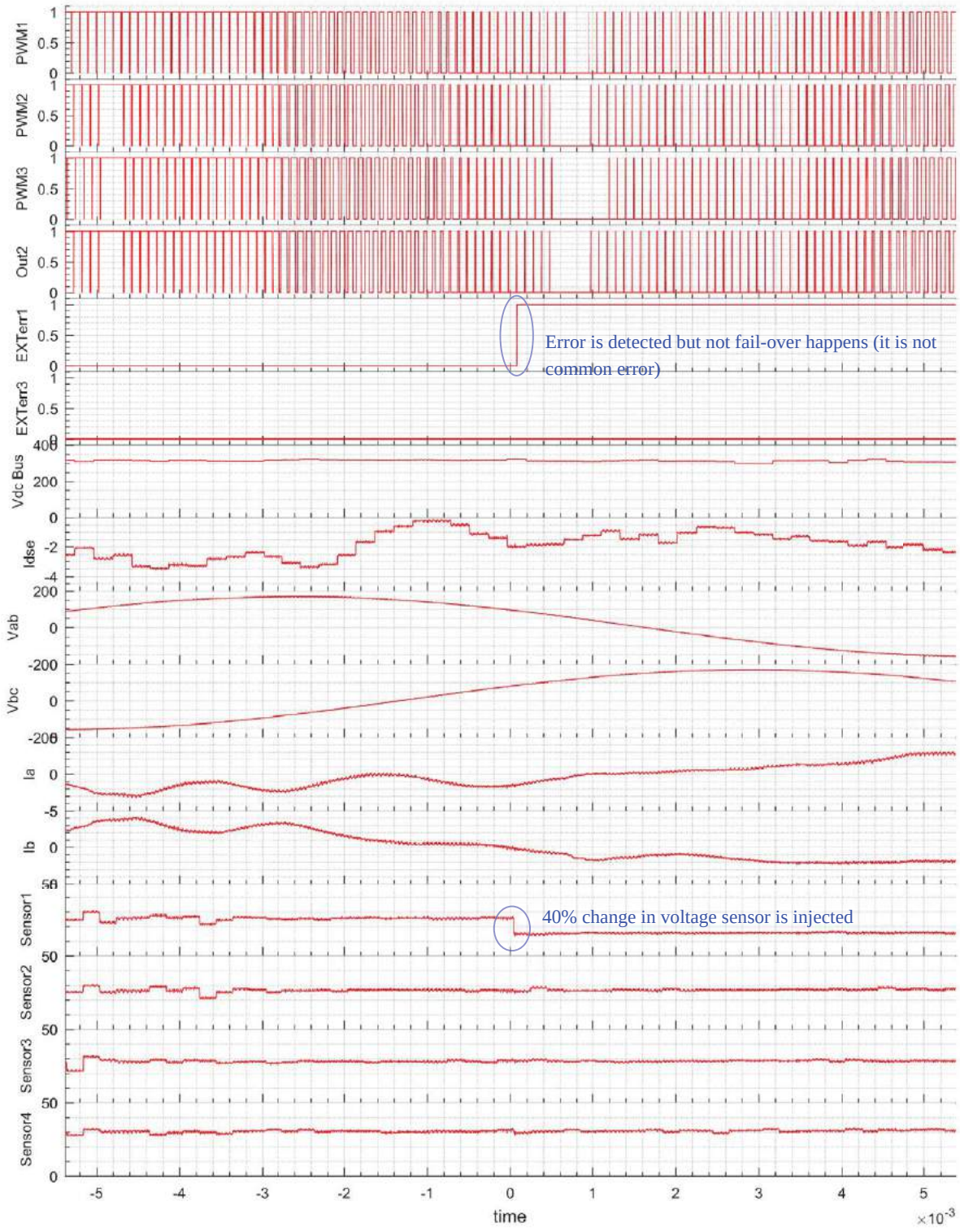


Figure 6.59: Experimental Result of Voltage Measurement Sensor Failure in Controller 1 and its Effect on the System

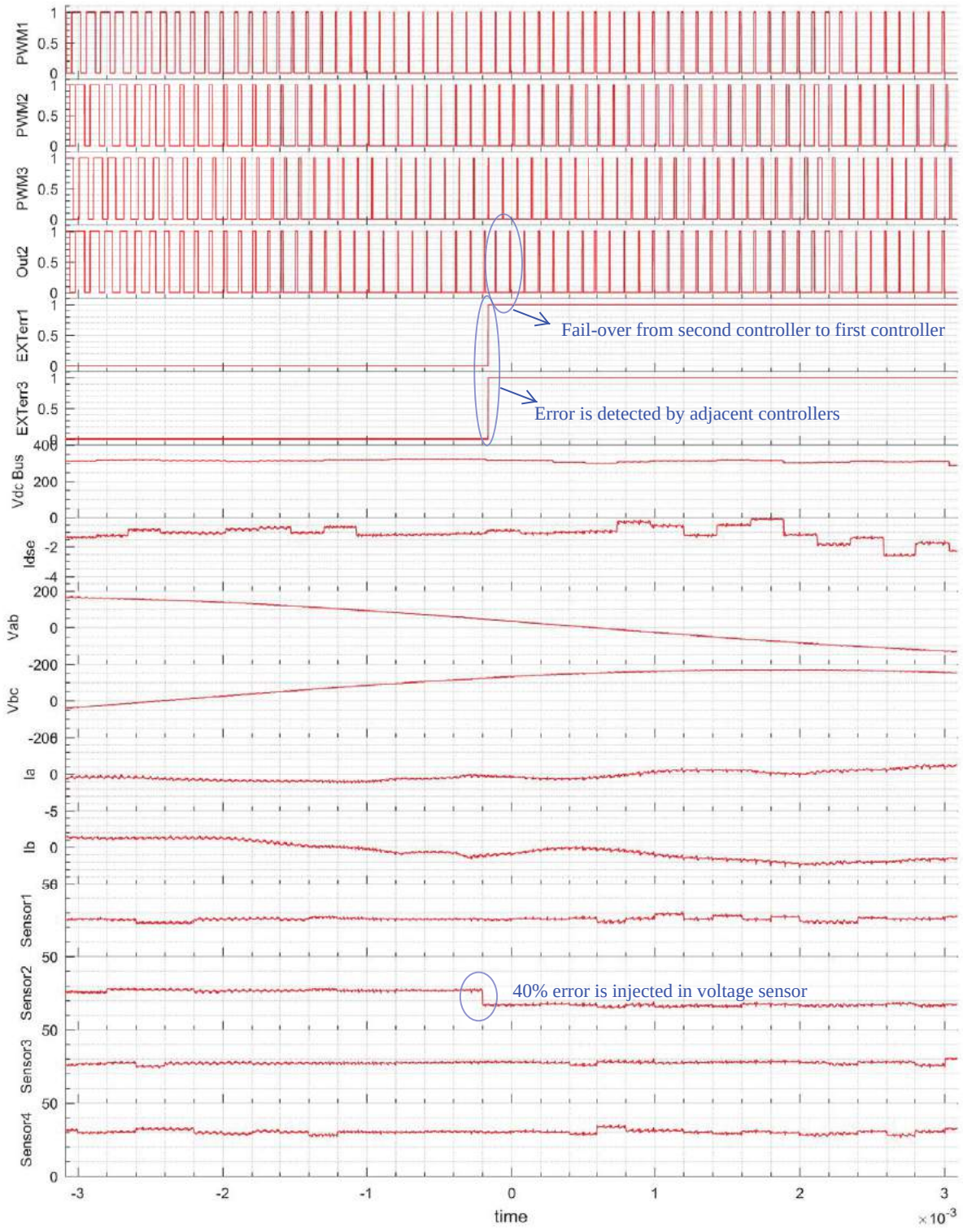


Figure 6.60: Experimental Result of Voltage Measurement Sensor Failure in Controller 2 and its Effect on the System

Chapter 7

Conclusion and Future Works

7.1 Conclusions

- Although implementing small multi-level converters is possible using a single converter, as number of modules increase there is no possibility of controlling the whole converter system with a single controller. In this case, controllers must be distributed among the converters and all of them must be synchronized to a master controller.
- The other benefit that distributed control can provide is reconfiguration of the control blocks. In the event of failure, modules can be re-arranged and the failed module can be bypassed.
- The second generation fault-tolerant controller proposes fault-tolerancy in the network architecture. By using a grid of controllers, it is possible to bypass failed master controller and networks link which are not functioning anymore. This architecture is more complex but it gives ultimate resiliency in modular multi-level converters.
- Simulation results show that the proposed controller is suitable for hardware implementa-

tion. The same control algorithms can be implemented in hardware controller to get the same results.

- Reliability assessment result provides good result for the proposed controller in comparison with the single controller method. These assessment may also be improved by including more details of system operation in computer model.

- Fault-tolerancy must be applied to the software design too. It is common thinking that software is already broken, therefore fault avoidance techniques may lower the probability of failure in the firmware.

- Experimental result approves the feasibility of the controller architecture that it can isolate faults in the system with the best dynamics.

- Experimental result on cascaded H-bridge converter (CHB) and modular multi-level converter (MMC) approves that the proposed fault-tolerant controller may be used for vast majority of multi-level converters with slight modification in the control structure for the master and slave controllers.

7.2 Proposed Future Works

- The next big step in the fault-tolerant controller for MMC is to complete the research and implementation of the second generation fault-tolerant controller. The same hardware test-bed may be used to implement the controller and more simulation result may be gathered to approve the functionality of the system.
- The proposed controller has been used for CHB and MMC converters. It is a good idea to use the controller for other types of multi-level converters or converter topologies like matrix converter that consist of several modules.
- Reliable firmware design is desired in the industry. There is a huge subjects that may be researched in this field with application in the proposed controller architecture.
- Hardware implementation of the controller in CHB was accomplished by the author. It is also possible to implement the same results with modular multi-level converter (MMC).
- Reliability assessment of the controller has been accomplished using Markov chain process and software simulation. It is a good idea to do real non-destructive tests and record data in long time using hardware in the loop (HIL) simulation.

References

[1] Mil-hdbk-217f, reliability prediction of electronic equipment.

[2] *www.infineon.com*.

[3] H. Akagi. Classification, terminology, and application of the modular multilevel cascade converter (mmcc). *IEEE TRANSACTIONS ON POWER ELECTRONICS*, 2011.

[4] N. Amenta. Proof for dijkstra algorithm.

[5] Multiple authors. Power blackout risks. *CRO forum*, 2011.

[6] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 1, NO. 1*, 2004.

[7] M.H. Azadmanesh and R.M. Kieckhafer. Exploiting omissive faults in synchronous approximate agreement. *IEEE TRANSACTIONS ON COMPUTERS*, 2000.

[8] A. Azidehak. Synchronization and architectures of distributed controllers in advanced modular multi-level converters. Master's thesis, North Carolina State University, 2014.

[9] H. S. Bø. Estimation of reliability by monte carlo simulations. Master's thesis, Norwegian University of Science and Technology, 2014.

[10] J. Bartlett, W. Bartlett, R. Carr, D. Garcia, J. Gray, R. Horst, R. Jardine, D. Lenoski, and D. McGuire. *Fault Tolerance in Tandem Computer Systems*. Tandem Computers, 1990.

[11] W. Bartlett and L. Spainhower. Commercial fault tolerance: A tale of two systems. *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, 1, JANUARY-MARCH 2004.

[12] D. Bernick, B. Bruckert, P. D. Vigna, D. Garcia, R. Jardine, J. Klecka, and J. Smullen. Nonstop advanced architecture. *Hewlett Packard Company*, 2005.

[13] S. Bhattacharya. Resilient multi-terminal hvdc networks with high-voltage high-frequency electronics. Technical report, North Carolina State University, 2015.

[14] J. R. Black. Electromigration-a brief survey and some recent results. *IEEE TRANSACTIONS OF ELECTRON DEVICES*, April 1969.

