Comparative Study of PES Net and SyCCo Bus: Communication Protocols for Modular Multilevel Converter

Hao Tu, Srdjan Lukic
FREEDM Systems Center
Department of Electrical and Computer Engineering
North Carolina State University, Raleigh, US
Email: {htu, smlukic}@ncsu.edu

***Abstract*—Due to its modularity and scalability, modular multilevel converter is a promising topology for medium voltage applications, such as energy storage or motor drives. Because of the large number of modules, the wiring can become complex if the conventional star topology communication network is used. To solve this issue, a communication link that uses a ring topology to connect the central controller and the modules provides an efficient solution. The information exchanged on this link includes real time data, such as duty cycle, measurements and fault feedback. Thus, a dedicated high-speed communication protocol is needed. This paper reviews two possible protocols for modular multilevel converter communication, PES Net and SyCCo bus. After introducing the operating principles, their performance is compared and discussed. SyCCo bus is implemented using Altera Cyclone IV FPGA. Tests are conducted to analyze the communication delay of SyCCo bus. The implementation is made open source.**

I. Introduction

Modular multilevel converter (MMC) is a promising topology for many medium voltage applications [1]. It offers several distinct advantages over traditional converters. First, with its high number of modules, the harmonic content in the output voltage is low. Smaller output filter can be used compared to traditional converter. Also, if a sufficient number of modules are stacked in series, the converter can be connected to the medium voltage grid directly. The bulky transformer which is needed for traditional converters can be eliminated. Furthermore, because of its modularity, MMCs can achieve fault tolerant operation by bypassing the faulty modules. All of those advantages are possible because of the large number of modules in the MMC. For example, the more the modules, the more the voltage levels thus the better the harmonic performance. However, the control of MMCs becomes challenging with the increasing number of modules.

In a conventional MMC topology, a central controller is used to run the control algorithm [2]–[4]. The controller and power electronics modules are connected using a star topology, i.e., the controller has direct connections to all the modules. Every piece of information exchanged between the central controller and module has its own dedicated communication channel. When the number of modules becomes large, the complexity, cost, and reliability of the star communication link becomes an issue [5].

One alternative way to control the MMC is to employ ring topology communication network between modules and the main controller. In this communication network, a module only communicates with its neighbor. The information exchange between the controller and the module is realized with the help of other modules. Compared with conventional star topology, this leads to a reduction of the number of I/Os of the controller and simplification of the wiring. Another advantage of this topology is that some part of the control algorithm can be distributed to the modules, reducing the computational burden of the central controller [6]. However, for this communication network to work, a real time communication protocol is needed. The efficiency of the protocol greatly influences the performance of the converter.

In this paper, two communication protocols suitable for MMCs are reviewed. They are Power Electronics System Network (PES Net) and Synchronous Converter Control bus (SyCCo bus). PES Net was developed at Virginia Polytechnic Institute. for the concept of power electronics building block (PEBB) [7]. The idea is to build converter systems using one basic module called PEBB [8]. Communication as part of the PEBB should support system modularity. Hence, a ring topology communication network is chosen. SyCCo bus was developed at ETH Zurich for modular converter systems [9]. It shares some common features with PES Net such as ring topology and FPGA-based implementation. However, their operating principles and synchronization methods are different, which leads to difference in performance in terms of communication speed and synchronization accuracy.

The contribution of this paper is twofold: First, this paper provides an apples-to-apples comparison of the two common protocols for control and communication in power electronics applications, pointing out the implementation saliencies and challenges of each. Second, the paper presents a software and hardware reference design for the implementation of the protocols. The implementation is made available to the community through an open source repository located at [10].

The reminder of this paper is arranged as follows: Section II introduces the basic topology of the MMC and its communication network topology. In section III the operating principles of PES Net, protocols are reviewed. In section IV the features of SyCCo bus are discussed. In section V the performance of PES Net and of SyCCo bus are compared. The hardware implementation is introduced in section VI. Section VII concludes the paper.

II. MMC Topology and Communication Network

Fig. 1 shows a simplified three-phase MMC diagram. Due to its topology, MMC control mainly has three control targets. First, the output current must be controlled to the reference value. Second, the circulating current between phases should be controlled. Third, the capacitor voltages in one arm should be balanced. To fulfill those control objectives, the minimal information exchange between the controller and the circuit includes gate signals and measurement data such as the arm current and capacitor voltage. The information exchange can happen either in a star topology or ring topology communication network.

