ATLANTIS PRESS International Journal of Networked and Distributed Computing, Vol. 5, No. 1 (January 2017) 12–21

Wireless Extension Mechanism and Logic Design for FPGA-based Ethernet Powerlink Node

Kailong Zhang, Panfei Zuo, Liang Hu, Xiao Wu, Kejian Miao

School of Computer Science and Engineering, Northwestern Polytechnical University, Youyi West Road, Xi'an, Shaanxi,
Xi'an, 710072, China

E-mail: kl.zhang@nwpu.edu.cn

Abstract

Real-time networks, such as industrial network, field bus and so on, have been becoming one vital component to develop large-scale and cooperative embedded systems. As one important branch, the wireless mode of realtime networks also raises more and more attentions in recent years since its conveniences to construct a flexible control system. Ethernet Powerlink is such a typical real-time industrial network protocol, and provides a master-slave, time-slot mechanism that can well avoid radio collisions. With the designed FPGA-based hardware node, in this paper, a new method to extend wireless capability of Powerlink is explored and described. Concretely, Powerlink architecture and especially its original mechanisms are analyzed at first. For effectively connecting OpenMAC module of Powerlink and RF module, an interface logic at MAC layer is introduced and is typically designed in a multiplexing mode with a dual-FIFO logic according to the limited resource on FPGA, some key designs and mechanisms of which are detailed later. Finally, all designed mechanisms and logics are implemented as an extension part of IP core of Powerlink in VHDL language, and the communication functions and performance of such extended protocol are verified.

Keywords: Industrial Network, Powerlink, Real-time, MAC, Wireless Extension, Multiplexing, IP Core

1. Introduction

Nowadays, networking and intelligence have been becoming two remarkable emergent characteristics for novel industrial embedded systems with innovative information technologies. Thereinto, real-time network is one key infrastructure for its capability to flexibly connect electronic devices inside equipments and construct flexible product lines. With these obvious advantages, such networks and corresponding application technologies have increasingly raised engineering and research interesting.

In recent decades, several different industrial networks have been developed, in which ethernet-based real-time networks formed one main branch. For instance, EtherCAT, Profinet IRT and Ethernet Powerlink(EPL) are three typical ones[123]. Surrounding this topic, many studies to improve the performance of these networks are also carried out. Y-oon et al. [4] studied a redundancy ethernet based on ring topology and designed a new topology adaptation network management protocol, which can provide a recoverable control network. After analyzing the schemes and performances of EtherCAT, Rostan et al. [5] studied an EtherCAT enabled control architecture with extraordinary real-time performance and flexible topology. On the basis of Dynamic Frame Packing (DFP) algorithm in Profinet IRT, Schlesinger et al. [6] proposed an automatic packing mechanism with subframes, the positions of which are recognizable for each device. This design makes a scheduling dispensable and achieves better performance than DFP, especially in star and tree topology. Schlesinger et al. [7] compared the performance of three real-time ethernets, and thought that Profinet IRT gains the advantage with asymmetric throughput data payloads, while EtherCAT and VABS is better for very small data payloads. Limal et al. [8] employed a model-checking approach, concretely timed finite state automata, to validate the medium redundancy management part of the ethernet Powerlink high availability extension in the context of special power critical applications. From current work, we can find that these studies are mainly focused on the optimization of protocols and applications.

On the other hand, some researchers also attempt to migrate real-time networks to wireless mode because its convenience and flexibility to construct local control networks. Typically, Kjellsson et al. [10] discussed how the WISA(Wireless Interface for Sensors and Actuators) concept can be efficiently integrated into wired field networks, and proposed amendments to WISA to improve the 802.11b/g coexistence and harmonize the integration of WISA. Seno et al. [11] analyzed the possibility to realize the wireless capability of ethernet Powerlink protocol based on 802.11. These work show that wireless mode is feasible for some real-time networks, because the capability to avoid radio collisions with time-division multiple access (TDMA) or other no-competitive communication mechanisms.

The employment of a time-slot-based mechanism makes it's feasible to transplant Powerlink protocol from wired mode into wireless mode[9]. After analyzing the architecture and characteristics of Powerlink, in this paper we design an new interface logic at MAC layer to extend the wireless interface of a Powerlink node, without considering any secure problem described in [12]. Primarily, FIFO-based multiplexing mechanisms at the physical and data link layers are explained in detail. And finally, some key implementation methods are also presented.

2. Architecture of Wireless Ethernet Powerlink(WEPL)

Time-slot communication mechanism of Powerlink is the key foundation to realize a wireless mode. In this section, related features of this protocol and the self-designed FPGA-based hardware are explained firstly.