[15] C. Bron and J. Kerbosch. Algorithm 457-finding all cliques of an undirected graph. *Comm of ACM*, 1973.

[16] Alan Burns. *Real-Time Systems and Programming Languages*. Addison-Wesley Pub. Co, 1990.

[17] R. W. Butler. A primer on architectural level fault tolerance. Technical report, NASA, 2008.

[18] R. W. Butler and S. C. Johnson. Techniques for modeling the reliability of fault-tolerant systems with the markov state-space approach. *National Aeronautics and Space Administration*, 1995.

[19] L. Chen and A. Avizienis. N-version programming: A fault-tolerance approach to reliability of software operation. In *Proceedings of FTCS-25*, 1996.

[20] T. Cuatto, C. Passerone, C. Sansoè, F. Gregoretti, A. Jurecska, and A. Sangiovanni-Vincentelli. A case study in embedded systems design: An engine control unit. *Design Automation for Embedded Systems*, 2000.

[21] M. Dagbagi, L. Idkhajine, E. Monmasson, and I. Slama-Belkhodja. Fpga implementation of power electronic converter real-time model. In *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, 2012.

[22] W. Denson. The history of reliability prediction. *IEEE TRANSACTIONS ON RELIABILITY*, 1998.

[23] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik 1*, 1959.

[24] R. R. Errabelli and P. Mutschler. A fault tolerant digital controller for power electronic applications. In *Power Electronics and Applications (EPE 2011)*, 2011.

[25] G. Falahi and A. Huang. Low voltage ride through control of modular multilevel converter based hvdc systems. In *IEEE IECON*, 2014.

[26] Seyed Saeed Fazel. *Investigation and Comparison of Multi-Level Converters for Medium Voltage Applications*. PhD thesis, Technischen Universität Berlin, 2007.

[27] P. Parkinson FIET and L. Kinnan. Safety-critical software development for integrated modular avionicson. Technical report, WIND River Systems, 2015.

[28] M. D. Galanis, G. Dimitroulakos, and C. E. Goutis. Speedups from partitioning critical software parts to coarse-grain reconfigurable hardware. In *16th International Conference on Application-Specific Systems, Architecture and Processors (ASAPŠ05)*, 2005.

[29] F. Gao, D. Niu, H. Tian, C. Jia, N. Li, and Y. Zhao. Control of parallel-connected modular multilevel converters. *IEEE TRANSACTIONS ON POWER ELECTRONICS, VOL. 30, NO. 1*, 2015.

[30] Z. Gao, C. Cecati, and S. X. Ding. A survey of fault diagnosis and fault-tolerant techniques-part i: Fault diagnosis with model-based and signal-based approaches. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 62, NO. 6*, 2015.

[31] Z. Gao, C. Cecati, and S. X. Ding. A survey of fault diagnosis and fault-tolerant techniques-part ii: Fault diagnosis with knowledge-based and hybrid/active approaches. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 62, NO. 6*, 2015.

[32] Z. Gao, S. X. Ding, and C. Cecati. Real-time fault diagnosis and fault-tolerant control. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 62, NO. 6*, 2015.

[33] M. Glabowski, B. Musznicki, P. Nowak, and P. Zwierzykowski. Efficiency evaluation of shortest path algorithms. In *The Ninth Advanced International Conference on Telecommunications*, 2013.

[34] B. Golden. Shortest-path algorithm: A comparison. *Operation research*, 1976.

[35] J. Gray. Why do computers stop and what can be done about it? *In Proc. of 5th Symposium on Reliability in Distributed Software and Database Systems*, 1986.

[36] M. Grottke, R.Matias Jr, and K. S. Trivedi. Fundamentals of software aging. In *International workshop on software aging and rejuvenation*, 2008.

[37] M. Hagiwara and H. Akagi. Pwm control and experiment of modular multilevel converters. In *Power Electronics Specialists Conference*, 2008.

[38] M. Hagiwara and H. Akagi. Control and experiment of pulsewidth-modulated modular multilevel converters. *IEEE Transactions on Power Electronics*, 2009.

[39] M. Hagiwara, K. Nishimura, and H. Akagi. A modular multilevel pwm inverter for medium-voltage motor drives. In *Energy Conversion Congress and Exposition*, 2009.

[40] R. Horst, D. Jewett, and D. Lenoski. The risk of data corruption in microprocessor-based systems. *IEEE*, pages 576–585, 1993.

[41] Y. Huang, P. Jalote, and C. Kintala. Two techniques for transient software error recovery. *Lecture Notes in Computer Science*, 1994.

[42] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton. Software rejuvenation: Analysis, module and applications. *IEEE*, 1995.

[43] N. Jaalam, N.A. Rahim, A.H.A Bakar, C. Tan, and A.M.A Haidar. Renewable and sustainable energy reviews. *Elsevier Renewable and Sustainable Reviews*, 2016.

[44] S. N. Jean. *Mapping/Matching Algorithms to Reconfigurable Mesh Arrays*. PhD thesis, University of Southern California, 1988.

[45] S. N. Jean and S. Y. Kung. Necessary and sufficient conditions for reconfigurability in single-track switch wsi arrays. In *International Conference on Wafer Scale Integration*, 1989.

[46] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods*. Wiley, 2008.

[47] R. Kapoor and M. Sushama. Sensor fault detection for a repetitive controller based d-fact device. In *International Conference on Control, Automation, Robotics and Embedded Systems*, 2013.

[48] J. A. Katzman. The tandem 16: A fault-tolerant computing system. In *Proceedings of the 11th Hawaii Conference. on System Sciences (11th HICSS'78)*, pages 85–102, Honolulu, Hawaii,, 1978. IEEE Computer Society.

[49] M.F.R. Keuning. Software partitioning for safety-critical airborne systems in practice. Technical report, National Aerospace Lab (NLR), 2000.

[50] William Kocay and Donald L. Kreher. *Graphs, alogorithms, and optimization*. Chapman and hall/crc, 2005.

[51] H. Kopetz. Clock synchronization in distributed real time systems. *IEEE TRANSACTIONS ON COMPUTERS*, 1987.

[52] T. Krone, C. Xu, and A. Mertens. Fast and easily implementable detection circuits for short-circuits of power semiconductors. In *IEEE Energy Conversion Congress and Exposition (ECCE)*, 2015.

[53] S.Y. KUNG, S.N. JEAN, and C.W. CHANG. Fault-tolerant array processors using single-track switches. *IEEE TRANSACTIONS ON COMPUTER*, 1989.

[54] K. H. Lacommare and J. H. Eto. Cost of power interruptions to electricity consumers in the united states (u.s.). *Berkeley National Laboratory*, 2006.

[55] L. LAMPORT, R.SHOSTAK, and M. PEASE. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 1982.

[56] L. Langenhop. Embedded security analysis for an engine control unit architecture. Master's thesis, Christian-Albrechts-Universität zu Kiel, 2015.

[57] I. Lee and R. K. Iyer. Software dependability in the tandem guardian system. *IEEE Trans. on Software Engineering, pages 455-467, Vol. 21, No. 5*, 1995.

[58] A. Lesnicar and R. Marquardt. An innovative modular multilevel converter topology suitable for a wide power range. In *Power Tech Conference Proceedings*, 2003.

[59] M. Levesque and D.Tipper. A survey of clock synchronization over packet-switched networks. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 2016.

[60] F.L. Lian, J. Moyne, and D. Tilbury. Network design consideration for distributed control systems. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, VOL. 10, NO. 2:297–306, MARCH 2002.

[61] Y.i Lin, Y. Li, and E. Zio. A reliability assessment framework for systems with degradation dependency by combining binary decision diagrams and monte carlo simulation. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS*, 2016.

[62] B. Littlewood and J. Rushby. Reasoning about the reliability of diversetwo-channel systems in which one channel is "possibly perfect". *IEEE Transactions on Software Engineering*, 2012.

[63] H. Liu, P. C. Loh, and F. Blaabjerg. Review of fault diagnosis and fault-tolerant control for modular multilevel converter of hvdc. In *Industrial Electronics Society, IECON - Annual Conference of the IEEE*, 2013.

[64] F. Lombardi, M. G. Sami, and R. Stefanelli. Reconfiguration of vlsi arrays by covering. *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN*, 1989.

[65] D. Majstorovic, I. Celanovic, N. Dj. Teslic, N. Celanovic, and V. A. Katic. Ultralow-latency hardware-in-the-loop platform for rapid validation of power electronics designs. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 2011.

[66] R. Marquardt. Modular multilevel converter topologies with dc-short circuit current limitation. In *Power Electronics and ECCE Asia (ICPE & ECCE)*, 2011.

[67] A. Mason and V. Rytchenkov. Fiber optic repeater with compensation of pulse width distortion. In *Nuclear Science Symposium and Medical Imaging Conference Record*, 1995.

[68] M. Matar and R. Iravani. Fpga implementation of the power electronic converter model for real-time simulation of electromagnetic transients. *IEEE TRANSACTIONS ON POWER DELIVERY*, 2010.

[69] T.C. May and M.H. Woods. Alpha-particle-induced soft errors in dynamic memories. *IEEE Trans. on Electron Devices*, 1979.

[70] J.W. McPherson. Time dependent dielectric breakdown physics Ű models revisited. In *SPECIAL ISSUE 23rd EUROPEAN SYMPOSIUM ON THE RELIABILITY OF ELECTRON DEVICES, FAILURE PHYSICS AND ANALYSIS*, SeptemberŰOctober 2012.

[71] R. M.Mattox and J. B. White. Space shuttle main engine controller. *Nasa technical paper 1932*, 1981.

[72] Mul. *MISRA C:2012 Guidlines fo the use of the C language in critical Systems*. MIRA Limited, 2013.

[73] H. Niemann, J. Stoustrup, and N. K. Poulsen. Controller modification applied for active fault detection. In *American Control Conference (ACC)*, 2014.

[74] E. Normand. Single event upset at ground level. *IEEE TRANSACTIONS ON NUCLEAR SCIENCE*, 1996.

[75] M. A. Parker, L. Ran, and S. J. Finney. Distributed control of a fault-tolerant modular multilevel inverter for direct-drive wind turbine grid interfacing. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 60, NO. 2*, 2013.

[76] K. M. PASSINO and P. J. ANTSAKLIS. Fault detection and identification in an intelligent restructurable controller. *Journal of Intelligent and Robotic Systems*, 1988.

[77] P. PejoviC and D. MaksimoviC. A method for fast time-domain simulation of networks with switches. *IEEE TRANSACTIONS ON POWER ELECTRONICS*, 1994.

[78] J. Pukite and P. Pukite. *Modeling for reliability analysis*. IEEE press, 1998.

[79] M. Raynal and M. Singhal. Mastering agreement problems in distributed systems. *IEEE SOFTWARE*, 2001.

[80] D. A. RENNELS. Architectures for fault-tolerant spacecraft computers. *PROCEEDINGS OF THE IEEE*, 66:1255–1268, 1978.

[81] G. K. Saha. Software fault avoidance. 2011.

[82] A. K. Samal, A. K. Parida, S. K. Pani, and A. K. Dash. A novel fault-tolerant scheduling of real-time tasks on multiprocessor using discrete-elitist multi-aco. In *IEEE ICCSP conference*, 2015.

[83] M. Sami and R. Stefanelli. Reconfigurable architectures for vlsi processing arrays. *Proceedings of the IEEE (Volume:74 , Issue: 5)*, 1986.

[84] D. K. Schroder. Negative bias temperature instability: What do we understand? *Microelectronics Reliability*, December 2006.

[85] RL Sellick and M Åkerberg. Comparison of hvdc light (vsc) and hvdc classic (lcc) site aspects, for a 500mw 400kv hvdc transmission scheme. In *IET ACDC Conference*, 2012.

[86] S. Z. Shazli, M. Abdul-Aziz, M. B. Tahoori, and D. R. Kaeli. A field failure analysis of microprocessors used in information systems. Technical report, Northeastern University, 2009.