In a star topology communication network, one central controller is connected directly with all the gate drivers and measurement circuit. All the signals run in parallel. The star topology becomes cumbersome if it is used in the modular multilevel converter. The wires and interfaces become hard to manage if the number of

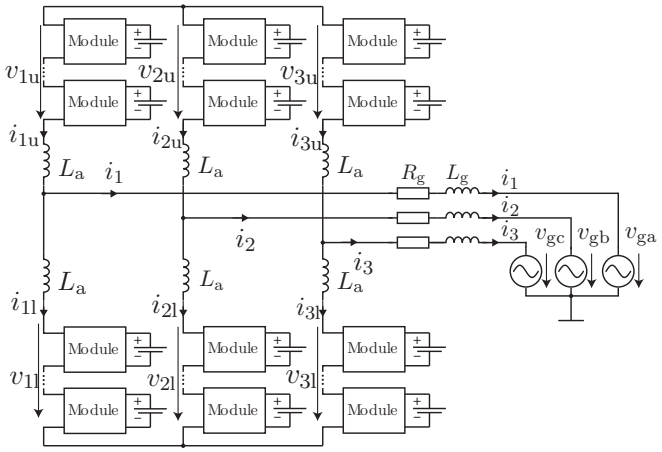

Fig. 1. Simplified circuit diagram for MMC

modules is large. Also, since all the data must be processed by the central controller, this proposes a big challenge for the I/Os and computational power of the central controller.

Unlike star topology, ring topology is an efficient alternative communication approach for MMCs. In ring topology, each module is connected in series as shown in Fig. 2. The central controller which is the master in Fig. 2 only needs one pair of I/Os to communicate with the first and the last module in the ring. The transmitter of one module is connected to the receiver of the next module. Thus, the serialized data can reach every module via this topology. It is also easy for ring topology to achieve redundancy. Adding another channel can avoid system failure caused by single point failure in the communication system [11].

Ring topology communication network is widely used in industrial automation. Two examples are MACRO developed by Delta Tau Data Systems and Ethercat by Beckhoff GmbH. Both of them define a high speed communication network using ring topology. However, because they are general protocol designed for industrial automation, the communication efficiency would be low if they are adopted directly for the power electronics system communication without any modification.

If a ring topology communication network is to be deployed for power electronics systems, specifically MMC, several challenges can be quickly identified. First, all the information exchange and computation must be completed in one switching period. As the switching frequency of today's power electronics device can be very high, the bandwidth of the communication should also be high enough. Second, as each module in the MMC receives and sends its information at different time instants, a synchronization mechanism must be implemented to make sure all the modules update their data at the same time. Last, to avoid system failure caused by single point failure in the communication system, this communication network should be at least single fault tolerant. To meet the specific needs for power electronics systems, research groups in Virginia Polytechnic Institute and ETH Zurich have adopted the basic operating principle of MACRO and Ethercat, respectively, to create their own communication protocols, namely PES Net and SyCCo bus.

III. PES NET

PES Net was developed by Virginia Polytechnic Institute based on communication protocol MACRO. It uses optical fiber as the transmission medium and a bit rate as high as 125 Mb/s.

