ABSTRACT

Wei Liu. Distributed modular controller architecture for high power converter applications. (under the direction of Dr. Alex Q. Huang)

Modular converter is of particular interest in high power applications in order to achieve higher level of flexibility, expandability and reliability. Power electronics building block(PEBB) and H-bridge building block(HBBB) are typical modular blocks proposed for this applications. However, for converters based on the modular converter concept, the conventional controller is still not modular. The controller typically only has one single central control unit. The central controller has direct connections with each modular converter. When the controller and the power converters are placed close to each other, there will not be problem to do so despite a lot of wire connections. For high voltage and power rating converters, it is true that the power converters will consist of many modular blocks and these blocks will be placed at some distance from the controller. In this case, digital switch signals must be sent to the converters via optical fiber to improve reliability. However, the required fiber connections are too many that increase the risk of fault. Moreover, the analog signals from the sensors such as the voltage and current are usually sending back to the central controller through analog wire connections that has low electromagnetic interference (EMI) susceptibility.

To solve the problems mentioned above for a high power converter system that may have multiple modular converter units, a modular controller architecture is needed and this

concept is starting to be accepted specially if the system is constructed with modular converters.

In this thesis, the proposed modular controller architecture includes a central controller and distributed local controllers. Each modular converter is accompanying with a local controller. The central controller performs the main control loop calculation and then sends the control commands to the local controllers. The central controller and the local controllers are communicated based on defined protocol via optical fibers. Then, the local controllers decode the command and generate the gate signals to the converter. Also, the local controllers have the function to convert analog information to digital such as voltages and currents and then feed back them to the central controller via optical fibers. The communication between the central controller and each local controller requires only two optical fibers. This largely increases the system reliability and also the flexibility to expand the system to higher power rating.

Distributed Modular Controller Architecture

For High Power Converter Applications

by

Wei Liu

A thesis submitted to the Graduate Faculty of

North Carolina State University

In partial fulfillment of the

Requirements for the degree of

Master of Science

In

Electrical Engineering

Raleigh, NC

2005

Approved by:

_____ _____

_____

Chair of Advisory Committee

DEDICATION

I would like to dedicate this work to my parents, Di Liu and Qiuxiu Chen. They are the people who love me most in this world. They take care of me from my childhood. They encourage me to do my best in my study and life. They are always giving me strength when I lose my faith. They are always staying around me when I am sick. They are always happier than me when they get good news from me. They are the best father and mother in this world. I love them so much.

BIOGRAPHY

Wei Liu was born in Jiangxi, China in 1978. He received his bachelor's degree in automation control engineering in 1998 from Huazhong University of Science and Technology. From 1998 to 2003, he joined the team to start the Chinese branch of MGE UPS Systems. He worked as a R&D engineer in the beginning and then was promoted to serve as R&D supervisor. He has led the R&D teams developing a couple of new uninterrupted power supply projects. From the August of 2003, he began to pursue his master's degree in electrical engineering department of Virginia Tech. He was also a research assistant in Center for Power Electronics Systems (CPES). In the summer of 2004, he did an internship as an application engineer in INTERSIL Corporation. From the August of 2004, he transferred to North Carolina State University together with his advisor Dr. Alex Huang. He continued pursuing his master degree in Electrical Engineering Department and worked as a research assistant in the Semiconductor Power Electronics Center (SPEC) at the same time. His research interests include DC-DC converter, PWM controller, DC-AC inverter and digital controller. Other than work, He likes reading and travel.

ACKNOWLEDGMENTS

To complete this thesis work, I got help and support from a lot of people. I would like to acknowledge the following individuals for their guidance and support.

I would like to express my gratitude to my advisor Dr. Alex Q. Huang. He is always giving me the right research direction. He is also always so patient to discuss with me no matter how busy he is. I learned a lot of knowledge, methods and ways of thinking from him.

I am also grateful to Dr. Mesut Baran and Dr. Maysam Ghovanloo for their interest in my research and for serving as my thesis committee.

I greatly appreciate the discussions with Dr. Zhong Du and Dr. Subhashish Bhattacharya. They gave me a lot of helpful suggestions on my research work.

I would like to acknowledge Mr. Rajganesh Jayakar,Mr. Wenchao Song and Mr. Zhaoning Yang for their efforts on my research project. It is a great pleasure to work with them.

I also want to thank everyone in my research group and the SPEC center. I am so happy to work together with so many talents. I also greatly enjoy the friendship with them.

I would also like to acknowledge the help from Daniel Ghizoni and Jerry Francis who are research assistants in CPES of Virginia Tech. Their supports are quite helpful to my research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Motivation

Modular Converter has drawn particular interest in high power applications to achieve high level of reliability and flexibility. The concept of power electronics building block (PEBB) is proposed to achieve modular converter [4][5]. However, high power converters are still centrally controlled. The centralized controller is very complex. It performs all the control function, collects all the analog sensing signals from the converters, and generates all switch signals for power devices. The large amount of wire connections reduces the reliability of the system. Moreover, it makes the expansion of the system quite difficult.

To solve the problems of the centralized control, a novel distributed modular controller is proposed and developed in this thesis. The proposed modular controller architecture includes a central controller and distributed local controllers. Each local controller is accompanying with one modular converter to form an intelligent modular converter.

1.2 Thesis Overview

Chapter 1 introduces the motivation and background of this thesis. High power converter topologies and applications are briefly described. Also, modular converter concept and topologies are illustrated.

Chapter 2 describes the existing power converter digital controller structures. Previous work on modular controllers is introduced. Then, several possible distributed modular controller architectures are investigated. Finally, a novel control architecture is identified for an example application.

Chapter 3 shows the development of the proposed distributed modular controller architecture. Hardware and software implementations are described in details. Communication protocol is also defined and clearly described.

Chapter 4 presents the experimental validation of modular controller with modular converter. Close loop tests are set up to validate the modular controller design.

Chapter 5 investigates the synchronization issues among the local controllers. Several solutions are proposed to solve this problem. Experiment results are shown for the validation.

Chapter 6 is the summary of the work presented in the thesis. Future plans with the distributed modular controller are outlined.

1.3 High Power Electronics Applications

The highest power electronics application is for transmission power including High Voltage Direct Current (HVDC) power transmission and Flexible AC Transmission (FACTS). HVDC involves AC-DC and DC-AC conversions. To be more cost effective, current source converters are usually used for HVDC. For FACTS, voltage source converters are more preferred [1].

FACTS controllers can be generally divided into the following categories [1]:

- Series Controllers
- Shunt Controllers

- Combined series-series Controllers
- Combined series-shunt Controllers

The series controller injects voltage in series with the transmission line. It can be achieved by a variable impedance or a power electronics based variable source. Static Synchronous Series Compensator (SSSC) is one kind of series controllers.

The shunt controller injects current into the system at a coupling point. It can be achieved by a variable impedance or a variable source. Static Var Compensator (SVC) and Static Synchronous Compensator (STATCOM) are typically shunt compensators.

The combined series-series controllers provide reactive power compensation for each line and also transfer the real power among lines. This kind of controller is also known as Interline Power Flow Controller (IPFC) [1].

The combined series-shunt Controller is also called a Unified Power Flow Controller (UPFC). It combines one shunt controller and one series compensator [1].

1.4 High Power Converter Topologies

For the above high power electronics applications, shunt controllers has the easiest connection with the power line. SVC and STATCOM have therefore been widely accepted by the industry. The VAR compensation generated from a SVC or STATCOM helps the voltage regulation at the coupling point hence mitigating the voltage instability, as well as damping of power oscillation [1].

In this thesis, STATCOM will be the main concern of the applications. The appropriate high power converter topologies for its application are reviewed in the following. The converter

topologies could be divided into two level and multi-level converters. To achieve higher power rating and less harmonic, only multi-level converter topologies are shown.

The multilevel converter topologies can be classified as following: [2] [3]

A) diode-clamped multilevel converter

B) flying-capacitor multilevel converter

C) cascaded converters with separated DC sources

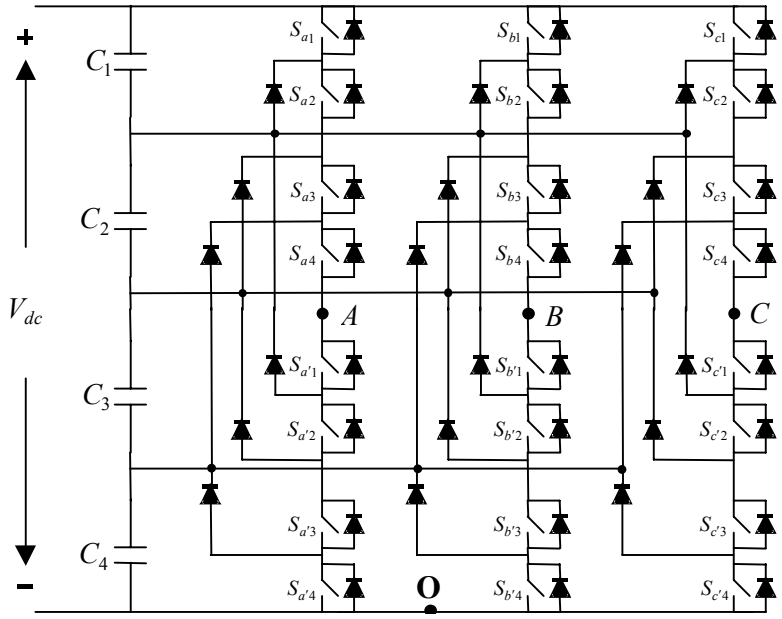


Fig.1.1. three-phase, five-level diode-clamped converter

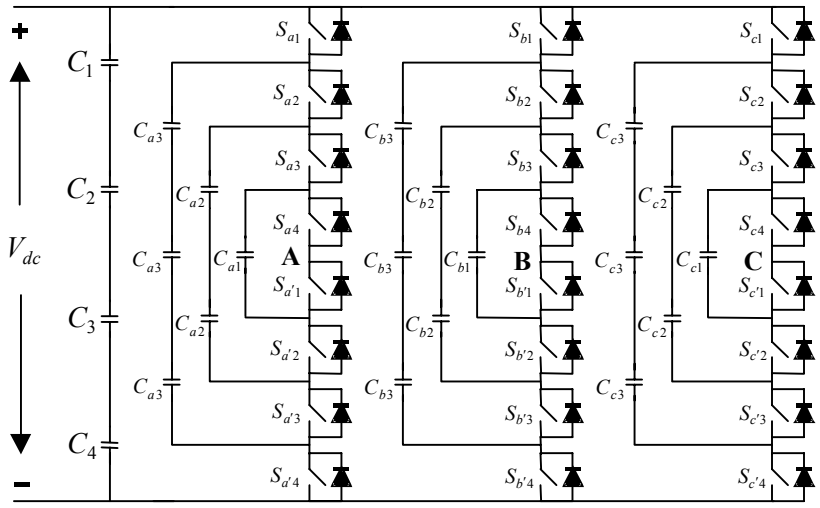


Fig. 1.2 three-phase, five level flying capacitor converter

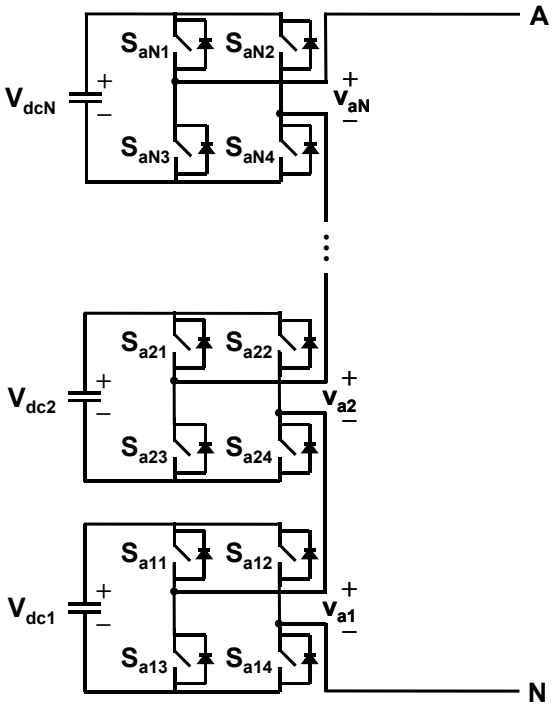


Fig. 1.3 Cascaded Multilevel Converter with separated DC sources

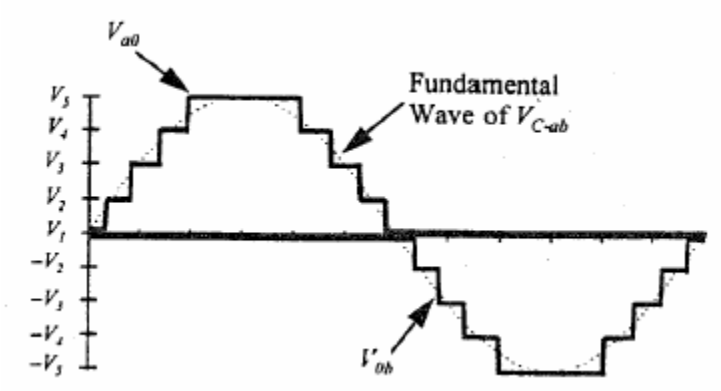


Fig. 1.4 the multi level converter synthesized voltage waveform

The multilevel voltage converter topologies are shown in Fig 1.1-1.3 respectively. Fig 1.4 shows the synthesized output voltage waveform of a 9 level converter.

1.5 Modular Converter Topologies

For high power electronics applications, modular converters will greatly reduce the engineering development cost. Power Electronics Building Block (PEBB) and H-Bridge Building blocks (HBBB) are possible modular converters that could be easily reconfigured to various applications. [4][5][6]

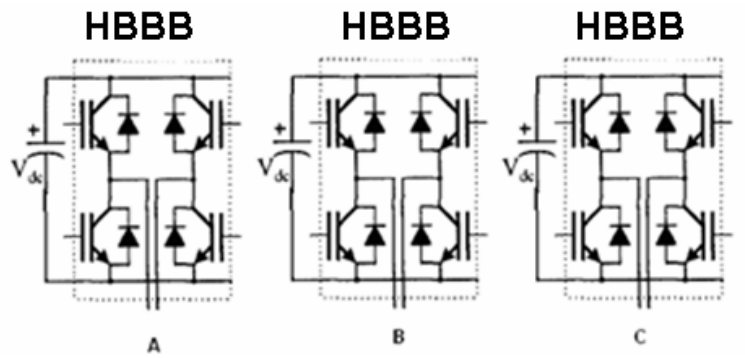


Fig. 1.4 H-Bridge Building Block

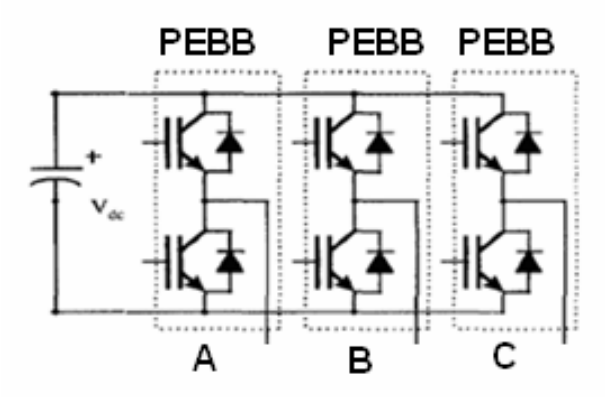


Fig. 1.5 Power Electronics Building Block (PEBB)

For the multi-level converter topologies shown above, cascaded multilevel converter is the most modularized topology. Besides its modularity, it has also a lot of advantages as following:

- Least number of components compared with other multi level topologies
- Easy expansion
- Easy to achieve higher power rating
- Easy to add redundancy

Therefore, in the following design examples, cascaded multilevel converter is used as the example.

CHAPTER 2

Power Converter Controller Architecture Investigations

2.1 Conventional Power Converter Controller

Conventionally, in most of the high power applications, the controller for the power converters is centralized. Typical control architecture for a 13 level cascaded multilevel converter is shown in Fig. 2.1

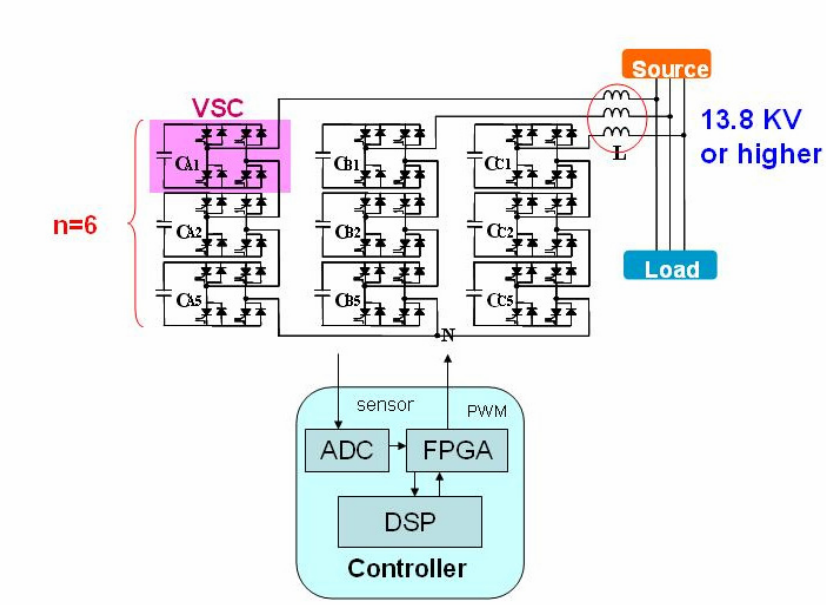


Fig. 2.1 Conventional Centralized Controller Structure

In this control architecture, the central controller performs all the control functions. It receives the analog sensing signals from each converter that includes voltage, current, temperature and fault information. Then it generates the modulation commands based on the received information. After that, the PWM signals are generated for each device in the converters. The control function is quite complex that decreases the reliability.

For the above architecture, between the controller and the power converters, there are a large number of connections including electrical wires for analog sensing signals. The connection interface is quite dependent on the converter rating and complexity. Although the converter has already been modular in this example, the system still could not achieve modularity because of the complex connection interface between the controller and the power stage. As the power converter becomes more complex, the controller will be even harder to be controlled.

The centralized controller structure has the following main drawbacks:

- Less system expansion capability
- No system modularity
- High engineering development cost
- Large number of wire connections between the power stage and the controller
- Less noise susceptibility
- Low system reliability as each wire is a failure node

2.2 Modular Controller Definition

In order to overcome the problems with the conventional centralized controller, modular controller concept is proposed. A local controller is designed for each modular converter. It provides intelligence to the converter. For difference power processing such as AC-DC, dc-DC and DC-AC, the local controller could be reconfigured to make the modular controller adaptive to various applications.

The modular controller may have the following features:

1. Easy reconfiguration capability

2. Flexibility for expansion
3. Less connections between the central controller and the local controllers
4. True optical fiber connections between central and local controllers
5. Analog sensing signals conditioning
6. Simple communication protocol between central and local controllers

In [6], the following features are considered for the defined modular controller

- Integrated current, voltage and temperature measurement.
- Isolated gate drive for both top and bottom switches of the phase leg.
- Device ID based communications protocol.
- Analog over-current protection (self contained).
- Digital implementation of pulse width modulation and deadtime.
- Analog to digital conversion of measured current for controller and feedback to outer-loop.
- Expandable for parallel device operation.
- Global control codes, for example shutdown and synchronize.
- Software selectable switching frequency, current and temperature trip and PWM phase sync.
- Isolated voltage measurement for voltage regulation and/or protection.

2.3 Literature Review on Modular Controller Development

In [6], a distributed controller topology is proposed for de-coupled phases control as shown in Fig.2.2. Each phase leg is controlled by a separate module controller. The module controller is defined to perform the voltage or current loop control. They are defined as inner loop control. The proposed controller is more effective while the converter topologies are phase de-coupled.

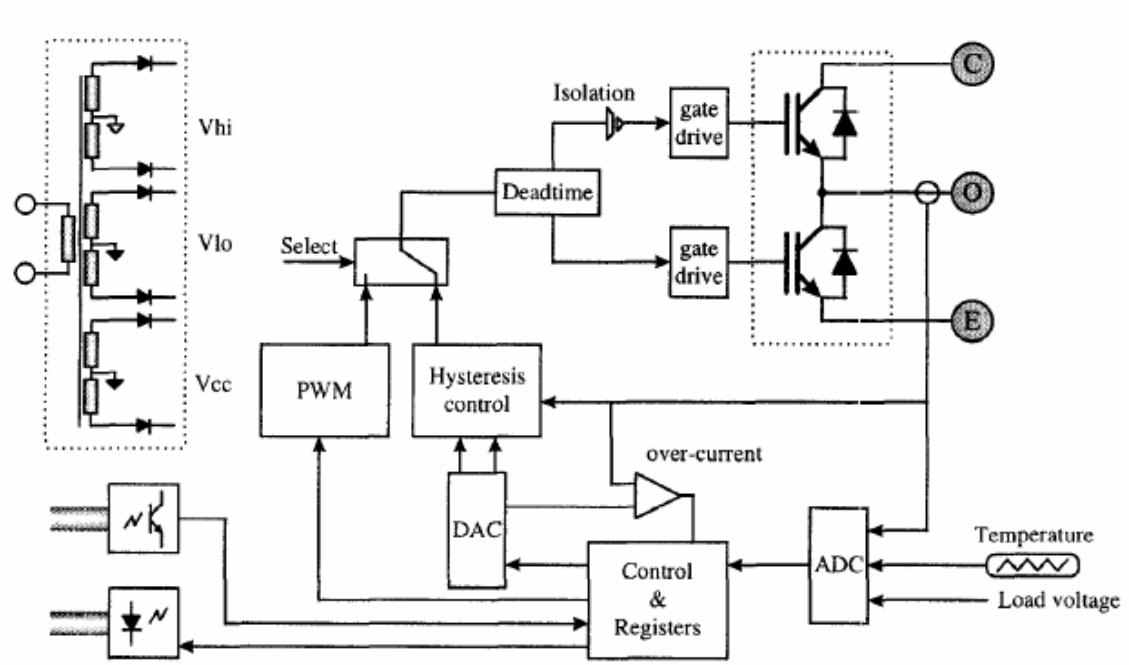


Fig. 2.2 Proposed modular controller in [6]

In [7], a high speed, programmable modular controller is investigated for large drives. Its simplified control architecture is shown in Fig 2.3.

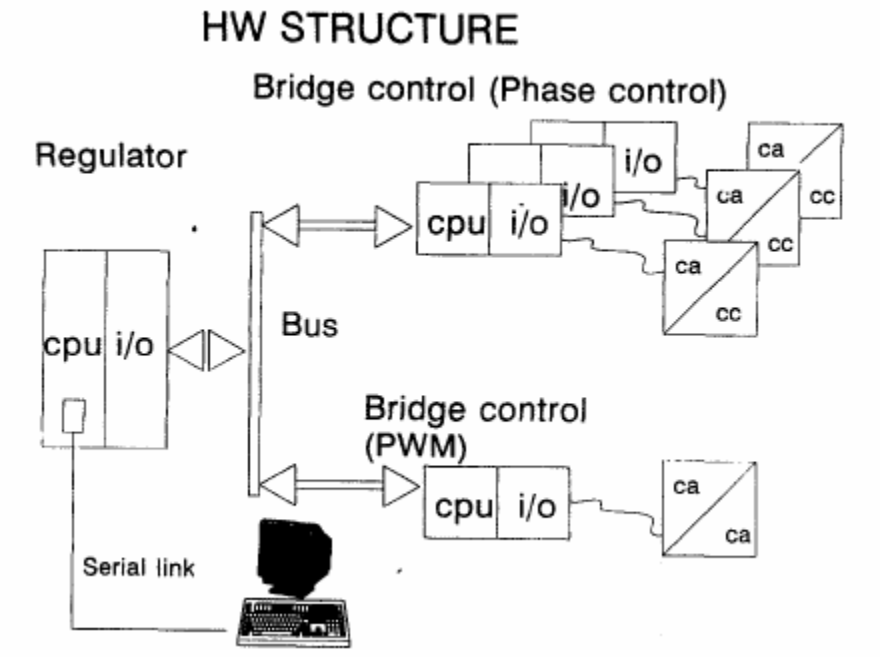


Fig. 2.3 Proposed modular controller for large drives in [7]

The regulator block includes the control functions dedicated to the true control of the motor, such as currents, voltages, speed...; the bridge control block contains all the algorithms to generate the power stage gate signals. So, the regulator performs tasks not limited to the topology of the specific converter, while the bridge controller block will define the gate sequence and the protection function of the power converter.

The communication bus of this control system links the regulator and the bridge control units and is also used for connecting the interfaces boards. The format chosen for the bus comes from the allowable lines in a backplane conceived according to the E2 standard, when DIN 41612 connectors are mounted. The lines available on these connectors are 192. The board power supply occupies 64 lines, so other 128 lines are left free for signals [7].

The data link between the master and slave controller is a slow parallel data bus.

A PEBB plug and play controller architecture is developed in [8]. The main concept is actually employing a ring communication structure. Its diagram is shown in Fig. 2.4.

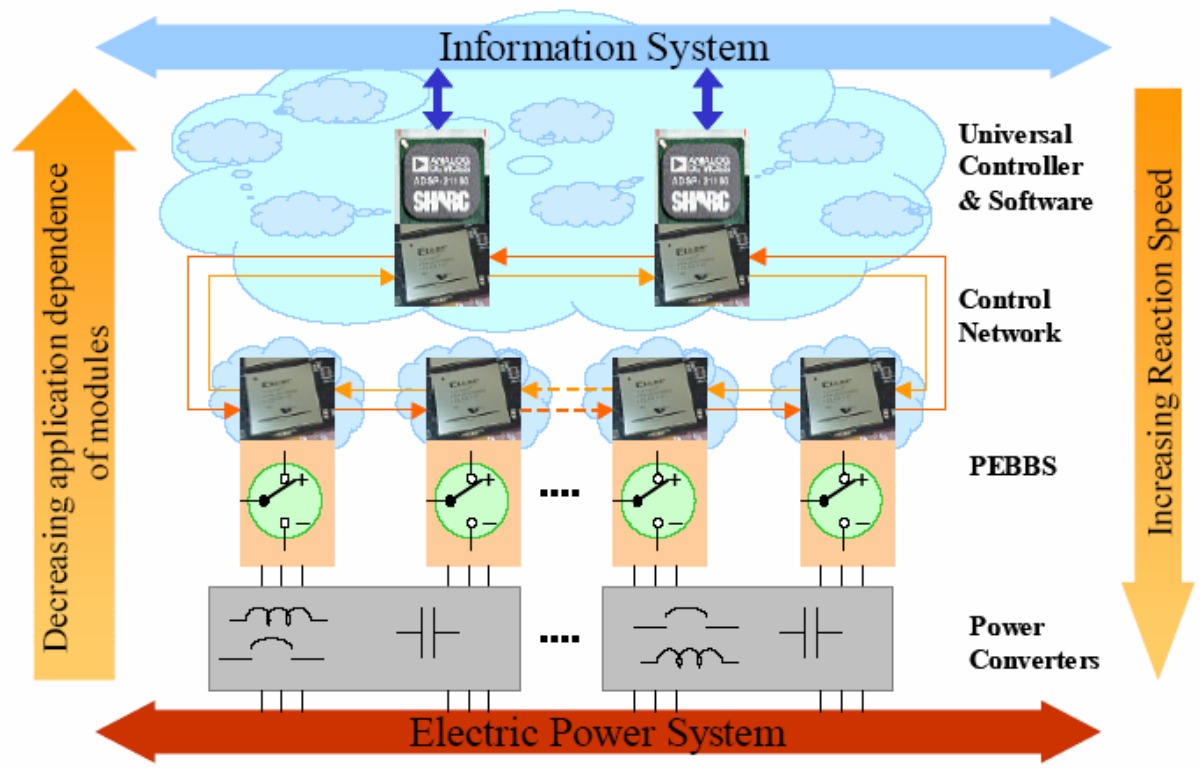


Fig. 2.4 PEBB plug and play system architecture in [8]

To connect the modules together into a working system, a synchronous control oriented communication protocol called PESNET is used. A series communication is employed between controllers and hardware managers. This modular control architecture is supposed to be able to handle a wide variety of power electronics applications varying from DC/DC converters to multilevel, multiphase and matrix topologies, requiring tight synchronization between switches [8].

Another modular controller investigation is conducted in [9] as shown in Fig. 2.5. Each of the H-bridges is provided with its own digital signal processor (DSP) controller so that the

required control and pulse-width modulation (PWM) capabilities can be evenly distributed among the numerous H-bridges connected in series (distributed computation). The resulting cascaded inverter can then be implemented through the interconnection of multiple integrated power bridges without requiring major modifications every time bridges are used in a new application or replaced/removed for service. A non-isolated serial peripheral interface (SPI) is implemented in the module for the controller communication purpose. The SPI is a high-speed communication interface suitable for transferring data between multiple processors and is used in this work for implementing a modular cascaded inverter.

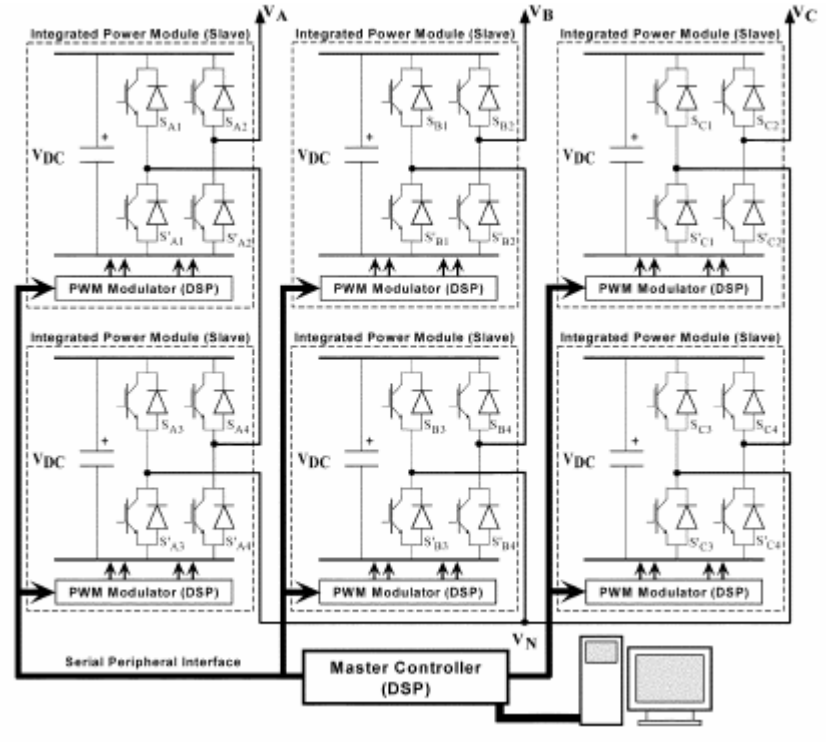


Fig 2.5 Proposed DSP-DSP modular controller in [9]

2.4 Candidate Modular Controller Architecture Comparison

2.4.1 Communication Network Topology Comparison

The modular controller architecture is actually determined by the communication network. According to the modular controller concept, a central controller is connected with multiple local controllers. To achieve better modularity and system reliability, simpler communication topology is a plus. So, let's first review the basic network topologies. The topology is also considered as the layout of the central controller and the local controllers in the network. Network topologies are categorized into the following basic types. More complex topologies could be built up with two or more of the following topologies.