[87] K. Shimamura, Y. Morita, Y. Takahashi, T. Hotta, S. Ueda, M. Nohara, M. Kido, S. Tanaka, K. Imaie, K. Sakamoto, and T. Nakajima. A triple redundant controller which adopts the time-sharing fault recovery method and its application to a power converter controller. In *Real-Time Technology and Applications Symposium*, 1998.

[88] J. R. Sklaroff. Redundancy management technique for space shuttle computers. *IBM J.Res Develop*, pages 20–28, 1976.

[89] T. L. SKVARENINA, editor. *The power electronics handbook*. CRC Press, 2002.

[90] J. Sneyers, T. Schrijvers, and B. Demoen. DijkstraŠs algorithm with fibonacci heaps: An executable description. In *In 20th Workshop on Logic Programming*, 2006.

[91] L. Snyder. Introduction to the configurable, highly parallel computer. *IEEE Journal of Computer*, 1982.

[92] J. Specht. Terminology proposal: Redundancy for fault tolerance. Technical report, University of Duisburg-Essen, 2013.

[93] W. Steiner and H. Kopetz. The startup problem in fault-tolerant time-triggered communication. In *International Conference on Dependable Systems and Networks*, 2006.

[94] N. Storey. *Safety-critical Computer Systems*. Addison-wesley, 1996.

[95] T.J. Summers, R.E. Betz, and G Mirzaeva. Phase leg voltage balancing of a cascaded h-bridge converter based statcom using zero sequence injection. In *13th European Conference on Power Electronics and Applications*, 2009.

[96] M. Takeshi. A monte carlo simulation method for system reliability analysis. *Nuclear Safety and Simulation*, 2013.

[97] G. Tang, Z. Xu, and Y. Zhou. Impacts of three mmc-hvdc configurations on ac system stability under dc line faults. *IEEE TRANSACTIONS ON POWER SYSTEMS*, 2014.

[98] T.Kailath and A. Paulraj. Algorithms and architectures for high speed signal processing. Technical report, Stanford University, 1993.

[99] J. E. Tomayko. Computers in spaceflight the nasa experience. Technical report, NASA History Office, 1988.

[100] K. S. Trivedi and K. Vaidyanathan. Software rejuvenation - modeling and analysis. Technical report.

[101] K. Vaidyanathan and K. S. Trivedi. Extended classification of software faults based on aging. 2001.

[102] N. A. Valentim, A. Macedo, and R. Matias Jr. A systematic mapping review of the first 20 years of software aging and rejuvenation research. In *IEEE 27th International sysposium on software reliability engineering workshops*, 2016.

[103] T. Varvarigou, V. P.Roychowhury, and T. Kailath. Some new algorithms for reconfiguring vlsi/wsi arrays. In *Wafer Scale Integration*, 1990.

[104] Ben L. Di Vito. A formal model of partitioning for integrated modular avionics. Technical report, NASA, 1998.

[105] V.P.Roychowdhury, J.Bruck, and T. Efficient algorithms for reconfiguration in vlsi/wsi arrays. *IEEE TRANSACTIONS ON COMPUTERS*, 1990.

[106] A. Webber. Calculating useful lifetimes of embedded processors. Technical report, Texas Instruments, Nov 2014.

[107] T. Yoshii, S. Inoue, and H. Akagi. Control and performance of a medium-voltage transformerless cascade pwm statcom with star-configuration. In *Industry Applications Conference*, 2006.

[108] N. Yousefpoor. *Control of Advanced Power Converter Topologies for Transmission Grid Management*. PhD thesis, North Carolina State University, 2014.

[109] N. Yousefpoor, B. Parkhideh, A. Azidehak, S. Kim, and S. Bhattacharya. Control of high-frequency isolated modular converter. *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, VOL. 51, NO. 6*, 2015.

[110] J. Zhao, Y. Wang, G. Ning, K. S. Trivedi, R.Matias Jr, and K. Cai. A comprehesive approach to optimal software rejuvenation. *Elsevier*, 2013.

Appendices

Appendix A

Distributed Control Stages for Cascaded H-bridge Multi-level Converter

There are different architectures for multi-level converters and cascaded H-bridge converter with high-frequency isolation at output stage is one of the methods to implement a multi-level converter (figure A.1)[109]. In this converter, each leg consists of several module connected in series together to increase the maximum breakdown voltage of the leg. A high-frequency isolated DC/DC converter isolates the DC voltage of the H-bridge capacitor from grid voltage (figure A.2). It can transfer power bidirectionally with transfer ratio of 1:1. Since the operating frequency of the transformer is high, the core size can be reduced and the total weight of the system would be less. Since capacitors are floating, their voltages must be balanced by the controller. Control procedure for this converter consist of four layers. The first three layers are implemented in the master controller and the last layer is implemented on the slave controllers.

- **DC terminal voltage controller**:this controller is responsible for the high voltage DC side. The input is the DC grid voltage and the output is dq-current that converter must supply. Since it is the highest level controller, it must have slow response time to make

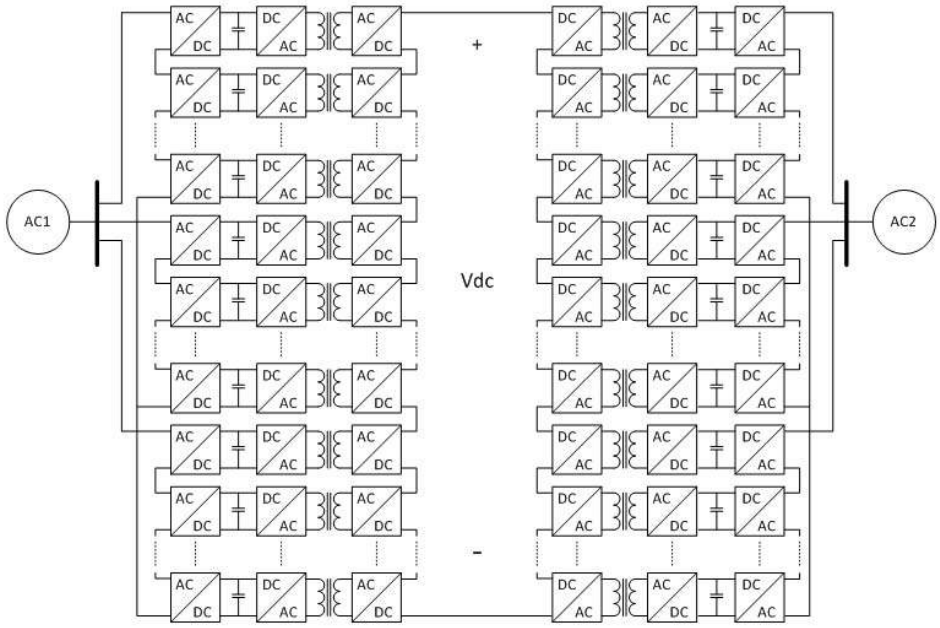


Figure A.1: Cascaded H-bridge Multi-level Converter with Isolated Output Stage

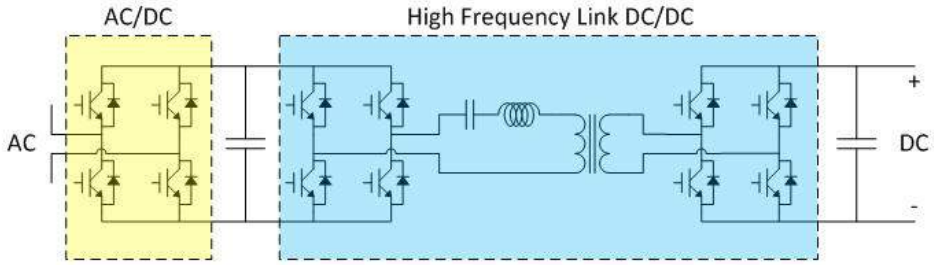


Figure A.2: Detailed Schematic of Converter Modules in Cascaded H-bridge Multi-level Converter

the inertia of the controller high.

- **Current controller**: Unlike the terminal voltage controller, current controller has higher controller response speed and therefore, its inertia is much less. The decoupling factor has been included in the control and third harmonics has been injected to the output voltage to get the highest rail to rail output voltage in the linear region.

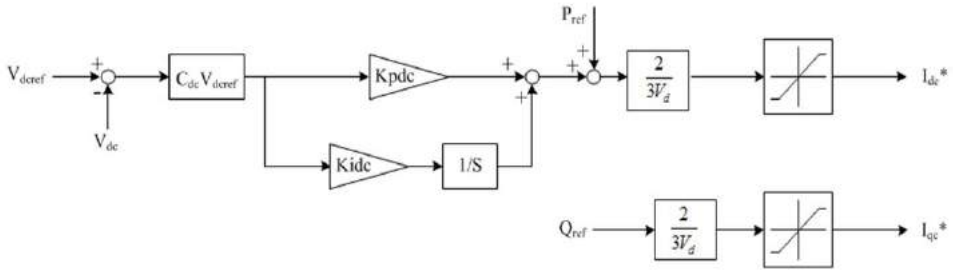


Figure A.3: DC Grid Voltage Controller for the Converter

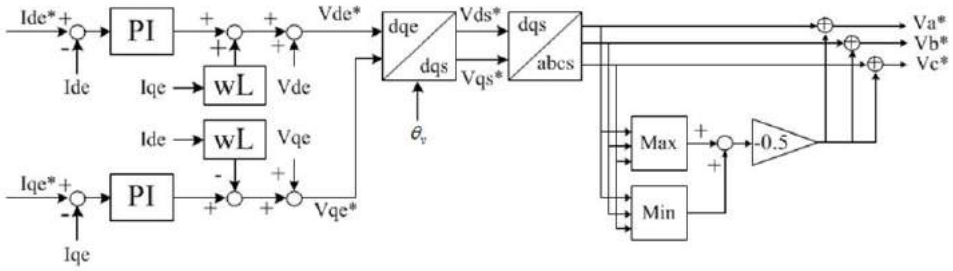


Figure A.4: Current Controller for Cascaded H-bridge Converter

- **Phase voltage balancing**: It is important to have balanced voltages on DC capacitors of the converters and therefore have balanced voltages on each phase of the converter. Phase voltage balance controllers achieve this by adding a zero sequence to the modulating waveform and force more current in one phase. The result is increase in the voltage of the phase that has unbalanced voltage [95]. To understand the control theory, power equations must be written as follow:

$$v_a = V_m cos(wt)$$
$$v_b = V_m cos(wt - 2\pi/3)$$
$$v_c = V_m cos(wt + 2\pi/3) \quad \text{(A.1)}$$

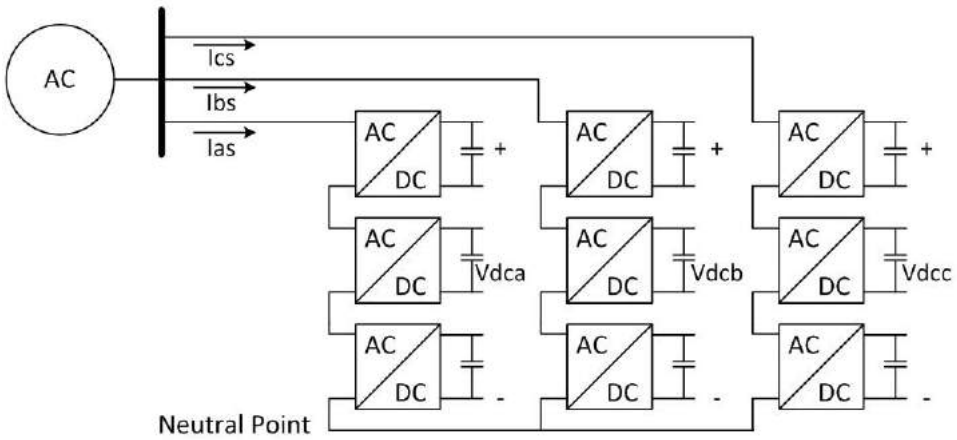


Figure A.5: Block Diagram of Three Module per Phase Cascaded H-bridge Converter

$$\begin{aligned} i_a &= I_m cos(wt) \\ i_b &= I_m cos(wt - 2\pi/3) \\ i_c &= I_m cos(wt + 2\pi/3) \end{aligned} \tag{A.2}$$