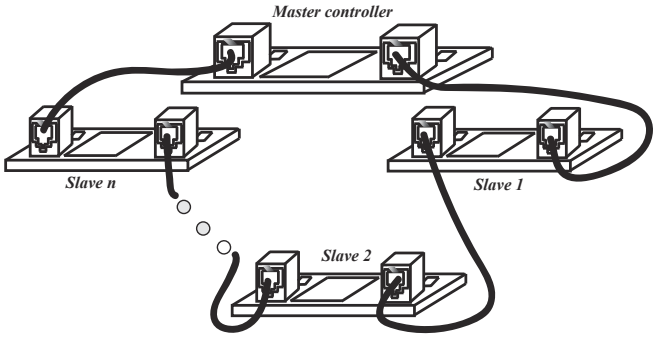

Fig. 2. Ring topology communication network

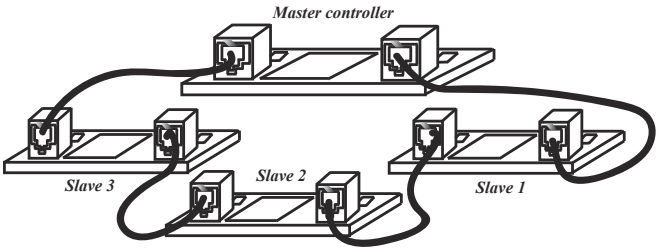

Fig. 3. Simplified communication chain with a master controller and three slaves

A. Fundamental operating principle

PES Net uses a ring topology. A simplified diagram with a master controller and three slaves is in shown in Fig. 3. If the master wants to send a message to a slave, for example slave 3, a data frame as shown in Fig. 4 is constructed. The frame starts with an address identifier to indicate it contains message to a specific slave. Then, the address of the slave is given in the address field which is followed by the data field. The frame ends with 2 bytes cyclic redundancy check (CRC) field.

B. Fundamental operating principle

After the address identifier (not the whole frame) arrives at slave 1, it checks the identifier and knows it is a data frame. When it receives the address field, the slave will check if the address matches its own address. If not, it will forward the frame to the subsequent slave. The same action is performed by slave 2 and finally slave 3 receives this frame. After slave 3 extracts the data in the frame, it will replace the data part of the frame with its own measurement data and send it out. Then the local measurement data from slave 3 would be received by the master controller.

C. Synchronization mechanism

For PES Net, the master controller will send a data frame to each slave in every switching period. After the slaves receive the messages, it will not become effective instantly. Instead the slave will wait for a synchronization signal. Upon receiving the synchronization signal, the data received by all the slaves will be updated at the same time. The synchronization is achieved by sending a synchronization frame as shown in Fig. 5 . According to the frame structure, the address of the last slave will be sent out first. Then, filler bytes are sent out followed the address of the second last slave. If the number of filler bytes is carefully chosen such that the transmission time of the

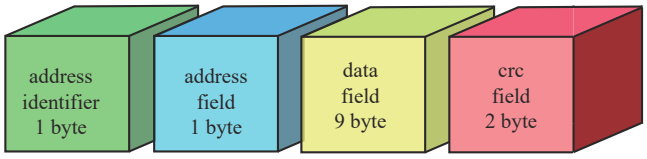

Fig. 4. Data frame defined by PES Net

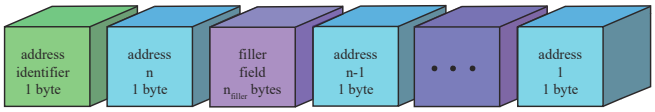

Fig. 5. Synchronization frame defined by PES Net

filler bytes is equal to the delay between the last and the second last slave, the two slaves will receive their addresses at the same time. This signals the module to update the new data at the same time. However, this technique has one drawback. The delay must be multiple of the transmission time for a byte. With 125 Mb/s bit rate, the transmission time for one byte (4B/5B) is $T_{\text{byte}} = 80$ ns. The worst case synchronization accuracy is limited to half of that, 40 ns

D. Cycle time calculation

For PES Net, the switching period must satisfy:

$$T_{\text{sw}} \geq T_{\text{DSP}} + T_{\text{data}} + T_{\text{sync}} + T_{\text{tran}} \tag{1}$$

where T_{DSP} is the time that the master controller requires to finish calculating the control algorithm; T_{data} is the time to send the data frame to all the slaves; T_{sync} is the time to send the synchronization frame. T_{tran} is the transmission delay from the master controller to the last slave.

Since each slave must receive its data frame from the master controller, the total transmission time can be calculated as,

$$T_{\text{data}} = nt_{\text{df}} \tag{2}$$

where n is the number of slave nodes. t_{df} is the time to transmit one data frame. If there are 13 bytes per frame then,

$$nt_{\text{df}} = 13T_{\text{byte}} \tag{3}$$

where $T_{\text{byte}} = 80$ns.