2.1. Ethernet Powerlink Architecture

Ethernet Powerlink (EPL), regarded as a combination of Ethernet and CANopen, was originally defined by the specification subsequently included in the IEC 61784 International Standard[9], and its architecture, shown as Figure.1, is consistent with the OSI standard. EPL is completely compatible with legacy Ethernet since it is based on the definition of a data link layer protocol placed on top of the Ethernet Medium Access Control(MAC) layer. This means EPL frames are encapsulated and transmitted by means of Ethernet protocol data units. Among different physical layers encompassed by the original Ethernet specification, EPL standard refers explicitly to 100BASE-X with half-duplex transmission. And at higher layers of the communication stack, it includes an application layer protocol based on CANopen profiles. Especially, the Procedure Data Object (PDO) and Service Data Object (SDO) of object dictionary are kept to make EPL open and flexible[11].

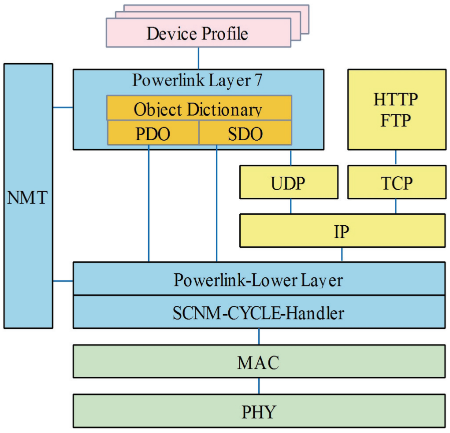

Fig. 1. Architecture of EPL.

In EPL networks, all EPL stations are connected via either hubs or switchers, and one managing node (MN) is responsible for polling a set of controlled nodes (CNs). Actually, the operation of EPL is based on a cycle of predefined fixed duration, continuously repeated on the network and handled by the MN. As shown in Figure.2, MN broadcasts a Start-of-Cycle frame (SoC) at the beginning of each cycle to inform all CNs that a new cycle is started. After this initialization, an isochronous period is launched in which MN polls all the CNs according to a round-robin technique. Concretely, a Poll-Request frame (PReq) is issued by MN to each CN which carries command data, and when received a PReq the addressed CN responds a Poll-Response frame (PRes) with data. Typically, the communication procedure is always organized as the "Cycle i-Classic" shown in Figure.2. During this period, if MN does not receive a correct PRes from a CN within a predefined interval (EPL time-out), it marks that query as failed and moves on the next CN, and if one CN does not respond for a predefined number of consecutive cycles, it is removed from the isochronous cycle. At the end of each isochronous period, MN broadcasts SoA to inform CNs that an asynchronous period has started, in which a station that made a request during one of the previous isochronous periods may be granted to transmit an asynchronous message. Additionally, in the optimized EPL procedure, Poll-Request-Chaining (PRC) technology is adopted based on the synchronized distributed timer on each EPL node, as "Cycle i-PRC" shown in Figure.2. Thus, most repeated hand-shaking between PReq and PRes will be eliminated, and the transmission efficiency of the whole EPL network increases at about 40%.

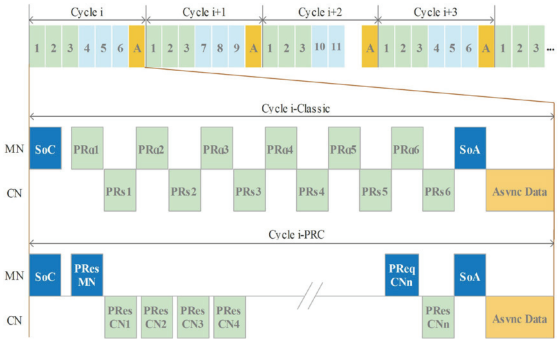

Fig. 2. TDMA Mechanism at EPL Data Link Layer.

2.2. *FPGA-based WEPL Hardware Structure*

In the industry, Powerlink protocol are typically implemented with standard C or VHDL, being able to run on heterogeneous platforms. In our previous work, we have designed a FPGA-based hardware, as shown in Figure.3, in which a CC2530 RF module is adopted for the possible wireless extension.

According to the requirements of performance, when we designed the hardware one Altera Cyclone IV FPGA is choosed as the central protocol processor, and a 50MHz crystal oscillator and other peripheral circuits are also attached. Meanwhile, one CC2530, which owns built-in MCU and a RF transceiver at 2.4GHz, is employed to serve as RF module, with a 32MHz crystal oscillator. According to our analysis, these two module can satisfy the expected requirement well. While, oscillator frequencies and data formats between FPGA and the RF module are apparently different, it requires the IP core logic of EPL must be extended. This is obviously the key problems we should consider.