The power of the source would be:

$$\begin{aligned} P_a = v_a i_a &= 0.5 V_m I_m cos(\theta_i) + 0.5 V_m I_m cos(2wt + \theta_i) \\ P_b = v_b i_b &= 0.5 V_m I_m cos(\theta_i) + 0.5 V_m I_m cos(2wt + \theta_i + 2\pi/3) \\ P_c = v_c i_c &= 0.5 V_m I_m cos(\theta_i) + 0.5 V_m I_m cos(2wt + \theta_i - 2\pi/3) \end{aligned} \tag{A.3}$$

Assuming that the phase voltage balancing voltage is $V_{pb} = V_{cm}cos(wt + \phi)$, the power flow by injecting this voltage is:

$$\begin{aligned} P'_a &= P_a + 0.5V_{cm}I_m cos(\phi - \theta_i) + 0.5V_{cm}I_m cos(2wt + \theta_i + \phi) \\ P'_b &= P_b + 0.5V_{cm}I_m cos(\phi - \theta_i + 2\pi/3) + 0.5V_{cm}I_m cos(2wt + \theta_i + \phi + 2\pi/3) \\ P'_c &= P_c + 0.5V_{cm}I_m cos(\phi - \theta_i - 2\pi/3) + 0.5V_{cm}I_m cos(2wt + \theta_i + \phi - 2\pi/3) \end{aligned} \quad (A.4)$$

Therefore, the average value of the power would become:

$$\begin{aligned} P'_{aavg} &= 0.5V_m I_m cos(\theta_i) + 0.5V_{cm}I_m cos(\theta_i + \phi) \\ P'_{bavg} &= 0.5V_m I_m cos(\theta_i) + 0.5V_{cm}I_m cos(\theta_i + \phi + 2\pi/3) \\ P'_{cavg} &= 0.5V_m I_m cos(\theta_i) + 0.5V_{cm}I_m cos(\theta_i + \phi - 2\pi/3) \end{aligned} \quad (A.5)$$

Converting it from abc frame to dq frame the power equations are:

$$\begin{bmatrix} P'_{davg} \\ P'_{qavg} \end{bmatrix} = 0.5V_{cm}I_m \begin{bmatrix} cos(\phi - \theta) \\ sin(\phi - \theta) \end{bmatrix} \quad (A.6)$$

Now let's look at the voltage imbalance in phases. By finding the energy imbalance in the phases and inject the necessary power, the voltages will become balanced too:

$$\begin{bmatrix} \Delta E_{aavg} \\ \Delta E_{bavg} \\ \Delta E_{cavg} \end{bmatrix} = C_{dc}V_{pavg} \begin{bmatrix} \Delta V_{aavg} \\ \Delta V_{bavg} \\ \Delta V_{cavg} \end{bmatrix} \Rightarrow \quad (A.7)$$

$$\begin{bmatrix} \Delta E_{davg} \\ \Delta E_{qavg} \end{bmatrix} = \frac{2}{3} \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{pmatrix} \begin{bmatrix} \Delta E_{aavg} \\ \Delta E_{bavg} \\ \Delta E_{cavg} \end{bmatrix} \tag{A.8}$$

$$\begin{bmatrix} P^*_{dpavg} \\ P^*_{qpavg} \end{bmatrix} = \frac{K_p + K_i s}{s} \begin{bmatrix} -\Delta E_{davg} \\ -\Delta E_{qavg} \end{bmatrix} \tag{A.9}$$

Therefore, phase balancing voltage is:

$$V_{pb} = cos(wt + \theta^*_i) P^*_d pavg - sin(wt + \theta^*_i) P^*_q pavg \tag{A.10}$$

The final control block diagram has been shown in the Figure 2.3. This controller can balance the total dc voltages in each phase.

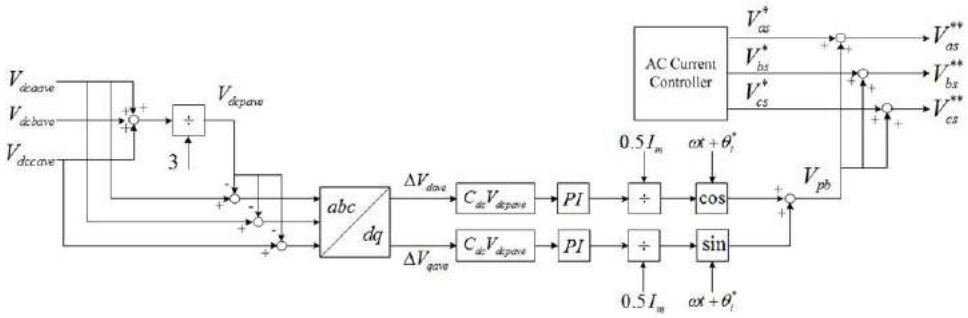


Figure A.6: Phase Voltage balancing Controller block diagram

- **Voltage Balancing Controller**: The last controller balances the voltages on all the DC capacitors of each module. It acts like a circuit protector and prevents overcharging of

capacitors. Just like the DC terminal controller, the proportional gain of this controller is low. Since this controller get direct feedback from the power electronic and measurement signals, it must be implemented on the slave controllers. The way this controller works is

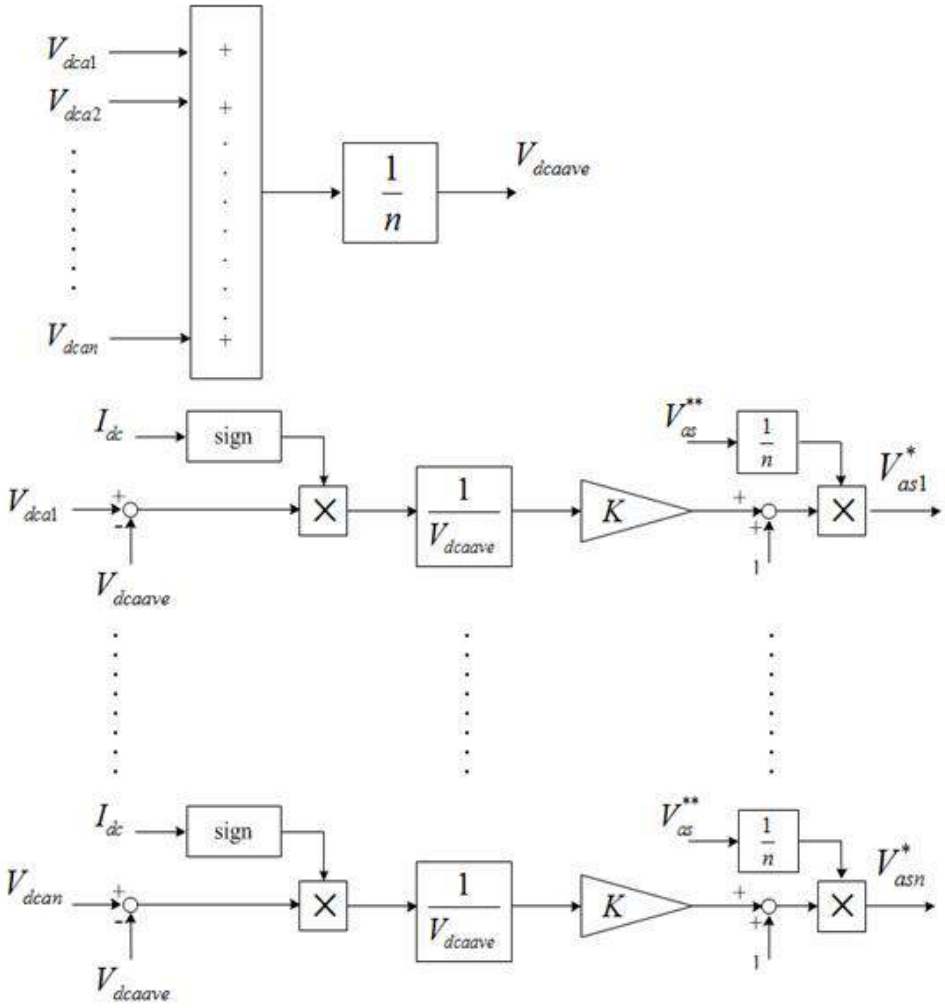


Figure A.7: Module Voltage Balancing Controller Block Diagram Implemented in Slave Controllers

by changing the effective active cycle of each controller, therefore the average current

that each capacitor supplies will change and the circuit will become balanced. When the grid is supplying current to the converter, the higher active duty cycle the capacitor has, the more it is going to be charged. When the converter is sourcing current to the grid, the action would be reversed. The higher active duty cycle the capacitor has, the more it gets discharged. By using this principle, a controller based on Figure A.7 can be designed.

Appendix B

Distributed Control Stages for Modular Multi-level Converter

Modular Multi-level Converter (MMC) is one of the utilized converter topologies in high voltage DC (HVDC) power transmission and had become practical from 1990s by different corporations. MMC has better performance in comparison to the conventional tyristor based converters. The total harmonic distortion (THD) has largly been improved due to the gradual change in the output voltage (ommision of AC filter in the system). The semiconductor has been changed to IGBT and it is possible to turn off the device without any problem. This will increase the controlability of the converter and add capabilities like black start to the converter [85].

There are different topologies for MMC and the connection style of the converter to the electrical grid (figure B.1)[3][37]. MMC can be single sided in each leg (like the cascaded H-bridge converter in the previous chapter) or double sided. Single-sided MMC topologies are usually suitable for STATCOM applications or in systems in which the DC voltage of each phase can be connected to the load separately (Battery energy storage system). Since the single sided converters may not be connected to DC grid without any conversion at the capacitor side, only

double sided converter may be used for direct conversion of AC to DC or vice versa (BTB system).

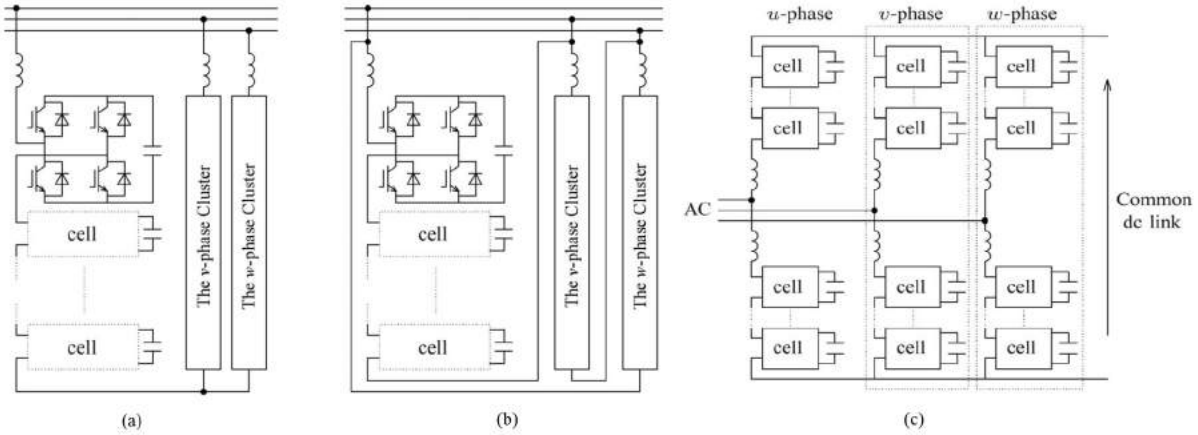


Figure B.1: Different Architectures of Modular Multi-level Converters (MMC) a)Single-Star Bridge-Cells (SSBC) b)Single-Delta Bridge-Cells (SDBC) c)Double-Star Bridge-Cells (DSBC)