For the synchronization frame there are $n+1$ bytes for addresses and $n-1$ fillers. Therefore,

$$T_{\text{sync}} = (n+1)T_{\text{byte}} + (n-1)T_{\text{filler}} \tag{4}$$

where T_{filler} is the time to transmit the one filler between two slave addresses. If there are n_{filler} bytes in one filler,

$$T_{\text{sync}} = (n+1)T_{\text{byte}} + (n-1)n_{\text{filler}}T_{\text{byte}} \tag{5}$$

For the transmission delay, it consists of three components, the propagation time through the optical fibers, the delay introduced by transceiver and the delay of FPGA logic. The first component is very small compared with the other two. Therefore, it is ignored. Transmission delay may vary in a wide range depending on the transceiver chip used and FPGA logic implemented. In [8], the delay between two slaves $T_{\text{s-s}}$ is measured to be 468 ns. With n slaves, the delay from the master controller to the last slave is,

$$T_{\text{tran}} = nT_{\text{s-s}} \tag{6}$$

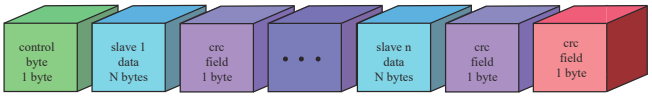

Fig. 6. Data frame defined by SyCCo bus

Combined above equations,

$$T_{\text{sw}} \geq T_{\text{DSP}} + (14n + 1 + (n-1)n_{\text{filler}})T_{\text{byte}} + nT_{\text{s-s}} \tag{7}$$

For an MMC with 30 modules in total, assume n_{filler} and $T_{\text{DSP}} = 10$ μs,

$$T_{\text{sw}} \geq 71.64 \ \mu\text{s} \tag{8}$$

Therefore, the maximum switching frequency,

$$f_{\text{sw,max}} \leq 13.9 \text{ kHz} \tag{9}$$

In practice, the switching frequency should be much lower than $f_{\text{sw,max}}$ considering a safety margin and other unexpected delays.

IV. SYCCO BUS

The SyCCo field bus protocol is based on the Ethercat communication protocol. At the lowest level, SyCCo bus uses the physical layer defined by Ethernet. In this way it takes advantage of the high speed of Ethernet and does not require a custom lower level implementation. As define by Ethernet, the bit rate ranges from 10 Mb/s to 10 Gb/s. In this paper 100 Mb/s is used.

A. Fundamental operating principle

Similar to PES Net, SyCCo bus utilizes a ring structure to connect the master controller and the slaves. Unlike PES Net, SyCCo bus uses a concept called summation frame. Instead of sending frames to each slave individually, the master controller only sends out one frame which contains the messages to all the slaves. When the frame arrives at a slave, the slave extracts the data that are addressed to it, replaces that part of frame with its own data and then forwards the frame to the next slave. A data frame for SyCCo bus is shown in Fig. 6. The first byte is called control byte. Its function is to indicate the frame to come containing data. Then a fixed number of bytes are assigned to each slave followed by a CRC byte. Since this protocol is highly customized, there is no need to provide the slave address in the frame. The address of the slave is implied by the position of the bytes. At the end of the frame, there is one byte CRC for the entire frame.

In the original design [9], SyCCo bus has a different topology from that in Fig. 2. Instead of connecting the last slave to the master, the last slave is connected to the second last slave. The frame has to go through all the slaves one more time before it is back to the master. As we will show in section VI, this will greatly decrease the communication speed because of the delay introduced by the physical layer chips. Thus, the following analysis is done for SyCCo bus using topology shown in Fig. 2.

B. Synchronization mechanism

SyCCo bus uses time-stamped method for synchronization. A 100 MHz clock is used for this purpose. During the configuration, the master sends out a synchronization frame and notes the sending time $t_{\text{tx},0}$. Then, a slave in the ring, for example slave i, would note the time $t_{\text{rx},i}$ when the frame first arrives and the time $t_{\text{tx},i}$ when the frame is forwarded. The delay caused by the slave i can be calculated by,

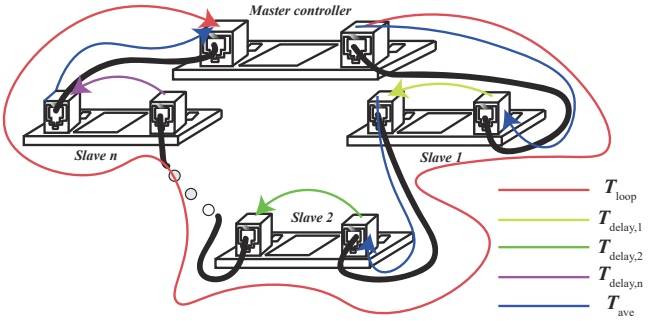

Fig. 7. SyCCo bus time measurement for synchronization

$$T_{\text{delay},i} = t_{\text{tx},i} - t_{\text{rx},i} \tag{10}$$

When this frame goes through the ring and back to the master, the master notes the time $t_{\text{rx},0}$. The loop time of the frame is calculated by,