1. Bus Topology

Bus networks use a common backbone to connect all devices. A single cable functions as a shared communication medium, which devices attach or tap into with an interface connector. A device wanting to communicate with another device on the network sends a global message onto the wire. Then all other devices see this message, however, only the intended recipient actually accepts and processes the message.

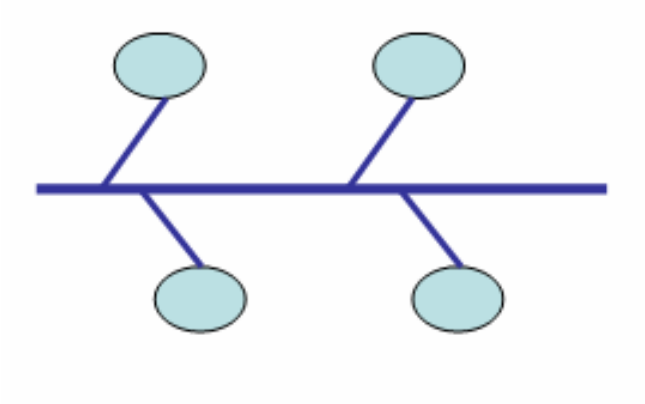


Fig. 2.6 Bus network communication Topology

Bus networks work best with a limited number of devices. If more than a few dozen devices are added to a bus, performance problems will likely result. In addition, if the backbone cable fails, the entire network effectively becomes unusable.

2. Ring Topology

In a ring network, every device has exactly two neighbors for communication purposes. All messages travel through a ring in the same direction (effectively either "clockwise" or "counterclockwise"). The diagram for a ring topology is shown in Fig.2.7. A failure in any cable or device breaks the loop and can take down the entire network. Series communication is employed in this topology. A specific communication protocol should be defined.

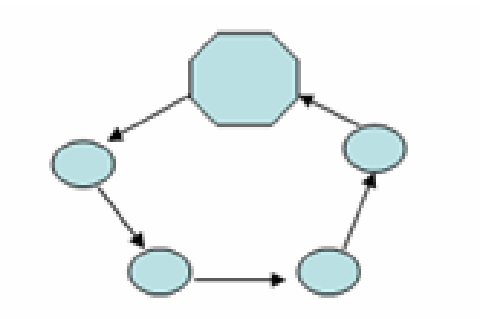


Fig. 2.7 Ring topology

3. Star topology

A star network features a central connection point called a "hub" that may be an actual hub or a central controller. Compared to the bus topology, a star network generally requires more cable, but a failure in any star network cable will only take down one device's network access and not the entire network. So, from this point of view, it has advantages over the bus and the ring structure.

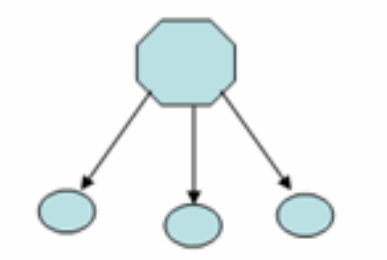


Fig. 2.8 Star Topology

4. Tree topology

Tree topologies integrate multiple star topologies together onto a bus. In its simplest form, only hub devices connect directly to the tree bus and each hub functions as the "root" of a tree of devices. This bus/star hybrid approach supports future expandability of the network much better than a bus (limited in the number of devices due to the broadcast traffic it generates) or a star (limited by the number of hub ports) alone.

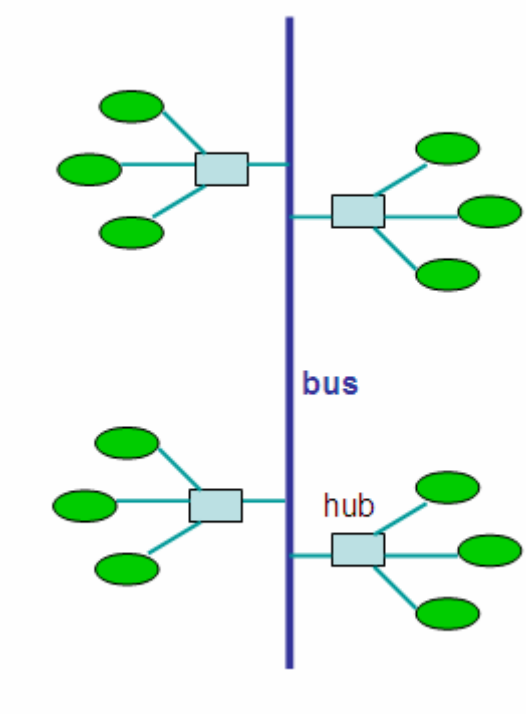


Fig 2.9 Tree Topology

The limitations of the tree topology are:

- Overall length of each segment is limited by the type of cabling used.
- If the backbone line breaks, the entire segment goes down.
- More difficult to configure and wire than other topologies.

2.4.2 Candidate Modular Controller Topologies Comparison

Since the modular controller design is mainly relying on the communication between the central controller and the distributed local controllers, the communication topology and protocol determines the whole controller architecture. There are many different topologies for data communication as shown above in a digital control system. In the following, three topologies will be investigated using the example of a thirteen level cascaded multilevel converter (3x6=18 modular converter blocks or nodes) as shown in Fig. 2.10. These are conventional star structure, ring structure and proposed optimized star structure respectively as shown in Fig. 2.11.

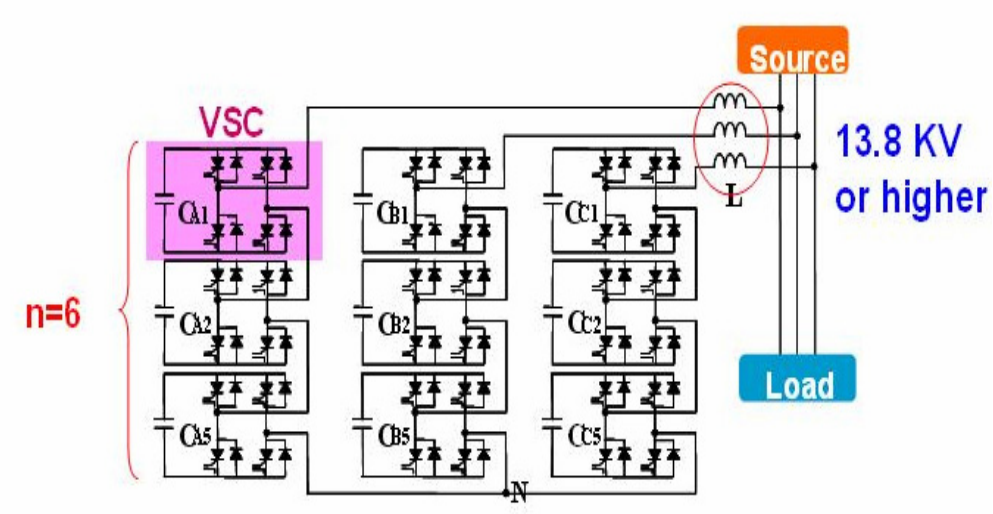


Fig. 2.10 Thirteen level Cascaded Multilevel Converter

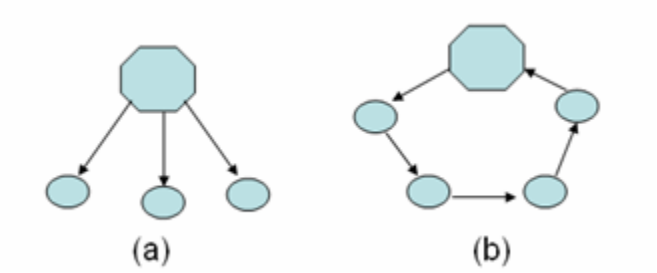


Fig 2.11 Candidate topologies for modular controller architecture

The topologies are compared according to the following aspects:

- Optical fiber numbers
- Reconfiguration capability
- Reliability
- Communication protocol
- Long distance electrical wire number

The comparison results are shown in Table 1.

Considering the following information from the converter are fed back:

- DC bus voltage for each H-bridge converter
- Temperature information for each H-bridge converter
- Over current status for each H-bridge converter
- Three phase output current
- Three phase coupling point voltage

1. Star Structure

With the conventional centralized star structure, the optical fibers required to send switch signals are 4 for each modular converter, with 18 converters to achieve thirteen level three phase system, the optical fiber numbers will be 72. For each optical fiber, a pair of optical transmitter and receiver is required.

In addition, a lot of long distance wires are required to feed back the voltage or current analog information to the central controller. There are 3 common coupling point voltage, 3 output current, 18 DC capacitor voltages, 18 temperature information and 18 over current status are required to be sent back to the central controller. So there are totally 60 long wires. These long wires lead to significant noise interference that might make the control loop malfunction and unstable.

2. Ring Structure

For a double ring structure, there is only a need of two optical fibers for the communication between each two nodes, so that only 42 optical fibers required for a 13 level CMC converter. With largely reduced connections, the system reliability will be improved. Also, there is advantage to achieve modularity with ring structure. It is easy to break the loop and add an additional node in without modifying the other nodes. However, to implement the ring structure, a complex communication protocol is required. In addition, the ring structure has not so good fault tolerant capability. If one local controller failed, it means all the communications are terminated and the whole ring failed. What is more, each local controller will receive information from the central controller with different prorogation delay due to the ring.

Table 1. Topology Comparison

	Centralized Star	Ring	Optimized Star
Optical number	4*6*3=72	2*7*3=42	2*6*3=36
reconfigure	normal	good	good
reliability	normal	normal	Good
protocol	no	complex	Simple
Long distance wires	60	no	no
others		Different delay for local nodes	

3. Optimized Star Structure (Proposed Modular Controller Architecture)

An optimized star structure for the modular controller architecture is proposed here. It is a simplified star structure. The modulation command is sending from the central controller to the local controller with only one connection in the form of serial digital data. The feedback path is implemented with an inverse star structure. Each local node has one connection with the central unit. So for a 13 level cascaded multilevel converter application, only 36 optical fibers are required. There are no long electrical wires between the central controller and the distributed local controllers.

2.5 Proposed Modular Controller Architecture

Based on the above investigations and comparisons, a novel modular controller architecture is proposed for high power electronics applications. It is shown in Fig. 2.12.

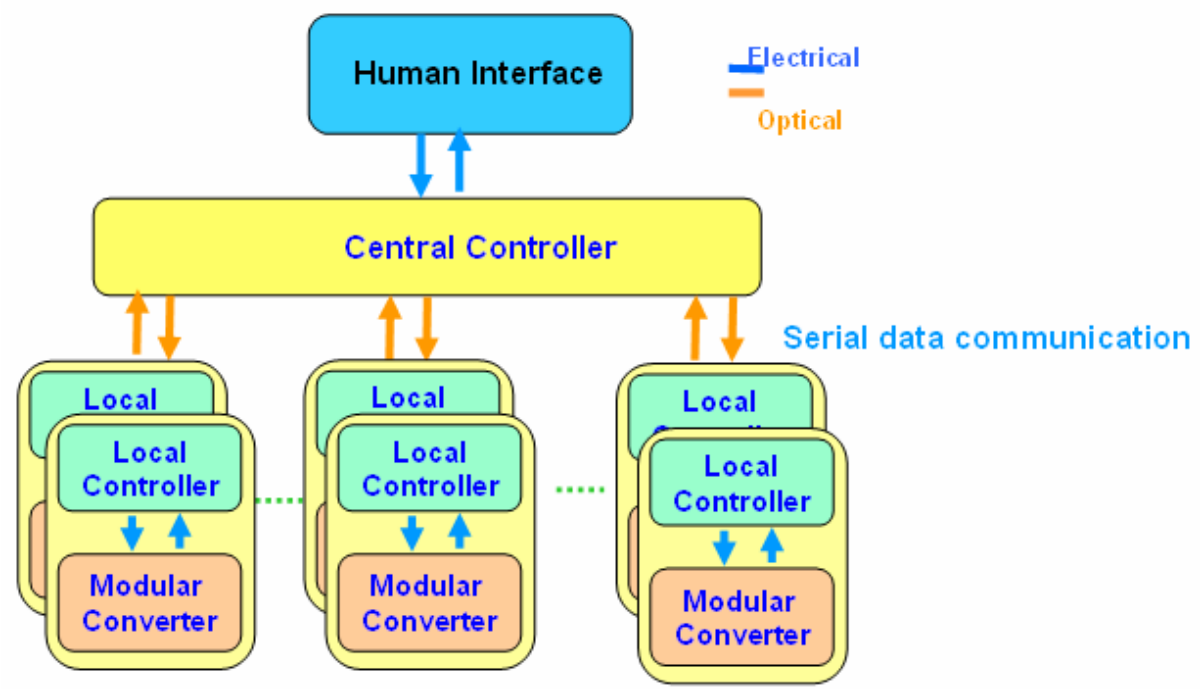


Fig 2.12 Proposed modular controller architecture

The central controller is a digital controller that gathers all the required system information and performs the control operations to generate control commands. The local controller is a local intelligent unit that includes a DSP or FPGA and analog conditioning and ADC functions. The local controller will generate the switch signals to control the switches in the modular converter.

The communication between the central controller and the local controller can be implemented with simple asynchronous serial communication. Each local controller share the same command from the central controller or take the data according to the address information and then generate various switching patterns for the modular converters. For the data return path, each local controller has an optical link with the central controller, this optical fiber will include all the voltage, current, temperature and fault information for each modular converter. They are sending back according to the predefined sequence.

The proposed modular controller architecture has the following advanced features:

- **Improved reliability & isolation**

For medium to high power applications, the converter is usually far away from the central controller as shown in Fig. 2.13. In the proposed architecture, the central controller is communicating with the local controller via optical link for data communication. One fiber is to send control commands to the local units, while the other is to get back to central controller the system voltage, current, temperature and fault information. The connection numbers are greatly reduced and this also eliminates the need of long analog wires. Thus the system reliability is highly improved. Moreover, with optical fiber link, the transmitted data is less susceptible to noise. In addition, optical link provides great isolation between the converter and the central controller. This is very important for safety operation.

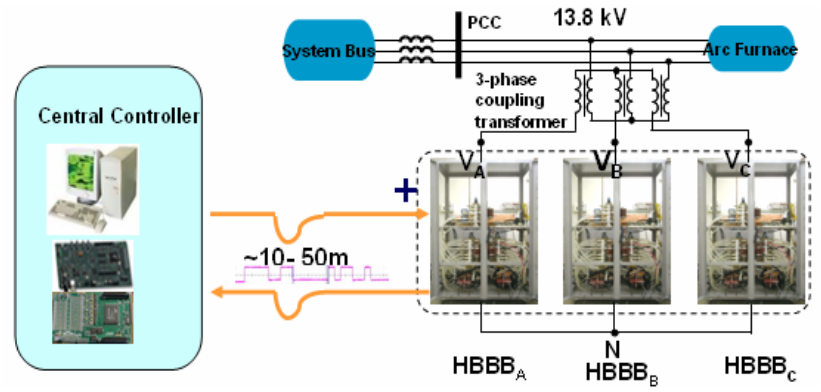


Fig. 2.13 Long distance control

- **Improved Expansion Flexibility**

With the local controller, it is much easier to expand the system. To give the example of expanding a five level cascaded multilevel converter to a seven level, only six more optical fibers are required to make the connections between the central and local controller.

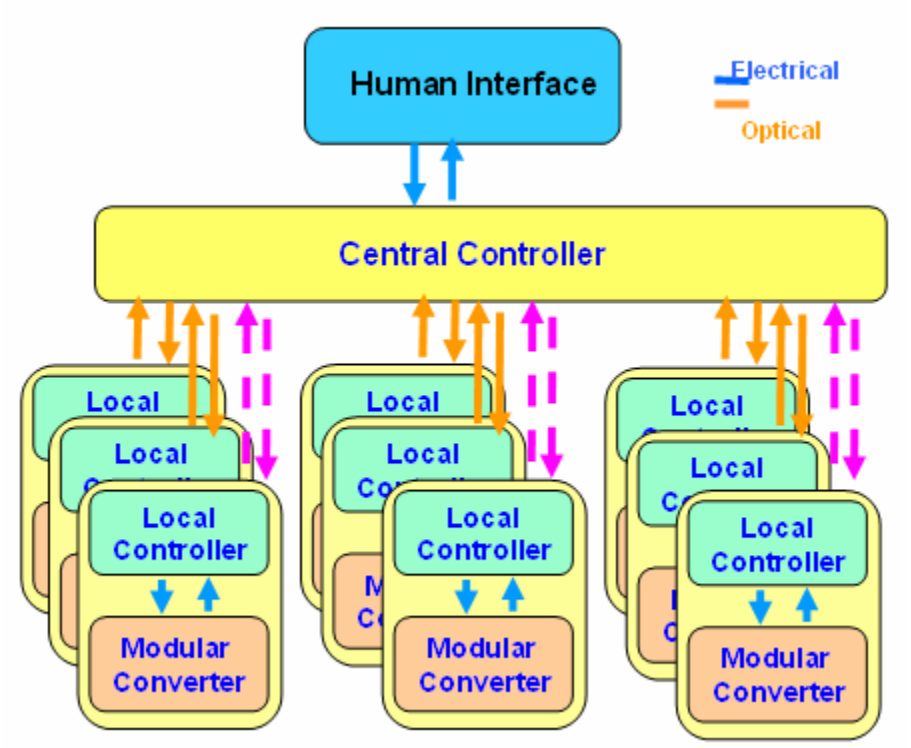


Fig. 2.14 System expansion from a 5 level to a 7 level converter

- **Asynchronous Serial data communication**

The data transmitted between the central controller and the local controller uses the asynchronous serial communication protocol that makes it possible to have largely reduced wire connections. Also, the serial transmission speed is fast enough for high power applications. The applied protocol has the features as following: simple and easy to implement in the code, no central and local controller synchronization problem, fault checking capability.

CHAPTER 3

Distributed Modular Controller Development

3.1 Overview

As described in chapter 2, a hybrid multi-tap and inverse star structure has been proposed for the distributed modular controller development. With this structure, only two optical fibers are connecting the local controller with the central controller. Very high modularity could be achieved with this architecture. Moreover, the system reliability will be largely increased.

In this chapter, the development of this distributed modular controller will be illustrated in details. Firstly, the controller hardware part will be described. The controller hardware includes the central controller that is implemented with a DSP EVM board and a FPGA daughterboard, and the local controller which is implemented with a FPGA board with ADC function. The specification of the DSP, FPGA and ADC will be shown. The selected controller hardware is just one option of many possible designs. The proposed architecture and function diagram could be implemented with other kind of DSP and FPGA.
Then, the software diagram and implementation will be shown in details. The software includes the DSP, central controller FPGA and local controller FPGA. Also, the communication protocol between the central and local controller will be illustrated. FPGA simulation in MODELSIM and experiments will also be shown to validate the design.

3.2 Hardware Section

As shown in Fig. 3.1, the central controller is composed of a DSP EVM board and a FPGA daughterboard. The local controller consists of a FPGA and ADC chips.

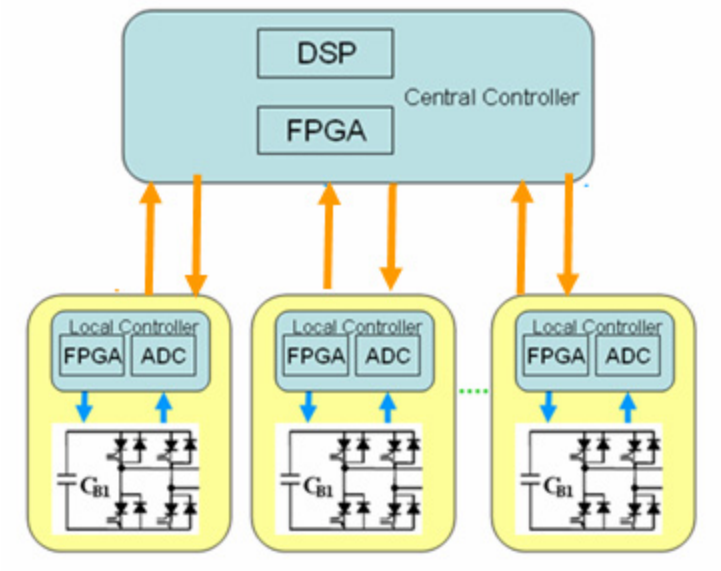


Fig. 3.1 Distributed Modular Controller Architecture

3. 2. 1 Central Controller – DSP EVM

The selected DSP EVM board is TMS320C6701 developed by Texas Instrument. TMS320C6701 is a floating point digital signal processor. The function block and CPU diagram is shown in Fig. 3.2. [10]. This DSP has the following features:

- 120, 150, 167 M Hz clock rate
- Floating point digital signal processing
- 1M-Bit on chip SRAM
- 32 bit External Memory Interface (EMIF)
- 2 32-bit general purpose timers
- JTAG boundary-scan-compatible

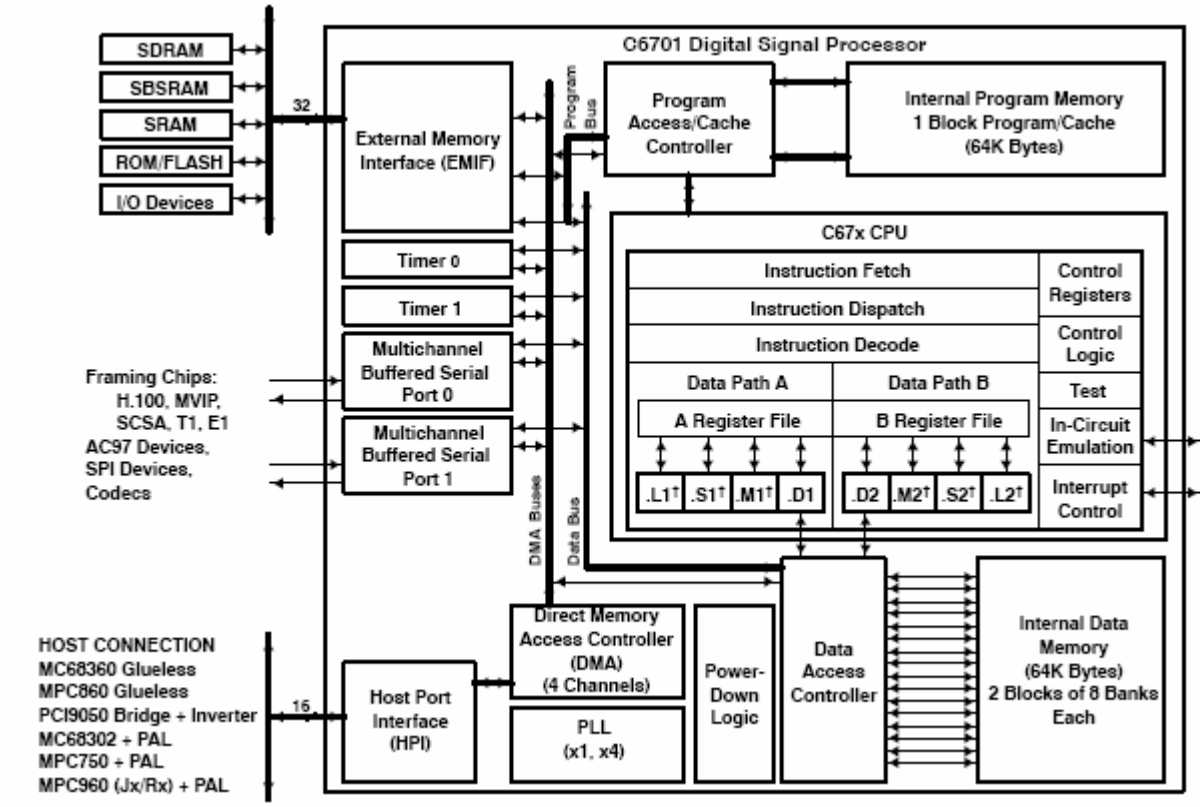


Fig. 3.2 DSP 6701 function block and CPU diagram [10]

3.2.2 Central Controller – FPGA daughterboard

The FPGA daughterboard is commercially designed by SIGNALWARE for DSP6701 EVM board. The EVM board has an expansion slot that has both the expansion memory interface (EMIF) and an expansion peripheral interface which allows the daughterboard full access to all of the DSP's resources [11]. Also, the FPGA works as an expansion memory of the DSP. Its registers are assigned addresses in the DSP, so the FPGA register data could be directly accessed in the DSP.

The FPGA board AED-106 developed by SIGNALWARE corporation includes a powerful XILINX XCV300 FPGA chip. Moreover, there are 4 A/D chips on the board that provides 16 A/D channels which could be sampled up to 6MS/s with 12 bits.

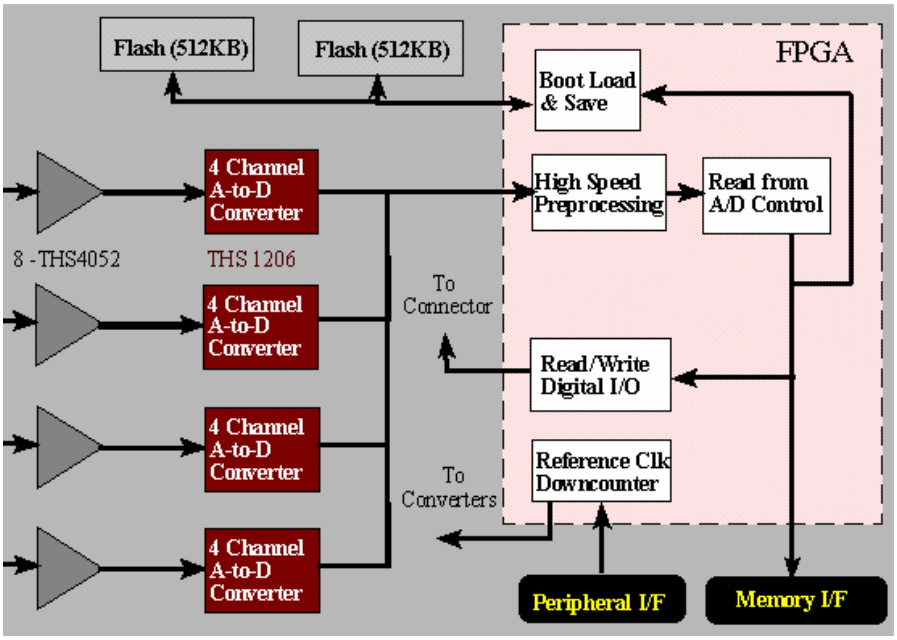


Fig. 3.3 FPGA daughterboard AED-106 function diagram [11]

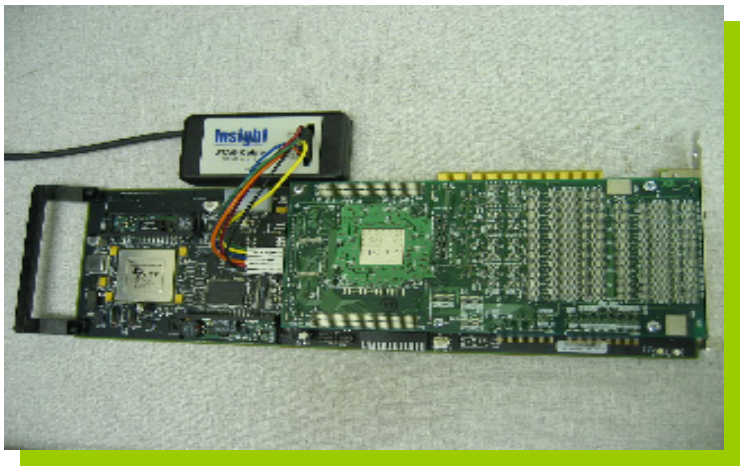


Fig. 3.4 Central controller

All the signals from the EVM external interface will be fed into the FPGA. The FPGA is also connected to the control lines on the flash, and all digital lines of the A/D converters. The FPGA controls the interface between the EVM board and the daughterboard. It can use the address lines from the EVM to decode reads/writes to the flash, to the digital I/O, and to the converters. For the C6701EVN, the addresses are 0x01400000, 0x01500000, and 0x01600000 respectively.

There are 24 digital I/Os on this board. They can be configured to be input and output. The 24 digital I/Os are buffered with 3 digital transceivers 74ABR245. The buffers can be enabled or disabled all together, and the directions are set by the FPGA in groups of eight. The FPGA could be programmed directly with the JTAG cable. However, the program in FPGA will be cleared while the power is off. The flash memory XCV20 accompanying with the FPGA could also be programmed. It can be configured to load the program to FPGA automatically. Also, the program in the flash memory will not be cleared while the power is off.

3.2.3 Local Controller

The local controller is the distributed modular controller. It is required to be reconfigured to support power level specific power stage. So it should have an intelligent unit. Here, a FPGA is used to perform this intelligent function. Also it should be able to collect and process the analog sensing data. As an example of the design, a controller that has been designed and developed by Virginia Tech[8] is selected. Its picture and function diagram are shown in Fig. 3.5 and Fig. 3.6 respectively.