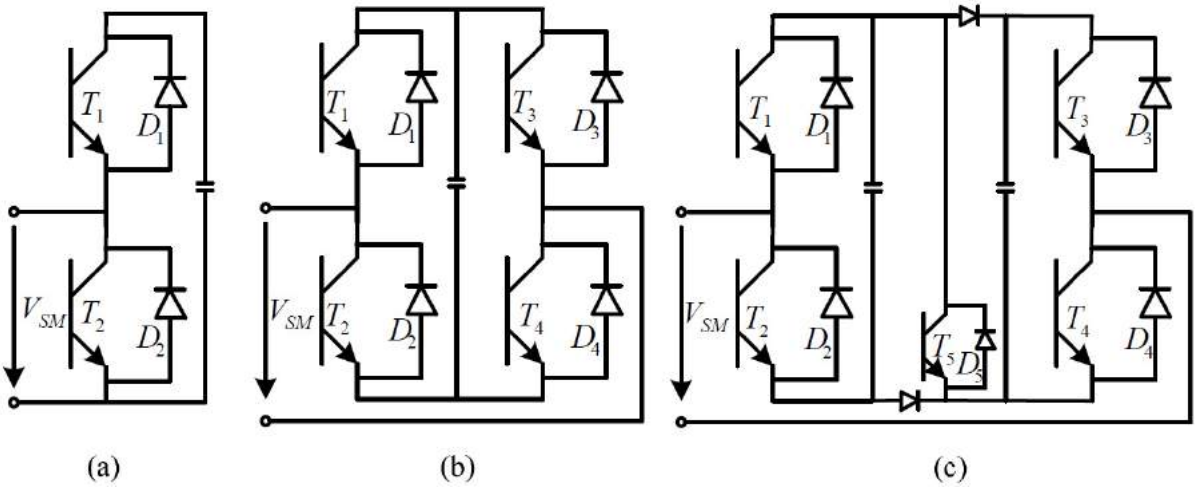
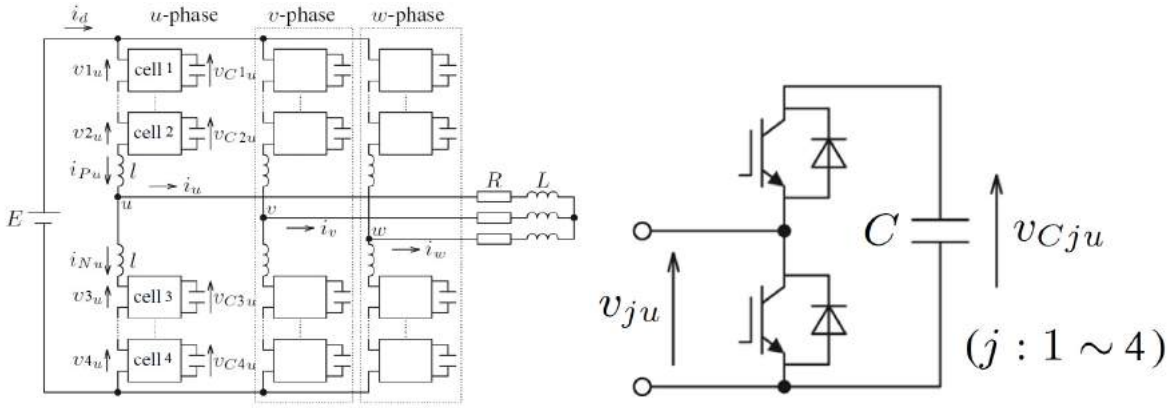


Figure B.2: Architectures of MMC Sub-modules a)Half-bridge b)Full-bridge c)Double-clamped

There are different possible converter architectures for sub-modules in MMC[63]. Figure B.2 shows three most common architectures. Half-bridge sub-module has the lowest loss compared to other architectures, but it can only conduct current in one direction. It is not possible to form a single-sided MMC with this sub-module. The full-bridge sub-module adds bi-directional capability at the cost of more switches. At each moment two switch must be turned on for conduction, therefore the total efficiency of the MMC would be less. The double-clamped submodule is combination of two half bridge converter and can perform better at DC faults[66][97]. At dc fault, T5 (which is normally on) will be turned off resulting in voltage clamping and energy absorption. During voltage clamping, both capacitors are in parallel, ensuring minimized over-voltage.

In this chapter, the focus is on MMC with double-sided converter arm and half-bridge sub-modules. The DC grid voltage controller and converter current controller for MMC are the same as the cascaded H-bridge converter but there is different in the averaging control and the balancing control of the converter [107, 37, 39, 58, 38].

Figure **??** demonstrate a three phase converter based on MMC. In this converter, each leg



(a) MMC Power Circuit (b) Half-bridge PWM chopper-cell

Figure B.3: Circuit Configuration of a Double-star MMC

consists of eight sub-modules. Each sub-module consists of half-bridge connected to a floating capacitor at dc side. All the equations are given for u-phase with 8 sub-modules and it can be used for other phases and different number of sub-modules per phase. The following equation exist at the DC voltages:

$$E = \sum_{j=1}^{4} v_{ju} + l\frac{d}{dt}(i_{Pu} + i_{Nu}) \tag{B.1}$$

In this equation, E is the supply DC voltage, v_{ju} is output voltage of each chopper cell, l is buffer inductance, i_{Pu} and i_{Nu} are positive and negative arm currents. A circulating current (i_{Zu}) through the dc power supply and the converter arm may be defined as follows:

$$i_{Zu} = i_{Pu} - \frac{i_u}{2} = i_{Nu} + \frac{i_u}{2} = \frac{1}{2}(i_{Pu} + i_{Nu}) \tag{B.2}$$

The voltage balancing for the capacitors is divided in to two control stages. The averaging control (phase voltage balancing) and balancing control. The first controller tries to equalize the dc voltages in each phase and the later controller equalize the voltage of each cell.

- **Averaging control**: Figure B.4 shows the control diagram of the averaging control. This block equalize the average voltage of capacitors in each phase. This controller forces the average voltage $\bar{v}_{Cu}$ to follow the reference point v_C^*. The equation for $\bar{v}_{Cu}$ is given by:

$$\bar{v}_{Cu} = \frac{1}{4}\sum_{j=1}^{4} v_{Cju} \tag{B.3}$$

  The current command for balancing the voltage can be defined as following:

$$i_{Zu}^* = K_1(v_C^* - \bar{v}_{Cu}) + K_2 \int (v_C^* - \bar{v}_{Cu})dt \tag{B.4}$$

The voltage command for balancing control is as following:

$$v_{Au}^{*} = K_3(i_{Zu} - i_{Zu}^{*}) + K_4 \int (i_{Zu} - i_{Zu}^{*})dt \tag{B.5}$$

Whenever v_C^* is higher than $\bar{v}_{Cu}$, i_{Zu}^* increases to compensate it. The main task of the current minor loop is forcing the i_{Zu} to follow the i_{Zu}^*. Controlling the i_{Zu} will enable the $\bar{v}_{Cu}$ to follow v_C^* without getting any effect from the load current i_u.

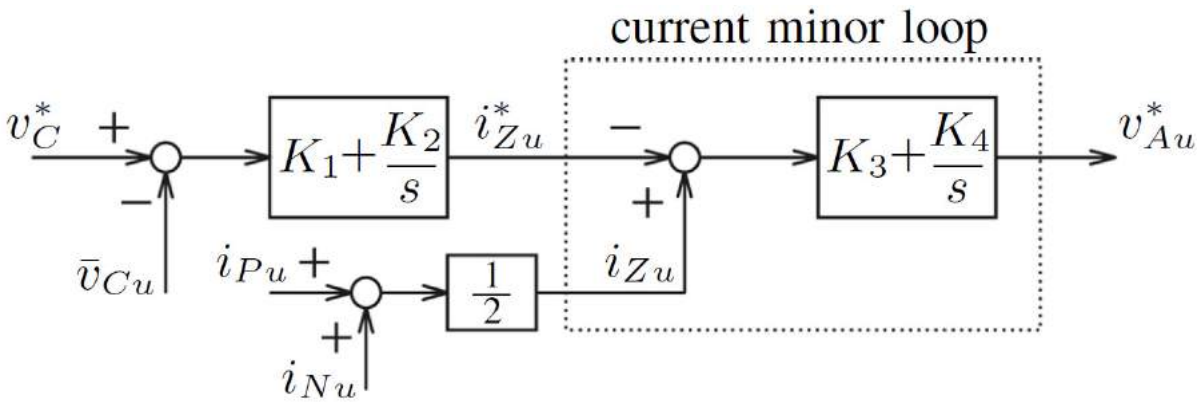


Figure B.4: Averaging Control of the Capacitor Voltages

- **Balancing control**: Figure B.4 demonstrate the structure of the balancing control. This controller tries to balance the voltage of capacitors individually. Since the balancing is based on either i_{Nu} or i_{Pu}, polarity of v_{Bju}^* should be changed accordingly. When $v_C^* \geq v_{Cju}$, positive active power must be taken from the dc power supply and be fed to the chopper-cell capacitors. When i_{Pu} (or i_{Pu}) is positive, multiplication of v_{Bju} and i_{Pu} forms positive active power. When i_{Pu} is negative, the polarity should be inverse to take positive active

power from the capacitors. The following equations can be derived for the upper leg:

$$v^*_{Bju} = \begin{cases} K_5(v^*_C - v_{Cju}) & (i_{Pu} \geq 0) \\ -K_5(v^*_C - v_{Cju}) & (i_{Pu} \leq 0) \end{cases} \tag{B.6}$$

The same equations may be applied for the lower leg:

$$v^*_{Bju} = \begin{cases} K_5(v^*_C - v_{Cju}) & (i_{Nu} \geq 0) \\ -K_5(v^*_C - v_{Cju}) & (i_{Nu} \leq 0) \end{cases} \tag{B.7}$$

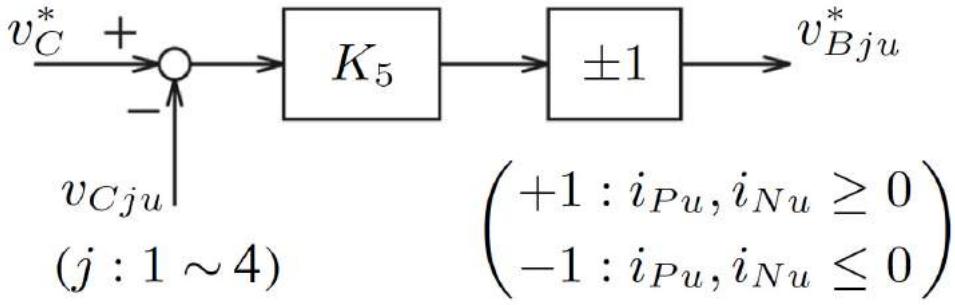


Figure B.5: Balancing Control of the Capacitor Voltages

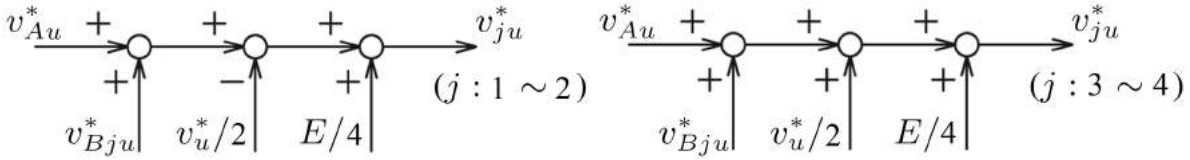
The final reference voltage each sub-module (v^*_{ju}) of a MMC with eight module per phase has been shown in figure **??** and is given by:

$$v^*_{ju} = v^*_{Au} + v^*_{Bju} - \frac{v^*_u}{2} + \frac{E}{4} \quad (upper\ leg) \tag{B.8}$$

$$v^*_{ju} = v^*_{Au} + v^*_{Bju} + \frac{v^*_u}{2} + \frac{E}{4} \quad (lower\ leg) \tag{B.9}$$

In this equations, v^*_u is the reference voltage for the u-phase load. For the upper leg, this voltage

must have negative sign and for the lower leg, positive sign is necessary. These equations include a feed-forward control from the dc supply voltage to the output ($\frac{E}{4}$). This element improves the dynamic performance of the converter. The final controller stage in figure **??** may be



(a) Positive arm (b) Negative arm

Figure B.6: Voltage Command of each MMC Arm

implemented by distributed controllers. Data representing global variables may be supplied by the master controller (synchronizer). Each slave controller has access to local variables and can implement the whole control block.