$$T_{\text{loop}} = t_{\text{rx},0} - t_{\text{tx},0} \tag{11}$$

With $T_{\text{delay},i}$ and T_{loop}, the average delay caused by signal propagation between two slaves can be calculated,

$$T_{\text{ave}} = \frac{T_{\text{loop}} - \Sigma T_{\text{delay},i}}{n+1} \tag{12}$$

where n is the number of slaves in the ring. After T_{ave} is calculated in the master, it is sent to all the slaves by another frame. With this information, each slave i can calculate the time between slave i receiving the frame and the last slave receiving the frame,

$$T_{i-n} = (n-i)T_{\text{ave}} + \sum_{j=i}^{n-1} T_{\text{delay},i} + T_{\text{frame}} \tag{13}$$

where T_{frame} is the time needed to transmit the frame. It only depends the length of the frame and the bandwidth of the channel.

After slave i receives one frame, the data addressed to it is extracted and stored in a shadow register. The data will be updated after a delay T_{i-n}. In this way all the slaves will update the data at the same time, and synchronization is achieved. Because the clocks used to measure the time points is 100 MHz, the synchronization error is only 5 ns.

C. Cycle time calculation

Similar to (1), the minimal cycle time of SyCCo bus is calculated by,

$$T_{\text{sw}} \geq T_{\text{DSP}} + T_{\text{data}} + T_{\text{tran}} \tag{14}$$

Note that there is no delay caused by synchronization frame in SyCCo bus. T_{data} is the time needed to transmit the summation frame,

$$T_{\text{data}} = (2 + \sum_{i=1}^{n}(N_{\text{byte}}(i) + 1))T'_{\text{byte}} \tag{15}$$

where $N_{\text{byte}}(i)$ is the number of bytes of data for slave i. To compare with PES Net, $N_{\text{byte}}(i)$ is 9 bytes here. If the bit rate is 100 Mb/s, the time to transmit one byte T'_{byte} is 80 ns.

Like PES Net, the third component in (14) T_{tran} depends on the implementation and has to be measured. In the original design [9] the transmission time for one master and two slaves is measured to be 4.25 μs. The transmission delay between 2 modules T'_{s-s} is 2.125 μs. In [12], the transmission delay between two slaves has been reduced to 720ns. If we assume that the transmission delay is proportional to the number of slaves, the T_{tran} is given by:

$$T_{\text{tran}} = nT'_{\text{s-s}} \tag{16}$$

Combine (14), (15), and (16)

$$T_{\text{sw}} \geq T_{\text{DSP}} + (10n+2)T'_{\text{byte}} + nT'_{\text{s-s}} \tag{17}$$

For an MMC with 30 slaves and $T_{\text{DSP}} = 10$ μs,

$$T_{\text{sw}} \geq 55.76 \ \mu\text{s} \tag{18}$$

Therefore, the maximum switching frequency,

$$f_{\text{sw,max}} \leq 17.9 \text{ kHz} \tag{19}$$

TABLE I
CYCLE TIME COMPONENTS FOR PES NET AND SYCCO BUS

	PES Net		SyCCo bus	
T_{data}	$13nT_{\text{byte}}$	31.2 μs	$(10n+3)T'_{\text{byte}}$	24.24 μs
T_{tran}	$nT_{\text{s-s}}$	14.04 μs	$nT'_{\text{s-s}}$	21.60 μs
T_{sync}	$(7n-5)T_{\text{byte}}$	16.4 μs	-	-
T_{DSP}	-	10 μs	-	10 μs
T_{sw}	-	71.64 μs	-	55.84 μs
$T_{\text{sync,err}}$	-	40 ns	-	5 ns

The key parameters of PES Net and SyCCo bus are listed in Table I. The first term T_{data} is the absolute minimum time for information exchange. The second term T_{tran} is the transmission delay caused by signal propagation, transceiver, and FPGA logic. Among them, the delay caused by signal propagation is negligible compared with that of the transceiver and FPGA logic. A good design should yield a small FPGA logic delay. The third term T_{sync} is unique in PES Net.