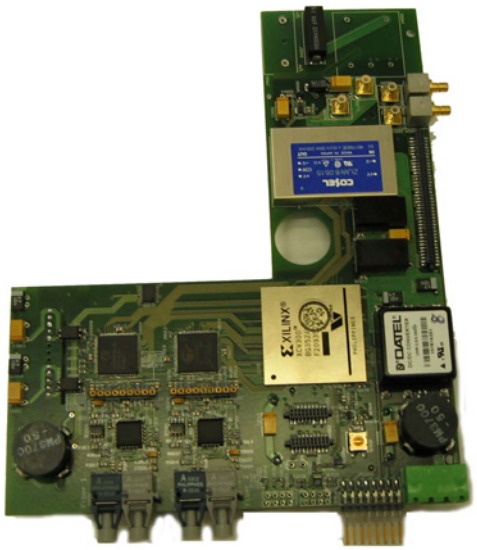


Fig. 3.5 Local Controller

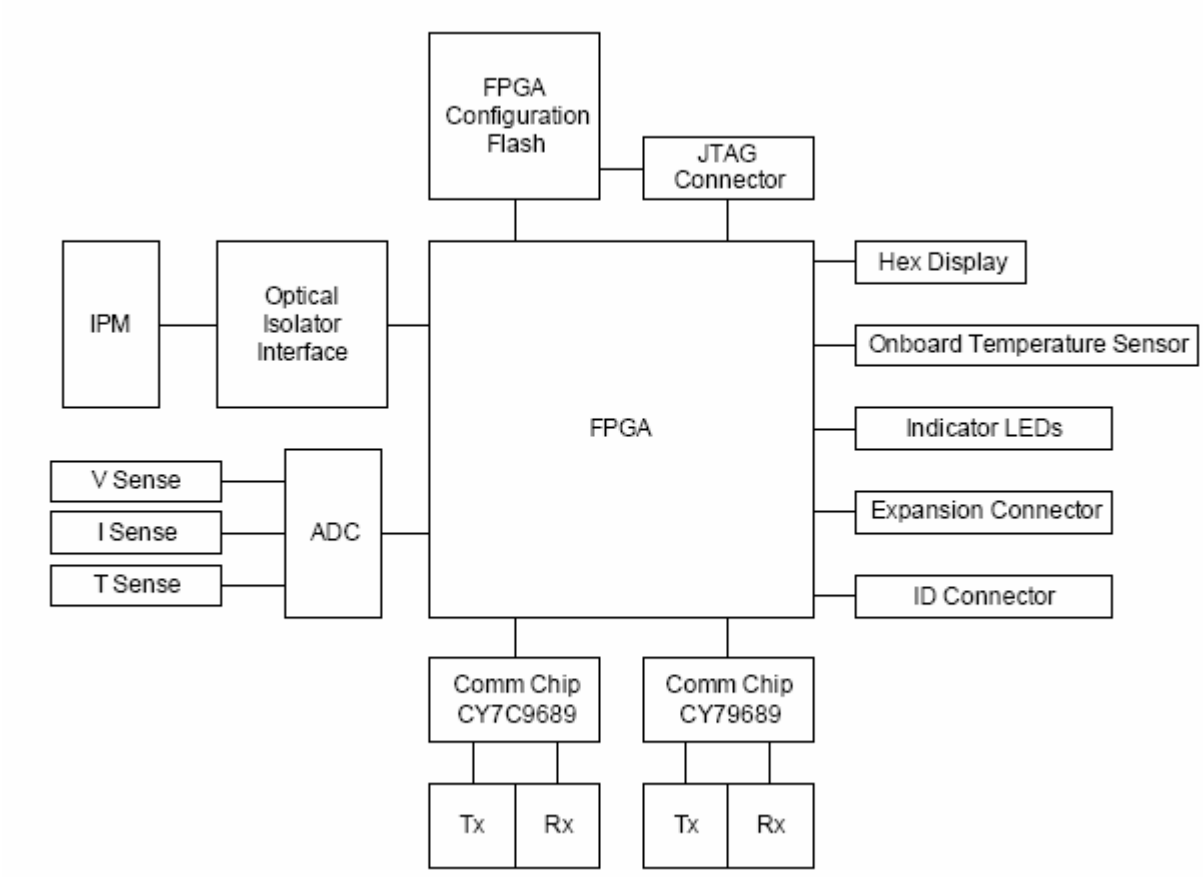


Fig. 3.6 Local Controller Function Diagram

The FPGA is XCV300 that is the same as the central controller. It is a very powerful FPGA sufficient to implement the PWM generator, ADC control and signal processing functions. The maximum clock rate is 80MHz. There are two ADC chips AD7862 on the board that includes 8 A/D channels. The highest sampling rate is 125KS/s that is fast enough for the high power electronics data sensing.

3.3 Software Section

In this section, the central controller and the local controller software function and implementation will be shown in details. Before this, a clear communication protocol should be defined between the central and local controllers.

3.3.1 Communication Protocol

To develop a distributed modular controller, a reliable and simple communication protocol is quite critical. Through the literature research, it is found that some people develop a specific communication protocol for the specific modular controller architecture. A good example is the PESNET protocol defined for a ring structure in Virginia Tech [9]. Before we come to see some features of this protocol and define the desired protocol we want to use, let's firstly review what the general requirements for a communication protocol are.

3.3.1.1 Communication Protocol Definition and Requirements

A communications protocol is the set of standard rules for data representation, signaling, authentication, and error detection required to send information over a communications channel. An example of a simple communications protocol adapted to voice communication is the case of a radio dispatcher talking to mobile stations. The communication protocols for digital computer network communication have many features intended to ensure reliable interchange of data over an imperfect communication channel.

To design a communication protocol, the following requirements should be met:

1. **Effectiveness**

   For a protocol to be effective it needs to be specified in such a way that engineers, designers, and in some cases software developers can implement and/or use it.

2. **Reliability**

   Assuring reliability of data transmission involves error detection and correction, or some means of requesting retransmission. It is a truism that communication media are always faulty. The conventional measure of quality is the number of failed bits per bits transmitted. This has the wonderful feature of being a dimensionless figure of merit that can be compared across any speed or type of communication media.

   Communication systems correct errors by selectively resending bad parts of a message. For example, in TCP (the internet's Transmission Control Protocol), messages are divided into packets, each of which has a checksum. When a checksum is bad, the packet is discarded. When a packet is lost, the receiver acknowledges all of the packets up to, but not including the failed packet. Eventually, the sender sees that too much time has elapsed without an acknowledgement, so it resends all of the packets that have not been

acknowledged. At the same time, the sender backs off its rate of sending, in case the packet loss was caused by saturation of the path between sender and receiver.

3. Resiliency

Resiliency addresses a form of network failure known as topological failure in which a communication link is cut, or degrades below usable quality. Most modern communication protocols periodically send messages to test a link. In phones, a framing bit is sent every 24 bits on T1 lines. In phone systems, when "sync is lost", fail-safe mechanisms reroute the signals around the failing equipment.

So, with the above descriptions, a more clear idea is obtained for the protocol to be applied on the distributed modular controller. It should be simple and reliable.

3.3.1.2 Previously Defined Communication Protocol

To start with the defined communication protocol, let's have a look at the PESNET protocol defined in Virginia Tech for a daisy chain ring structure. [9]

Table 2. PESNET data packet [9]

	ADDR	BYTE1	BYTE2	BYTE3	BYTE4	CRC
Size	4 Bits	8 Bits	8 Bits	8 Bits	8 Bits	8 Bits

Table 3. PESNET synchronization packet [9]

	CMD	ADDR3	PADDING	ADDR2	PADDING	ADDR1
Size	4 Bits	8 Bits	8 Bits	8 Bits	8 Bits	8 Bits

There are two types of packets sent in PESNET. The first is a data packet, which is designed to deliver information to a specific slave device on the network. The master initiates all packets. The format of the data packet is shown in Table 2.

Each slave receives data from the master, and returns sensor data back to it in a packet with the same ADDR field. The master collects this data, and uses it for closed loop control.

After information is sent to all the nodes, the synchronization packet is sent to activate the data. The synchronization packet has the structure as shown in Table 3. The concept is that each node will synchronize when they receive their own address in the data field.

This defined protocol has the following drawbacks:

- Strong dependency on the node order

  If any extra node is added into the ring, the extra transmission delay would desynchronize the nodes due to the method used to synchronize them.

- Each node is a failure point

  If any point is failed, the whole loop from the master to slave and back to master is broken, that means the whole controller is “dead”.

- Lack of variety of the data packet

  For example, duty cycle is always located in the position 1 and 2 of the data packet. If more than 4 bytes are needed to completely control the node, the node would have to have two addresses.

- The number of nodes is limited

  The number of nodes are limited to the number of the commands. If more nodes are needed, it is hard to add them. This limits the application of multiple modular controllers.

3.3.1.3 Proposed Communication Protocol (UART)

In the proposed distributed modular controller architecture, a simple and easy implemented protocol is defined. The basic idea is coming from the **Universal Asynchronous Receiver Transmitter (UART).**

In the high power electronics applications, long distance between the central controller and the distributed local controllers is quite possible. To send the digital bits over a long distance, sending parallel data will be more expensive than sending serial data. Moreover, less wire connections will reduce the failure nodes for the whole system, hence increasing the system reliability. So, in my proposal, serial communication is the first choice.

The possible transmission medium for serial communication are traditional wires, optical fiber, infrared, wireless blue-tooth in its serial port profile (SPP) and the DC_LIN for power line communication. To get low signal attenuation and good signal integrity and isolation, optical fiber is applied in the proposed modular controller application.

Serial Communication includes synchronous serial communication and asynchronous serial communication.

- Synchronous Serial Communication

Synchronous serial transmission requires that the sender and receiver share a clock with one another, or that the sender provide a strobe or other timing signal so that the receiver knows

when to “read” the next bit of the data. In most forms of serial Synchronous communication, if there is no data available at a given instant to transmit, a fill character must be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, and synchronous communication can be more costly if extra wiring and circuits are required to share a clock signal between the sender and receiver.

- Asynchronous Serial Communication

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word which are used to synchronize the sending and receiving units.

By convention, teletype-style UARTs send a "start" bit, five to eight data bits, least-significant-bit first, an optional "parity" bit, and then a "stop" bit. The start bit is the opposite polarity of the data-line's normal state. The stop-bit is the data-line's normal state, and provides a space before the next character can start. The parity bit can either make the number of bits odd, or even, or it can be omitted. Odd parity is more reliable because it assures that there will always be a data transition, and this permits many UARTs to resynchronize.

Asynchronous Serial Communication is proposed for the central controller and local controller. Actually there are two paths between the controllers. Let’s define the name as Forward path and Feedback Path. The forward path is from the central controller to the local

controller. The data being sent rely on the modulation scheme in the controller design. For example, the modulation schemes may include fundamental switching, phase shift SPWM, SVM and etc. For a phase shift SPWM modulation method, usually the modulation signals will be sent from this path to the local controllers. It requires higher accuracy. Moreover, along with the modulation signals, the synchronization signal should also be sent to the local controllers. The feedback path is from the distributed local controllers to the central controller. Usually the A/D data including voltage, current, temperature and fault status will be sent in this path. A simpler protocol could be defined for it.

In Fig. 3.7, the serial communication protocol for the forward path is defined.

Fig. 3.7 Forward Serial Communication Protocol

If there is no data for transmitting, the transmit line will be in idle. It is permitted because the asynchronous serial communication is a “self synchronization” scheme. In idle mode, the transmit line is always ‘1’.

When a modulation signal is sent to be transmitted, a start bit ‘0’ will be the first bit to be sent. It is used to alert the receiver of the local controller that a new data is to be sent, and force clock in the receiver to be synchronized with the transmitter. These two clocks should be accurate enough in the data transmission. More than 10% of the frequency drift will affect the data transmission accuracy.

After the start bit, for a 16 bit modulation data, the least significant 8 bits (LSB) will be sent. Each bit will be transmitted with the same speed. As mentioned before, the transmission medium in the application is optical fiber, so the propagation delay on the optical fiber will be very small.

A parity bit for the first 8 bit data will be sent in the next. After the first 8 bits, two bits stop and 2 bits idle are defined. Actually it is not so necessary to have so many bits stop and idle here, it could be changed to 1 bit stop and 1 bit idle.

Then, the most significant 8 bits (MSB) are sent according to the same protocol as the LSB. The parity of these 8 bits data will also be checked in the receiver.

When the whole data has been sent, two bits of stop '1' will indicate the completeness of the transmission. The receiver will calculate the parity of the received data and compare with the received parity bit. If they are not the same, the received data will be disregarded.

The feedback path serial communication protocol is defined in Fig 3.8.

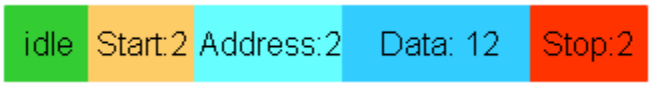


Fig. 3.8 Feedback Communication Protocol

When there is no data transmitting, the line is always idle. The receiver will always get '1'. In this development, all the feedback analog sensing signals are converted to a 12 bit data in the local controller and then they are sending back to the central controller in sequence. Each data will be appended with an address information for identification.

If there is a new data to be sent, 2 start bits will be firstly received by the central controller. It is for the purpose of detecting the starting of a new data accurately. Then, 2 bits of address will be sent before the 12 bits data. To be simple, parity information is not included. There is also another reason for eliminating the parity checking for the ADC data. A/D converter has its accuracy and the input analog signals have noise interference, so the least significant bits may be time varying. So parity checking will give a wrong result for these kinds of situations. After the 12 bits data are transmitted, 2 stop bits will be sent to the central controller to indicate that a data is transmitted completely.

3.3.2 Central Controller Software Module Design

3.3.2.1 DSP EVM

The DSP in the central controller will perform the main control function and communicate with the FPGA. The control scheme is application specific. A control diagram for a 3 level voltage source converter (VSC) for STATCOM application is shown in Fig. 3.9.

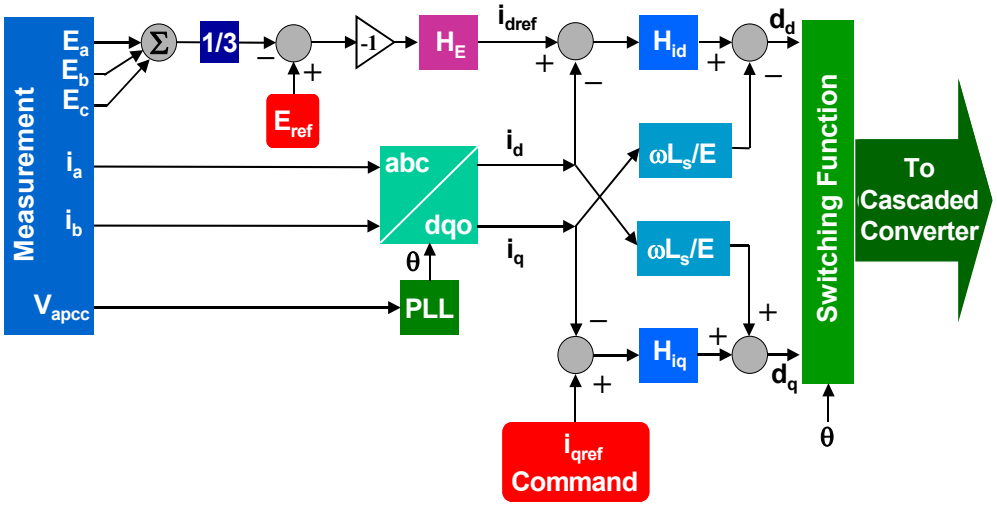


Fig. 3.9 Control Scheme for a 3 level VSC for STATCOM application

This is a general decoupled power controller for STACOM application. The DSP will collect the voltage and current data from the FPGA registers, then generate the modulation signal d. This signal will be sent to the FPGA for more processing.

Moreover, the DSP will perform the system operation. It is also application specific.

A STATCOM operation flowchart is shown in Fig. 3.10.

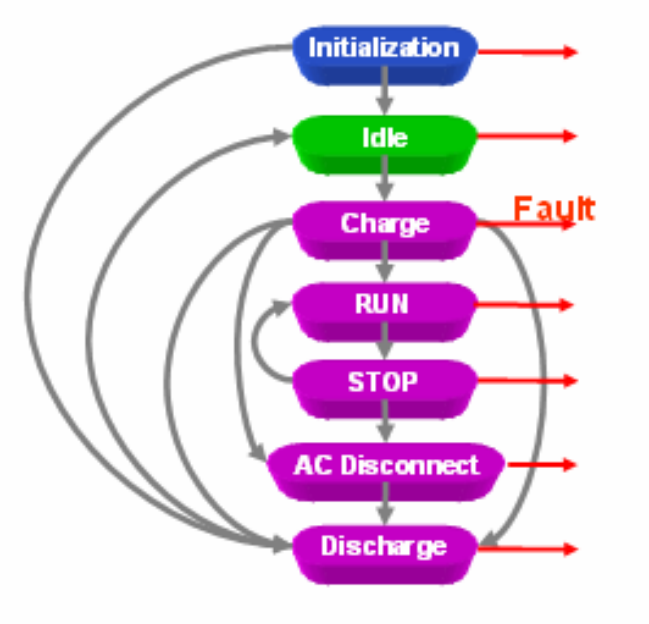


Fig. 3.10 STATCOM Operation Flowchart

The operation modes include initialization, idle, charge, run, stop, ac disconnect and discharge. The DSP will send out commands to charge the dc capacitors in the STATCOM upon the “charge” command is initiated. After the DCBUS voltage is built up, with the “run” button pressed, STATCOM will operate in standby, capacitive or inductive modes. “stop” mode is to stop the STATCOM operation and “ Ac disconnect” is to cut off the connections between the converter and the ac power grid by turning off the ac circuit breaker. After that, the DC capacitors will be discharged.

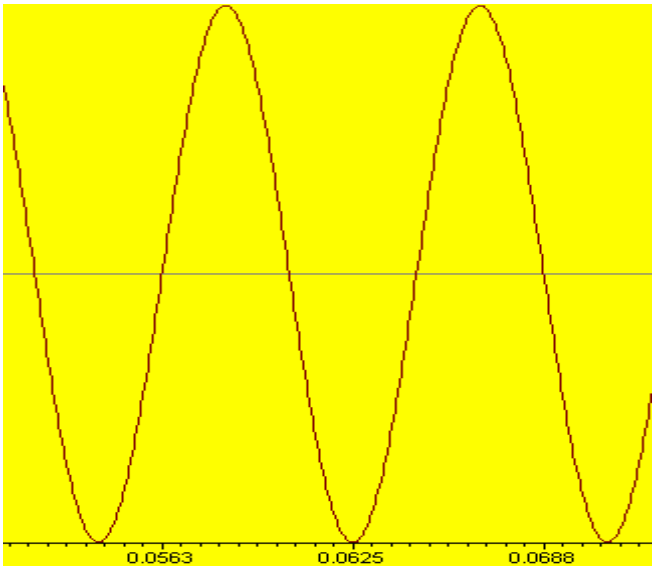


Fig.3.11 Modulation signal generated by DSP control loop

In Fig 3.11, the modulation signal generated by the DSP close loop control is shown. It is a sinusoidal PWM modulation signal that is used for the local controller to generate SPWM switch signals.

3.3.2.2 Central Controller FPGA Software Module

1. **Block Diagram**

The software block diagram of the central FPGA is shown in Fig. 3.12. The central FPGA is applied with XILINX XCV300. It has 322,970 system gates, 6912 logic cells and 316 maximum I/Os. Its working frequency is 80MHz. In the selection of FPGA, resource is quite important. If the hardware resource is not enough, the difficulty to write a successful VHDL code for the FPGA will be largely increased. Sometimes, resource optimization in the source code is not so effective. Some problems have been met with the software design for a 3 level converter testbed. Due to the FPGA resource limitation, some of the A/D channels are unable

to be effective. So in this design, a powerful FPGA is selected to avoid such kind of problems.

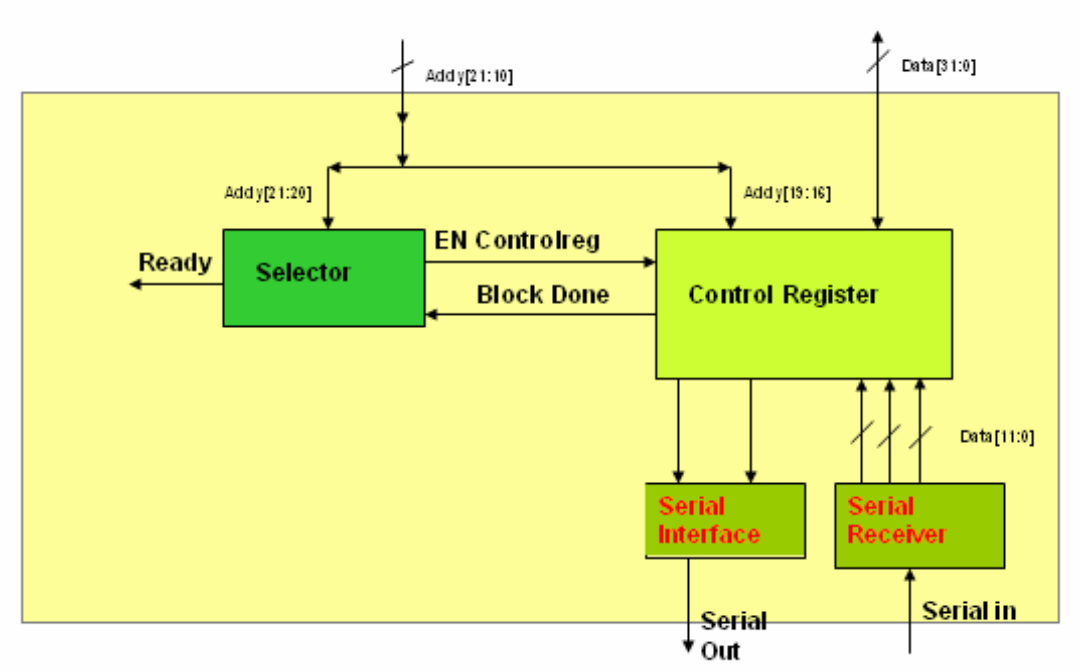


Fig. 3.12 Central Controller FPGA Block Diagram

The software in the central FPGA includes 4 modules. The code is written in VERILOG Hardware Description Language(VHDL). Due to its special tight connections with the hardware, some people also call it as “very hard description language” for fun. It is true that the code will not definitely work even the simulation is correct. Synthesis must be done to transfer the behavior model in VHDL to register level. After this, implementation and routing will be done to reach the gate level configuration in the FPGA.

All the 4 modules are included in a top entity. That is the highest hierarchy in the VHDL code. In this top entity, the I/O interface of the FPGA is defined. Each port is assigned a name and attributes.

The function for each module is shown in Table 4.

Table 4. Function of the software modules in Central FPGA

Module	Function
Selector	Receive commands from DSP and enable the control register
Control Register	Store all the data in registers including A/D, duty commands, and other operation commands.
Serial Interface	Convert the modulation signal from parallel to a serial data and send out according to the defined protocol
Serial Receiver	Detect and store the serial data from the local controllers, then send the converted parallel data to control register

2. Selector & Control Register

The selector works like the chip enable for FPGA. The selector takes in ADDR(21:10) from the address line of the DSP and enables the other modules in the FPGA.

The control register defines the address registers and data registers. The A/D data and PWM duties are stored in the data registers. When the DSP address line sends the address, the related data register will be sent to the data bus.

The data transaction diagram is shown in Fig. 3.13. When a write operation is initiated, the selector will enable the control register, then the control register write the received data in the register and set the Done_controlreg to '0' at the same time. When the data store is completed, Done_controlreg is set to '1' that indicate the operation is finished. Then, the selector will disable the control register. The whole write operation is completed.

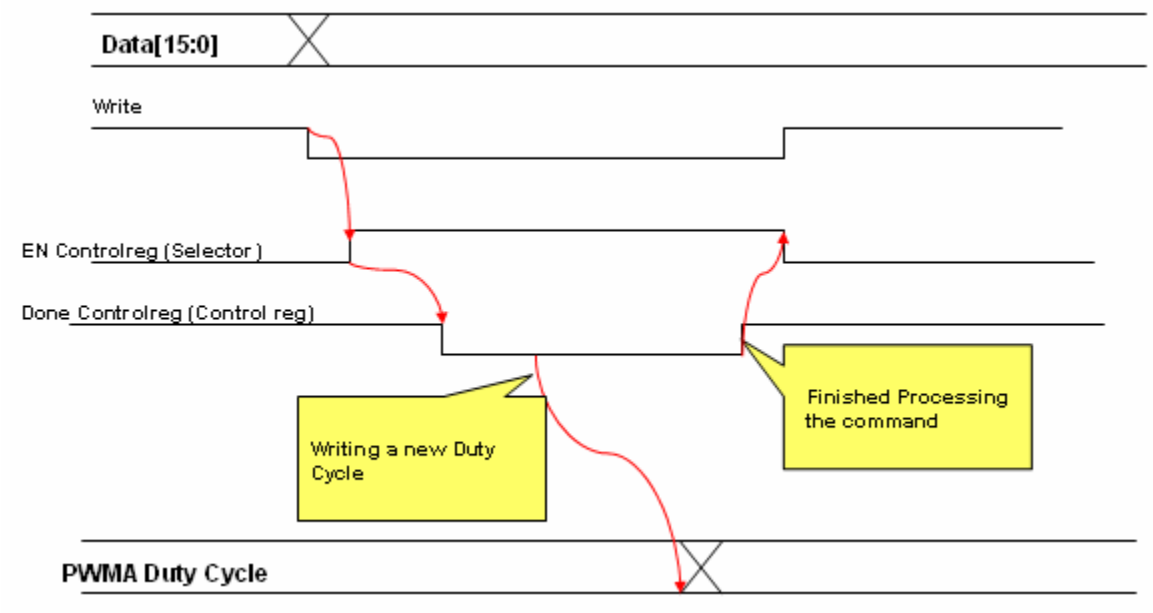


Fig. 3.13 Data Transaction for a Write Operation

3. Serial Interface

The serial interface receives new data from the control register, and then transforms this parallel data to a serial data, and sends this data out in sequence based on the pre-defined protocol.

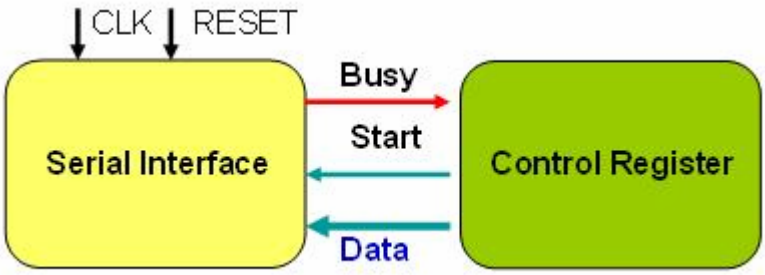


Fig. 3.14 Serial interface I/O

As shown in Fig. 3.14, the serial interface have the following input ports: clock, reset, start and data. The data is a 16 bit data that is sent from the control register. In this application,

this data is the modulation signal sent from the DSP. Whenever a new modulation signal is received in the control register, the data is stored and transferred to the serial interface. At the same time, the control register will send out a start signal to the serial interface to start the conversion. While the serial interface receive the data and the start command, the block starts to work and send back a busy signal to the control register to prevent the other data being sent in.

The serial interface is implemented with a state machine. A state type is defined in the VHDL code. It is shown in Table 5.

Table 5. Finite State Definition

```
type SERIAL_STATE is (idle,start_tran1,
                      transmit_0,
                      transmit_1,transmit_2,
                      transmit_3,transmit_4,transmit_5,
                      transmit_6,transmit_7,parity_tran1,
                      stop1,stop2,
                      idle1,idle2,
                      start_tran2,
                      transmit_20,transmit_21,transmit_22,
                      transmit_23,transmit_24,transmit_25,
                      transmit_26,transmit_27,
                      parity_tran2,
                      stop21,
                      stop22
                      );
```

The state machine has 27 states in the data transformation. An internal clock with 1/16 of the global clock frequency is generated to control the state transition. As the global clock in the application is half of the DSP frequency and the DSP frequency has been configured to be 100MHz, so the global frequency is 50MHz, and the transition clock is 3.125MHz. Then the whole time to complete sending a 16 data will take 8.64us. This speed is fast enough for the

high power applications. The high power converter is usually switching at low frequency as kilo hertzs. The state machine transition diagram is shown in Fig. 3.15.

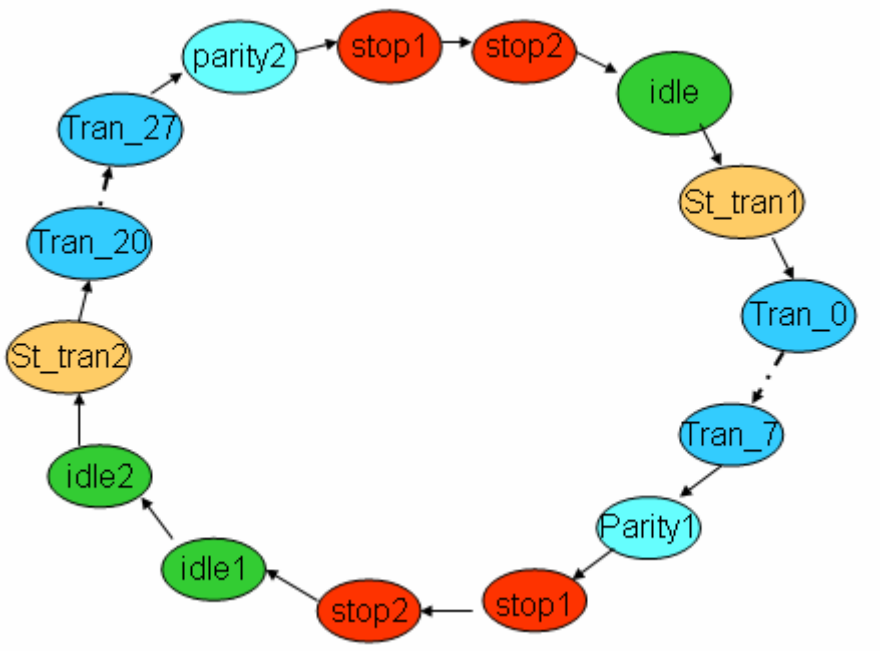


Fig.3.15 Serial Interface State Transition Diagram

To validate the design, a test bench is set up and simulated in MODELSIM. The available MODELSIM version is working under UNIX environment.

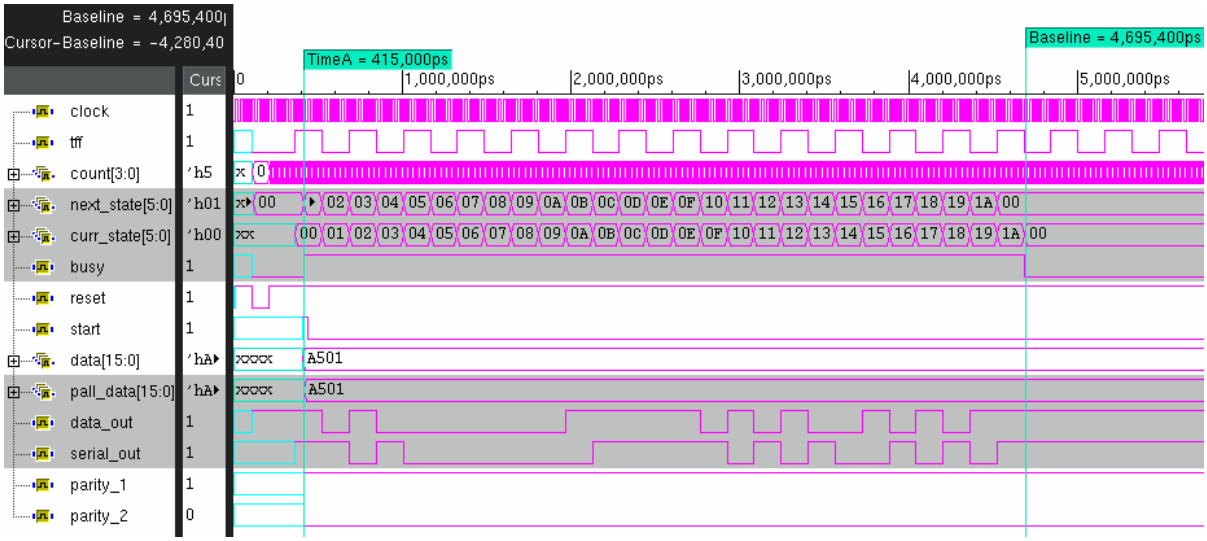


Fig. 3.16 Serial Interface MODELSIM Simulation

In the simulation, a parallel data “A501” is sent to the serial interface. At the same time, the start signal is enabled. The serial out is the data after the transformation.

As VHDL is a hardware language, the function may not work after synthesizing into the FPGA hardware even the simulation is working well. Experiments are conducted to validate the design. Firstly, The DSP code is modified to send out a fixed duty command to the FPGA. In the DSP code, a data is sent out according to the equation 3.1 shown in the following.

CDUTYA=(DUTYA*0.5+0.5)*(FFFF)…….Equ. 3.1

So, if DUTYA=0.0, then the send out duty command is “7FFF”. If the DUTYA=0.7, then the CDUTYA= “D998”.

The experiments are conducted with the DSP and FPGA board connected. A DSP command is sent from the DSP CCS software environment. The serial interface output is connected to one digital output pin of the FPGA. This pin is measured with an oscilloscope as shown in Fig. 3.17. Two duty commands 0.0 and 0.7 are tested for validation.