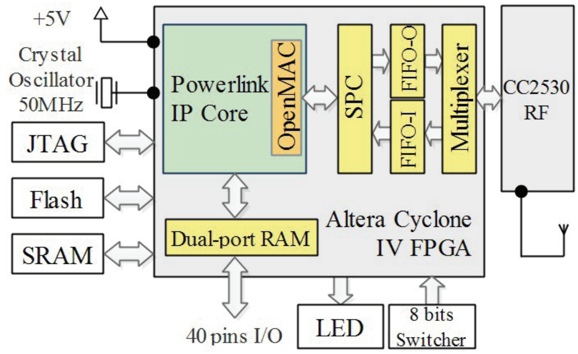

Fig. 3. FPGA-based Wireless EPL Hardware Logic.

3. Multiplexed Interface Logic Design

By the EPL protocol stack, OpenMAC is a particular unit that connects to Packet Buffer via DMA channel, and meanwhile with the physical interface via OpenFILTER. And in fact, it separates the protocol layer from interface layer. So, only logics and interfaces between OpenMAC and physical layer need to be changed. In our design, the extended logics mainly include one S2P/P2S unit, two data FIFOs (FIFO-O and FIFO-I), and one I/O Multiplexer unit.

Additionally, a dual-port RAM logic and 40 pins are set up on FPGA for the parallel communication with master system.

3.1. *Dual-FIFO Mechanism*

To eliminate the differences of data formats and communication frequencies, data buffers are introduced between OpenMAC and RF unit. Concretely, FIFO-I and FIFO-O are employed to buffer data for or from RF unit. These two buffers are implemented with existing DCFIFO model in QuartusII because this mode can support the synchronized adjustments of different clocks via its inside *wrclk* and *rdclk* signals.

In concrete implementation, each FIFO is set as the Legacy synchronous FIFO mode, which can satisfy the possible transmission delay in WEPL hardware. And two 8-bit data buses, named q[7:0] and data[7:0], are adopted as the input and output bus respectively. Meanwhile, *wrreq*, *wrfull*, and *wrempty* are defined as the writing request, FIFO full, and FIFO empty signals, separately; And, emphrdreq, *rdfull*, and *rdempty* are indicated for reading operation as the reading request, FIFO full, and FIFO empty signals, respectively. With these signals, the other designed logics can operate FIFOs in effective operation series.

3.2. *Serial-Parallel-Coverter Logic*

Serial-Parallel Converter (SPC) logic is further designed as a internal interface between OpenMAC and FIFOs. Its main function is to convert 2-bit data stream from OpenMAC to 8-bit data stream, write converted data into FIFO-O, and notice Multiplexer to read data from FIFO-I. When received a read signal from Multiplexer, it reads a 8-bit data frame, converts to a 2-bit data stream, and transfers to OpenMAC. The detailed logic of SPC is presented in Figure.4, where the Tx_Block and Rx_Block are designed to carry converting functions, and the details of I/O definitions are defined in Table 1.

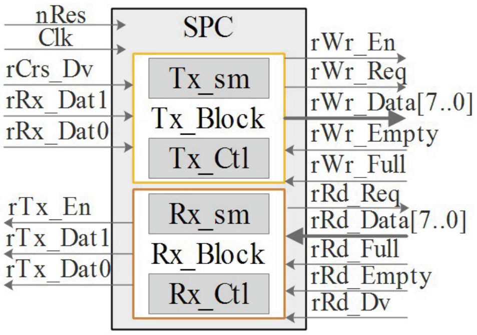

Fig. 4. SPC Logic and Signal Definition.

Table 1. I/O Definition of SPC.

Symbol	Description
Clk	Clock signal.
nRes	Transmission reset signal.
rCrs_Dv	Carrier monitoring signal.
rRx_Dat1	High bit received from OpenMAC.
rRx_Dat0	Low bit received from OpenMAC.
rTx_En	Carrier signal to OpenMAC.
rTx_Dat1	High bit sent to OpenMAC.
rTx_Dat0	Low bit sent to OpenMAC.
rWr_En	Write-end signal to Multiplexer.
rWr_Req	Req-signal for writing FIFO-O.
rWr_Data[7..0]	8-bit output data interface.
rWr_Empty	FIFO-O empty signal.
rWr_Full	FIFO-O full signal.
rRd_Req	Req-signal for reading FIFO-I.
rRd_Data[7..0]	8-bit input data interface.
rRd_Full	FIFO-I empty signal.
rRd_Empty	FIFO-I full signal.
rRd_Dv	Read enable signal to SPC.