Appendix C

Schematics and Design of Fault-tolerant Controller Testbed

This appendix represents the design of the test-bed for evaluating the fault-tolerant controller. Figure C.1 through C.8 demonstrate the schematics of the controller board. The main design sheet is in figure C.1 and other sheets are addressed by this schematic. Figure C.9 represent the final design of the printed circuit board which has been populated and assembled for implementation and testing of the controller architecture.

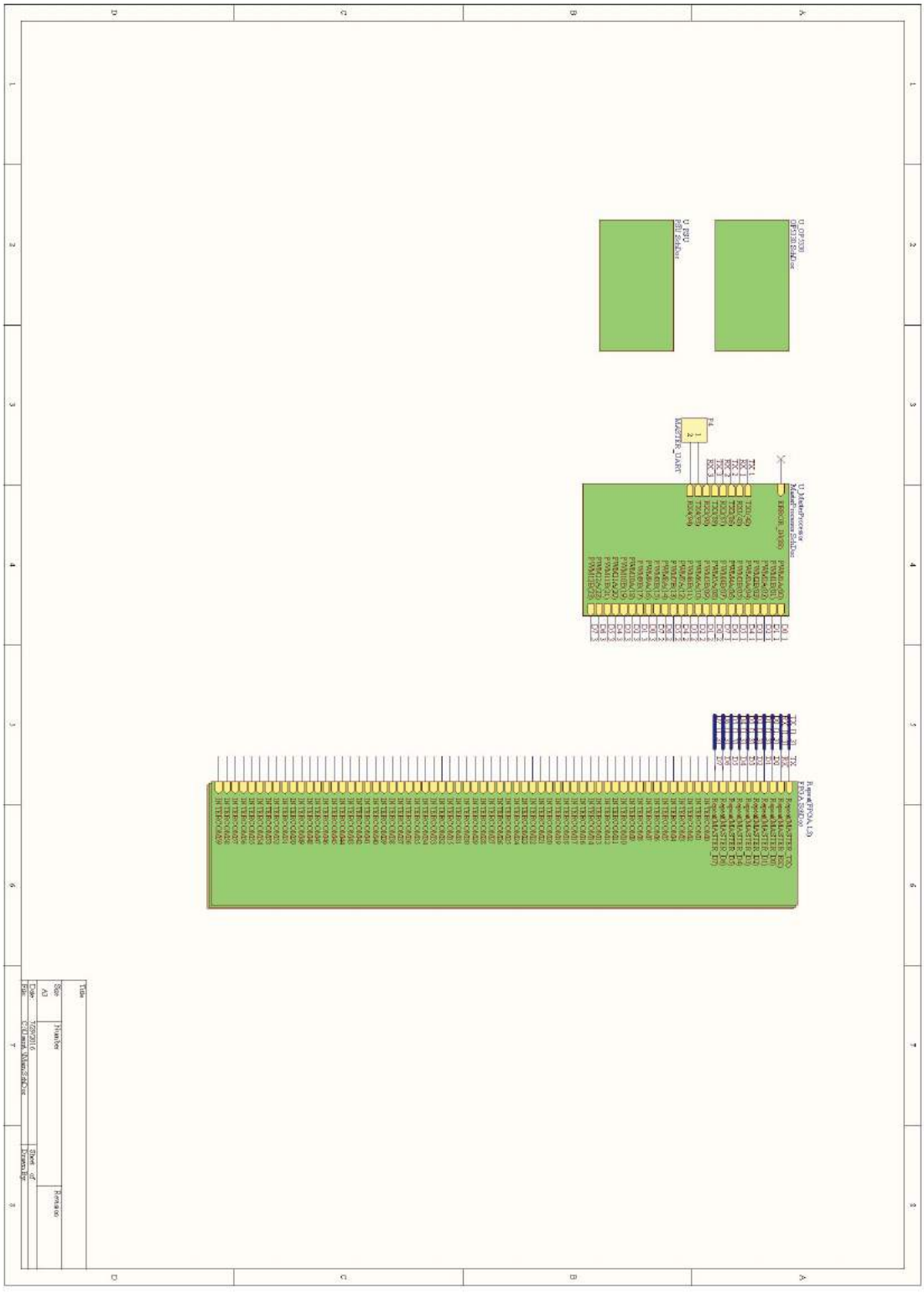