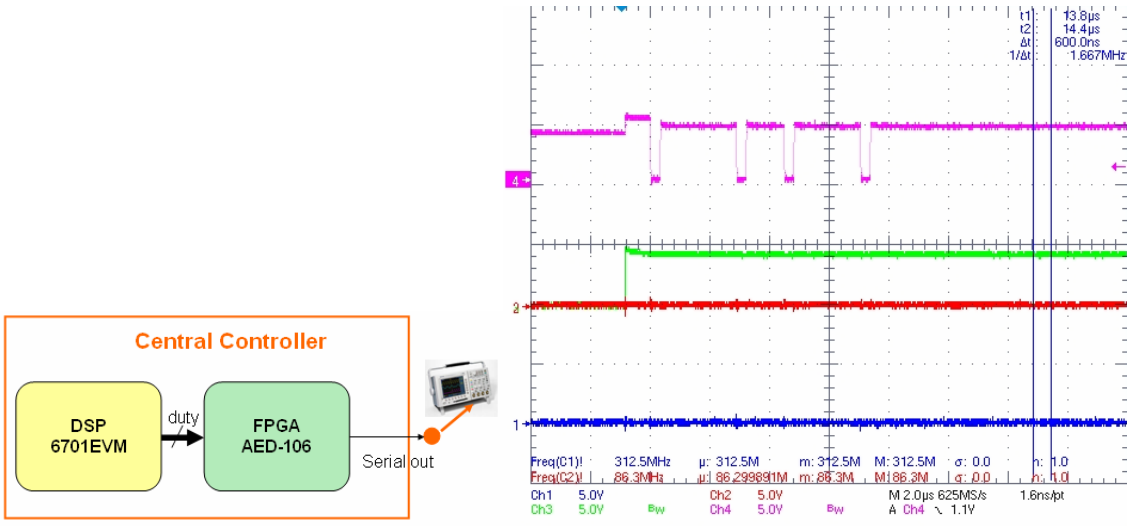


Fig. 3.17 Serial Output for Duty=0.0

In Fig. 3.17, Ch4 is the serial out signal for Duty 0.0 in the above waveform.

Da:0.0 => X"7FFF" => 0111111111111111

idle	start	Data:LSB 8	parity1	Stop:2	Idle:2	start	Data:MSB 8	parity2	Stop:2

Data measured from the scope waveform

1	0	11111111	0	11	11	0	11111110	1	11

Fig. 3.18 Data Validation for Duty = 0.0

From the waveform, the data for each bit are gotten and shown in Fig 3.18. The measured data is exactly matched with the data sent from DSP.

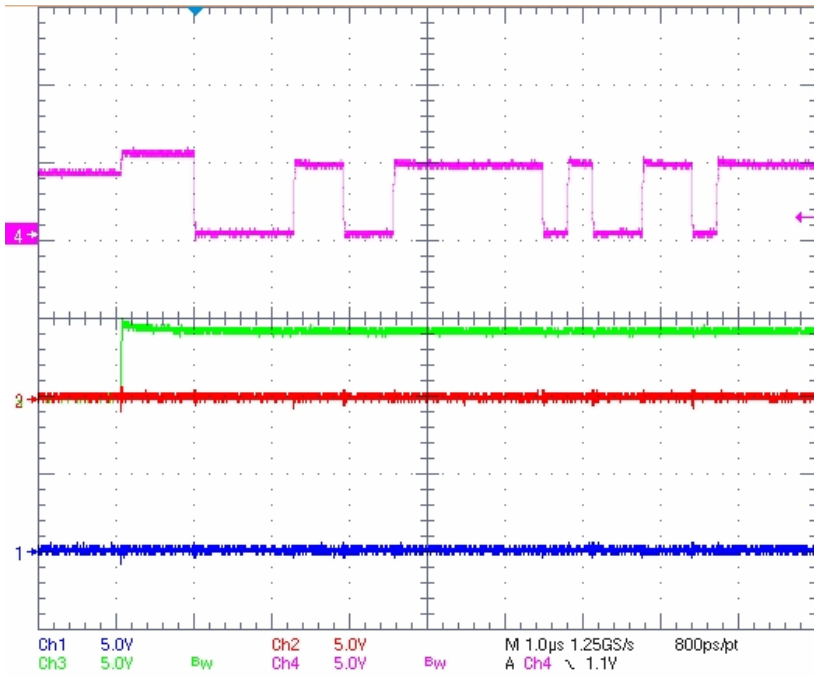


Fig. 3.19 Serial Interface Output for Duty = 0.7

Ch4 is the measured the serial interface output waveform.

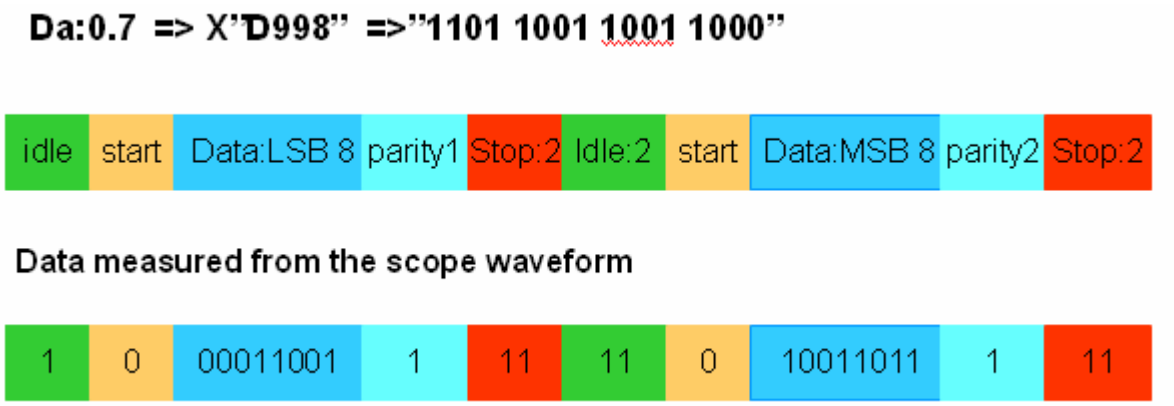


Fig. 3.20 Data Validation For Duty = 0.7

The measured bit by bit are shown in Fig. 3.20. It is exactly matched with the duty sent from DSP. So with the two experiments validation, it is sure that the serial interface is working well and the data is accurate.

4. Serial Receiver

The serial receiver receives the serial data from one designated FPGA input pin. The serial data is transmitted from the local controller. It is based on the defined asynchronous serial communication protocol.

The most important problem in the serial receiver is to detect the start bit of each data and find the address information. To make sure the start bit is detected precisely, each data is sent out a certain time period after the previous data. In this certain period, the transmission line stays in idle mode. A counter is defined to detect the '1' on the line. While the '1's are counted over a setting number and the input is transited to '0', the serial receiver will consider it is the starting point of a new data. If the input is transited to '0' while the ones counter does not reach the setting number, the ones counter will be reset because it should be in the middle of the data transmission.

The function diagram of the serial receiver is shown in Fig. 3.21.

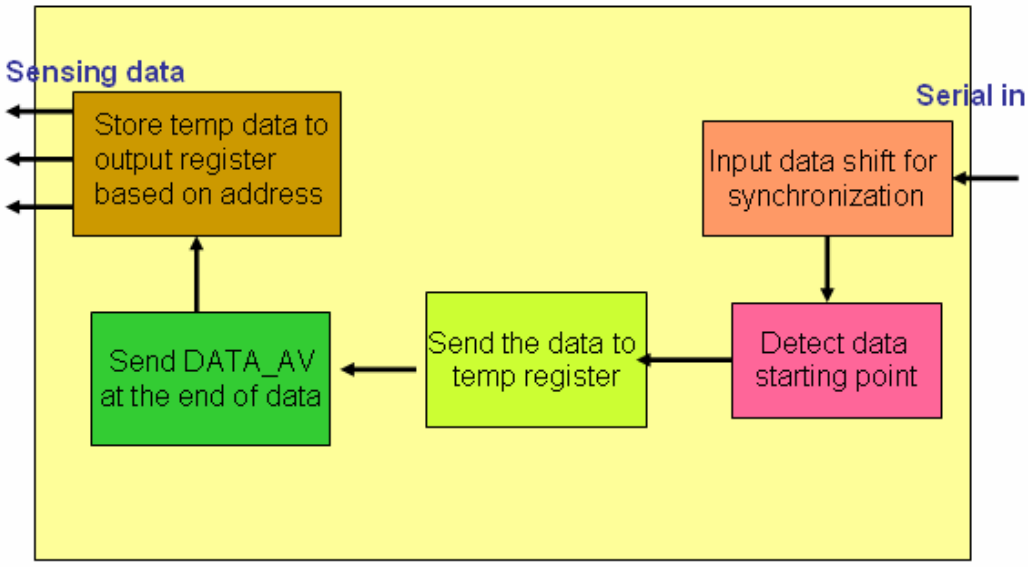


Fig. 3.21 Serial Receiver Block Diagram

In the following experiments, the setup diagram is shown as in Fig. 3.22. DC power source is connected with the local controller to adjust the sensing voltage. The central controller and the local controller are connected with serial lines. The serial data output of the local controller is measured. At the same time, to validate the design, some status control signals are connected to the FPGA output pins and measured.

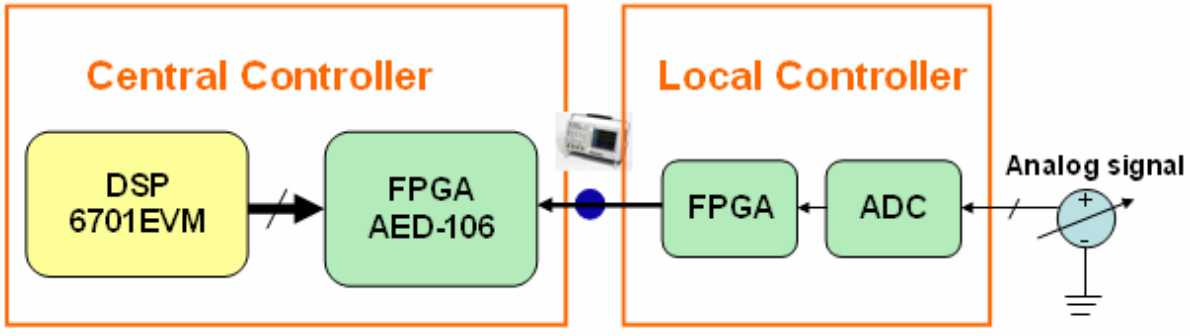


Fig. 3.22 Experiment setup diagram

Prior to detecting the data starting point, there is an operation to shift the input data for synchronization. The waveforms are shown in Fig. 3.23.

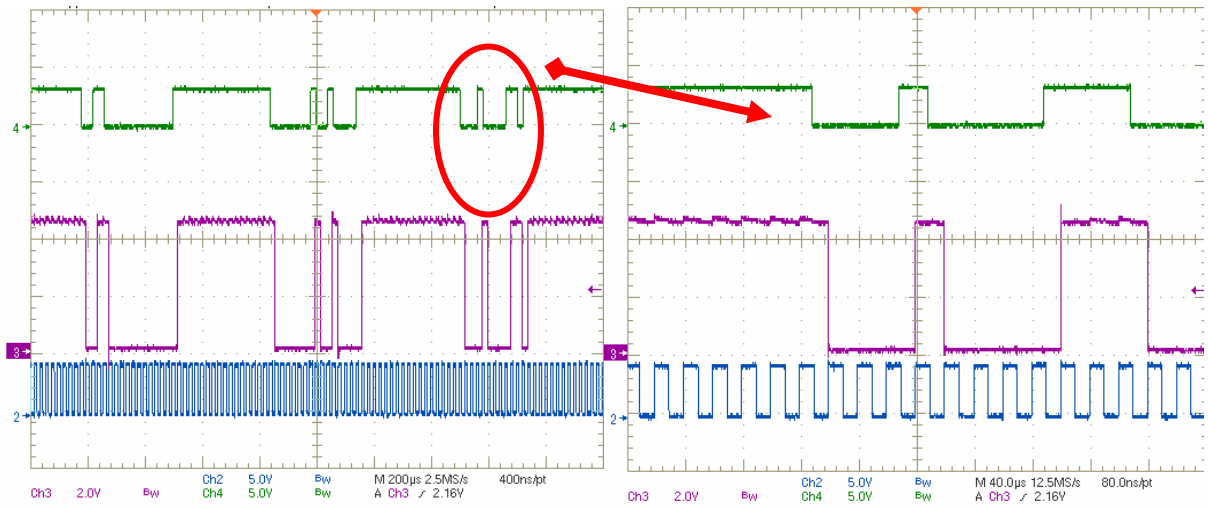


Fig. 3.23 Input Data Phase Shift

Ch2 is the receiver clock, Ch4 is the input serial data, Ch3 is the serial data after in-phase operation.

To receive the serial data, the receiver and the sender frequency should be the same. As the sender frequency in the local controller is set to be 50kHz, the receiver has also the same frequency. However, the two clock frequencies are not in phase. From Fig. 3.23, Ch4 is the serial data from the local controller. Compared with the receiver's clock, the serial data edge transition point is not at the receiver clock's edge transition point. To avoid this phase shift, a phase synchronization is done. Ch3 shows the serial data after processing. Now it is exactly transiting along with the receiver clock.

While the starting bit of a data is detected, the serial receiver will set the start signal to be '1', at the same time, the trans_en signal for the transmission is also enabled. While the data transmission starts, each bit of the data will be detected and stored in the temporary registers. At the end of the transmission, the data_av will be set to be '1' to indicate a new data is ready.

The waveform in Fig. 3.24 shows the operation. Ch4 is the input serial data, Ch3 is the start signal, Ch1 is the data_av signal, Ch2 is the tran_en signal. It could be observed from the waveform, the data is correctly detected and stored.

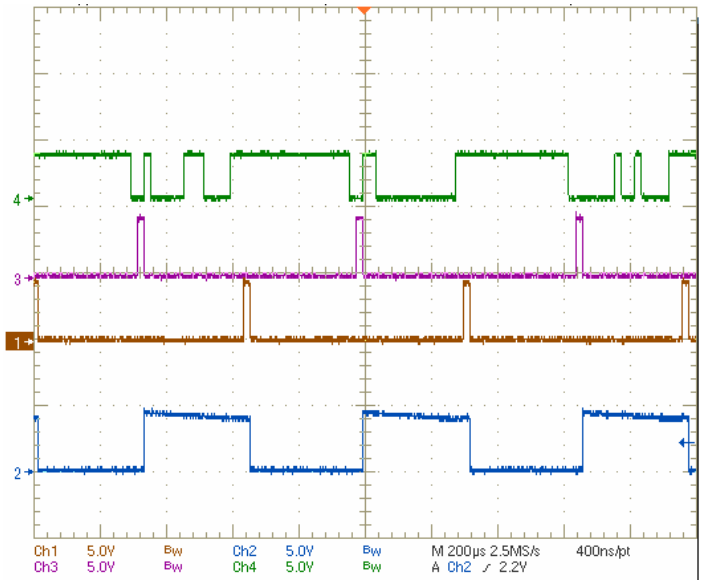


Fig. 3.24 Start & Data_ready &Tran_en in serial receiver

A temporary register is defined to store the data in the transmission. It is a 16 bit register, the highest two bits will store the address information. The lowest two bits will store the two stop bits, so they should be “11”. The left 12 bits store the transmitted A/D data.

To make sure the data stored in the temporary register is correct, a couple of bits are checked to validate the accuracy. In Fig. 3.25, the highest two bits are checked. Firstly checking the serial input data address information and then comparing it with the temporary register data. They are matched correctly.

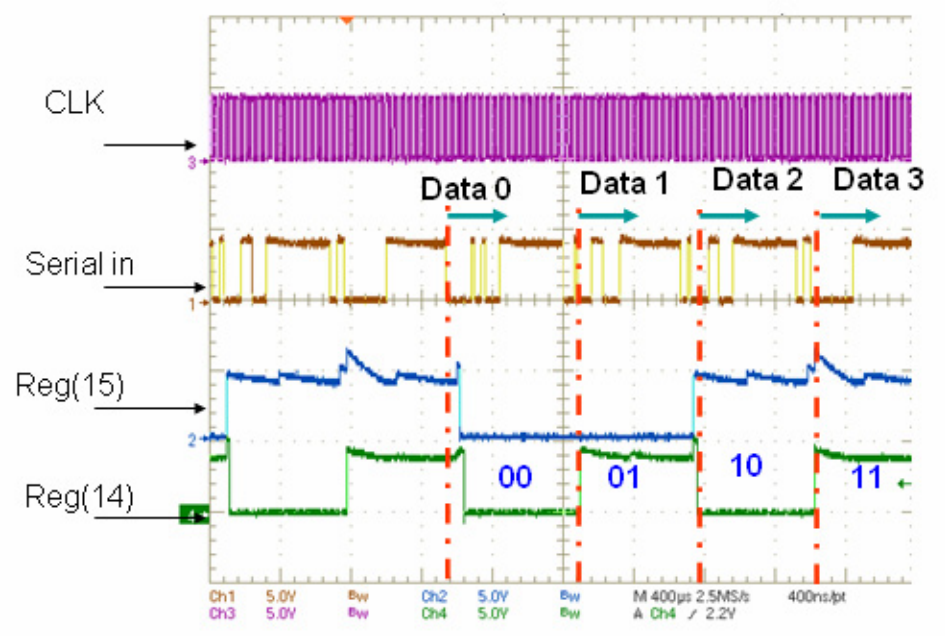


Fig. 3.25 Highest two bits in the temporary register

The lowest two bits are also checked. According to the design, the lowest two bits should always be high due to they are the two stop bits. The measured waveform is shown in Fig 3.26. Ch3 is the receiver clock, Ch1 is the input serial data, Ch2 and Ch4 are the lowest two bits. The measurement results show the design is correct.

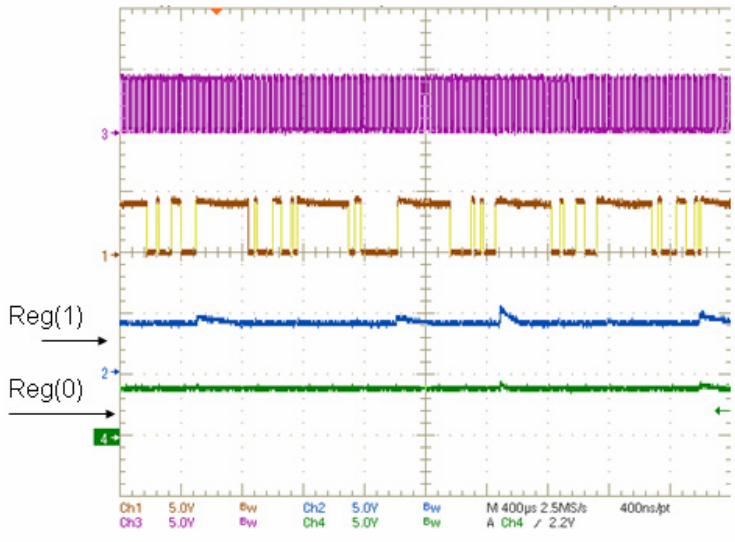


Fig. 3.26 Lowest two bits of the temp register in serial receiver

Every bit of the transmitted data is also checked and the transmission is accurate. To save space, only the checking of bit 6 and 7 are shown in Fig. 3.27.

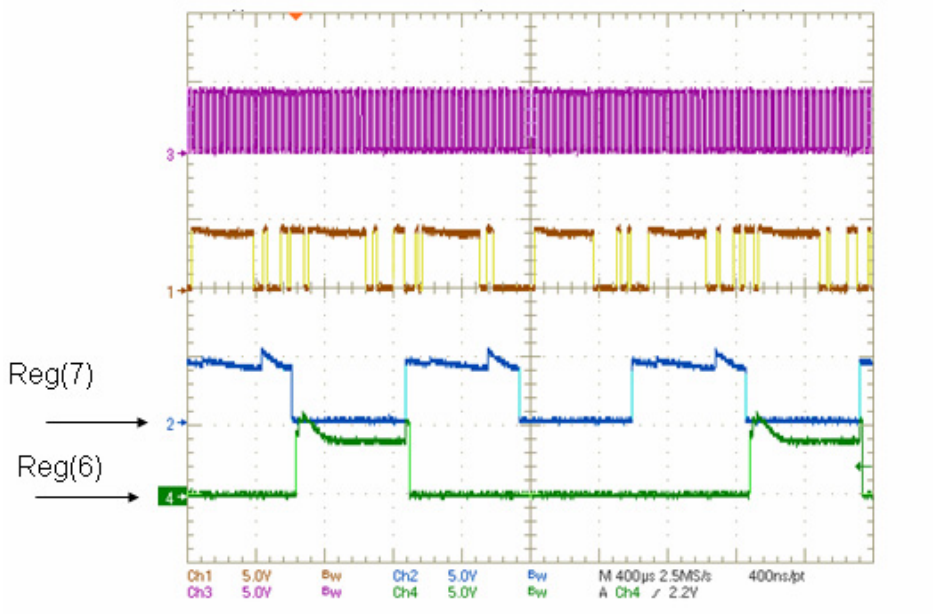


Fig. 3.27 Bit 6 & 7 of the temporary register

While storing the data in the temporary register, the data accuracy is checked in the code. The lowest two bits are checked if they are “11”. If the data accuracy is validated, the data will be stored in the output registers based on the address information. The output registers are connected to the control register module in the FPGA. DSP will access to these data with the defined memory address. In the experiment verification, ADC1, 2 and 3 are defined to receive the data. A formula is done in DSP to calibrate the received parallel data. The equations for the three channels are set to be a little different. After the calibration, the test results are shown in Table 6.

Table 6. DSP ADC Channel Displayed Value and Actual Input Voltage

ADC3

Input(V)	1.09	2.05	3.05	3.82
DSP	0.784	0.687	0.582	0.5

ADC2

Input(V)	1.32	2.62	3.27	4.04
DSP	0.877	0.74	0.673	0.592

ADC1

Input(V)	1.32	2.52	3.38	4.03
DSP	0.867	0.74	0.65	0.58

The input voltages to the A/D channels are changing as shown above from 1v to 4v. The DSP captured value is changing following the input. As the A/D converter has inverting configuration, the DSP value will be decreased while the input voltage is increased.

With the above validations, the serial receiver is able to precisely detect and store the data and send it to DSP.

What about multi-receivers in the FPGA? What is the possible effect on the data transmission?

After 6 receivers are implemented in the central FPGA, it is observed from the experiment that the 6 receiver channels are working well.

5. Central FPGA Synthesize and Resource Usage Summary

The synthesis, place and routing in the FPGA is completed correctly. The FPGA register level synthesis is shown in Fig. 3.28.

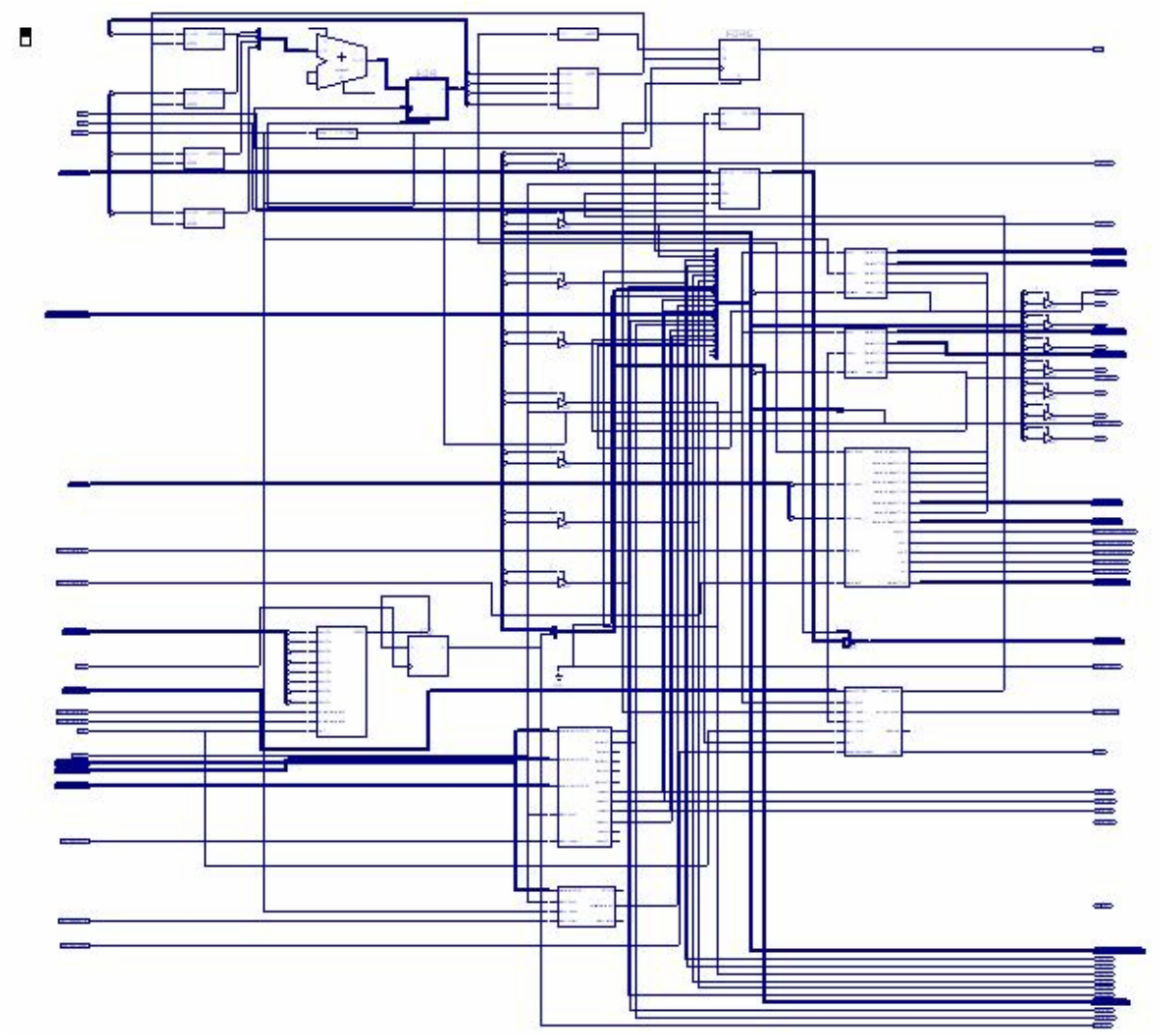


Fig. 3.28 Central FPGA Register Level Synthesis

The FPGA device resource utilization is shown in Table 7. From the data, the selected XCV300 has enough gate resources for the design.

Table 7. Central FPGA Resource Utilization Summary

Number of External GCLKIOBs	3 out of 4	75%
Number of External IOBs	121 out of 166	72%
Number of LOCed External IOBs	121 out of 121	100%
Number of SLICEs	692 out of 3072	22%
Number of GCLKs	3 out of 4	75%

3.3.3 Local Controller Software Module

The local controller receives the serial data from the central controller and converts it to a parallel data. This data is then being sent to the PWM generator module to generate switch signals for the power stages. The analog sensing signals are connected to the local controller. These signals are converted to a 12 bit digital data through A/D converters. The 12 bit data is processed by the local controller and appended with the address information. After that, the multiple analog sensing signals are sending out via one FPGA pin to the central controller.

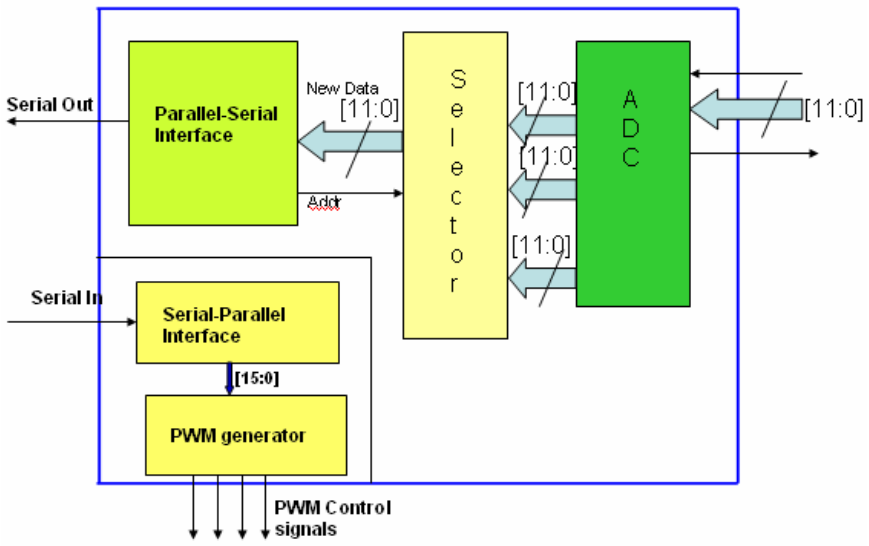


Fig. 3.29 Local Controller Software Function Diagram

The function diagram of the local controller is shown in Fig. 3.29. It is consisted of 5 VHDL modules: ADC, selector, parallel-serial interface, serial-parallel interface and PWM generator. The function of the modules are shown in Table 8.

Table 8. Local Controller Software Module Function Definitions

Software Modules	Function
ADC	Control interface to the physical A/D converter, store the sampled A/D data to the output registers
Selector	Store the ADC data and send them to the parallel-serial interface based on the input command
Parallel-serial Interface	Send all the sampled analog data in serial data and in one FPGA output pin
Serial-Parallel Interface	Convert the received serial data to parallel data
PWM Generator	Generate PWM signals with the received duty command

In the following, the details for the implementation of each module will be illustrated.

3.3.3.1 ADC Controller

The ADC controller is implemented to control the ADC chips on the local controller. As described before, the local controller has two ADC chips that will support 8 channel analog signal conversion.

The routine procedure of the ADC controller module is shown in Fig. 3.30. Firstly initialize the ADC chip, and then enable single conversion for all the ADC channels simultaneously. After the A/D conversion is completed (that means DATA_AV is enabled), a state machine

is designed to read the data from the FIFO of the ADC chips and store them into output registers.

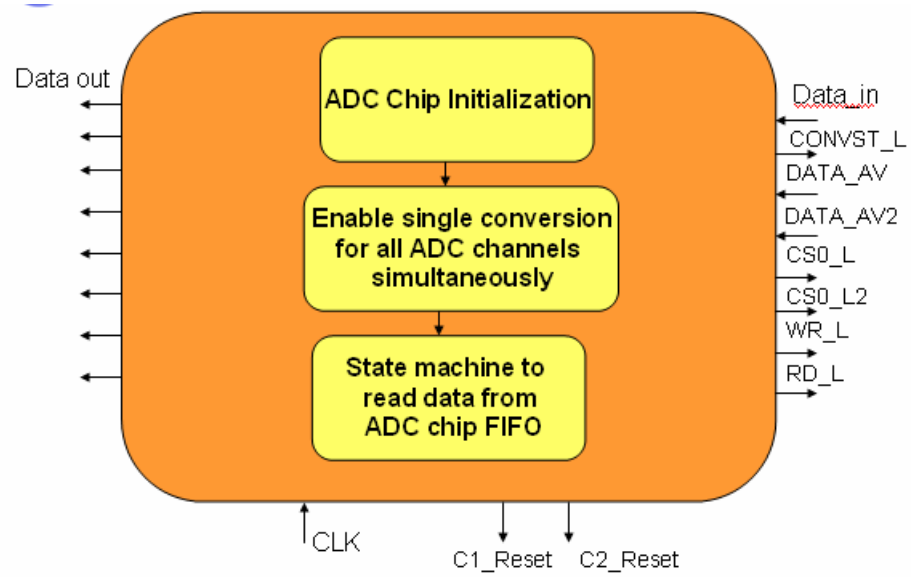


Fig. 3.30 ADC controller modular routine procedure

The state machine to read the data from the A/D chip FIFO register is defined in Fig. 3.31. A couple of transition states are defined to read all the data in sequence.