3.2.1. *Tx_Block Processes*

Tx_Block of SPC is composed of two hardware process modules: Tx_Sm process and Tx_Ctl process. The former process receives control signals in real-time, and shifts the status of state automata according to received signal. The later process executes corresponding functions under a special state. Concretely, control signals, states and their relationships of Tx_Block are presented in Figure.5, where R_Idl, R_Crs, R_Sof, R_Rxd, and R_Stat are the idle state, data monitor state, start state, receiving state and terminate state, respectively.

It should be cleared that the signal Wr_En sent to I/O Multiplexer in R_Stat must be kept in high level for a long enough time because the clock frequency of I/O Multiplexer is different from that of SPC. To realized accurate control, we design a 128-period time counter, Stat_Count[7..0], and only when its highest bit Stat_Count[7] switches to 1 the automata shifts to R_Idl. With this counter, a qualified Wr_En signal in 2.56μs can be generated. On the other hand, rCrs_Dv signal plays a role of data beginning synchronization, and via monitoring its upedge R_Crs can learn the correct beginning time of the data frame from OpenMAC. In fact, this is an empirical design to eliminate the lost of partial data that we have observed during experiments. To transform four 2-bit data frames to a 8-bit data frame, a shift register logic Rx_Sr is also designed in Tx_Ctl process. At each clock period, two bits Rx_Dat[1] and Rx_Dat[0] will be serially shifted into this register, and after every 4 clock periods a 8-bit data is formed.

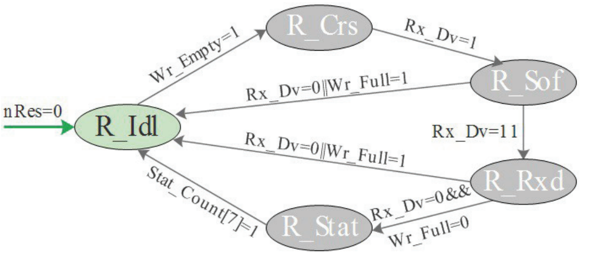

Fig. 5. State Automata in Tx_Block.

3.2.2. *Rx_Block Processes*

Similar to Tx_Block, Rx_Block also contains two blocks, the state automata logic Rx_Sm and the read-related control logic Rx_Ctl. As shown in Figure.6, R_Idl, R_Bop, R_Pre and R_Txd indicate the idle state, initial state, pre-sending state and sending state, separately. In R_Bop state, all required registers and timers will be initialized, and then Rx_Block shifts into R_Pre state. And in R_Pre state, Rx_Ctl process sends the beginning ethernet frame, meanwhile keeps detecting the timer signal Tx_Time. When Tx_Time signal changes to 1, the state automata of Rx_Block switches to R_Txd. It needs to be cleared that in Rx_Ctl a shift register is also adopted to transform received data to 8-bit format. After a reading operation, the state automata shifts to Rd_Idl state again.

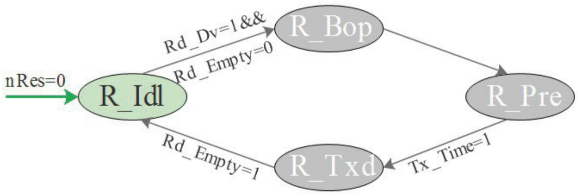

Fig. 6. State Automata of Rx_Block.

3.3. *I/O Multiplexer Logic*

To realize duplex communication with limited I/O resources between FPGA and RF module, a I/O Multiplexer logic is designed, which connects to RF module, FIFOs and SPC logic, as presented in Figure.7 and Table 2. The main role of this module is, fir detecting data direction between FIFOs and RF module via detecting control signals, and further dynamically scheduling I/O channels to serve each communication.

Specifically, F_Rd_Dv and C_Wr_Start are two defined important signals. When applying the use of multiplexed data bus C_Data, OpenMAC triggers F_Rd_Dv signal, and RF module will trigger C_Wr_Start signal when applying C_Data. Obviously, there must exist a resource competition problem between such two mutually-exclusive operations. In our work, it is resolved via introducing the state automata Mult_Sm, Rx_Ctl and Wr_Ctl processes with decision-making logics detailed as below.

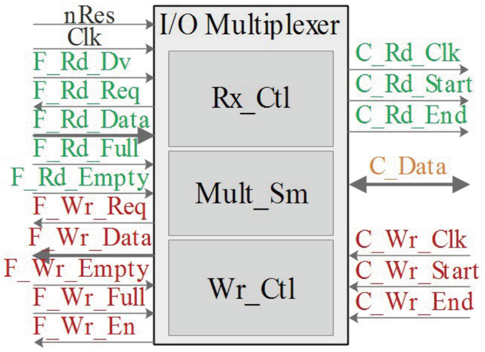

Fig. 7. I/O Multiplexer Logic and Signal Definition.

Table 2. I/O Definition of Multiplexer.

