Development of CRC Block on FPGA for Zigbee Standard

Rafidah Ahmad, Othman Sidek, Shukri Korakkottil Kunhi Mohd.
Collaborative MicroElectronic Design Excellence Centre (CEDEC)
Engineering Campus, Universiti Sains Malaysia
14300 Nibong Tebal, Penang, Malaysia.

ABSTRACT

CRC (Cyclic Redundancy Check) block was developed on FPGA (Field Programmable Gate Array) in order to meet the needs for simple, low-power and low-cost wireless communication such as Zigbee. Zigbee operates primarily in the 2.4 GHz band, which makes the technology easily applicable and worldwide available. This paper gives a short overview of CRC block in the digital transmitter based on Zigbee Standard. CRC is the most preferred method of encoding because it provides very efficient protection against commonly occurring burst errors, and is easily implemented. The purpose of the research is to diversify the design methods by using the Verilog code entry through Xilinx ISE 8.2i. The Verilog code is used to characterize the CRC block behavior which is then simulated, synthesized and successfully implemented on Spartan3E XC3S500E FPGA. Here, the simulation and measurement results are also presented to verify the functionality of the CRC block. The data rate of CRC block is 250 kbps.

INTRODUCTION

Zigbee is the name of the specification for short range wireless communication that requires a low data rate, long battery life, low power consumption, secure networking, and cheaper technology. The Zigbee standard is also known as the IEEE 802.15.4 standard for WPANs (Wireless Personal Area Networks). The standard specifies that a compliant system will operate in three license-free bands: 2.45 GHz (250 kbps maximum data rate) for worldwide use, 868 MHz (20 kbps) for North America, and 915 MHz (40 kbps) for Europe [1]. The transmission range is 10 to 100 meters based on the environment [2].

The major applications of the Zigbee focus on sensor and automatic control such as military application, industrial control, smart buildings and environment monitoring [3]. The Zigbee architecture recognizes three devices: coordinator, router and end device [4]. Meanwhile, the Zigbee network layer supports three networking topologies: star, mesh, and cluster tree [5, 6]. The star networks support a single Zigbee coordinator with one or more Zigbee end devices and provides a long battery life operation. Mesh or peer-to-peer networks enable high levels of reliability and scalability by providing many paths through the network. Cluster tree networks utilize a hybrid star or mesh topology that combines the benefits of both for high levels of reliability and support for battery-powered nodes.

The Zigbee standard has evolved standardized sets of solutions called "layers". The layers are network and application support layer, PHY (Physical) layer, and MAC (Media Access Control) layer [7]. There are four packet frame types: data, acknowledgment, MAC command and beacon.

In this paper, the design includes the behavioral coding, simulation and implementation of CRC (Cyclic Redundancy Check) block for Zigbee Standard on FPGA (Field Programmable Gate Array) through Xilinx software. The Spartan3E XC3S500E family has been used as FPGA device. This device is specifically designed to meet the needs of high volume and cost-sensitive consumer electronic applications. The device enhancements, combined with advanced 90 nm process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry [8].

This paper is organized as follows. In second section, an overview of the Zigbee digital transmitter is given followed by the characteristic of the CRC block in the third section. The fourth section explains the design methodology. In fifth section, the results and discussion of final simulation and measurement for CRC block is presented. Finally, the paper is concluded.

ZIGBEE TRANSMITTER

Sixteen channels are available for the 2.4 GHz band application with ample channel spacing of 5 MHz. This standard employs a DSSS (Direct Sequence Spread Spectrum) that uses a digital spreading function representing PN (Pseudo-random Noise) chip sequences [9].

As this project is focused on acknowledgment frame, it is used to confirm successful frame reception. Figure 1 shows the structure of the acknowledgment frame which originates from within the MAC sub-layer [10]. This frame is constructed from an MHR (MAC Header) and an MFR (MAC Footer). The MHR contains the MAC Frame Control field and the DSN (Direct Sequence Number) while the MFR is composed of a 16-bit FCS (Frame Check Sequence). Together, the MHR and the MFR form the MAC acknowledgment frame and passes to the PHY as the PSDU (PHY Service Data Unit), which becomes the PHY payload. The PHY payload is prefixed with the SHR (Synchronization Header), containing the Preamble Sequence and the SFD (Start of Frame Delimiter) fields, and the PHR (PHY Header). The SHR, PHR and PHY payload form the PHY packet known as the PPDU (PHY Protocol Data Unit).