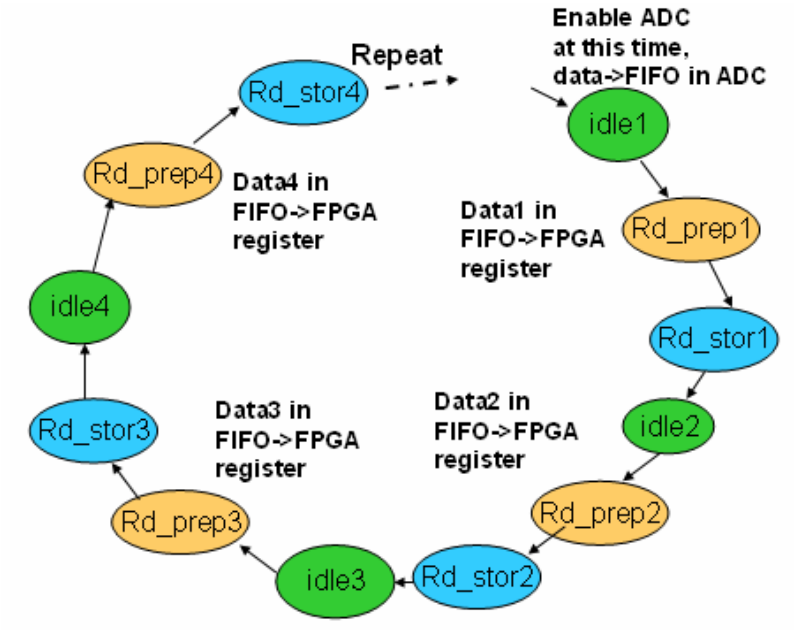


Fig. 3.31 ADC Controller State Machine

3.3.3.2 Parallel-Serial Interface

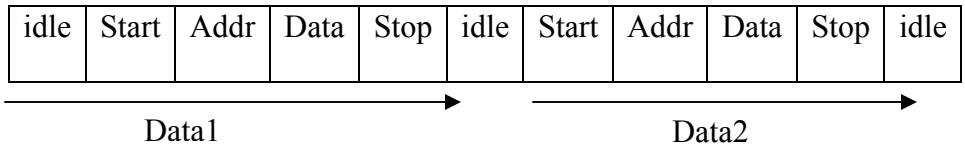
The parallel-serial interface performs the function of sending multiple analog sensing signals to the central controller via one pin. A concise design has been developed to fulfill this function.

Four analog sensing signal input are defined here. They are voltage, current, temperature, and fault status respectively. Also, they are appended with the address information from “00” to “11” respectively.

The address counter in the design is changing from “00” to “11”, the data for the designated address is updated from the selector module. While the selector receives the address information, it will immediately send the latest related data to the parallel-serial interface.

In the design, an internal 50kHz clock is generated to send the serial data. According to the defined communication protocol, for each data, the first two bits are start bits “00”, then they are followed with 2 bits address, after that, 12 bits of the real data are sent. In the end, there are 2 stop bits “11”. So, for each data packet, the data length is 18. A

Table 9. Data In Series



As Shown in Table 9, the sensing data are sending out in sequence. As mentioned above the 2 bits address information are changing from “00” to “11” periodically in the design, the 4 sensing data are also sent out periodically.

To simplify the design, an 18 bit output register is defined to be the shift register. It will perform the parallel to serial function. The A/D data after appending start bits, address

information and stop bits is stored in the output register. Then at each rising edge of the clock, the output register is shifted. The lowest bits are firstly sent to the output port.

To make the receiver of the central controller detect the new data accurately, the idle mode between two data should be set to be higher than 15 sender clock cycles. Otherwise, it would be hard for the receiver to find the starting point of a new data. An example is shown in Fig. 3.32. The idle mode after the stop bits is set to be 3 clock cycles. The data start and stop points could be observed by comparing with the control signals, however, the central and local controller have no other control signals communication except the serial data line. So with the method of ones counter and start bit detection, the data will be wrong.

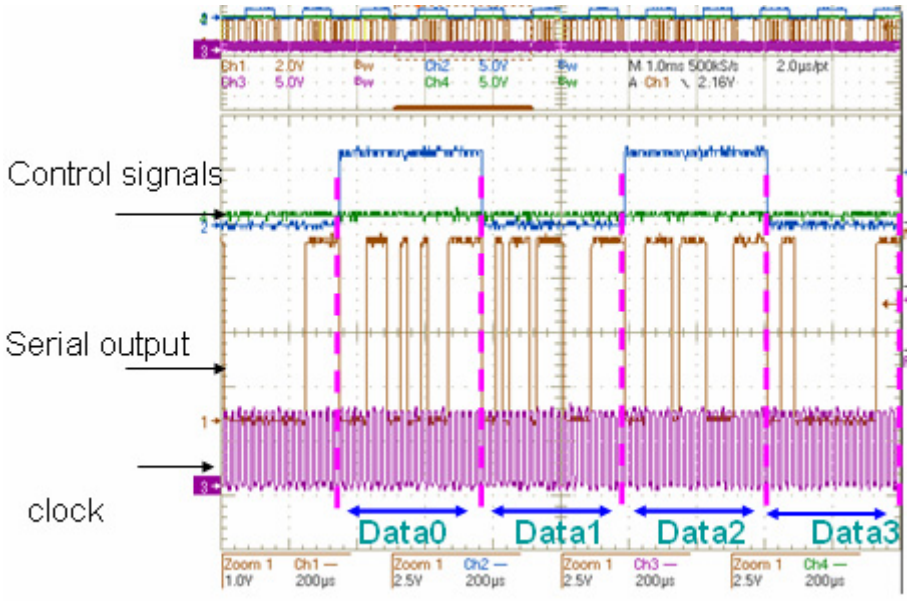


Fig. 3.32 Serial Out Data from Local Controller

In Fig. 3.33, the time period between the two data are set to be 15 clock cycles. So, now it is quite easy for the central controller receiver to capture the starting point.

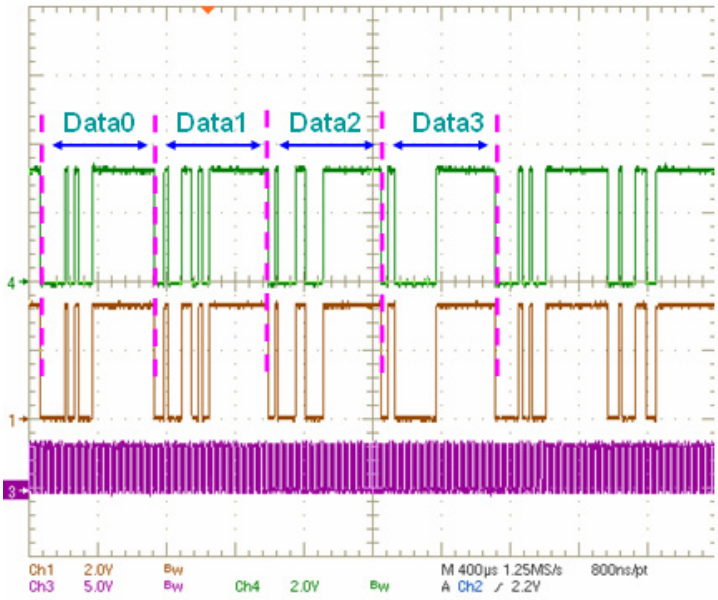


Fig. 3.33 Better Serial Output Data For Receiver

In the experiment, the serial output data change following the input sensing signal change has been validated. A HEX displayer is designed to display the input signal change. While adjusting the voltage level of the input signal, the HEX display will show the data change that is directly the register data in the FPGA. At the same time the change on the output serial data is also observed.

3.3.3.3 Serial-Parallel Interface

The function of the serial-parallel interface is to detect the received serial data and then converter it to a parallel data. Moreover, the accuracy of the received data will be checked. In this application, this parallel data from serial to parallel interface is the modulation signal for the PWM generator. While the new data is checked, the serial-parallel interface will issue a data available signal to inform the PWM generator. The PWM generator will then take this new data to generator switch signals.

This receiver has the same issues with the receiver in the central controller. It should detect the starting point of the modulation signal correctly. In the design, the same method is applied as the central controller receiver. A relatively longer time is set between two data.

For UART communication, the central controller and the local controller clock are not synchronized. The condition to correctly receiving the data is that the frequency of the two clocks should not have large drift. In my design, the DSP set the central controller FPGA frequency to be 50MHz. The serial data is sent out with 1/16 of the clock frequency. However, in the local controller, the FPGA clock is determined by its own crystal that is 80MHz. So, a receiver clock is generated at 80M/26. It is 3.077MHz. Look at this, the receiver clock is 3.077MHz, and the sender clock is 3.125MHz. The clock frequency shift is around 0.048MHz that is 1.6% of the receiver clock. It is validated with the experiment that the receiver could get the data correctly. More discussions on this kind of synchronization will be discussed in the next chapter.

The parallel-serial interface block diagram is shown in Fig. 3.34.

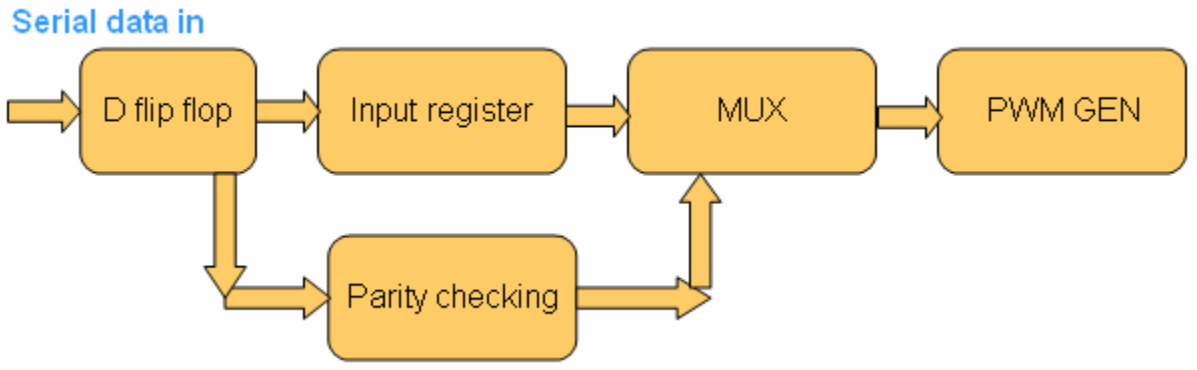


Fig. 3.34 Parallel-Serial Interface

The input data is firstly stored in a D flip flop, then it is stored to the input register based on the clock. The detected parity bits are checked. If the following conditions are met: the two parity bits are matched with the received data and the received stop bits are ‘1’s, the MUX will send the available data to PWM generator.

A Test bench is set up in MODELSIM to validate the design. The simulation is shown in Fig. 3.35.

Fig. 3. 35 Serial-Parallel Interface Simulation In Modelsim

Three serial data that represents “0000”, “FF55” and “552A” are sent to the serial input. It is shown in the simulation that the PWM generator get these data correctly.

The experiments are set up to validate the design. A fixed duty command is sent from the central controller in serial form. The local controller receives the data and then converts it to PWM signals. Duty 0.0 and 0.7 are examined in this validation as shown in Fig. 3.36 &3.37.

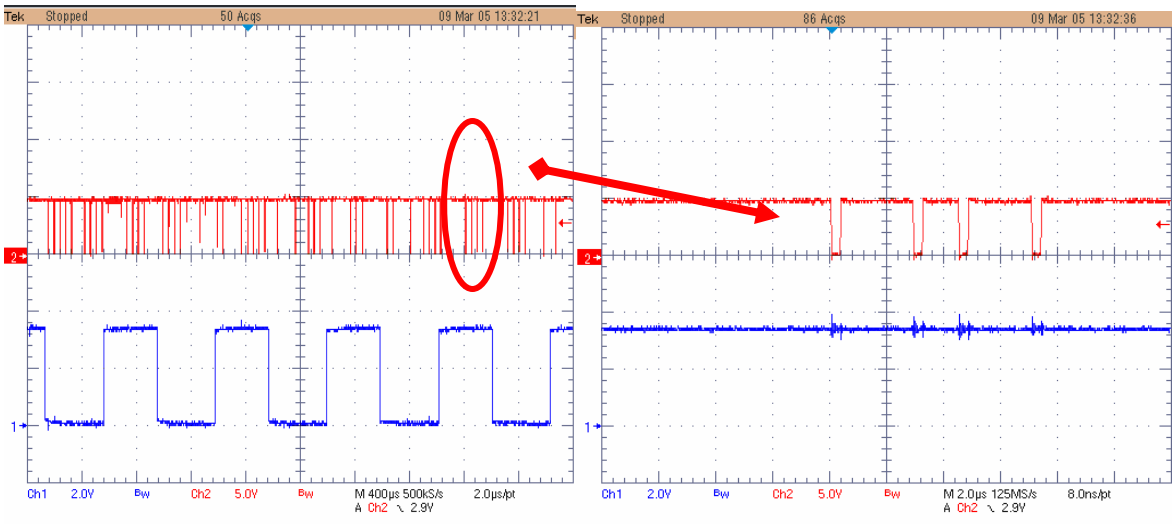


Fig. 3.36 Test with D=0.0 from Central Controller

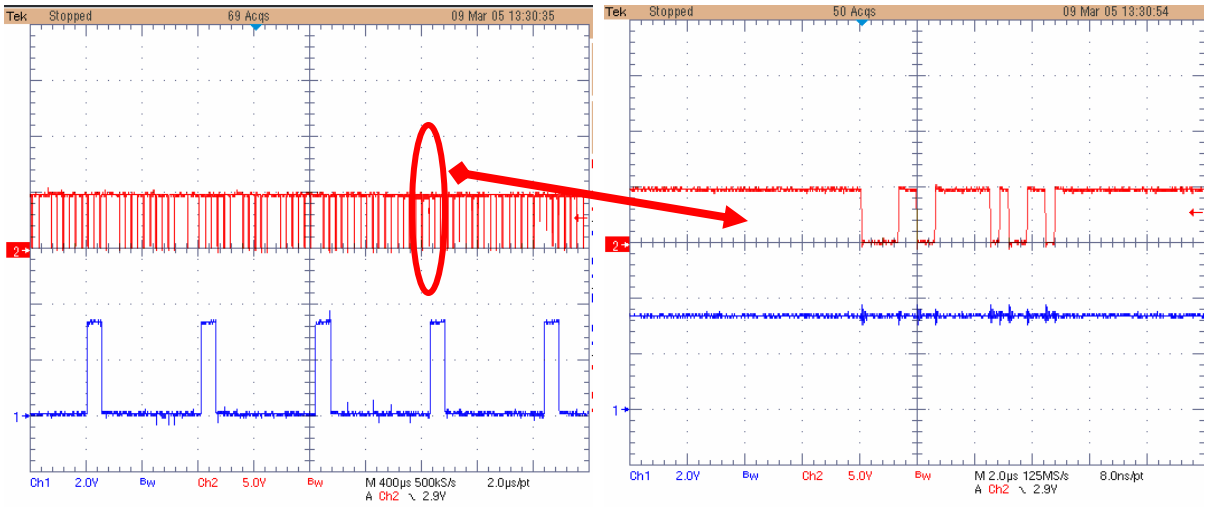


Fig. 3.37 Test with D=0.7 from Central Controller

In the above waveforms, Ch2 is the received serial data, Ch1 is the one of the generated switch signals. The frequency and the duty cycle of them are validated to be right.

3.3.3.4 PWM Generator

The PWM generator receives the modulation signal and then generates 4 switch signals for a H-bridge converter. It is composed of a triangle generator and PWM comparator in the VHDL module.

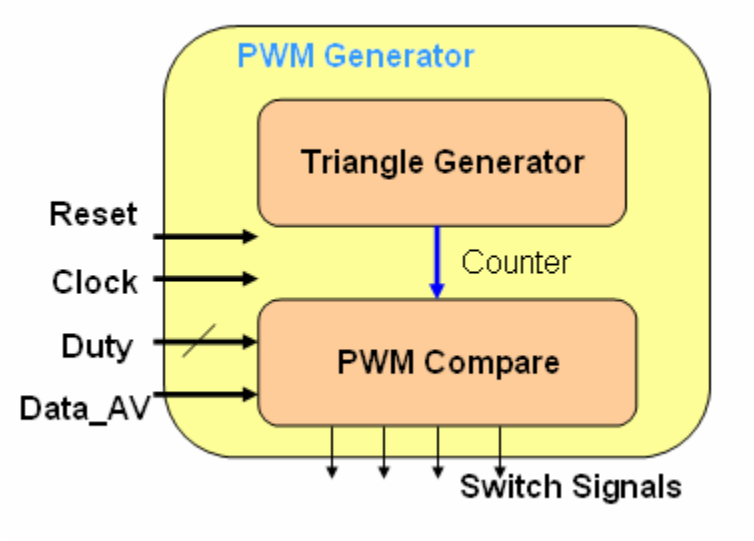


Fig. 3.38 PWM Generator Module

The triangle generator uses a counter to generate a triangle waveform. To facilitate the FPGA design, the counter is added up to a set value and then is reset to 0.

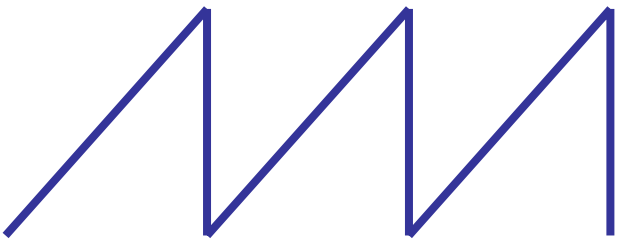
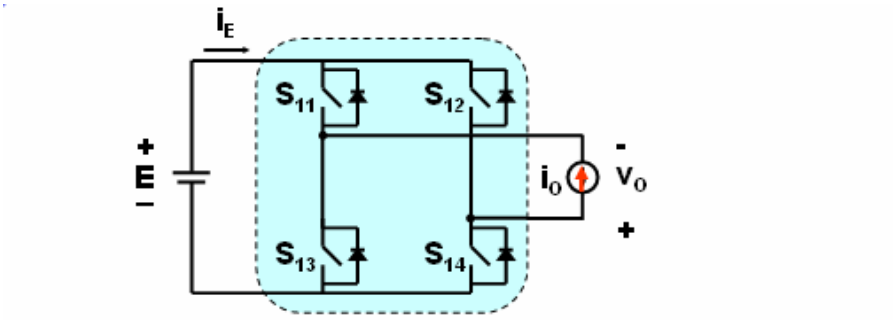


Fig. 3.39 Counter output of the Tri-Generator

An example is illustrated to understand the design in the following. A Fixed duty -0.7 is selected in the descriptions.

Firstly, to understand the implementation, the converter model should be clarified.



S_{11}	S_{12}	S_{13}=-S_{11}	S_{14}=-S_{12}	i_E	v_o
1	1	0	0	0	0
1	0	0	1	i_o	-E
0	1	1	0	- i_o	+ E
0	0	1	1	0	0

Note: "1" represents turn-on and "0" represents turn-off.

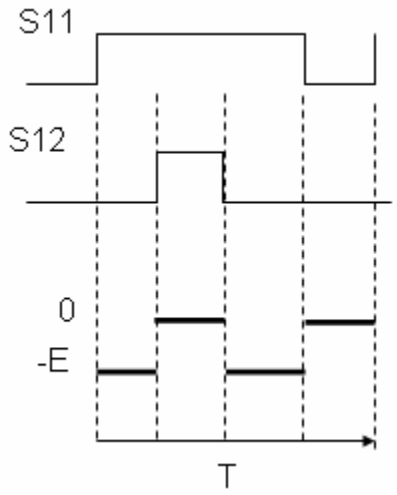


Fig. 3.40 H-Bridge converter switch combinations

The switch combinations for a H-bridge converter is shown in Fig. 3.40. It is a 3 level converter, the output voltage will be +E,-E and 0. In the average model derived in [3], the 4 switched are combined into one switch, its switch status is averaged to be represented by duty cycle d. Observed from the switching waveform, the converter switching frequency and the single switch switching frequency is different. The single switch frequency is double of

the converter switching frequency. To achieve 3 level operation, the duty cycle for each of the switch is also different from the converter duty cycle. Assume the central controller generate the duty cycle d for the converter based on the control scheme, the four switches will get the duty cycles as shown in Fig. 3.41.

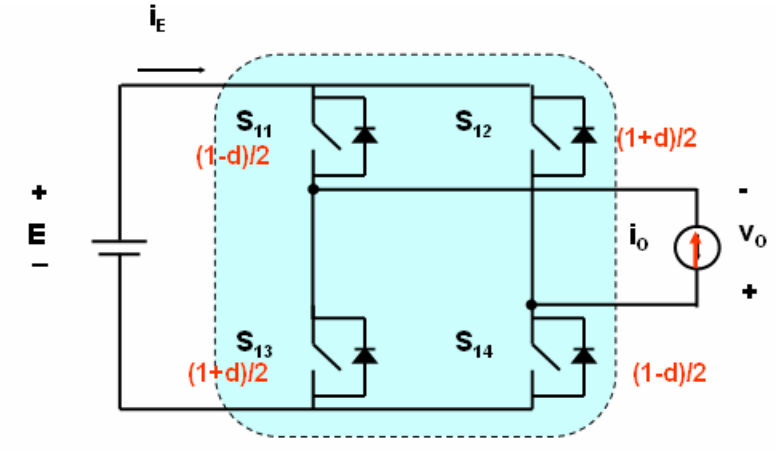


Fig. 3.41 Duty cycle for each switch in the H-bridge converter

Secondly, the DSP will generate a decimal data based on the control scheme. This data is converted to a 16 bit data as shown in Table 10 and 11.

Table 10. DSP Duty Scale

```
void write2pwm(const TVA_ABC_DATA* DUTY){unsigned int dutya, dutyb, dutyc;
        //Scaling the duty cycle to 16-bit integer
        dutya=((DUTY->A)*0.5+0.5)*(float)(0xFFFF);
        dutyb=((DUTY->B)*0.5+0.5)*(float)(0xFFFF);
        dutyc=((DUTY->C)*0.5+0.5)*(float)(0xFFFF);
        //store new value for PWM duty cycle in FPGA
        *pPWMA_DUTY = dutya;
        *pPWMB_DUTY = dutyb;
        *pPWMC_DUTY = dutyc;}
```

Table 11. Data Scale In DSP

Dutya	Scale value
0	**7FFF**
1	**FFFF**
-1	**0**
-0.7	**2666**

The duty cycle defined in the DSP code is from -1 to 1, the coordinate scaled value is shown in Table 11. The scale value is related with the counter setting in the triangle generator.

The sawtooth peak point is set to the scale value FFFF at which the duty is 1. The sawtooth bottom point is set to 0000 at which the duty is -1.

In the next, the PWM comparator working principle is illustrated. The received duty signal is compared with the output of the triangle generator.

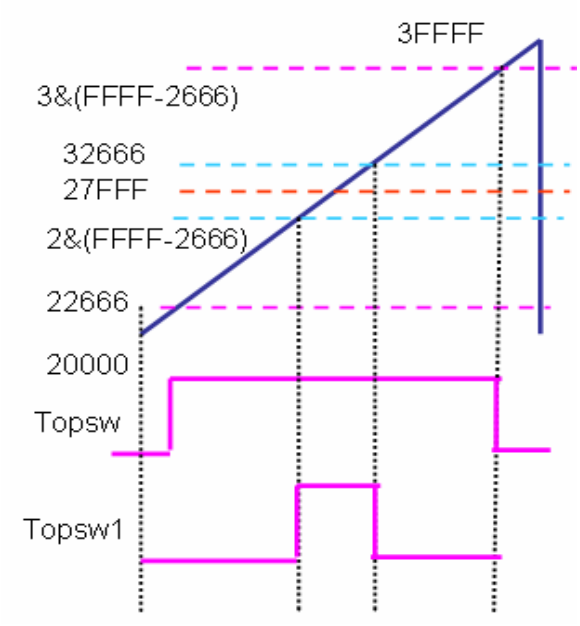


Fig. 3.42 PWM Comparator(Top SW is S11, Top SW1 is S12 in Fig. 3.40)

As shown in Fig. 3.42, the Sawtooth counts from 20000 to 3FFFF. As it is counted each two clock cycles, the actual counted number is FFFF(65535). Calculated with the clock frequency 50MHz, the carrier frequency is 760Hz. So the generated switch signals will also have the frequency of 760Hz.

The design follows the basic SPWM modulation scheme in that the sinusoidal modulation waveform is compared with a triangle carrier to generate the PWM signals.

The modulation signals for the two top switches in the H-bridge converter have phase shift of 180 degree. In Fig 3.42, there are 4 levels are set based on the received duty command. For fixed duty input d=-0.7, the scale value in DSP is X"2666" as shown in Table 11. The two pink levels are set for the top switch, and the two blue levels are set for the top switch 1.

After folding the upper section of the sawtooth down to form a triangle waveform, it is exactly the conventional Triangle-SPWM modulation. It is shown in Fig. 3.43.

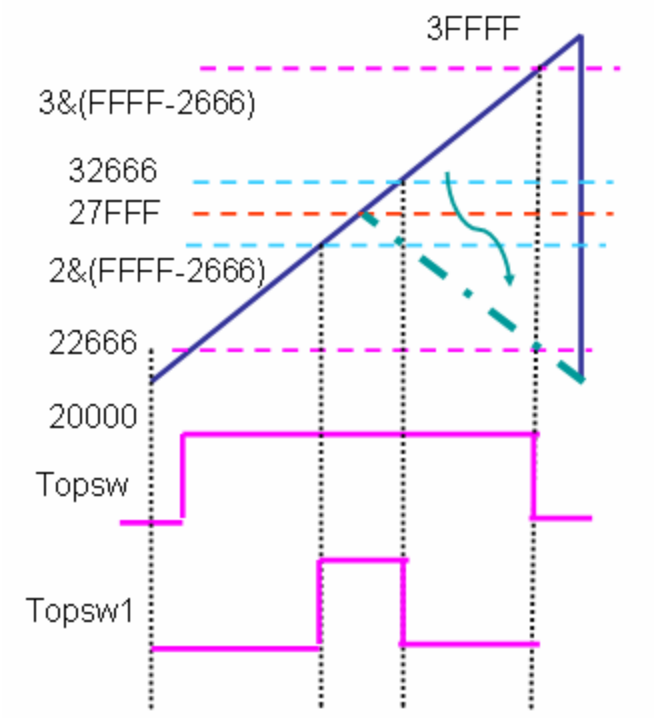


Fig. 3.43 Triangle SPWM Modulation

The top switch S11 duty cycle will be 0.85 and the top switch S12 duty cycle will be 0.15(here the dead time function is ignored). Then the whole converter will get a duty cycle of -0.7.

The dead time is for the top and bottom switches S11 and S13 as shown in Fig. 3.40 are implemented in the design. It is to prevent the shoot through of the converter. A pre-defined dead time is added to the levels mentioned above. Its implementation is shown in Fig 3.44

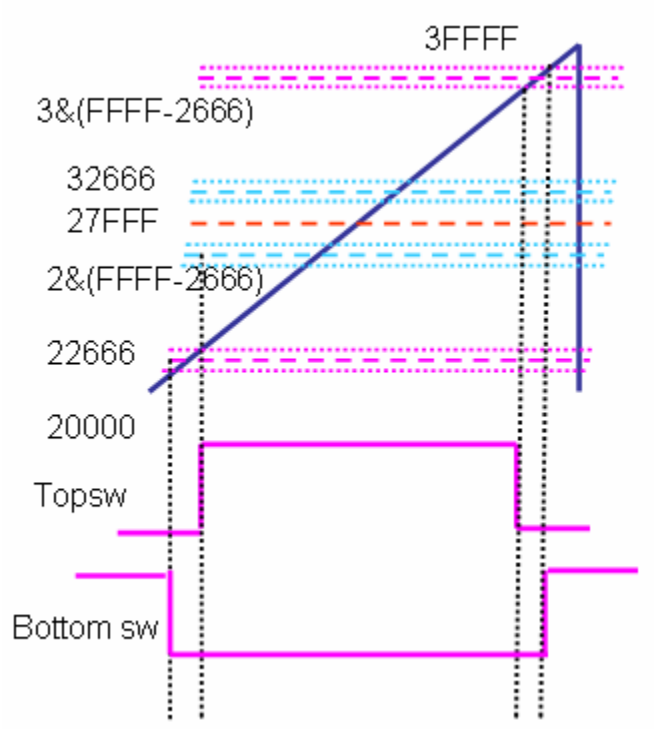


Fig. 3.44 Dead time implementation in the PWM generator

So, there are 8 levels set in the PWM generator to implement the function. With this design, the output switch signals are guaranteed to be able to prevent the shoot through problem.

A test bench is created and simulated in MODELSIM. The PWM generator is working fine with this scheme. The simulation result is shown in Fig. 3.45.

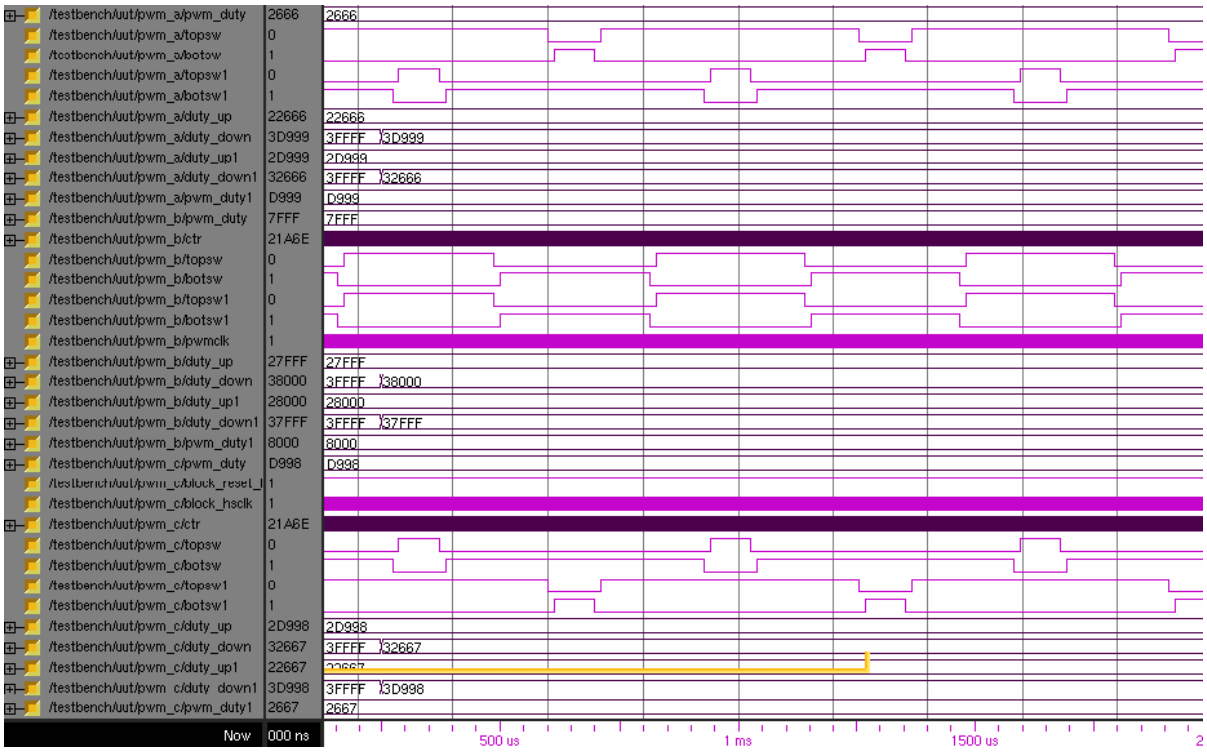


Fig. 3.45 PWM Generator MODELSIM Simulation Result

Experiments are set up to test the PWM generator, a couple of fix duties are sent to the PWM generator, the frequency, duty cycle and dead time of the output switch signals are measured.