Symbol	Description
F_Rd_Dv	Control sig from SPC.
F_Rd_Req	Read-req sig to FIFO-O.
F_Rd_Data	Read sig from FIFO-O.
F_Rd_Full	FIFO full sig of FIFO-O.
F_Rd_Empty	Idle sig of FIFO-O.
F_Wr_Req	Write data signal.
F_Wr_Data	Write sig to FIFO-I.
F_Wr_Empty	Idle sig of FIFO-I.
F_Wr_Full	FIFO full sig of FIFO-I.
F_Wr_En	Control signal to SPC.
C_Rd_Clk	Sync sig to RF module.
C_Rd_Start	Transmission-start sig to RF.
C_Rd_End	Transmission-end sig to RF.
C_Data	8-bit data bus.
C_Wr_Clk	Sync sig from RF.
C_Wr_Start	Transmission-start sig from RF.
C_Wr_End	Transmission-end sig from RF.

3.3.1. *Mult_Sm Process*

State automata Mult_Sm implements the main multiplexing logic, covering five statuses, two independent and mutually-exclusive state rings, as shown in Figure.8. In the left ring "R_Idl→R_Rd→R_RdE→R_Idl", Rd_Ctl process transfers data from F_Rd_Data to RF module through C_Data, and in the right ring "R_Idl→R_Wr→R_WrE→R_Idl" Rd_Ctl process transfers data from C_Data to FIFO-I via F_Wr_Data. In the R_Idl state, Multiplexer logic monitors aforementioned direction signals. When there's new data from RF module, Mult_Sm shifts to R_Wr state, and when Wr_Start=0 and F_Rd_Dv=1 are all satisfied it switches to R_Rd state. Impliedly, data from RF module will be preferentially transferred in our designed Mult_Sm logic. In R_RdE state, Multiplexer sends termination signal F_Rd_End to RF module, and then Mult_Sm turns into R_Idl state to wait next operation. To be clear, "req", "sig", and "sync" are abbreviations of "requirement", "signal", and "synchronous", respectively.

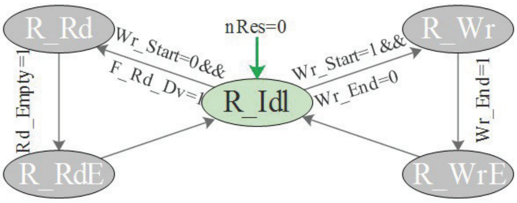

Fig. 8. Mult_Sm Logic.

3.3.2. *Rd_Ctl and Wr_Ctl Processes*

Rd_Ctl process is for processing signals and transmitting data to RF module in R_Rd and R_RdE states. When the state of Mult_Sm shifts to R_Rd, Rd_Ctl will generate a F_Rd_Req signal and monitor the status of FIFO-O: Empty or Full, continuously. At the same time, it sends control signals C_Rd_Clk and C_Rd_Start to RF module, guaranteeing a correct data reception. When in R_RdE, Rd_Ctl process will trigger the termination signal C_Rd_End, and switch Mult_Sm into R_Idl state.

Corresponding to Rd_Ctl process, Wr_Ctl process is in charge of things related to receive data from RF module in states R_Wr and R_WrE. When in state R_Wr, Rd_Ctl continuously receives signals C_Wr_Clk, C_Wr_Start, and C_Wr_End from RF module. Such mechanism is valuable to ensure all data from RF module will be received synchronically and correctly. Then, Wr_Ctl process will write received data into FIFO-I via triggering F_Wr_Req signal if FIFO-I is in correct status, such as not full. When Mult_Sm transfers to R_WrE state, Wr_Ctl notifies SPC logic to get data from FIFO-I via a termination signal F_Wr_En, and switches Mult_Sm into R_Idl state.

3.3.3. *Sampling Synchronized Signal*

As aforementioned, synchronization is an important factor for correct data transferring. For WEPL, it means the synchronization signal Wr_clk triggered by RF module must be sampled exactly. At beginning, we set the time of Wr_clk in high level be longer than one sampling period, and not exceeding two periods. However, the abnormal phenomena are observed that when a synchronization period is bigger than a sampling period, redundant synchronization signals are always sampled. This leads to false sampled data bits. And we also find another phenomenon that when a synchronization signal is sampled, a up-level signal with the duration equal to one sampling period will be generated. Thereout, we set two signal monitoring registers L1 and L2 to store two sampled synchronization signals in two continuous clock periods. And in Multiplexer logic, only when the signal in L1 is low level and signal in L2 is high, it outputs Wr_Req to FIFO-I, as the example shown in Figure.9.