V. COMPARISON BETWEEN PES NET AND SYCCO BUS

Both designed for high speed communication for power electronics systems, PES Net and SyCCo bus share a lot of similarities.

1) Both use a ring structure to minimize the number wires from the master controller to the slaves.
2) Both employ slaves that receive and transmit the data frame at the same time.
3) Both make use of FPGA to ensures the real time property of the communication.

However, their basic operating principles are different. In PES Net each slave has its own frame indicated by the address field while in SyCCo bus all the slaves share a summation frame. Because of the number of frames thus the number of overhead bytes sent by PES Net is more than SyCCo bus, T_{data} of SyCCo bus is smaller than that of PES Net. Although the transmission delay of SyCCo bus is slightly higher than that of PES Net, the synchronization frame introduces extra delay for PES Net. The achievable minimum cycle time of SyCCo bus is smaller than that of PES Net.

In PES Net the synchronization is achieved by sending a synchronization frame every switching period. Because the synchronization of the PES Net depends on the minimum filler size, the worst case synchronization error between two nodes can be as high as 40 ns. In SyCCo bus the synchronization is realized by calculating the delay from one slave to the last slave. Because SyCCo bus uses a 100 MHz clock to track the delay, its synchronization error can be much lower. However, with 20 kHz switching frequency, 40 ns synchronization error is only 0.08% of the switching period. Thus, both protocols yield good performances in terms of synchronization.

Fig. 8. Hardware test setup

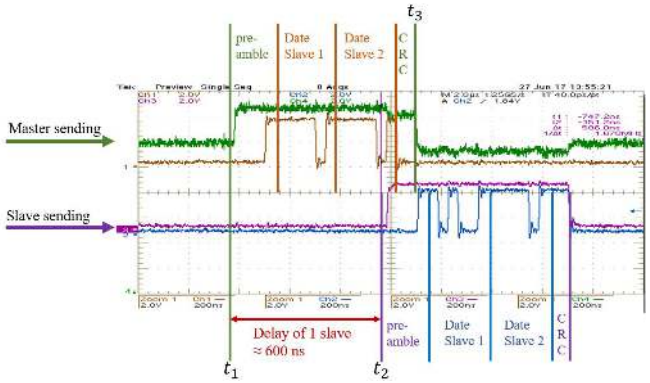

Fig. 9. Hardware test waveform

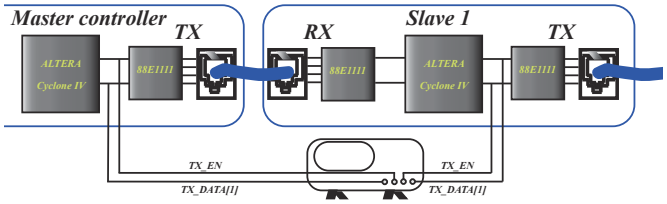

Fig. 10. Simplified test setup block diagram

VI. Hardware Implementation

From Table I, it can been seen SyCCo bus offers better performance than PES Net in terms of achievable minimum cycle time and synchronization accuracy. Therefore, it is implemented using Terasic DE2-115 development board with a Cyclone EP4CE115 FPGA. The Verilog source code can be found in Github repository as in [10].

The development board features two RJ45 ports with physical layer chip 88E1111 from Marvell. The PHY chip is selected to working in 100 Mb/s MII mode. In this mode, 4 pairs of twisted wires are used to transmit or receive data with a 25 MHz clock. Therefore, 4 bits of data(one nibble) are sent at the same time with a bit time of 40 ns.

To enable auto-configuration four types of configuration frames are implemented. They are numbering frame, distributing frame, delay-measuring frame and delay-distributing frame. During start-up, the master sends out a numbering frame to number all the slaves starting with zero. The first slave receives zero and store it as its ID. Then it will forward this frame but add one to the ID field. After it is received by the second slave, the second slave uses 1 as its ID and so on so forth. After the frame goes around the whole loop, all the slaves are numbered and the master receives the last slave's ID plus 1. This is the number of slaves in the loop. Then this number is sent out by a distributing frame. With those two frames, all the slaves know the number of slaves and what is its position in the communication chain. Delay-measuring frame and delay-distributing frame are used to measure the time delay and distribute it to all the slaves. As the synchronization method has been introduced in section IV, it is not repeated here.