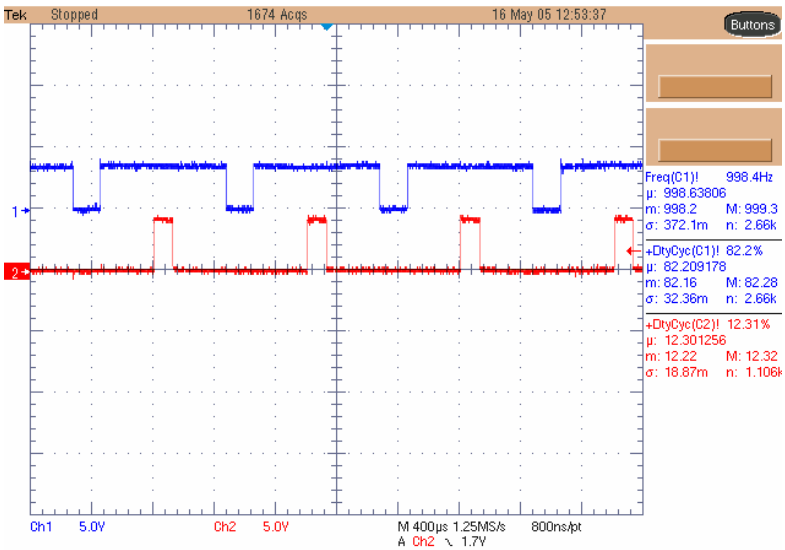


Fig. 3.46 Generated PWM signals with d=0.7

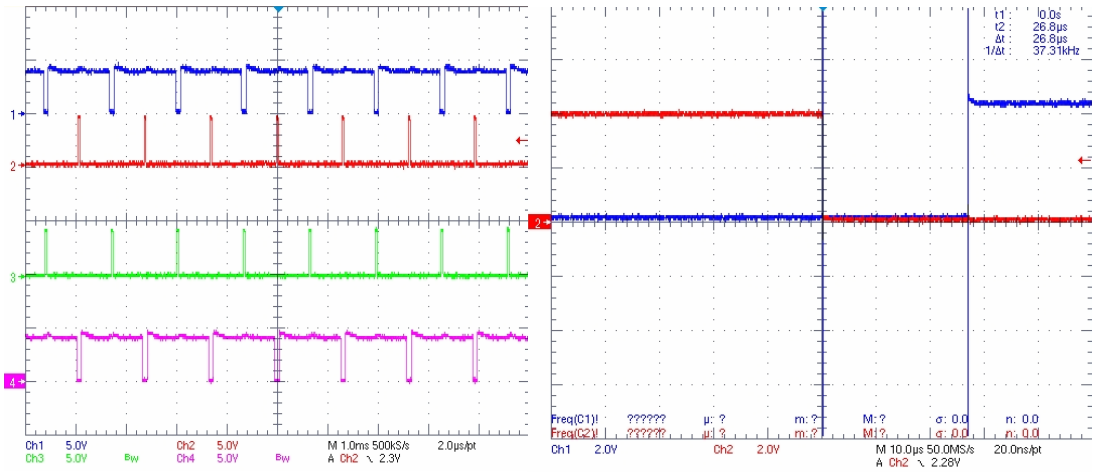


Fig. 3.47 Generated PWM signals with d=-0.9 and dead time checking

A trial to change the switching frequency has been done in the experiments, the switching frequency is changed to 1kHz. The measurement result is matched with this modification.

In Fig. 3.46, the duty command is 0.7, the top 1 switch (shown as Ch1) has the duty cycle of 82.2%, and the top switch (shown as Ch2) has the duty of 12.31%, the converter duty is 69.89% that is matched with the design.

In Fig. 3.47, the duty -0.9 is also tested. The 4 switch signals are measured. The frequency and the duty cycles are also matched with the design.

The dead time measurement is also shown in Fig. 3.47. The measured dead time is around 26.8us that matches the design.

3.3.3.5 Local Controller FPGA Synthesis and Resource Utilization Summary

The VHDL modules in the local controller is successfully synthesized, placed and routed in the XILINX XCV300 FPGA. The synthesis register level circuit is shown in Fig. 3. 48. The device resource utilization summary is shown in Table 12.

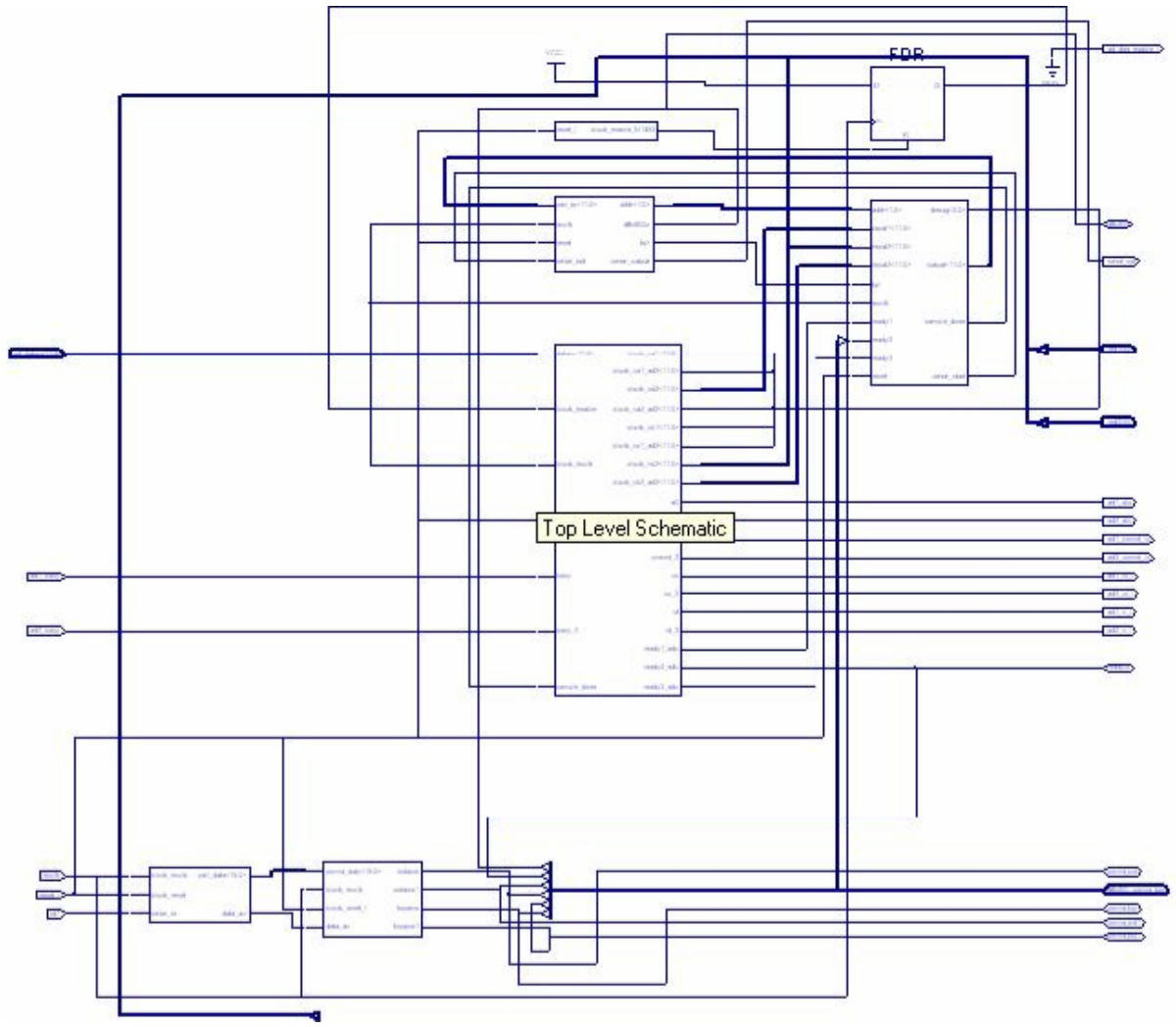


Fig. 3.48 Local Controller Register Level Schematic

Table 12 Local Controller Device Utilization Summary

Number of External GCLKIOBs	1 out of 4	25%
Number of External IOBs	42 out of 260	16%
Number of LOCed External IOBs	42 out of 42	100%
Number of SLICEs	411 out of 3072	13%
Number of GCLKs	1 out of 4	25%

3. 4 Summary of the Chapter

In this chapter, the development of the proposed distributed modular controller is described in details. A simple and easy-to-implement communication protocol UART is selected and defined for the proposed architecture. It makes the distributed controllers highly modularized. In this chapter, the software code that implement above protocol are also developed. The function and implementation methodology for all the software modules are described. Both simulation and experiments are used to validate the design.

CHAPTER 4

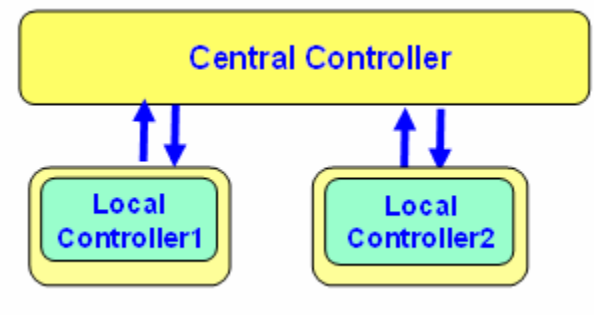
Intelligent Modular Converter Experiments

4. 1 Overview

An intelligent modular converter is defined as a modular converter incorporated with a local controller. In this chapter, firstly, tests of the communications from the central controller to multiple local controllers will be conducted. Next, a three level intelligent modular converter will be tested with a central controller. Fixed duty or SPWM duty commands will be sent from the central controller. The output load of the converter is resistive or RL load. The synchronization issues among the central controller and the distributed controllers are highlighted.

4.2 Central Controller Experiments with Multiple Local Controllers

In this experiment, two local controllers are connected with the central controller. They receive the same serial output data from the central controller. Then they convert the serial data to a parallel data and generate switch signals individually. At the same time, for the feedback path, they are sensing analog signals with A/D ports. Each of them will feedback one signal line to the central controller. Each signal line includes the data for 4 analog sensing signals. The experiment diagram and hardware are shown in Fig. 4.1.



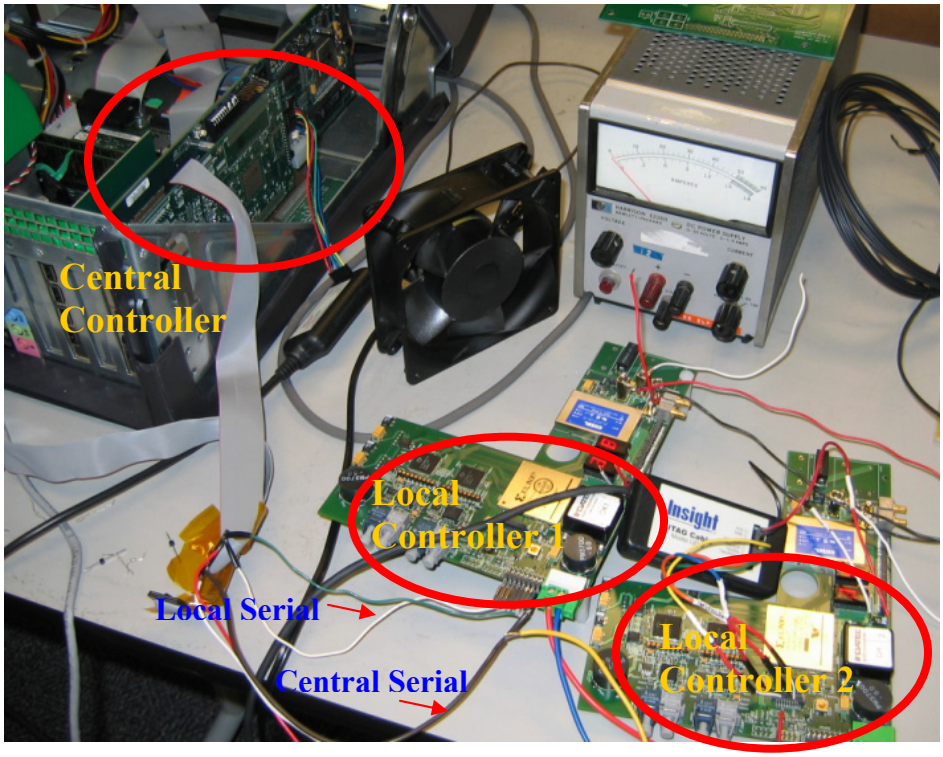


Fig. 4.1 Central Controller Connected with 2 Local Controllers

The measured waveform is shown in Fig. 4.2. Ch2 and Ch3 are top switch signal of the 2 local controllers respectively. Ch4 shows the serial data output from one of the local controllers. The duty sent from DSP in this case is -0.7.

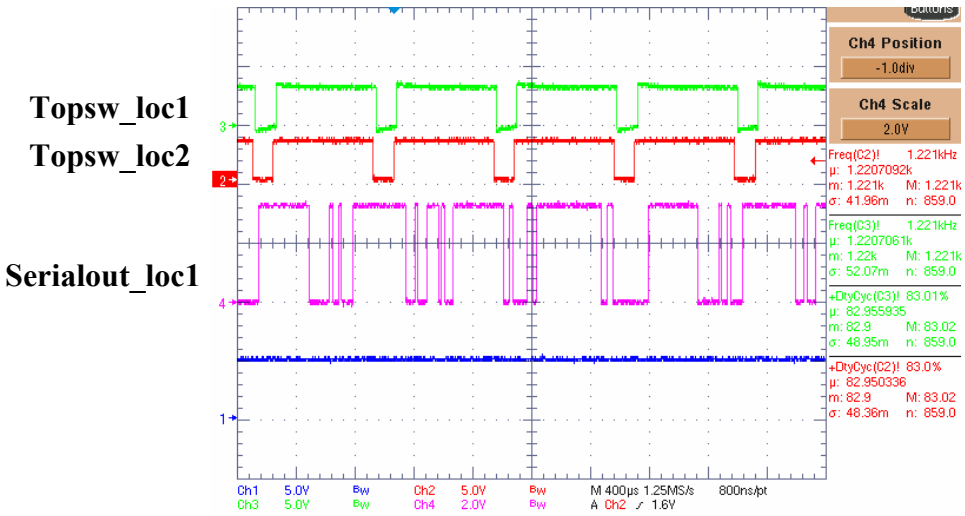


Fig. 4.2 Two Local Controllers Experiment

A simple close loop experiment is designed to test the forward and feedback path of the central-distributed local controller architecture.

In DSP code, it is written that the output duty is 0.7 if the feedback value from the local controller is over 0.7, otherwise, the output duty is -0.7. Adjusting the voltage level of the analog sensing signal at the local controller, the duty transitions of the switch signals are measured at the central controller side. The waveform in Fig. 4.3 shows that the duty of top switch signal changes from 83% to 12.99% when the DSP captured analog sensing voltage changes from less than 0.7V to over 0.7v. This proves that the close loop communication of the controller is working well.

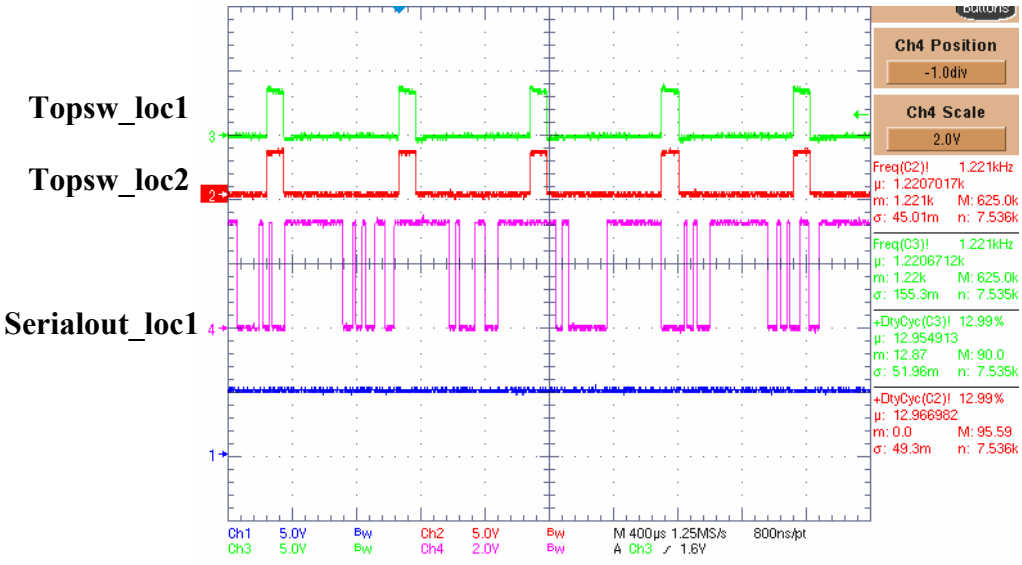


Fig. 4.3 Two Local Controllers Experiment with sensing voltage change

4.3 Synchronization between the Central and Local Controller

Sharing a common clock between the central controller and the local controller will surely eliminate the concern on the synchronization problem. However, it is not really so necessary for them to have one common clock.

The target of the common clock is to ensure the data are received and transmitted correctly. With the defined asynchronous serial communication protocol, a self synchronization scheme is accomplished. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver of the local controller into synchronization with the clock in the transmitter. These two clocks must be accurate enough to not have the frequency drift by more than 10% during the transmission of the remaining bits in the word. In the FPGA hardware used in the actual implementation, the central controller and the local controllers have different global clock frequencies. The clock frequency of the central controller is 50MHz, while the clock frequency of the local controller is 80MHz. Thus, clock

dividers are applied to make the transmitter and receiver frequency close to each other. In the central controller, each serial bit is transmitted with a frequency of 3.125MHz(50M/16). The receiver frequency is 3.077MHz(80M/26). To ensure the receiver can detect the right value, each data bit has a duration of 320ns and the detection happens in the middle of this data duration.

4.4 Data Propagation Delay Analysis

With the defined forward data communication protocol, the time delay to transmit one data is defined as shown in Fig. 4.4.

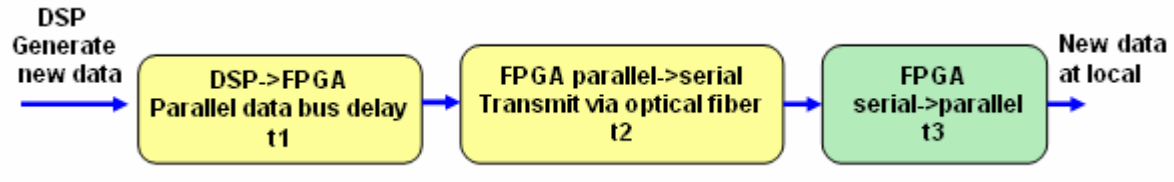


Fig.4.4 data propagation delay

After a new data is generated in the DSP, it is sent to the central FPGA with a preset interrupt frequency in DSP. The propagation delay to send the data from DSP to FPGA is negligible because there is a 32 bit parallel data bus between the DSP and FPGA. A 16 bit modulation command is transferred to FPGA within one clock cycle. After this operation, the FPGA starts the parallel to serial transformation. Each bit of the data is sent to the local controller sequentially via optical fiber. As optical fiber can transmit data with a very high speed rate, the propagation delay t2 is quite low here. The main propagation delay happens with t3. Each received bit at the local controller is stored into a register. A “receive” operation is completed while all the bits are stored. Thus, it takes time to get a new data ready. With a receiver

frequency of about 3MHz, t3 is approximately 8.3us. So, the total propagation delay will be around 8.3us. Considering the phase lock issue, 8.3us is just 0.05% of the modulation period 16.67ms. So, this delay should not affect the output phase of the local converter.

4.5 Synchronization among the Local Controllers

In the current design, each local controller has its own crystal to generate the clock. The clocks are rated at 80MHz. However, the clock frequency will have some small shift due to the tolerance of the crystals and the clocks among multiple local controllers are not exactly the same.

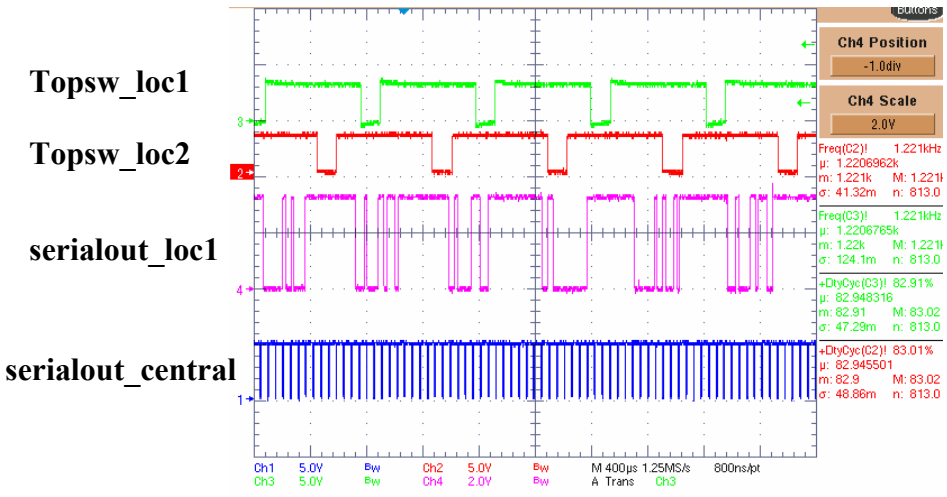


Fig. 4.5 Local Controller SW signals out of phase

In the experiment, the clocks of the two local controllers are observed not in phase as shown in Fig. 4.5. Ch 2 & 3 are the switch signals for local controller 1 &2 respectively. Ch4 is the serial out data from one of the local controllers. Ch1 is the serial out data from the central controller. Use the trigger function of the oscilloscope to fix Ch2, the Ch3 will be very slowly shifting referred to the fixed Ch2. The change rate is around 0.02Hz.

A couple of experiments are conducted to verify the reason for this phenomenon.

- Test the two 50kHz internal clock of the two local controllers

It is observed that the two clocks are shifting with respect to each other.

- Test the switch signals at different switching frequency such as 1.6kHz

To avoid the digital error resulted from the carrier frequency, the carrier frequency is set to be 1.6kHz that could be divided by the clock frequency of 80MHz. The same phenomenon is again observed as shown in Fig. 4.6.

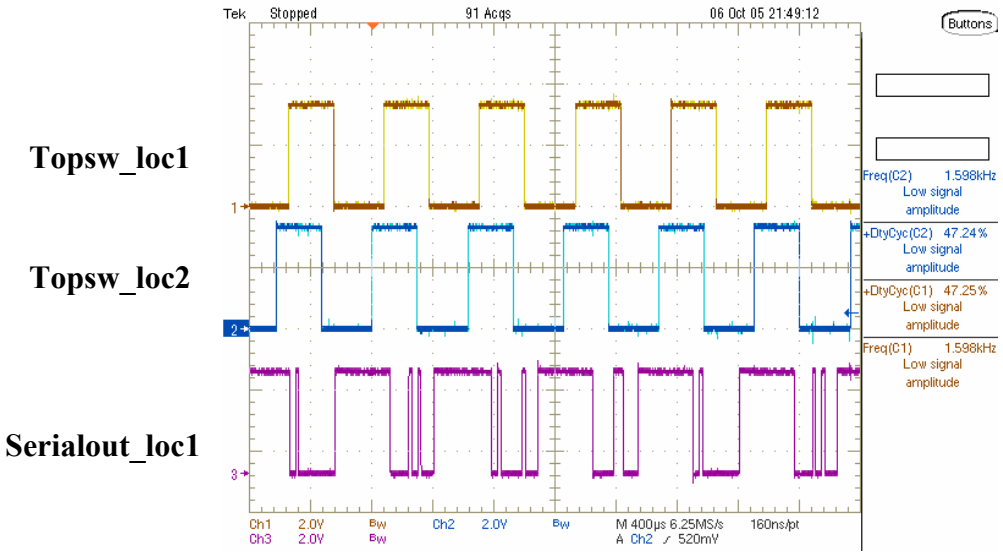


Fig. 4.6 Local Controller SW signals out of phase with f=1.6kHz

- Test the crystal

The crystal output waveforms are measured at its output pin and the buffer output pin. For either measurement, it is unable to get an accurate result. The measured frequency is varying and incorrect. The possible reason is the capacitive loading of the probe to the crystal.

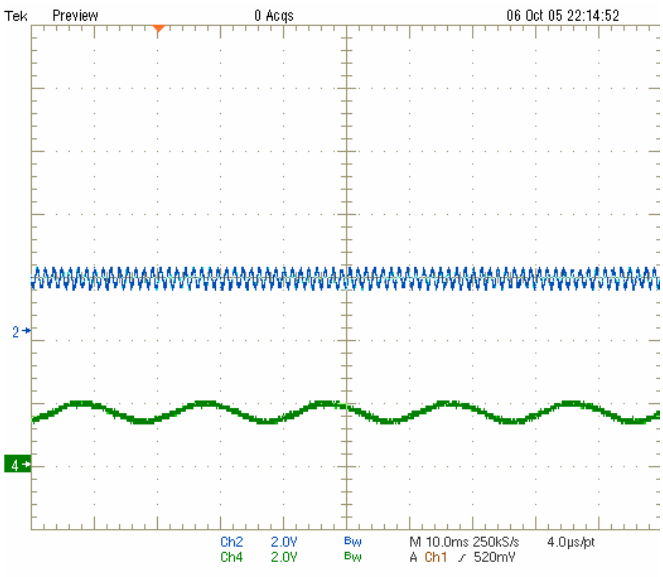


Fig. 4.7 Crystal output measurement

More investigations on this problem and the solutions will be provided in chapter 5.

4.6 Modular Converter Hardware Setup

A couple of H-bridge modular converters are developed. Each modular converter includes digital interface, driver board and a IGBT based H-bridge converter. In the following, the hardware parts are described.

4.6.1 Digital Interface

The digital interface is designed to provide buffer, I/O circuits and protection between the local controller and the power stage. Here only two top switch signals are required for the input. The bottom switches are generated in the board with hardware-implemented dead time. This will prevent the shoot through of the power stage. The output switch signals could be either electrical signal or optical signals. The board is shown in Fig. 4.8.

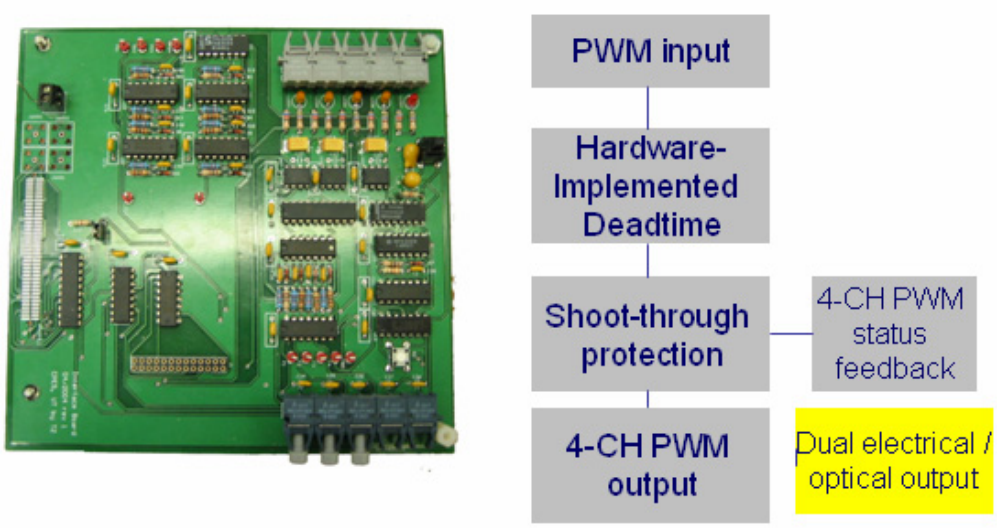


Fig. 4.8 Digital Interface Board

4.6.2 H-bridge Driver Board

The H-bridge driver board is shown in Fig. 4.9. This board includes a couple of isolated power supply provided for the top switch driver signals due to its isolation requirement. Optical coupler is used to isolate the input switch signal and the output drive signals.

4 channel switch signals are provided to drive the H-bridge devices in electrical or optical form.

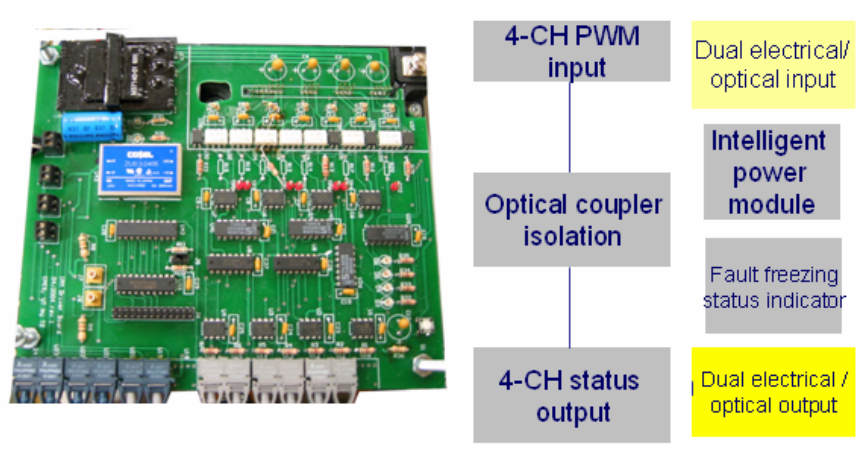


Fig. 4.9 H-bridge Driver Board

4.6.3 H-Bridge Converter

The H-bridge converter is constructed with an Intelligent Power Module (IPM) PM50RSA120. The IPM is an isolated base module designed for power switching applications operating at frequencies to 20kHz. Built-in control circuits provide optimum gate drive and protection for the IGBT and free-wheel diode power devices. The protection includes: Short Circuit, Over Current, Over Temperature and Under Voltage.

The IPM maximum block voltage is 1200V, and the maximum current rating is 50A.