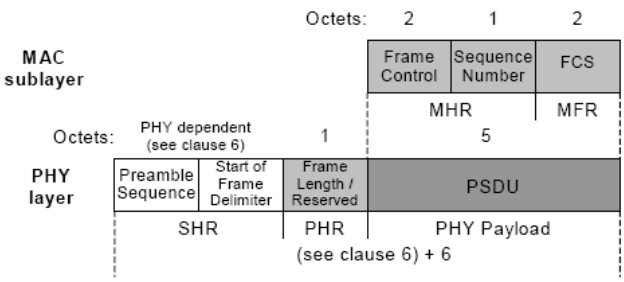


FIGURE 1. SCHEMATIC VIEW OF THE ACKNOWLEDGMENT FRAME AND PHY PACKET

In this paper, the Preamble Sequence contains 32 bits which are at logic '0'. For the SFD, the length is 8 bits with logic '1110 0101' as stated in the Zigbee standard. The PHR also contains 8 bits in length with logic '1010 0000' [10]. The Frame Control is 16 bits in length

978-1-4244-4342-0/09/$25.00 © 2009 IEEE

with logic '0100 0100 0000 0000'. The Sequence Number logic is '1000 0000'. Finally, the FCS is 16 bits in length which contains the CRC algorithm that depends on the MHR. This gives the total of 88 bits for the acknowledgment frame.

Figure 2 shows the block diagram of the proposed Zigbee digital transmitter based on [10] and [11]. Binary data from the PPDU packet is inserted into the CRC block and then to the bit-to-symbol block. After that, every 4 bits are mapped into one data symbol. The symbol-to-chip block performs the DSSS where each symbol is mapped into a 32-chip PN sequence [12]. In the 2.4 GHz band, the O-QPSK modulation is adopted. The fundamental O-QPSK method is to sum the in-phase signal with a quadrature phase signal delayed by half a cycle in order to avoid the sudden phase shift change [11]. Then, the modulated O-QPSK signal goes to the half-sine pulse shaping stage in order to reduce inter-symbol interference [13]. The resultant signal is transmitted by the RF (Radio Frequency) transmitter.

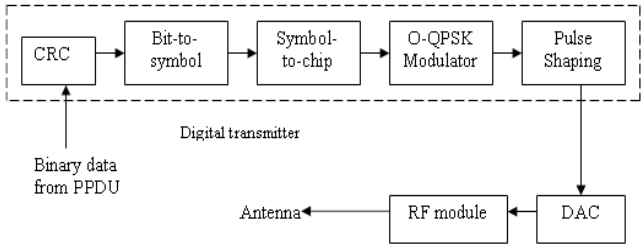


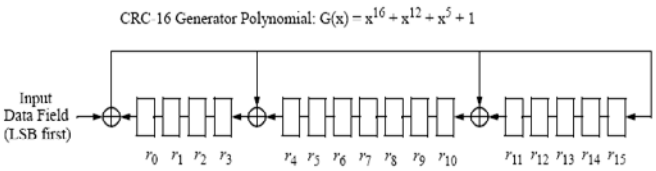
FIGURE 2. DETAILED BLOCK DIAGRAM OF THE PROPOSED ZIGBEE DIGITAL TRANSMITTER

CRC BLOCK

CRC is the most preferred method of encoding because it provides very efficient protection against commonly occurring burst errors [14], and is easily implemented [15]. CRC's can detect all one bits and two bits errors as well as all odd number of bits in error [16]. Since CRC is a technique for detecting errors, but not for making corrections when errors are detected, the whole packet data will be retransmitted if error occurs [17].

For Zigbee Standard, CRC involves a division of the transmitted packet data by a constant called the generator polynomial [18]. In this paper, the CRC block contains the SHR, PHR and PHY payload. For the acknowledgment frame, 11 octets which are equal to 88 bits will be the output of the CRC block.

In the PHY payload, the FCS mechanism employs a 16-bit CRC in order to detect errors [10]. The FCS is calculated over the MHR and MFR payload parts of the frame. The FCS will be calculated for transmission using the algorithm in Figure 3, based on [10].



1. Initialize the remainder register (r_0 through r_{15}) to zero.
2. Shift MHR and payload into the divider in the order of transmission (LSB first).
3. After the last bit of the data field is shifted into the divider, the remainder register contains the FCS.
4. The FCS is appended to the data field so that r_0 is transmitted first.

FIGURE 3. ALGORITHM FOR THE FCS

DESIGN METHODOLOGY

The CRC block is designed through Xilinx ISE 8.2i. Here, the behavior of the CRC block is characterized using the Verilog code. Next, this code is synthesized in order to convert the Verilog code to the logic gates and to check the syntax of the design to find any errors. Then, a simulation waveform is presented before the implementation process purposely to ensure the design's output waveform matches the theoretical expectation. After the CRC block was implemented on Spartan3E board, it was measured using Logic Analyzer as shown in Figure 4, once again to make sure that the design's output waveform matches the simulation waveform.