Figure C.1: Schematic of the Main Sheet

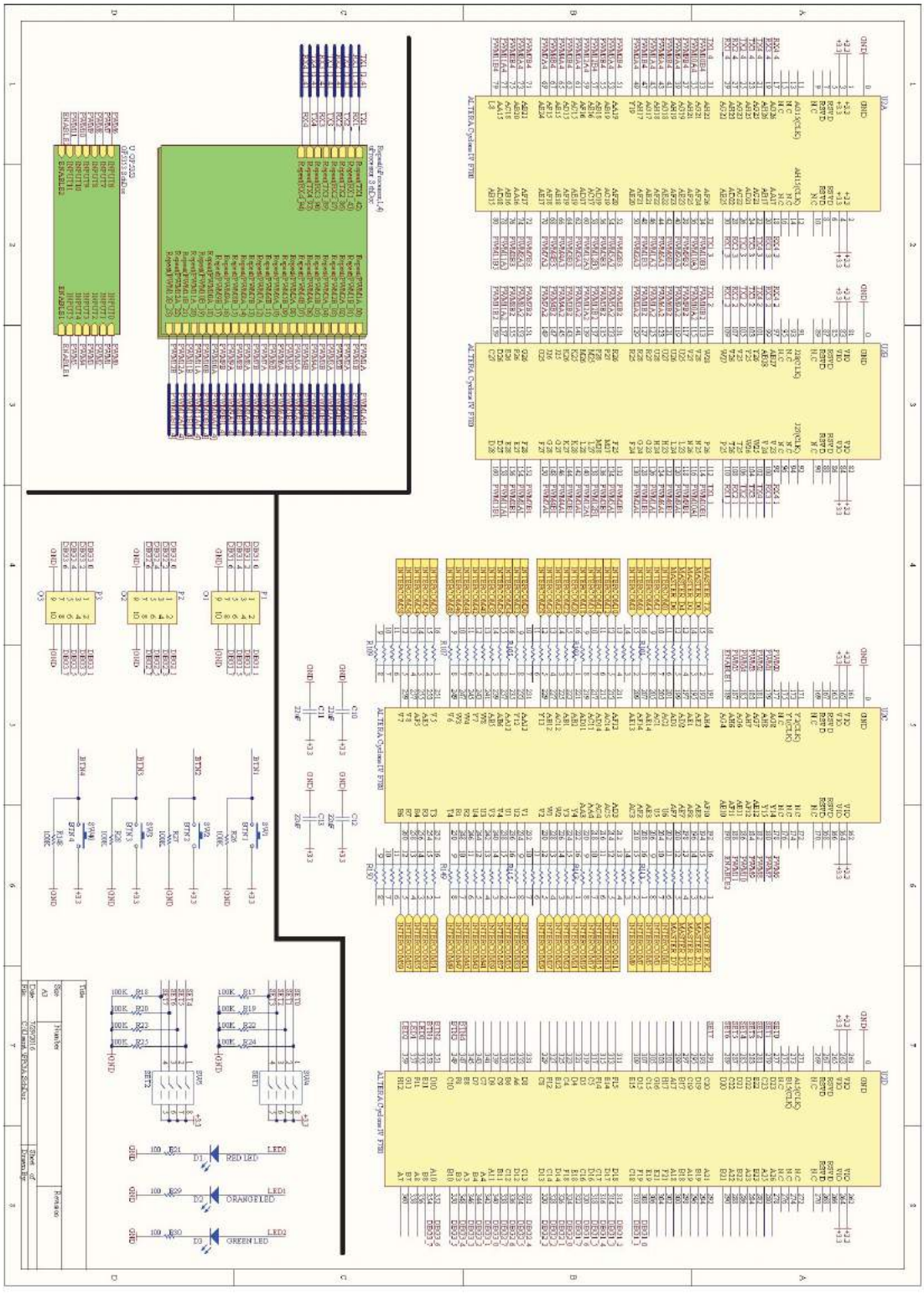


Figure C.2: Schematic of the FPGA Interconnection Configuration Circuit

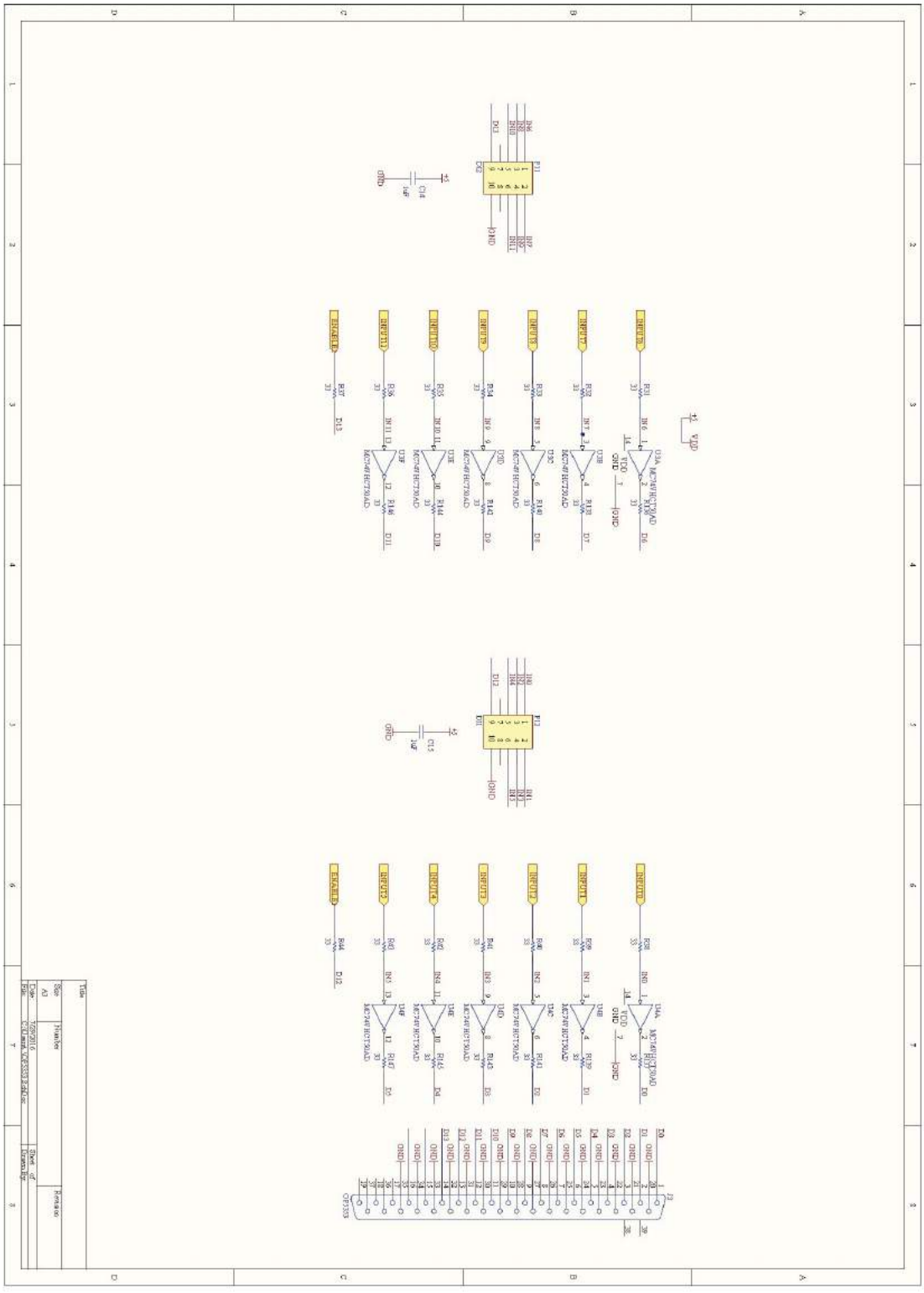


Figure C.3: Schematic of Switching Signal Buffers

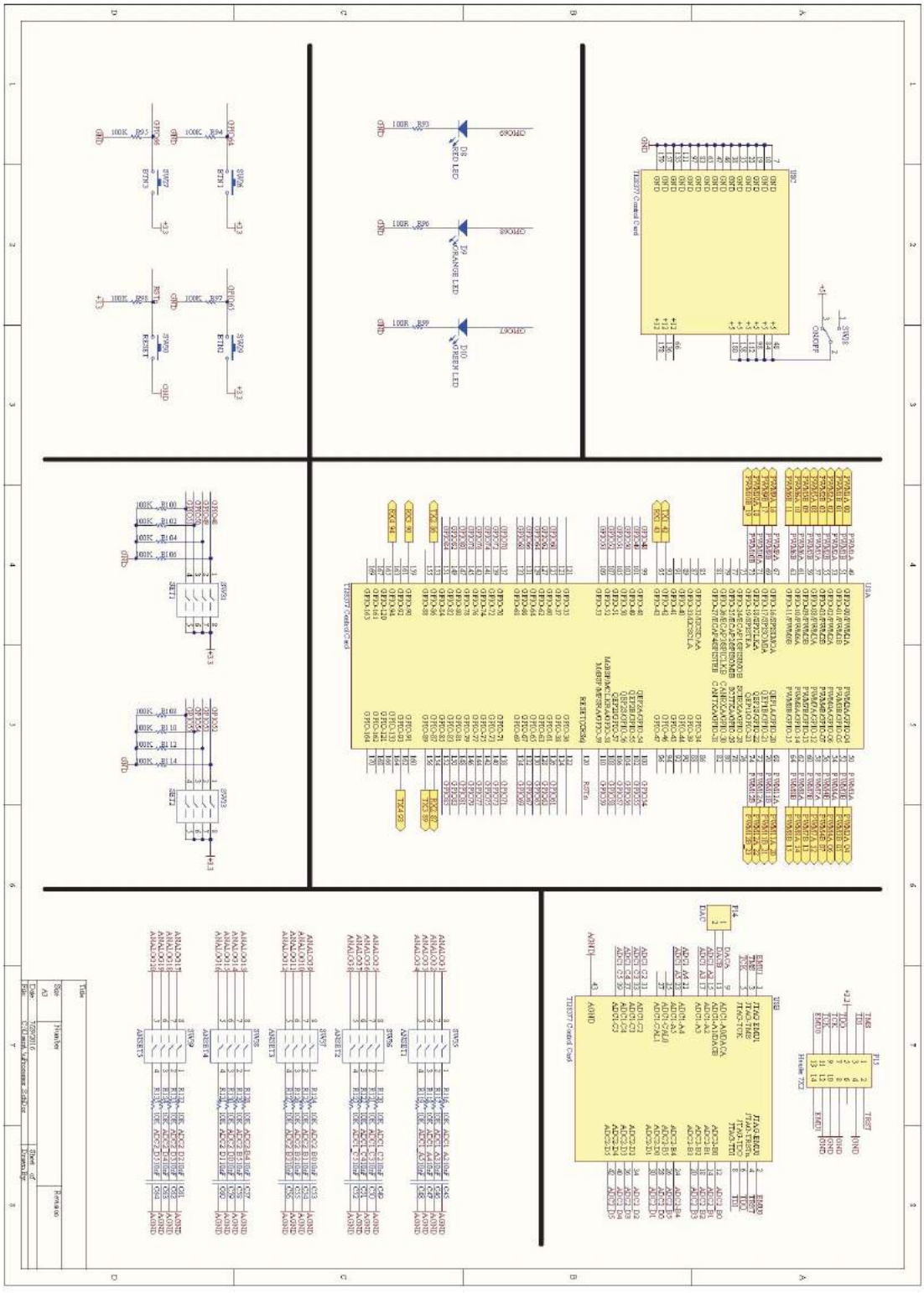


Figure C.4: Schematic of Slave Micro Controller

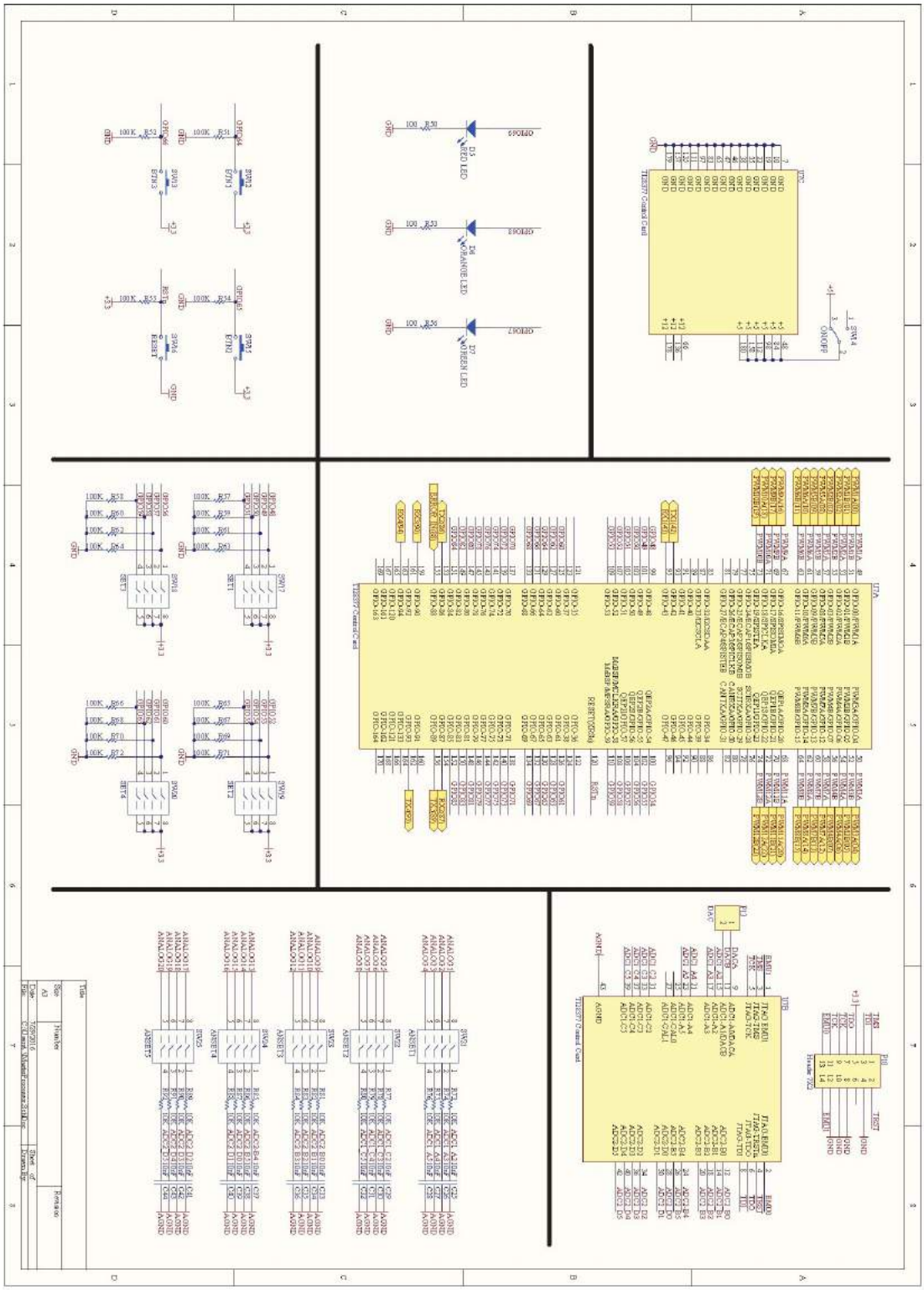


Figure C.5: Schematic of Master Micro Controller

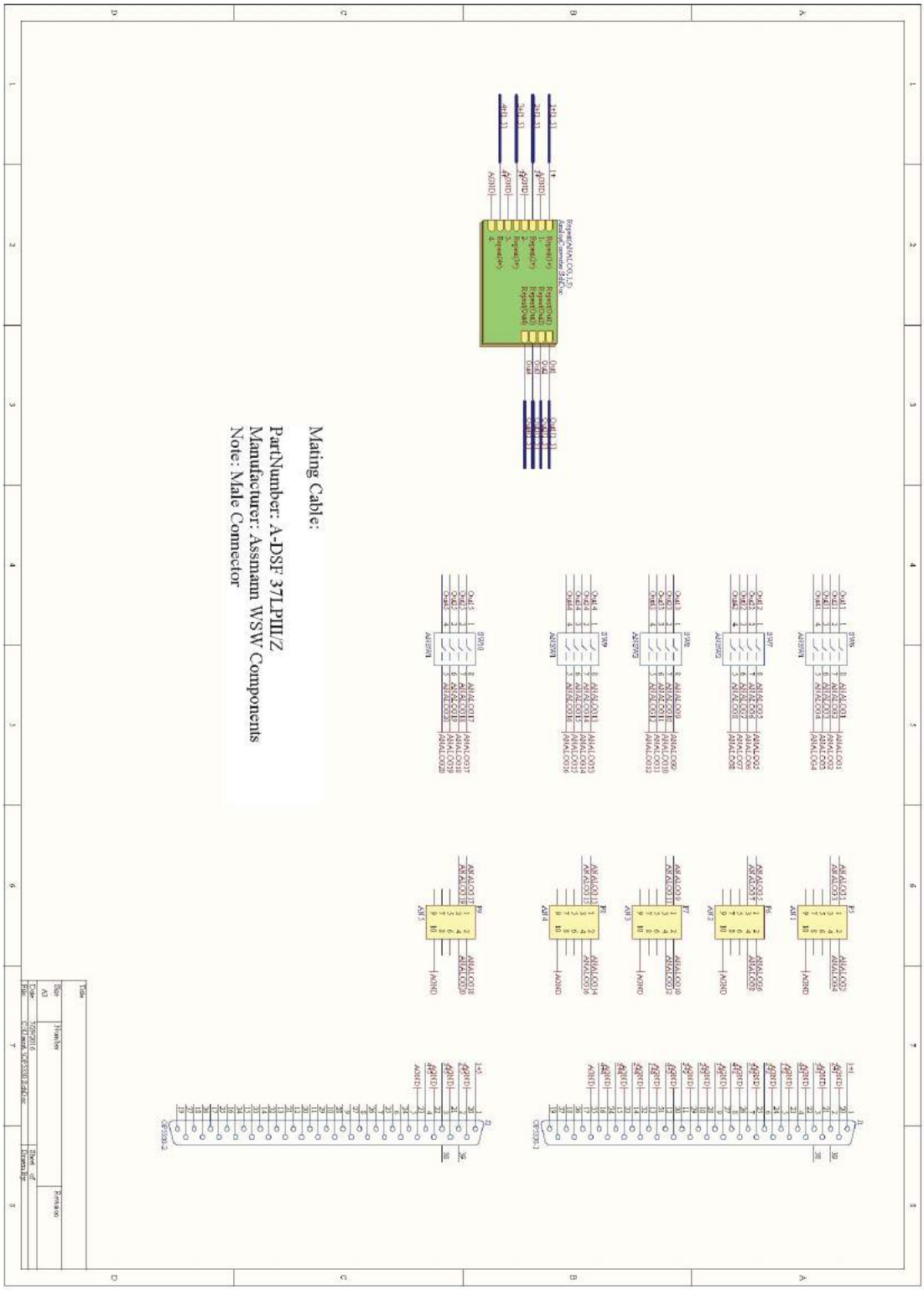


Figure C.6: Schematic of Analog Signal Conditioning (Top Level)