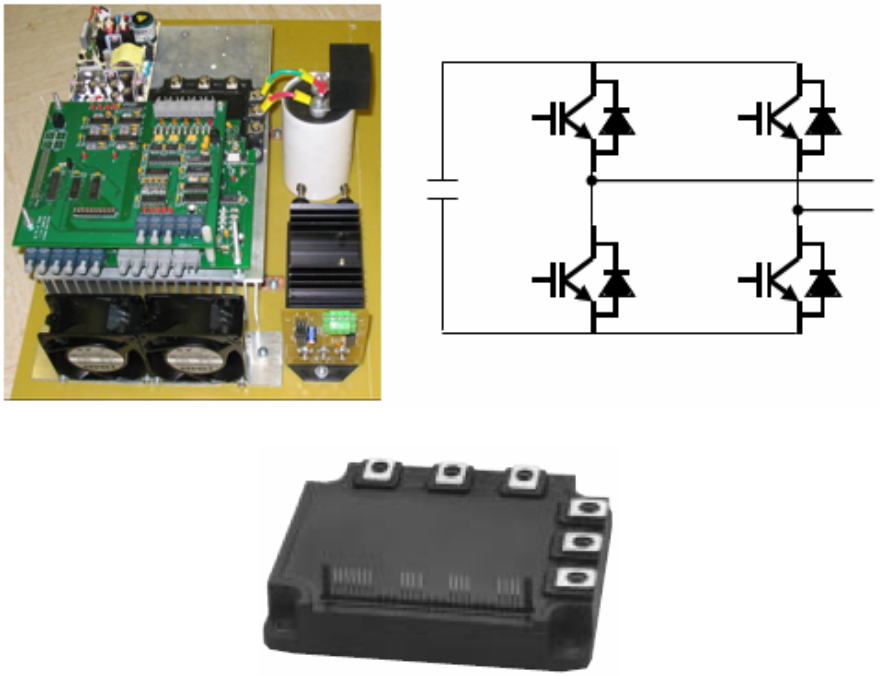


Fig. 4.10 H-bridge Converter & IPM

4.7 Intelligent Modular Converter Experiments

Experiments are set up with the modular controller and the modular converter.

The set up with one local controller is shown in Fig. 4.11.

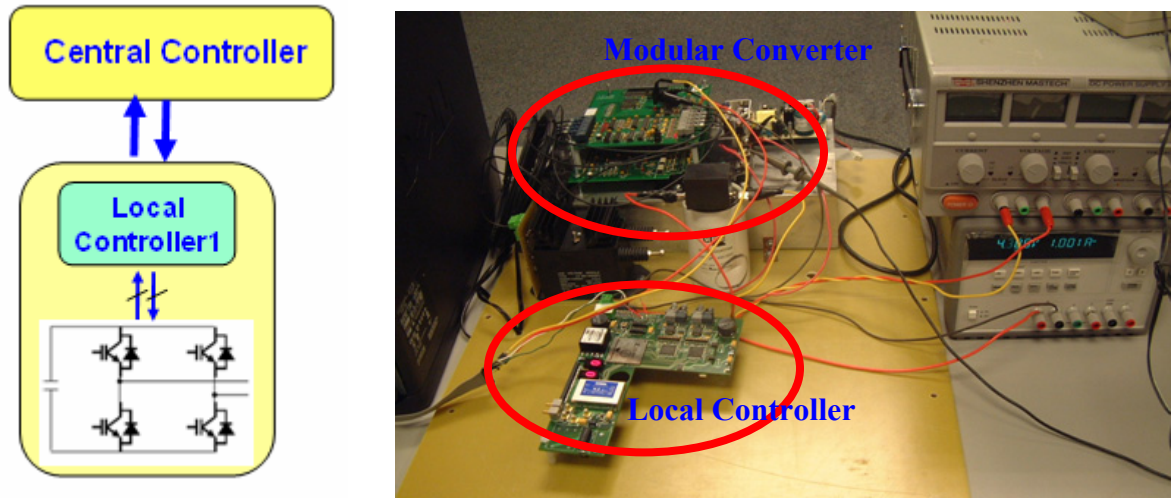


Fig. 4.11 Modular Controller/Converter Experiment Setup

In the experiment, DSP sends out a SPWM modulation signal to the local controller. The two top switches signals are shown in Fig. 4.12 .The measured converter output is a 3 level voltage waveform as shown in Fig.4.13. A 16 ohm R load is used as the output load.

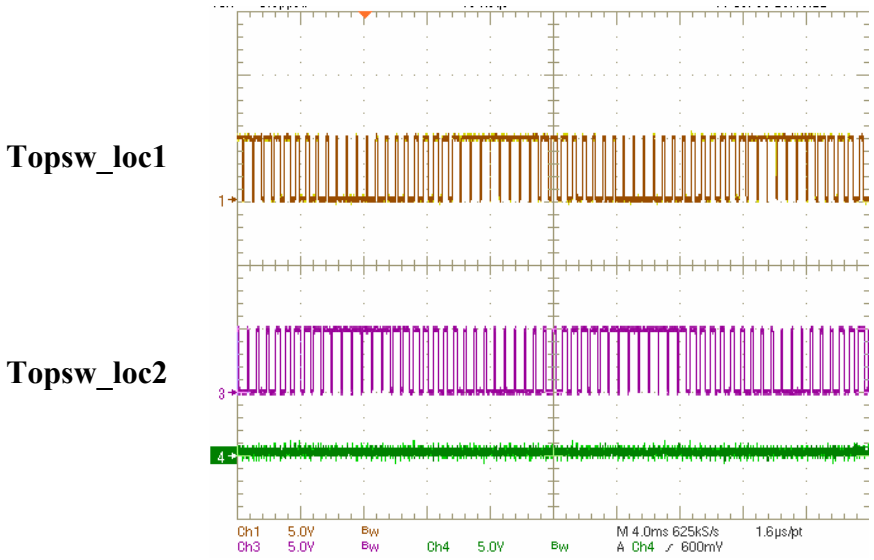


Fig. 4.12 SPWM signals

In Fig. 4.13, Ch2 is the measured output current and Ch1 is the converter output voltage.

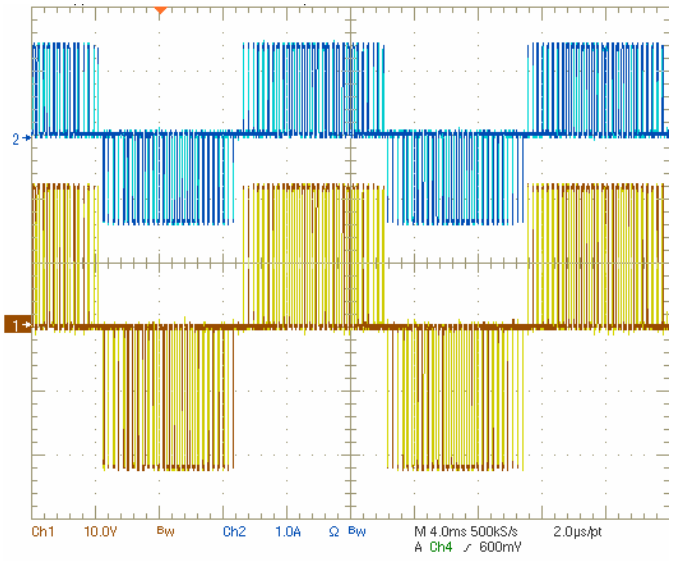


Fig. 4.13 converter output waveforms with R load

A RL load with L=2.5mH and R=6ohm is also tested and the results are shown in Fig. 4.14. Ch1 is the output voltage waveform and Ch2 is the output current waveform.

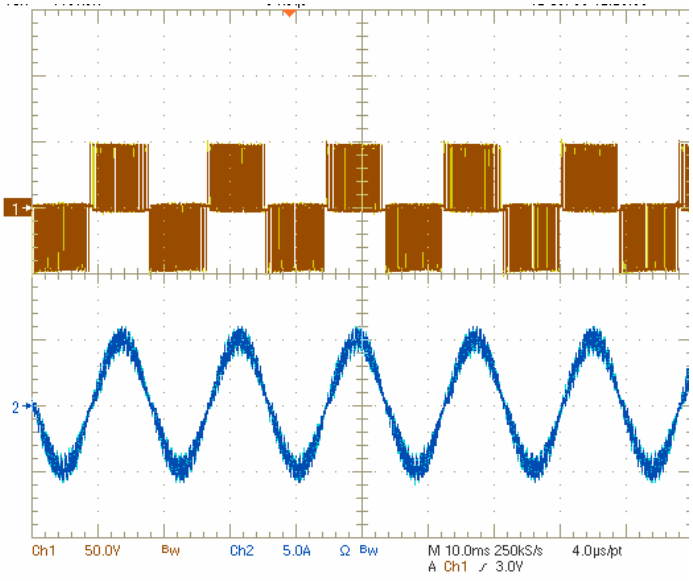


Fig. 4.14 converter output waveforms with RL load

Observed from the above analysis and experiment waveforms, the intelligent modular converter works well with the central controller.

CHAPTER 5

Synchronization of Local Controllers

5.1 Overview

In this chapter, synchronization among the local controllers is further analyzed in details. Firstly, the synchronization problem is described and confirmed with experiments. The real reason of the switch signal phase shift is explored. Simulations in Saber will also be shown.Also, literature study on the synchronization problem in modular controller architecture is conducted. Then, a couple of solutions to this problem are proposed. Finally, the implementation of the solutions and experiment validation is shown.

5.2 Description

As described briefly in the last chapter, there is a clock drift between the local controllers because the local controllers are not sharing a common clock. They have their own crystals to generate the clocks. The crystals are the same but with certain frequency tolerance. So the actual crystal frequencies have a small difference. This difference will lead to the phase shift of the generated PWM signals when multiple local controllers are used.

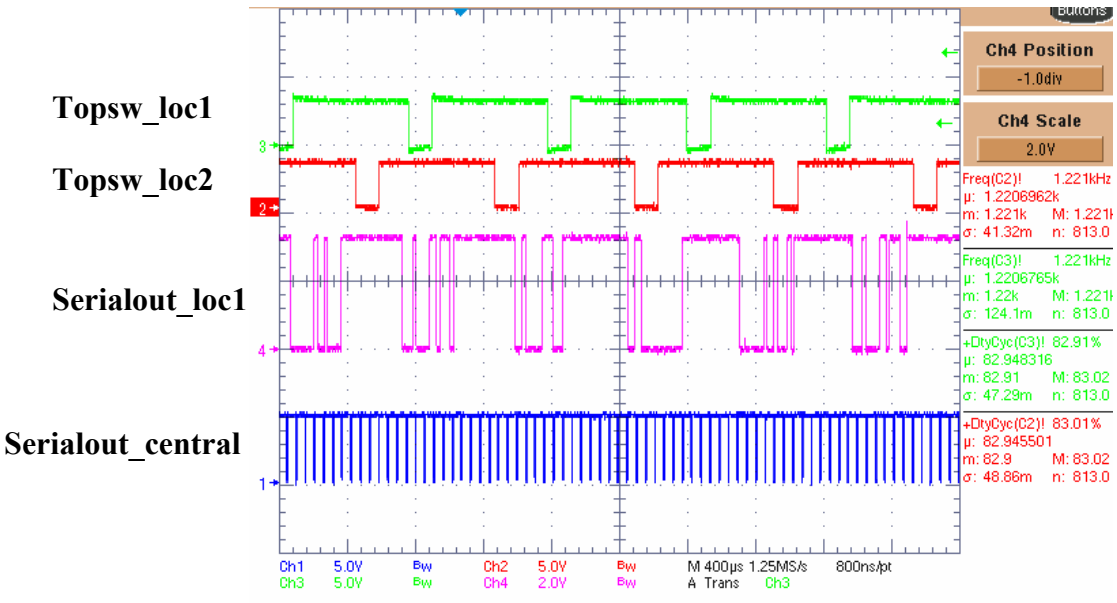


Fig. 5.1 PWM signals phase shift

In Fig.5.1, Ch1 and Ch2 are the switch signals generated by two local controllers. If the clocks of the two local controllers are in phase, the two switch signals should also be in phase. However, it is observed that the two signals will drift with a frequency of 0.05Hz that means they come back to be in phase after 20 seconds.

At converter level, if two intelligent converters are connected in series to construct a 5 level CMC converter, the output waveform will be a time varying waveform instead of a true 5 level waveform. The time varying waveforms are experimentally captured and shown in Fig.5.2. Observed from the waveforms, the worst output waveform is 3 level. The output Total Harmonic Distortion (THD) and the fundamental voltage magnitude will therefore have unacceptable variations.

More analysis are conducted using Saber simulations. An open loop simulation bench is setup to repeat this phenomenon. The simulation setup is shown in Fig. 5.3.

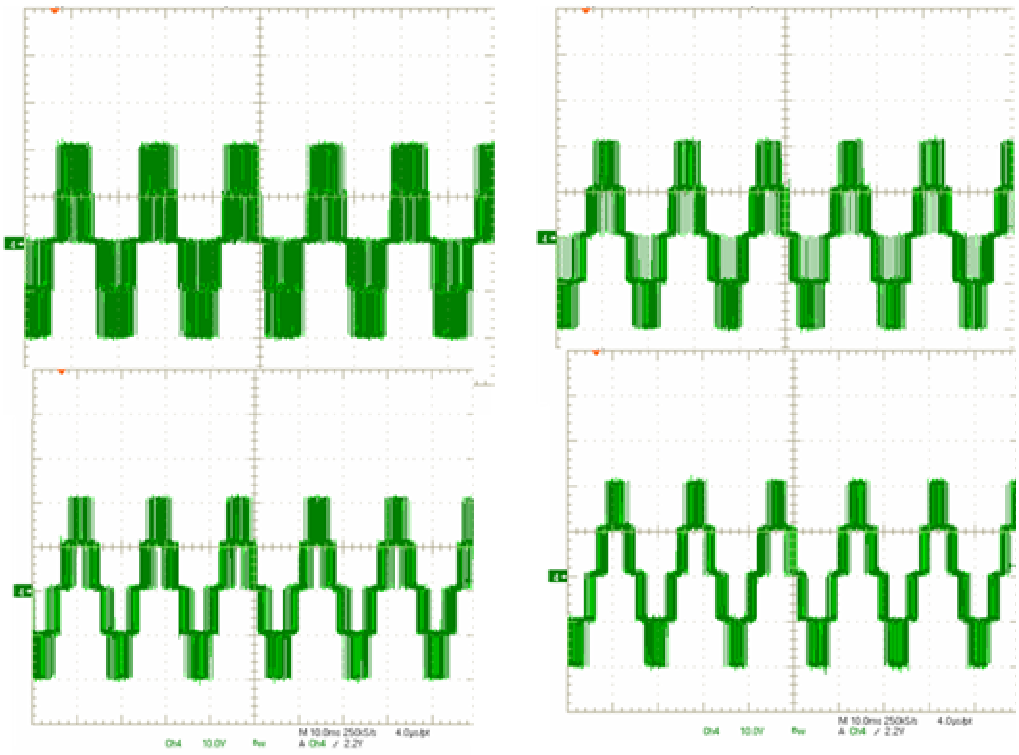


Fig. 5.2 Time varying output waveform

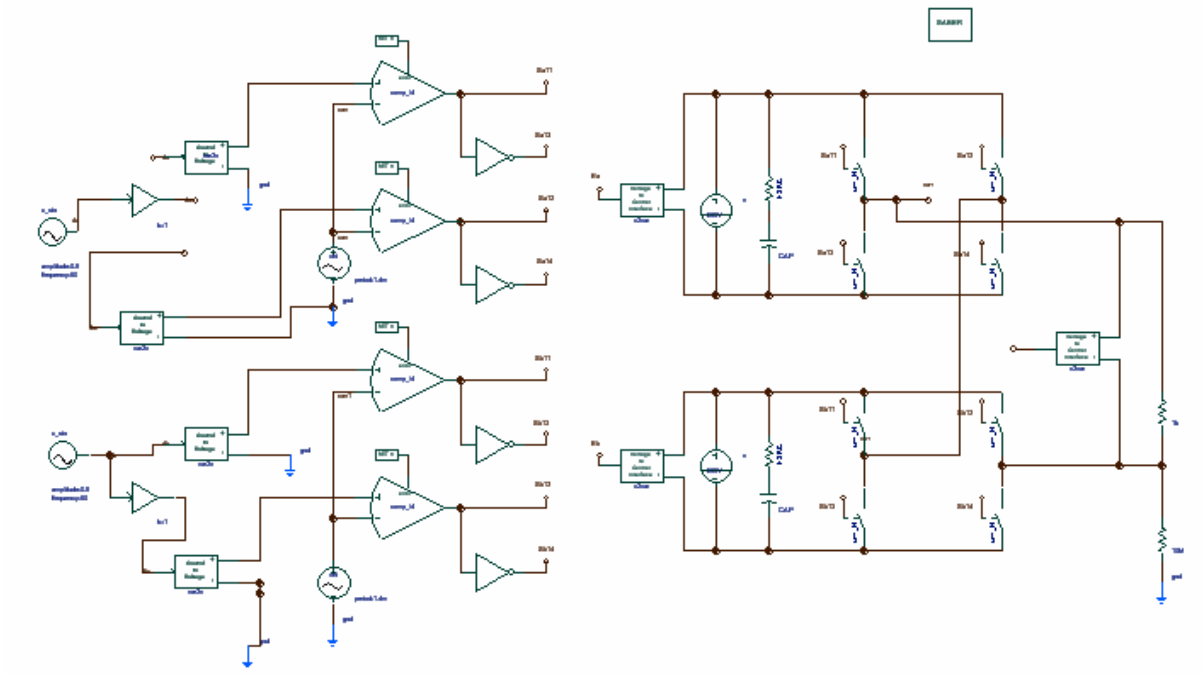


Fig. 5.3 Open loop simulation of clock drift

Carrier based modulation scheme is applied in the experiment and the simulation. So, the carrier phase shift will represent the clock drift. The simulation is done with carrier phase shift of 0, 25, 60, 90,180 degree. The parameters are shown in Table 13.

Table 13 Simulation Parameters for a 5 level CMC

DC Voltage(V)	Modulation	Carrier Frequency(Hz)	Carrier Phase shift
2000	0.9	720	0,25,60,90,180

The THD and fundamental values of the output voltage waveforms are measured with the difference phase shift. The results are shown in Table 14.

Table 14 Simulation Results for a 5 level CMC

Carrier phase Shift(degree)	Fundamental Magnitude(V)	Voltage THD(%)
0(three level)	3523	66.92
25	3595	52.56
60	3581	37.26
90	3513	34.03
180	3523	66.86

According to Table 14, the worst case THD is 66.92% and the optimized lowest THD is 34.03%. Moreover, the fundamental magnitude in these cases also has small variations.

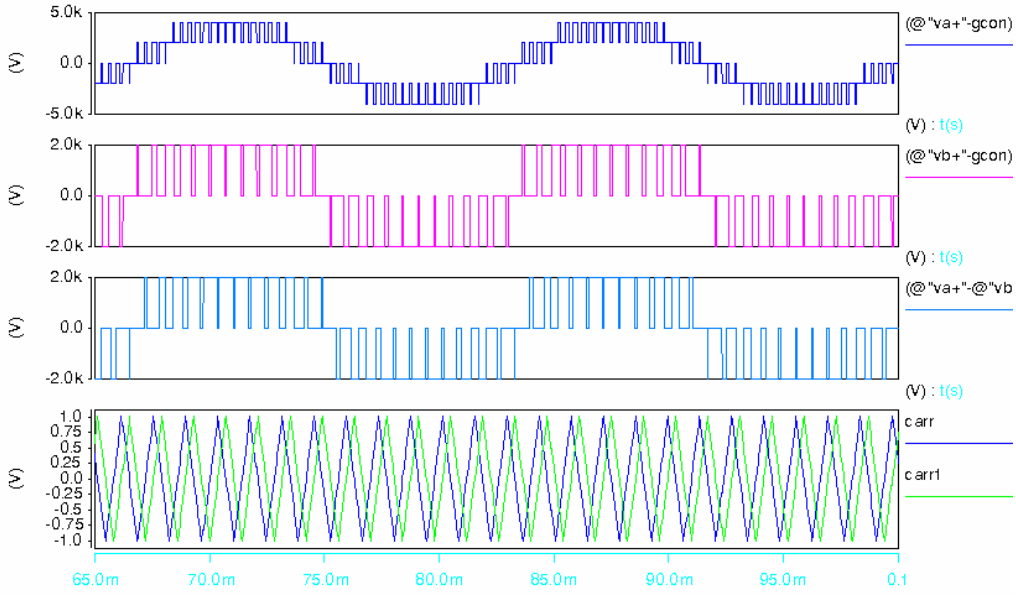


Fig. 5.4 Simulation waveforms with 90 degree carrier shift

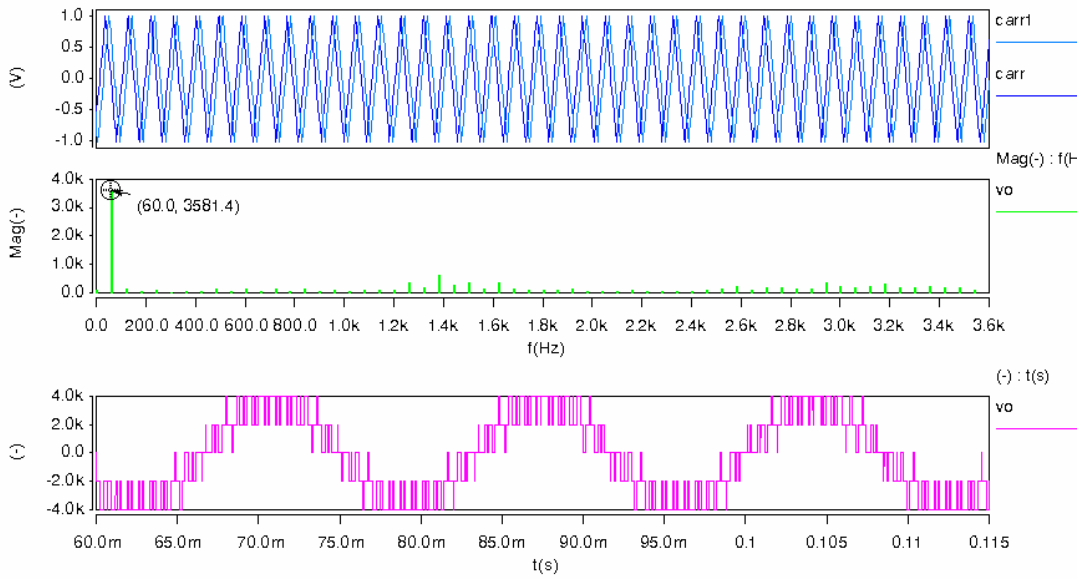


Fig. 5.5 Simulation waveforms with 60 degree carrier shift

The simulation waveforms are shown in Fig. 5.4-6 for 90, 60 and 0 degree phase shift respectively.

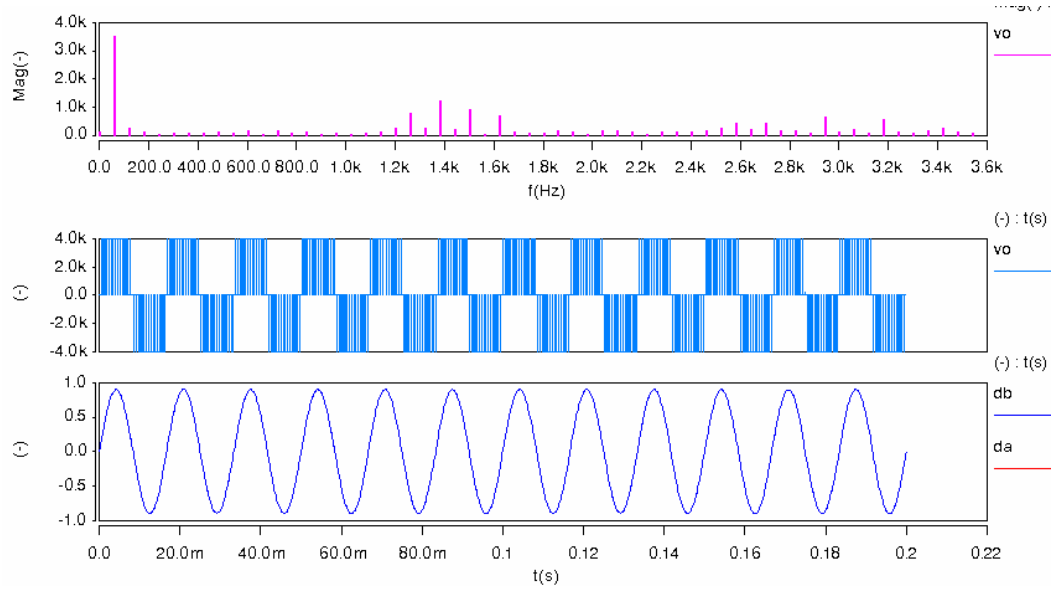


Fig. 5.6 Simulation waveforms with 0 degree carrier shift

From the above analysis, it can be stated that the phase shift problem is due to the clock frequency difference of the local controllers. Solutions to this problem are therefore needed.

5.3 Literature Research

Although there are some literatures about the modular controller architecture, only a few of them discussed the synchronization issue.

In [8], the ring structure is proposed for the distributed controller architecture. PESNET is defined as the communication protocol. In the PESNET 1.2, a synchronization packet is sent to the local nodes to activate the data after all the data information are sent. The synchronization packet is defined as shown in Fig.5.7.

	CMD	ADDR3	PADDING	ADDR2	PADDING	ADDR1
Size	4 Bits	8 Bits	8 Bits	8 Bits	8 Bits	8 Bits

Fig.5.7 PESNET 1.2 Synchronization Packet Structure

The concept is that each node will synchronize when they receive their own address in the data field. Since the addresses are sent in reverse order, all nodes should receive their own address at exactly the same time, hence synchronizing the nodes.

There is a problem if there are different requirements for synchronizing different nodes. If one group of nodes had to be synchronized every 100 microseconds, and another group of nodes had to be synchronized every 70 microseconds, then it will become complicated to synchronize these nodes. For example, when the synchronization times are nearly the same, and two packets have to be on the network at the same time, possibly at the same nodes (which would be impossible to realize). Moreover, the order of the packets would have to vary [8].

In PESNET 2.2, one of the most significant changes is the synchronization scheme. It is easy to explain the synchronous nature of PESNET 2.2 as a rotating gear with n teeth rotating in a circular cavity with n slots, as shown in Fig.5.8, where n is shown as 8. Between every two adjacent teeth is a packet of information. The gear rotates in a lock-step fashion, moving to the next of the n positions and locking. While locked, the packet is read and is either replaced by another packet of information, or is consumed (effectively replacing the original ball with a “NULL” packet.), or remains untouched. The gear will continue to rotate in this fashion, where at all times, all nodes have a packet to replace or ignore. All packets leave the current node and arrive at the next node at the same time. In the middle of the gear is a counter (clock) that is incremented every time the gear rotates to

the next notch. Every node has the same value of this clock.

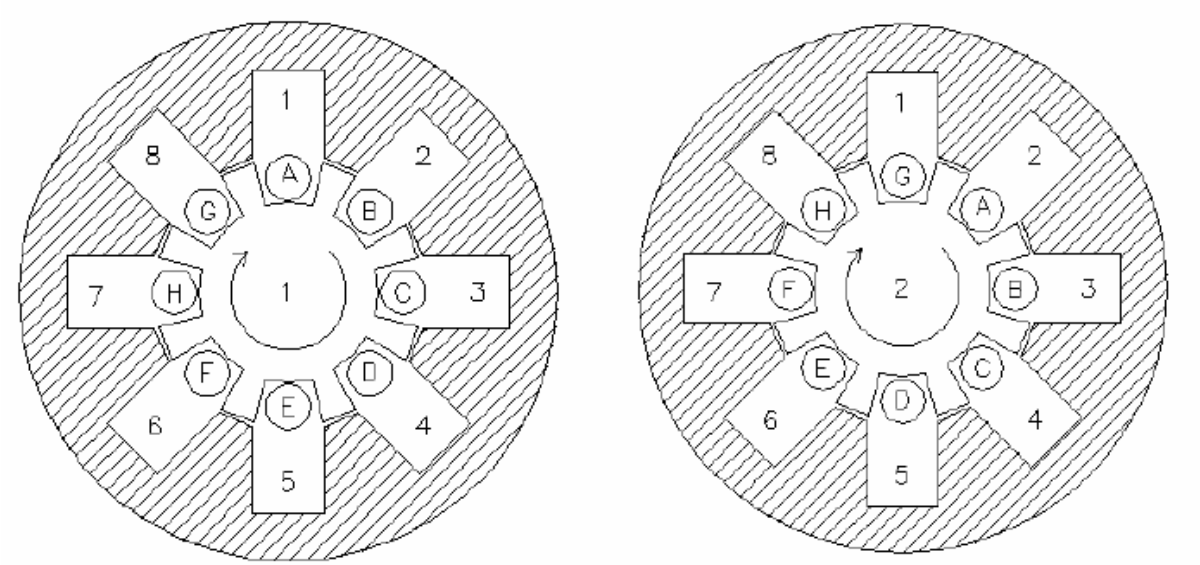


Fig. 5.8 Gear Conceptualization of PESNET [8]

For both PESNET 1.2 and 2.2, the synchronization scheme is quite complex and hard to implement with the hardware and software.

In [9], two schemes are discussed for synchronization between the local nodes that are controlled by local DSPs. Firstly, assuming that all slave DSP processors share a common master clock, the PWM counters of the slave controllers for each cascaded H-bridge need to be loaded with the appropriate phase shift values, and subsequently started and synchronized with the common master clock signal. This ensures that the triangular carriers for each cascaded H-bridge are always correctly and optimally phase-shifted, and stay synchronized as time continues. Secondly, if each slave power bridge has its own processor clock, carrier misalignment among the slaves can occur due to possible drifting of individual clock counting. To maintain the “correct” phase shifts among the carriers, the master controller now has to send out a common square synchronization signal relative to which all slave processors measure their respective carrier phase shifts and auto-compensate for any carrier

misalignment upon detected. This square synchronization signal can easily be implemented using a digital logic bit on the master controller, programmed to output a rising edge (falling edge) at the start (mid) of every carrier period [9].

5.4 Proposed Solutions, Implementation & Experimental Validation

The solutions found in previous literatures are unable to solve the problem in the proposed architecture. Firstly, too complex communication protocol is not good for real world implementation. Moreover, the first solution in [9] is to share the common clock among the local controllers. It will make sure of the synchronization of all the local controllers. However, a clock tree is required and it will increase the system complexity. The second solution in [9] is for DSP-DSP communication, it can not be transferred to a DSP-FPGA-FPGA system directly.

To achieve easy synchronization between the local controllers, the following scheme is proposed. The principle of the proposed scheme is to send a synchronization signal from the central controller to the local controllers.

5.4.1 Separate synchronization line from the data line

To achieve the proposed synchronization scheme, separate synchronization lines can be used apart from the data line.

As shown in Fig.5.9, one synchronization line is added for each converter. The on time of the synchronization signal is about 80ns and the frequency is 2 kHz.