In the FPGA a dual port RAM is implemented to enable the information exchange between the main controller and the transmitter(receiver). The main controller stores duty cycle for all the modules in the transmitter RAM once they are available. The transmitter state machine will fetch them automatically and construct the summation frame. The received measurement data are stored in the receiver RAM.

A. Test results

After the the protocol is implemented in FPGA, it is tested using two slaves. Fig. 8 shows the experiential set-up. One DE2-115 development board is used as the master while the other is the first slave. The second slave is implemented in Intel MAX 10 FPGA development board. This development board also has two RJ45 ports with physical layer chip 88E1111 from Marvell. To have a better view of the protocol operation, the length of data for each slave is set to 2. Also, the length of control byte is set to 2. The test result is shown in Fig.9.

As PHY chip is used as the lowest level transmitter, we do not have access to the actual signal transmitting on the cable. The lowest level signal we can get is the signal from the FPGA to PHY chip. Fig. 10 shows the signals are coming from FPGA to PHY chip.

The green waveform is the transmitter enable signal of the master. The brown waveform is the second bit of the transmitting nibble of the master. At time t_1 the master starts to transmit the data frame. As the control byte for data frame is hexadecimal 5557, the second bit is 0001 for those four nibbles. The following 2 bytes(4 nibbles) are the data for slave 1. Then, 1 byte(2 nibbles) CRC is added for it. Then same data and CRC field are repeated for slave 2. At the end of the frame, 1 byte(2 nibbles) CRC is added for the whole frame.

The purple waveform is the transmitter enable signal of slave 1. The blue one is the second bit of the transmitting nibble of slave 1. At time t_2 slave 1 starts to transmit the data frame. The first two bytes control byte are the same as it coming from the master. The data field for slave 1 has been changed. Because the slave extracts the original data and replaces it with its measurement. The CRC field for that data is also changed. The following 3 bytes remain the same as they are for the second slave. Slave 1 simply forwards the message without any changes. The last byte is different since any changes of the frame will cause the change of the frame CRC.

B. Discussion

One important feature of the protocol is that slaves receive and transmit data at the same time. This is can be seen from t_2 to t_3 in Fig. 9. Slave 1 is still receiving the data frame from the master while sending the preamble of the frame to the next slave. This greatly

improves the speed of the protocol. However, the bottle neck of the communication speed is the delay caused by slaves.

From time t_1 to t_2 is the delay cased by one slave. It is around 600 ns. As shown in Fig.10 this delay is caused by PHY transmitter delay, propagation delay, PHY receiver delay and FPGA logic delay. This delay is critical to the communication speed as adding every new slave will introduce an extra slave delay to the communication chain. The propagation delay is negligible compared with other three components. The PHY transmitter delay and PHY receiver delay depend on the PHY chip used. The last component FPGA logic delay is between 240 ns and 280 ns with current implementation. This is 6 to 7 nibbles time. The first five-nibble time is for receiving control byte as the slave will not do anything until it confirms the type of the frame. Another 1 to 2 nibble time is used to synchronize the data from receiver clock to the transmitter clock. This is necessary because most PHY chips working in in MII mode use different clocks for transmitting and receiving. Depending on the phase difference between the two clocks, the time is between 1 and 2 nibble time. The theoretical minimum delay for the FPGA logic is 3 nibble times, namely 120 ns. One nibble time delay to receive nibble and two nibble time to synchronize the nibble.

Subtracted the FPGA logic delay from the total slave delay, we can get the delay caused by the PHY chip transmitter and receiver to be between 320 and 360 ns. This can be confirmed by the Marvell 88E1111 datasheet which gives a maximum transmitter delay of 72 ns and maximum receiver delay 248 ns.

With the 360 ns delay from the PHY chip and 120 ns delay from the FPGA logic, the minimum delay for one slave can be reduced to 480 ns. If we re-calculate the minimum cycle time using (17),

$$T_{\mathrm{sw}} \geq 48.56\ \mu\mathrm{s} \tag{20}$$

Therefore, the maximum switching frequency,

$$f_{\mathrm{sw,max}} \leq 20.6\ \mathrm{kHz} \tag{21}$$

If higher switching frequency, and therefore higher communication speed is required, one option is to use GMII/RGMII mode with 1Gb/s bit rate. This can effectively reduce T_{data}. One challenge is that GMII/RGMII uses a 125 MHz clock to update the data. The transmitted nibble must be strictly aligned with the positive edge of the clock. However, as all the FPGA logic imposes some delay from register to register, aligning the correct nibble at the positive edge becomes cumbersome when the clock frequency is high. The logic needs to be designed very carefully if GMII/RGMII mode with 1 Gb/s is used.