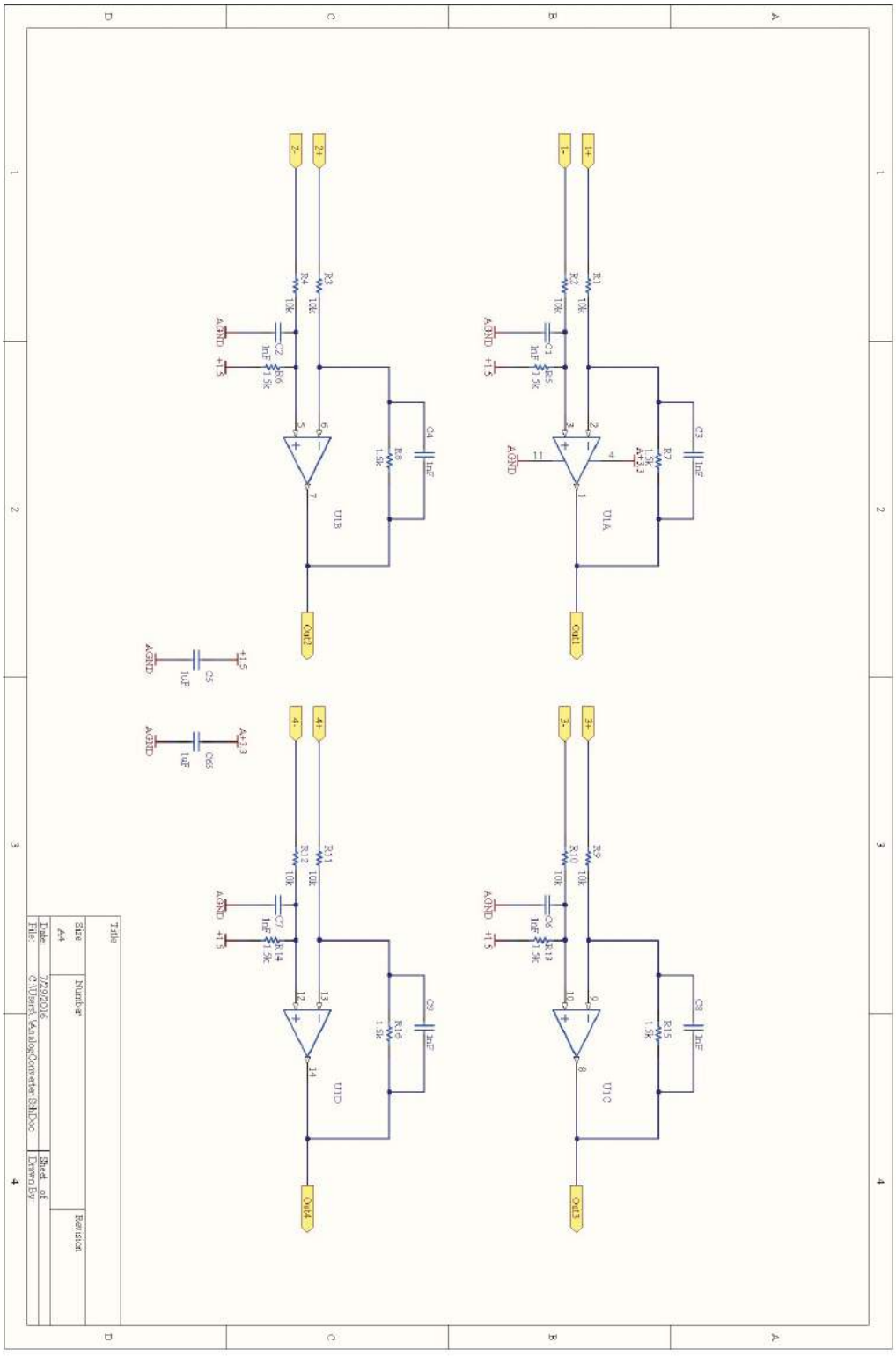


Figure C.7: Schematic of Analog Signal Conditioning (Low Level)

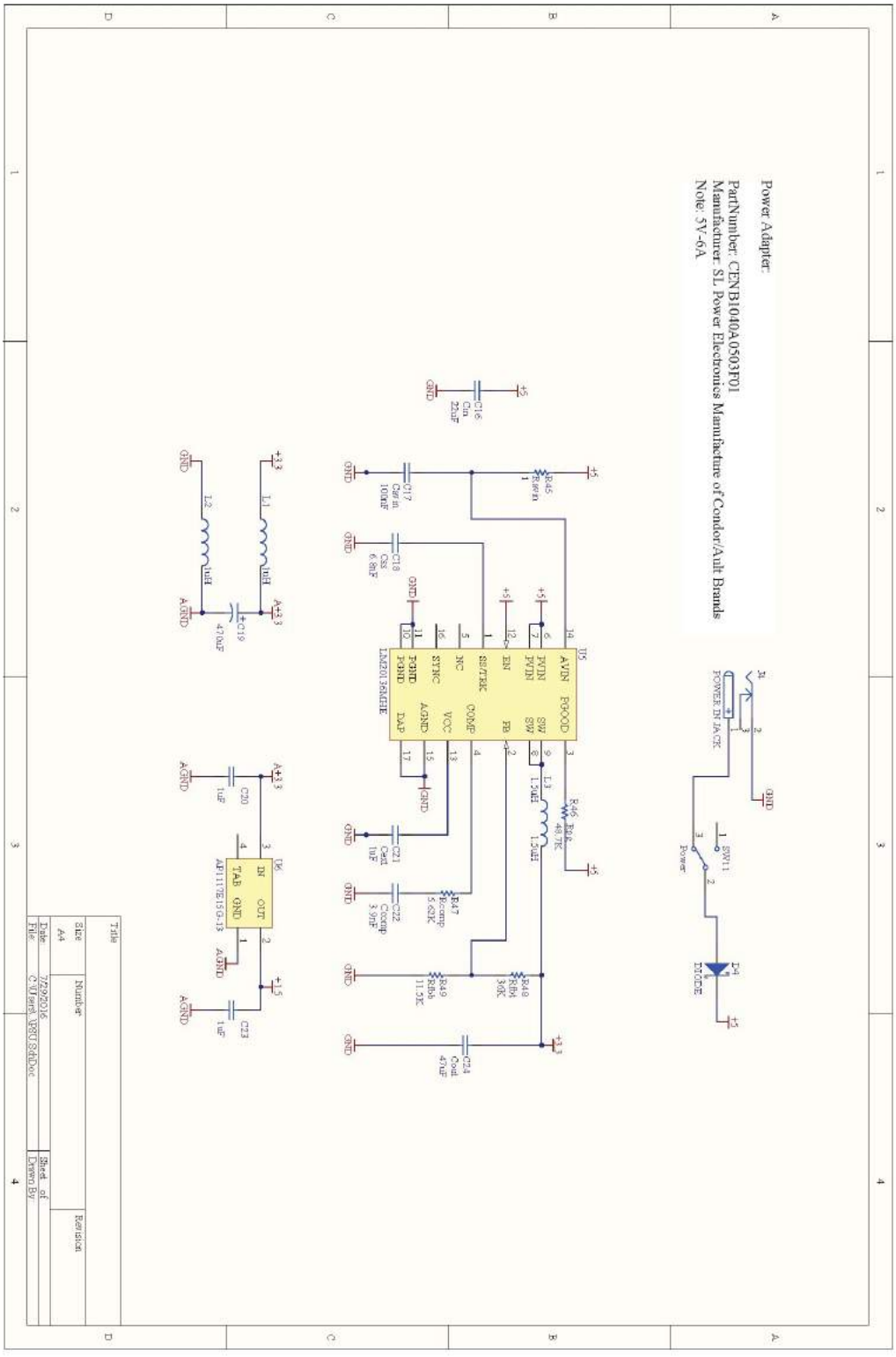


Figure C.8: Schematic of Power Supply Unit

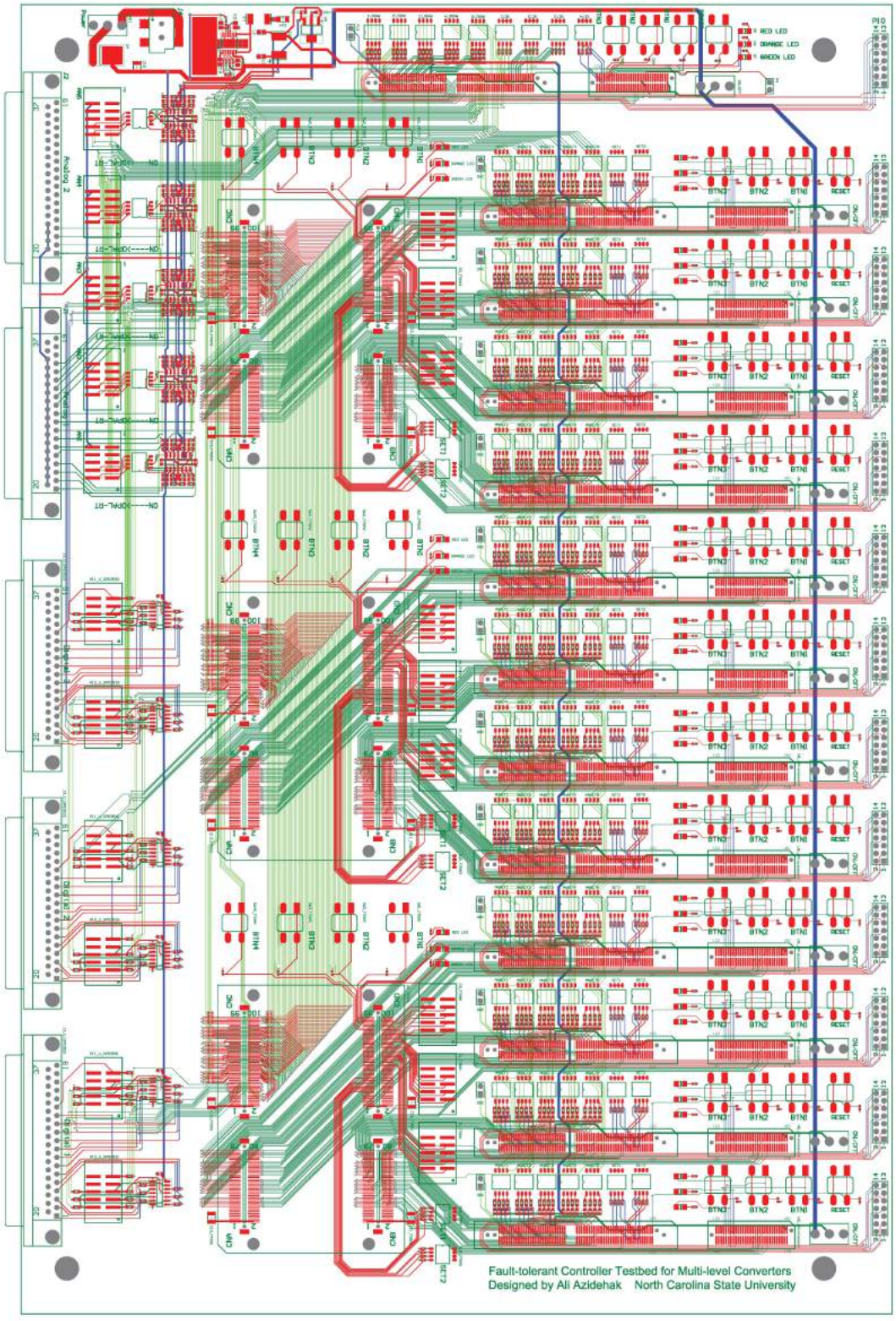


Figure C.9: Printed Circuit Board Design of the Fault-tolerant Test bed