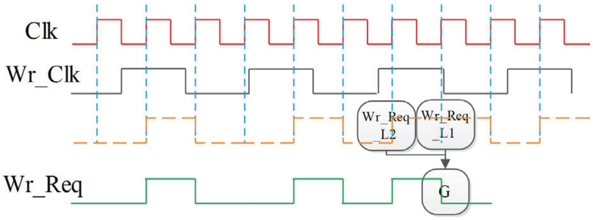

Fig. 9. Synchronized Sampling Singal.

3.4. *Driving RF Chip*

In the RF module, we design a group of functions, covering hal_McuInit(), hal_port_init(), and hal_RFInit(), to set I/O ports of CC2530 as a communication interface and initialize the corresponding functions. Concretely, P0 is in general I/O mode, P1 is in GPIO mode, and the lower three bits of P1 are set as output. Further, P1_0, P1_1, P1_2, P1_3, P1_4, P1_5, and the port P0 are defined as the Wr_Clk, Wr_Start, Wr_End, Rd_Clk, Rd_Start, Rd_End, and C_Data respectively. Based on these signals, rfRecvData(), hal_port_send(), hal_port_receive() and rfSendData() functions are designed to receive/send wireless data. Because the coupling relationship of CC2530 and Multiplexer, the logics of these functions are similar to that of Multiplexer.

Further, CC2530 allows the carrier frequency range from 2394MHz to 2507MHz and 1MHz step-width, and provide 16 channels conforming to IEEE802.15.4-2006. So, we expand the communication interface of RF module to be frequency-reconfigurable with programmable capabilities of WEPL nodes according to their requirements via hal_RFInit().

4. Node Integration and Verification

4.1. *Integration of WEPL Node*

All extended logics designed above are developed with VHDL language, via Quatus and ModelSim, and software in RF module is developed in IAR Embedded Workbench. Then, we solidified the implemented IP Core on WEPL node hardware, and the final WEPL node is shown in Figure.10. Based on such WEPL node, all fundamental functions of the designed IP Core are debugged and verified firstly via a logic analyzer Salea16, including the analysis and verification of the waveforms, frequencies, and time series of all signals.

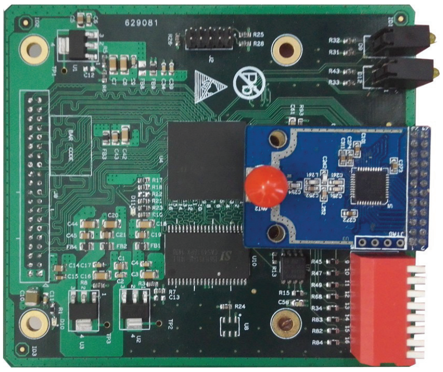

Fig. 10. Integrated WEPL Node.

4.2. *Experiments and Verification*

Further, we construct a prototype wireless network with three nodes, in which one node serves as the master and the other are slaves. On this basis, several experiments are carried out to verify the communication performance.

Figure.11 shows the waveform of a shortest data frame captured from openMAC, at 50MHz. It's obvious that the transmission-enable signal Tx En of openMAC is kept in high level for $6.08\mu s$, and during this period 8-byte Ethernet head, 64-byte data and 4-byte CRC code are transmitted on 2-bit data bus within $T_h(640ns)$, $T_d(5.12\mu s)$ and $T_c(320ns)$. It's easy calculated that one byte can be transmitted in $T_1(80ns)$, and the bandwidth of openMAC is about 12.5MB/s.

Figure.12 presents the data and time series when Multiplexer transmits 8-bit data from FIFO-O to CC2530. As shown in this figure, C_Rd_Start signal is triggered firstly to notify CC2530 to prepare a reception, while C_RD_End is the last one to be triggered, which indicates the transmission procedure is completed. C_Rd_Clk is the signal to active CC2530 to sample data on C_Data. The width of data signal is bigger than that of C_Rd_Clk signal ensure the signal correctness. It's also observed that waveforms of both rRD_Data and C_Data are totally same, which means the data is correctly transferred via Multiplexer. Trough experiments, we can measure the duration time of C_Rd_Start is 138*us*, and the effective transmission time is 136*us*. Thus, we know the transmission rate of Multiplexer is approximated 500KB/s.

Figure.13 presents one typical time series of waveforms when one 8-bit data is transferred from CC2530 to Multiplexer. It's obvious that setting the duration time of each bit being bigger than that of C_Wr_Clk, and the width of C_Wr_Clk in up-level being no less than the sampling period (1*us*) of Multiplexer, will guarantee the correct sampling of synchronous data well. which is Considering the delay led by software code, the widths of C_Wr_Clk and each data bit are set to be 1.06*us* and 4.26-4.22*us*, respectively. Thereout, we can know that sending one byte from CC2530 to Multiplexer needs almost 4.24*us*, and the data rate is approximately 235.85KB/s. Figure.14 indicates the time relations and data transmission procedure when Multiplexer writes FIFO-I. For the designed filtering mechanism in Figure.9, the average transmission time of one 8-bit data frame is almost 4.375*us*, so the receiving speed of Multiplexer is approximately 228.57KB/s.