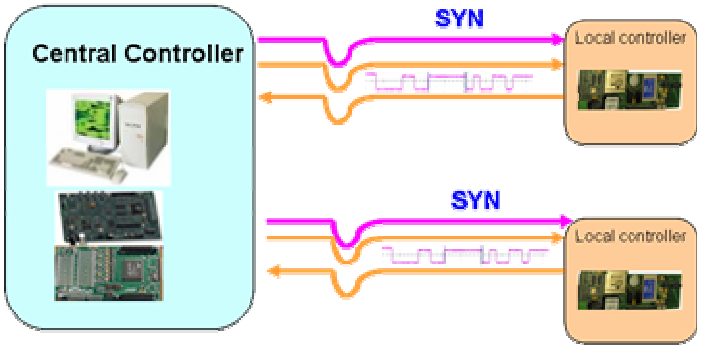


Fig. 5.9 Separate Synchronization line from the data line

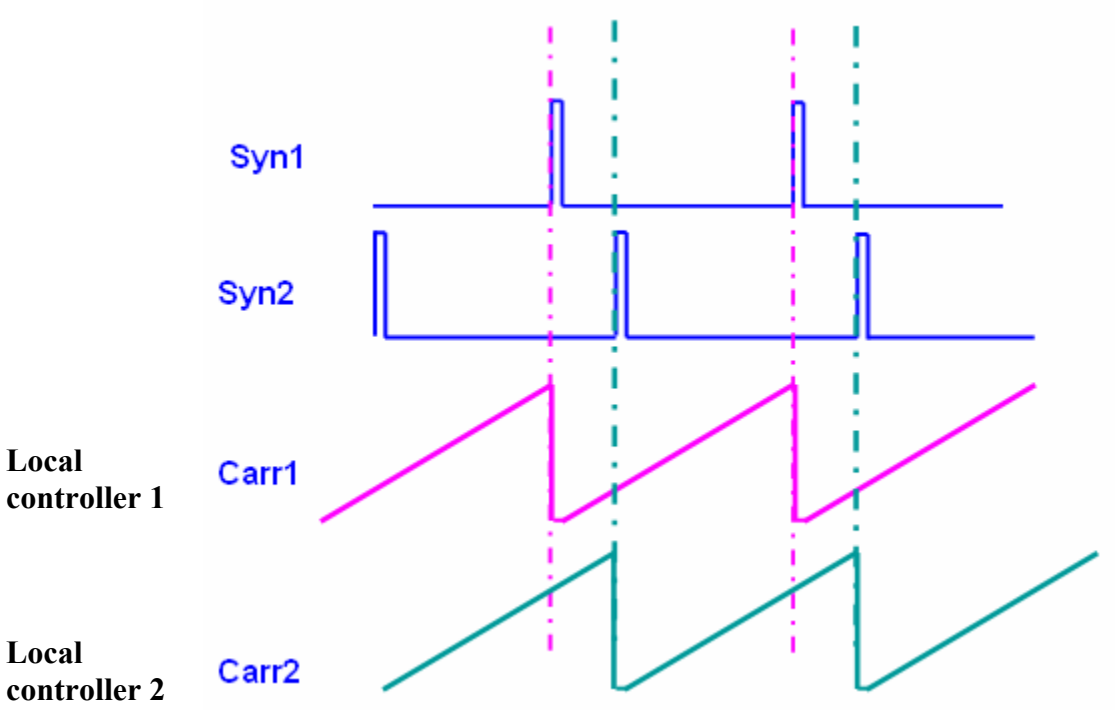


Fig. 5.10 Synchronization signals and the carriers in local FPGA

As shown in Fig.5.10, the local carriers will have a frequency of 2 kHz that is determined by the synchronization signals from the central controllers. When the synchronization signal is high, the carrier is reset to '0' and is kept until the synchronization signals go back to '0'.

As the synchronization signal is a very short pulse (much less than 0.1us) and the period of the carrier is 500us in this design, the carrier waveform distortion is less than 0.02%. So, this carrier distortion could be neglected.

A couple of experiments have been conducted to validate this design. The measured synchronization signals are shown in Fig. 5.11. The frequency of the synchronization signals is 2 kHz. The pulse width is 80ns. The switch signals are measured with a constant duty command from the central controller. Observed from the waveforms, ch3 and ch4 are the synchronization signals. They have 90 degree phase shift. The ch1 and ch2 are the top switch signals generated in the local controllers. They have 90 degree phase shift as desired. The clock drift phenomenon is not observed in the experiment.

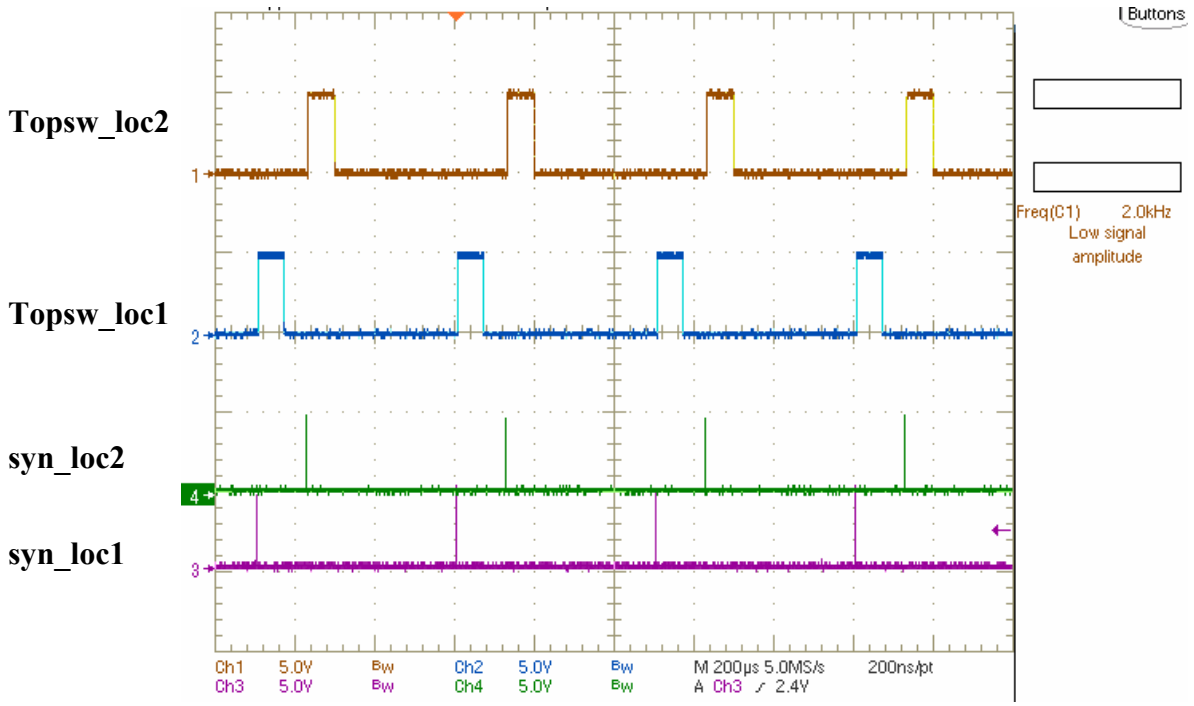


Fig. 5.11 Measured synchronization & switch signals

In Fig. 5.12, the experiment results of a SPWM modulated 5 level converter are shown. A SPWM modulation signal is sent from the central to the local controllers. The two local

controllers are sharing the same modulation signal but has a 90 degree phase shifted synchronization signal. With this synchronization scheme, the clock drift problem is eliminated and the output voltage is keeping stable as a 5 level waveform.

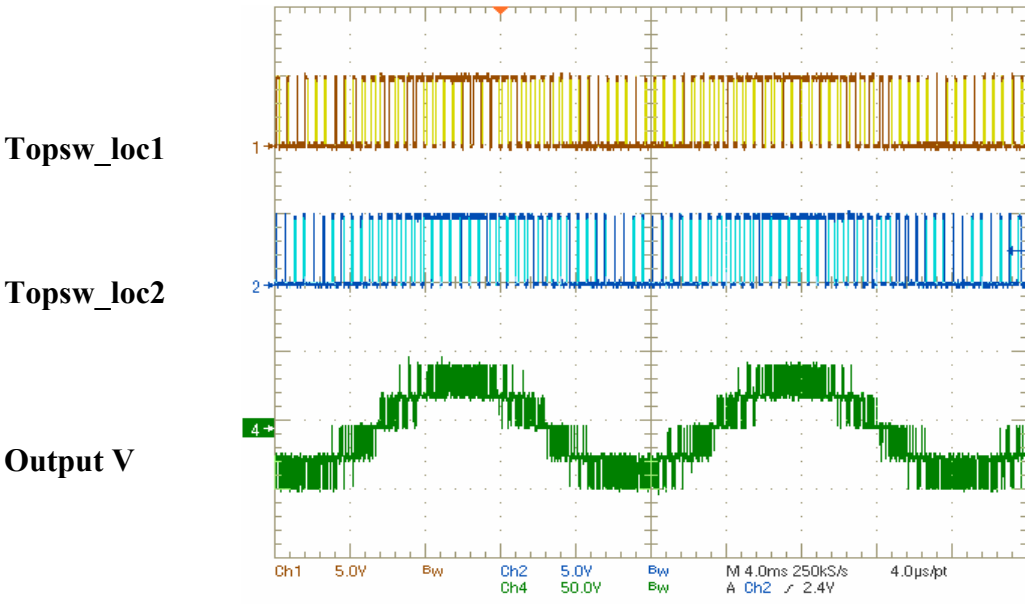


Fig. 5.12 SPWM signals and converter output waveforms

The data of the output voltage waveform is collected and analyzed with MATLAB.

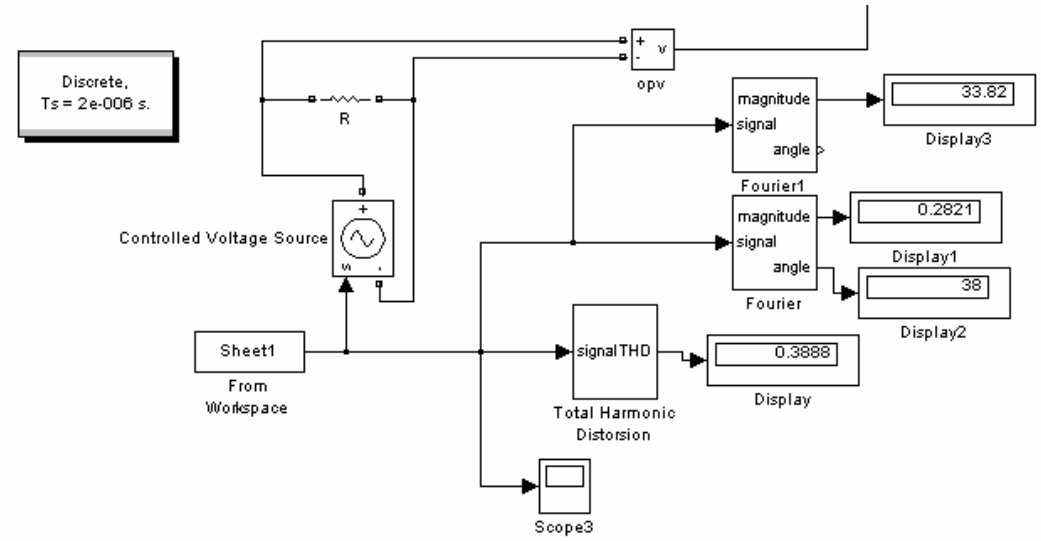


Fig. 5.13 MATLAB FFT analysis bench

Two methods are applied to analyze the data. One way is to use the THD and harmonic measurement blocks in MATLAB. The other way is to use POWERGUI to measure the THD. It is shown in Fig.5.13 that the measured THD is 38.88% and the fundamental magnitude is 33.82. In Fig. 5.14, the measured waveform is shown and the maximum frequency in the spectrum is 10 kHz (5 times of the switching frequency). The measured fundamental value is 32.86 and the THD is 34.22%. The results with the two approaches are quite close.

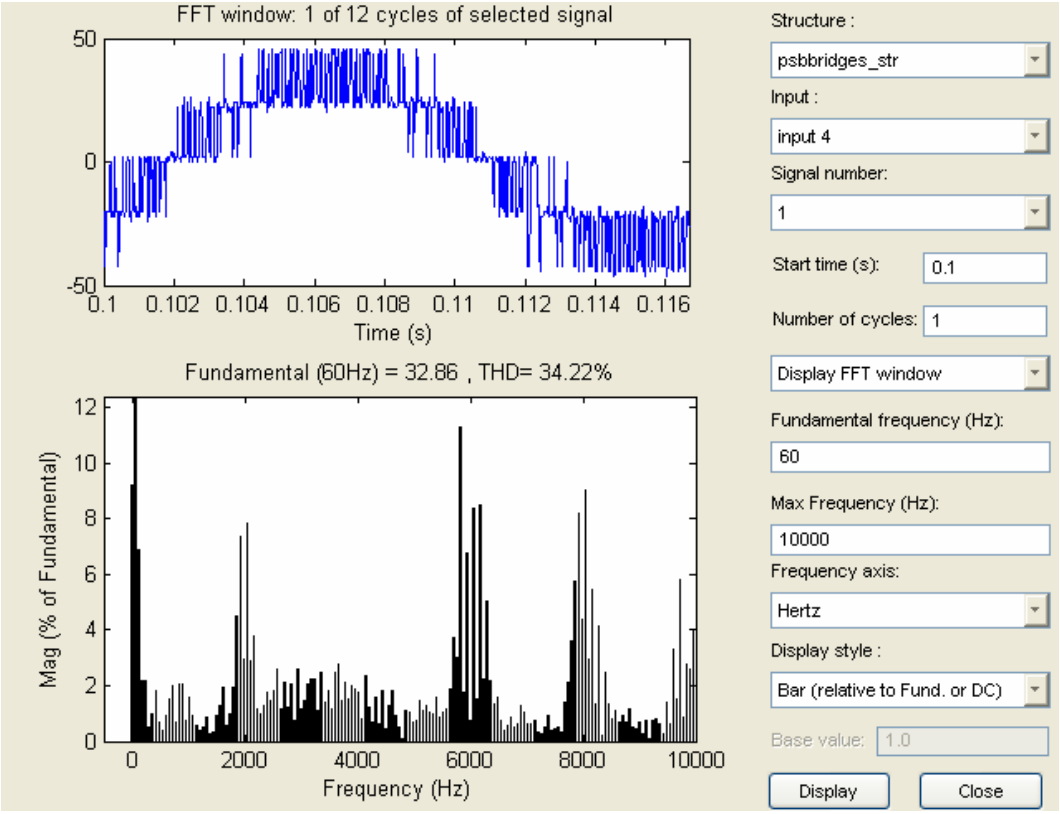


Fig. 5.14 FFT analysis of the measured voltage waveform

With the above analysis and experiments validation, this proposed synchronization scheme is confirmed to be able to solve the synchronization problem of the local controllers.

5.4.2 Incorporate synchronization in the data line

With the above approach, one more line will be added for each converter. It is therefore desirable to incorporate the synchronization signal into the data line.

To achieve this proposal is to set a bit in the forward path as the synchronization bit. The revised protocol is shown in Fig. 5.15.

Fig. 5.15 Communication Protocol with synchronization bit

The local controller receives the synchronization bit and converts it to a synchronization signal. The received synchronization signals for two local controllers are shown in Fig. 5.16. In this diagram, the phase shift is intentionally sent by the central controller.

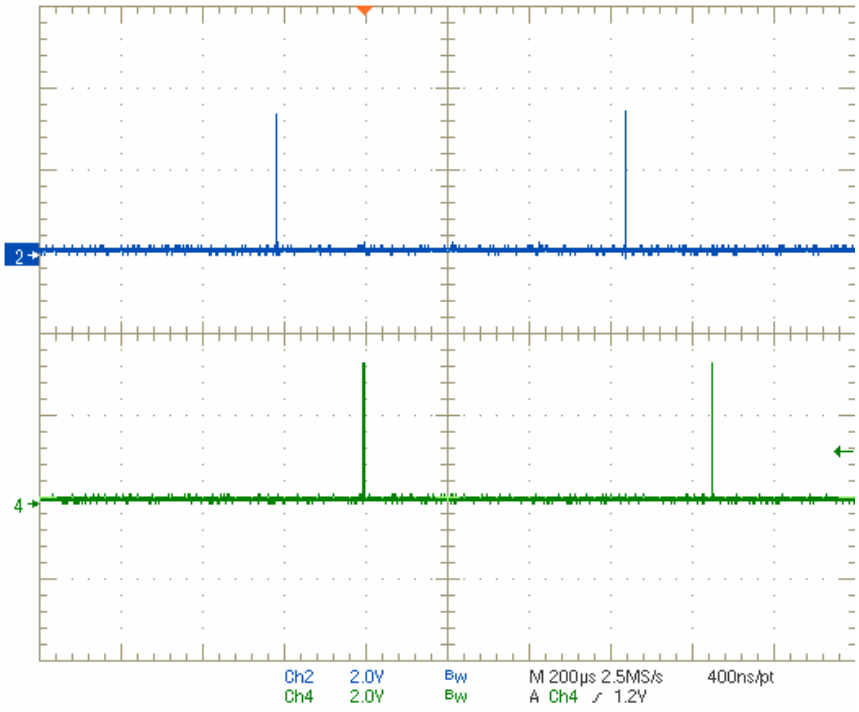


Fig. 5.16 Synchronization signals

As can been seen, the frequency of the synchronization signals determines the local controller carrier frequency. The phase shift of the synchronization signals will also determine the phase shift of the local carriers.

The resulting SPWM signals for the two local controllers are shown in Fig.5.17.

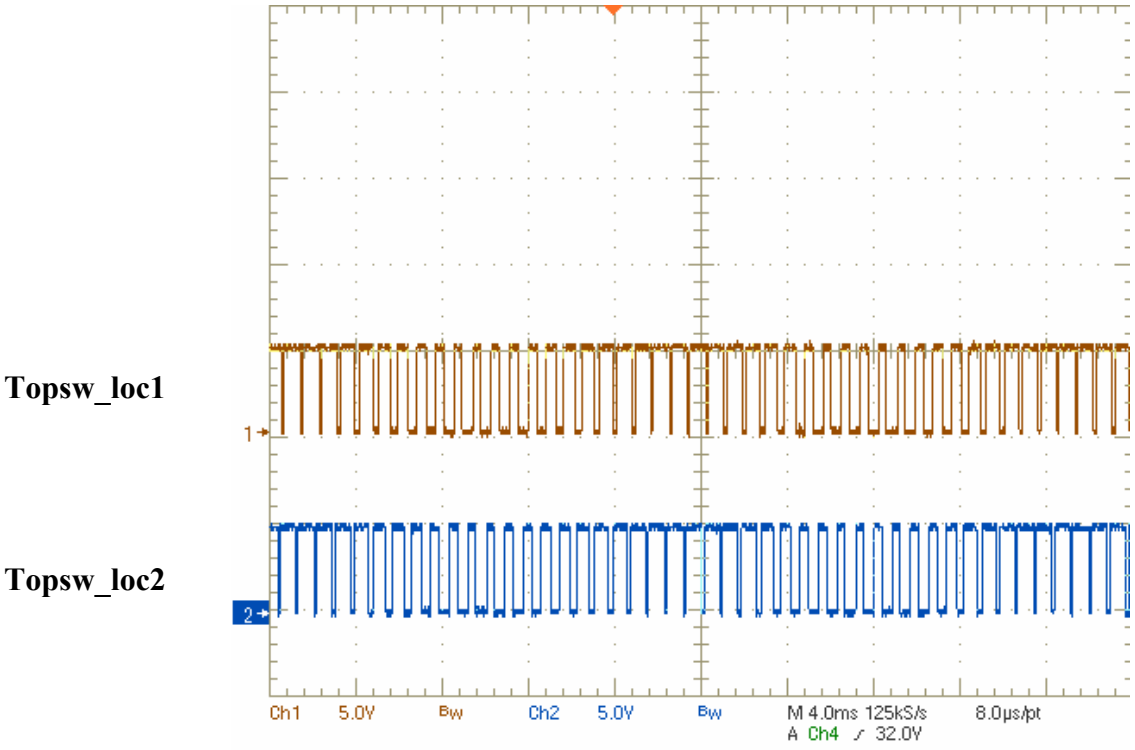


Fig.5.17 SPWM signals for two local controllers

In order to obtain further verification, the two local controllers are connected in series to achieve a five level converter. The output voltage waveform is shown in Fig. 5.18. Ch4 shows the output voltage waveform. Ch1 and Ch2 are two SPWM signals of the local controllers.

In this design, the synchronization signal is a 100ns pulse with a frequency of 1.17 kHz. The frequency is determined by the data transmitting rate on the DSP side. This will be discussed later in this chapter.

Topsw_loc1

Topsw_loc2

Output V

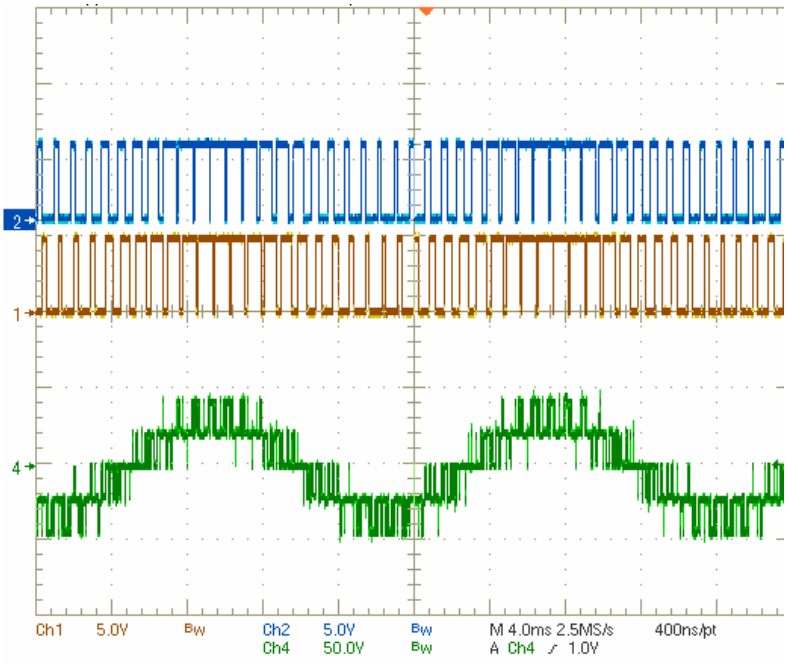


Fig. 5.18 Output voltage waveform and SPWM signals

The data of the output voltage waveform is analyzed with the MATLAB THD analysis bench mention previously. The maxim frequency of the spectrum is 10 kHz. One fundamental cycle of the waveform is analyzed. The calculated total harmonic distortion is 39.92%. It is matched with the desired result for 90 degree phase shift as shown in Table 14. The harmonics spectrum is shown in Fig. 5.19.

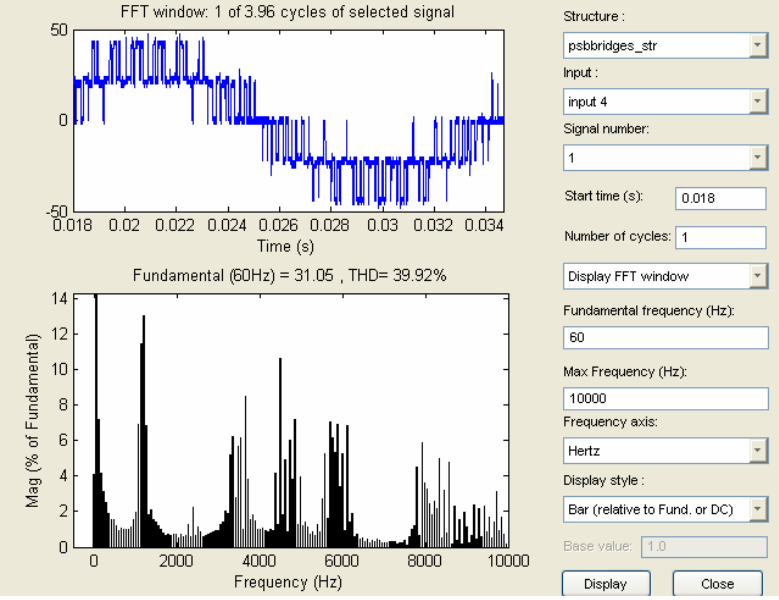


Fig. 5.19 THD Analysis

5.5 Summary & Discussions

In this chapter, the synchronization issue among the local controllers is investigated. A solution is proposed to solve this issue and the solution has been validated with experiments. The measured output voltage waveform is analyzed with MATLAB tools. The measured THD and spectrum also confirm the proposed solution.

The pulse width of the synchronization signal will lead to distortion of the local carriers. However, as the pulse is only around 100ns that is quite small compared with a switching period of 1ms or more, the carrier distortion could be neglected in such PWM converters with low switching frequency.

The proposed solution not only solves the synchronization problem, but also determines the switching frequency of the local carriers that means the central controller can change the switching frequency according to the applications. As the synchronization signal is sent out together with the modulation data, the frequency to send out the data will also affect the

frequency of the synchronization signal. In the above experiments, the data is sent out each $Tdata$ that is 220us determined by DSP. So for a 5 level converter, to achieve 90 degree phase shift, according to the following equation, the maximum switching frequency is 1.17 kHz.

$$f\max = \frac{1}{Tdata * 2 * N}, \quad \text{N is converter number, such as 2 for a 5 level converter}$$

For a 7 level, the maximum switching frequency is 758 Hz to achieve optimum harmonic elimination. For a 9 level converter, the maximum switching frequency is 585 Hz to achieve optimum harmonic elimination. The $Tdata$ is determined by DSP. To avoid this limitation, the $Tdata$ can be controlled by the central FPGA. No matter if there is new data or not from the DSP, the central FPGA sets the speed rate to transmit data. This approach has also been experimentally verified. The converter output waveform is not affected with this approach, and the switching frequency can be independently sets by the central FPGA.

From above analysis, in the final proposed solution, the local carrier switching frequency is determined by the synchronization signal from the central FPGA.

CHAPTER 6

Summary and Future Work

6.1 Summary

The main contribution of this thesis is the development of a novel distributed modular controller architecture. This structure has a number of advantages compared with the conventional structures. It increases the modularity, system reliability and reduces noise interference to the system.

A simple and reliable communication protocol is developed to achieve this structure. The central controller and local controller hardware and software implementations are described. Simulations and experiments have been conducted to validate the proposed architecture. The accomplished experimental validations of the proposed architecture are listed as following:

a. Forward path communication
    i. Send data from central controller and measure the PWM signals at the local controller to check accuracy
b. Feedback path communication
    i. Send analog signals to the local controller and measure the received values in the central controller to check accuracy
    ii. Test with multiple local controllers

c. Test a central controller with an intelligent modular converter, check the output voltage waveform with R and RL load
d. A two phase converter with modulation signals of 120 degree phase shift is tested and verified
e. A single phase 5 level converter is tested and verified

6.2 Limitations

In the developed modular controller architecture, there are two main limitations. The first one is that the clock frequency of the serial data receiver (local FPGA) and transmitter (central FPGA) should be close to each other. Otherwise, the transmitter will be unable to receive the right data. This is the limitation of the UART protocol. Therefore, in practical implementation, the transmitting speed (clock frequency) and receiving speed should be defined.

The second limitation is the DSP data A/D conversion. The conversion is done with the equation shown in Equ.3.1. The digital data FFFF is the peak value of the carrier in the local controller. It is determined by the switching frequency and the counter frequency of the local carrier. Assume the digital data is represented with X, the switching frequency is fsw, the counter frequency is fclk. The calculation equation is $X = \frac{fclk}{fsw}$. So the central controller designer should know the switching frequency and the counter frequency of the local controllers.

6.3 Future Work

Distributed modular controller is a quite attracting research topic. There are still a lot of things to investigate in this area.

Firstly, more studies could be done with other modulation schemes such as fundamental switching and SVM.

Secondly, the hardware part of the 5 level TNA has been completed. While the ac interface and all the control boards connected, the 5 level STATCOM is able to demonstrate the modular controller design and validate the 10MVar conceptual design.

What is more, the proposed architecture and communication is control scheme and power converter topology independent. So, more converter topologies could be applied to validate the design.

Finally, there are still some practical issues while the modular controller is actually mounted on a real STATCOM. Power supply of the local controller should be considered. Also the switching noise interference to the local controller should be avoided.

REFERENCES

1. Understanding FACTS Concept and Technology of Flexible AC Transmission Systems, NARAIN G. HINGORANI, LASZLO GYUGYI
2. Jih-Sheng Lai and Fang Zheng Peng, "Multilevel Converters-A New Breed of Power Converters", Industry Applications, IEEE Transactions on Volume 32, Issue 3, May-June 1996 Page(s):509 - 517 Digital Object Identifier 10.1109/28.502161
3. Siriroj Sirisukprasert, "The Modeling and Control of a Cascaded-Multilevel Converter-Based STATCOM", PhD Dissertation,2004, Virginia Tech
4. T. Ericsen, A. Tucker, "Power Electronics Building Blocks and Potential Power Modulator Applications," IEEE Conference Record of the Twenty-Third International Power Modulator Symposium,New York, hY, pp. p.12-15; 1998.
5. Van Wyk, F.C. Lee, "Power Electronics Technology at the Dawn the New Millennium-Status and Future," IEEE PESC Conference 1999
6. J.A. Du Toit, A.D. Le Roux, J.H.R. Enslin, "An Integrated Controller Module for Distributed Control of Power Electronics," IEEE APEC'98 Proceedings, pp. 874-880, February 1998
7. P.L.G. Malapelle, G. Tom, R. Moruzzi, A. Oliva, "A New, Modular, Programmable, High Speed Digital Control for Large Drives," IEEE IECON 20th International Conference on Industrial Electronics, Control and Instrumenfation 1994; pp. p.210-14 vol.1.
8. Gerald Francis," A Synchronous Distributed Digital Control Architecture for High Power Converters", master thesis, Virginia Tech

9. Poh Chiang Loh; Holmes, D.G.; Lipo, T.A.;" Implementation and control of distributed PWM cascaded multilevel inverters with minimal harmonic distortion and common-mode voltage", Power Electronics, IEEE Transactions on Volume 20, Issue 1, Jan. 2005 Page(s):90 – 99
10. Texas Instrument, "DSP 6701 datasheet", available from Texas Instrument website
11. AED-106 manual, available from Signalware corp.
12. Fang Zheng Peng, Jih-Sheng Lai, "Dynamic performance and control of a static Var generator using cascade multilevel inverters", Industry Applications, IEEE Transactions on Volume 33, Issue 3, May-June 1997 Page(s):748 - 755 Digital Object Identifier 10.1109/28.585865
13. Celanovic, I.; Celanovic, N.; Milosavljevic, I.; Boroyevich, D.; Cooley, R.;" A new control architecture for future distributed power electronics systems", Power Electronics Specialists Conference, 2000. PESC 00. 2000 IEEE 31st Annual Volume 1,18-23 June 2000 Page(s):113 - 118 vol.1
14. I. Milosavljevic, D., Borojevic., I. Celanovic, "Modularized Communication and Control Structure for Power Converters," European Conference on Power Electronics and Applications, EPE,September 1999 in press.
15. Milosavljevic, "Power Electronics System Communications," M.S. thesis, Virginia Polytechnic Institute and State University, 1999.