Another issue is that the delay caused by the transmitter and receiver in GMII/RGMII will not be reduced compared with MII. The maximum transmitter delay and the maximum receiver delay are 84 ns and 208 ns, respectively, for GMII mode. This limits the highest achievable communication speed.

VII. Conclusion

In this paper two ring topology communication protocols, PES Net and SyCCo bus, are reviewed. Their operating principle is introduced. Their performance is evaluated and compared in terms of minimal cycle time and synchronization accuracy. The implementation of SyCCo bus in Altera FPGA Cyclone IV is introduced in detail. The source files are made open-source in [10]. The delays in the implementation is discussed. The factors limiting the highest communication speed in SyCCo bus are identified.

References

[1] S. Debnath, J. Qin, B. Bahrani, M. Saeedifard, and P. Barbosa, "Operation, control, and applications of the modular multilevel converter: A review," *IEEE transactions on power electronics*, vol. 30, no. 1, pp. 37–53, 2015.

[2] A. Lesnicar and R. Marquardt, "An innovative modular multilevel converter topology suitable for a wide power range," in *Power Tech Conference Proceedings, 2003 IEEE Bologna*, vol. 3. IEEE, 2003, pp. 6–pp.

[3] A. Antonopoulos, L. Angquist, and H.-P. Nee, "On dynamics and voltage control of the modular multilevel converter," in *Power Electronics and Applications, 2009. EPE'09. 13th European Conference on*. IEEE, 2009, pp. 1–10.

[4] Q. Tu, Z. Xu, and L. Xu, "Reduced switching-frequency modulation and circulating current suppression for modular multilevel converters," *IEEE transactions on power delivery*, vol. 26, no. 3, pp. 2009–2017, 2011.

[5] Y.-M. Park, J.-Y. Yoo, and S.-B. Lee, "Practical implementation of pwm synchronization and phase-shift method for cascaded h-bridge multilevel inverters based on a standard serial communication protocol," *IEEE Transactions on Industry Applications*, vol. 44, no. 2, pp. 634–643, 2008.

[6] C. Toh and L. Norum, "A high speed control network synchronization jitter evaluation for embedded monitoring and control in modular multilevel converter," in *PowerTech (POWERTECH), 2013 IEEE Grenoble*. IEEE, 2013, pp. 1–6.

[7] I. Milosavljevic, Z. Ye, D. Boroyevich, and C. Holton, "Analysis of converter operation with phase-leg control in daisy-chained or ring-type structure," in *Power Electronics Specialists Conference, 1999. PESC 99. 30th Annual IEEE*, vol. 2. IEEE, 1999, pp. 1216–1221.

[8] T. Ericsen, "Power electronic building blocks-a systematic approach to power electronics," in *Power Engineering Society Summer Meeting, 2000. IEEE*, vol. 2. IEEE, 2000, pp. 1216–1218.

[9] C. Carstensen, R. Christen, H. Vollenweider, R. Stark, and J. Biela, "A converter control field bus protocol for power electronic systems with a synchronization accuracy of±5ns," in *Power Electronics and Applications (EPE'15 ECCE-Europe), 2015 17th European Conference on*. IEEE, 2015, pp. 1–10.

[10] H. Tu. (2017, June) Verilog source code for mmc communication. [Online]. Available: https://github.com/htuNCSU/MmcCommunicationVerilog

[11] C. Toh and L. Norum, "Implementation of high speed control network with fail-safe control and communication cable redundancy in modular multilevel converter," in *Power Electronics and Applications (EPE), 2013 15th European Conference on*. IEEE, 2013, pp. 1–10.

[12] A. Hillers, H. Tu, and J. Biela, "Central control and distributed protection of the dsbc and dscc modular multilevel converters," in *Energy Conversion Congress and Exposition (ECCE), 2016 IEEE*. IEEE, 2016, pp. 1–7.