Based on the prototype network, the full communication performance is also evaluated. When the average length of data frame is 64 bytes, the maximum transmission and receiving speeds of WEPL are 29.27KB/s and 27.56KB/s, respectively. Assume that the two slave nodes are independent, their shortest control period can be 5*ms*. For our designed Pattern-Sewing machine that compose a three-axis linkage electromechanical-mechanism, forming a cooperative movement between X-Y plane and a needle in vertical direction. For the three-axis linkage system based on WEPL, as long as its maximum control time of X-Y plane motion at each frame is not over 10*ms* the maximum sewing speed will reach 3000 needles per minute, which satisfies our requirements.

5. Conclusion

Real-time networks have been becoming one mainly technology for novel industrial systems and embedded applications. For the TDMA mechanism of Powerlink, extending it with a wireless interface is possible. In this paper, we proposed a wireless extension scheme at MAC layer on the basis of a self-designed WEPL hardware. Further, a dual-FIFO based multiplexer logic and interface are designed and implemented when several synchronization problems are well resolved. Experiments show that the basic communication performance can satisfy the requirements of different applications.

Our ongoing and future studies on this topic are mainly on the optimization of hardware design and protocols, applying this extension to other time-slot based protocol, and the applications of such real-time wireless networks in the design of novel industrial equipments.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (61572403, 61502394), the postdoctoral Research Project of Foundation Franco-China for Science and Application (FFCSA 2012), the Fundamental Research Funds for the Central Universities (3102015JSJ0002), Shaanxi Provincial Technical Innovation Project (ZDKG-83).

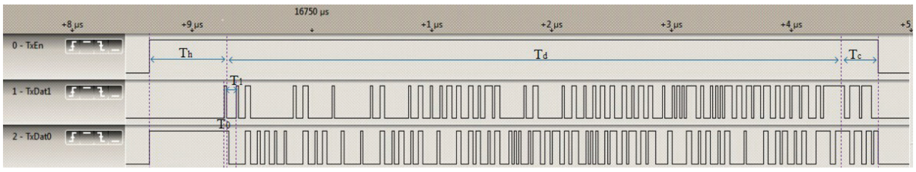

Figure 11: Data Waveform of openMAC, via Salea16.

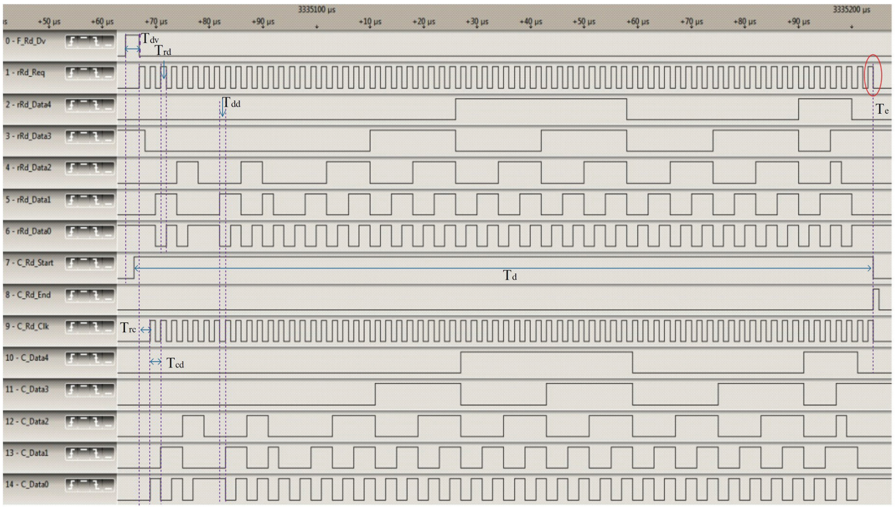

Figure 12: Data and Timing Waveform when I/O Multiplexer Sending Data to CC2530, via Salea16.

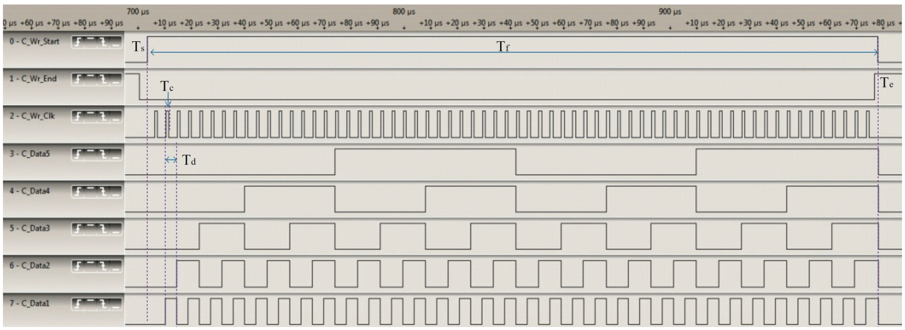

Figure 13: Data and Timing Waveform when CC2530 Writing to Multiplexer, via Salea16.

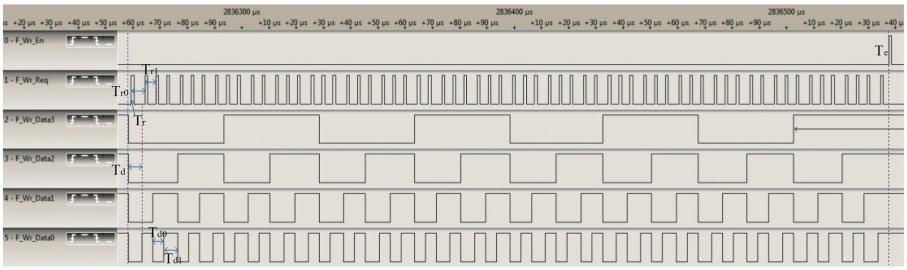

Figure 14: Data and Timing Waveform when Multiplexer Writing to FIFO-I, via Salea16.

References

1. G. Prytz, A performance analysis of EtherCAT and PROFINET IRT, IEEE International Conference on Emerging Technologies and Factory Automation, (Hamburg, German, 15-18 Sept 2008),pp.408-415.
2. J. A. Maestro and P. Reviriego, Energy Efficiency in Industrial Ethernet: The Case of Powerlink, IEEE Transactions on Industrial Electronics, (Vol:57(8), Aug 2010),pp.2896-2903.
3. C. Kunzle, D. Bursic and H. D. Doran, Embedding Real Time Ethernet: Examining feasibility of separat- ing bus master and application master in industrial POWERLINK implementations, IEEE 16th Confer- ence on Emerging Technologies & Factory Automa- tion, (Toulouse, France, 5-9 Sept,2011),pp.1-6.
4. G. Yoon, D. H. Kwon, S. C. Kwon and Y. O. Park, Ring Topology-based Redundancy Ethernet for In- dustrial Network, International Joint Conference SICE-ICASE, (Busan, South Korea, 18-21 Oct. 2006),pp.1404-1407.
5. M. Rostan, J. E. Stubbs and D. Dzilno, EtherCAT enabled advanced control architecture, IEEE/SEMI Advanced Semiconductor Manufacturing Conference, (San Francisco, Canada, 11-13 July 2010),pp.39-44.
6. R. Schlesinger, A. Springer and T. Sauter, Improv- ing profinet IRT frame packing using ethernet control characters, IEEE World Conference on Factory Com- munication Systems, (Palma de Mallorca, 27-29 May 2015),pp1-4.
7. R. Schlesinger, A. Springer and T. Sauter, New ap- proach for improvements and comparison of high performance real-time ethernet networks, IEEE Emerg- ing Technology and Factory Automation, (Barcelona, Spain, 16-19 Sept. 2014),pp1-4.
8. S. Limal, S. Potier, B. Denis and J. Lesage, Formal verification of redundant media extension of Ethernet PowerLink, IEEE Conference on Emerging Technolo- gies and Factory Automation, (Patras, Greece, 25-28 Sept, 2007),pp.1045-1052.
9. Ethernet PowerLink Standardization Group: Ether- net PowerLink Communication Profile Specification V. 2.0, , Ethernet PowerLink Standardization Group Std., 2003. [Online]. Available: http://www.ethernet-powerlink.org
10. J. Kjellsson, A. E. Vallestad, R. Steigmann and D. Dzung, Integration of a Wireless I/O Interface for PROFIBUS and PROFINET for Factory Automa- tion, IEEE Transactions on Industrial Electronics, (Vol:56(10), 2009),pp.4279-4287.
11. L. Seno, S. Vitturi and C. Zunino, Analysis of Ether- net Powerlink Wireless Extensions Based on the IEEE 802.11 WLAN, IEEE Transactions on Industrial In- formatics, (Vol:5(2), 2009),pp.86-98.
12. S. C. Smith, R. J. Hammell, T. W. Parker and L. M. Marvel, A Theoretical Exploration of the Impact of Packet Loss on Network Intrusion Detection, Inter- national Journal of Networked and Distributed Com- puting, (Vol:4(1), 2016),pp.1-10.