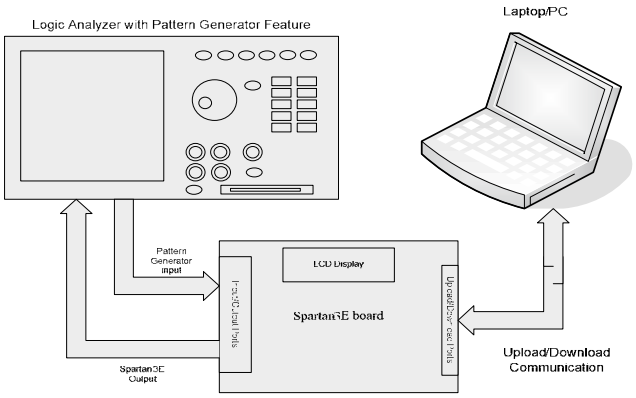


FIGURE 4. THE CONNECTION BETWEEN SPARTAN3E BOARD AND LOGIC ANALYZER

Figure 5 shows that the CRC block has 5 input ports: "clk", "reset", "write", "shift_enable", and "data_in". The only output port is "data_out". The "clk" is used for input and output data timing. For the CRC block, the frequency is set to 250 kHz.

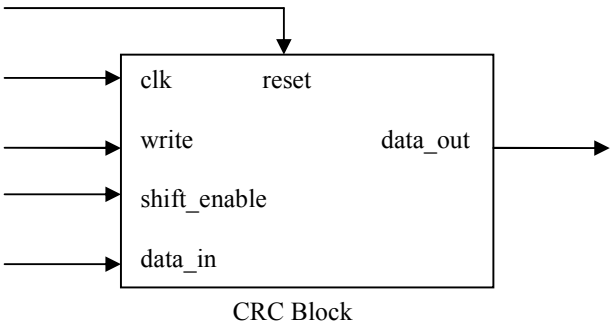


FIGURE 5. THE STRUCTURE OF THE CRC BLOCK

By referring to Figure 6, a flow diagram for the CRC block operation is shown. Beginning from the starting point, if the "reset" is at logic '1', all the registers in the CRC block will be reset. However, with the opposite condition, if "write" is at logic '1', then the input data will be processed and stored in the "count_in" and "count_out" registers. After that, if "shift_enable" is at logic '1', the data from "count_out" will be shifted serially to "data_out". If not, the data of "count_out" will always be at logic '1'.

One part of the Verilog code for the CRC block is shown in Figure 7. The code will be synthesized, simulated and then implemented on FPGA to verify its functionality.

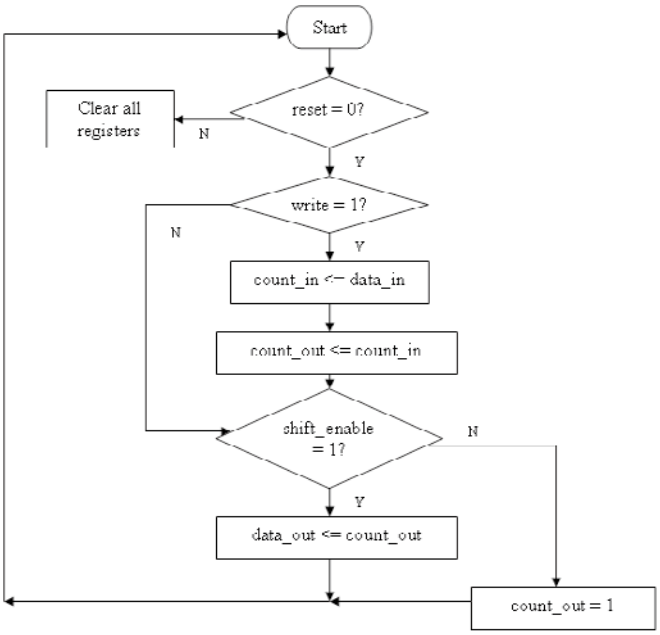


FIGURE 6. THE CRC BLOCK DESIGN FLOW

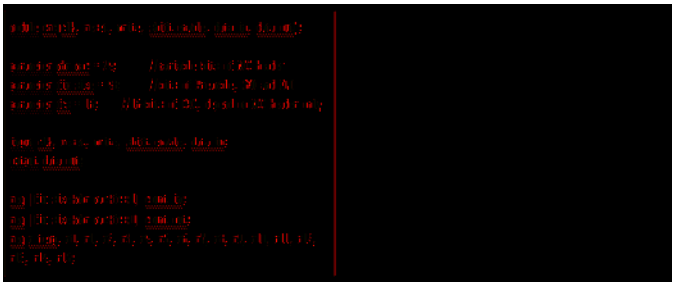


FIGURE 7. VERILOG CODE FOR THE CRC BLOCK

RESULTS AND DISCUSSION

Final simulation was carried out for CRC block which was designed based on its characteristic. Then, the design code was implemented on Spartan3E XC3S500E FG320 as FPGA device in order to get the measurement result. All the input ports were forced to have certain logic value based on the Zigbee Standard, and the output data was totally dependent on the input data. The frequency for CRC block was set to 250 kHz. This means that 1 clock cycle is equal to 4 µs.

Simulation Waveform

As shown in Figure 8, when "shift_enable" is set to logic '1' with the "reset" and "write" at logic '0', the output data is shifted serially to the "data_out" from 306 µs until 658 µs. The 16-bit FCS is shifted starting at 594 µs with logic '0110 1011 0101 0110'. The total of the output bits are 88 within 352 µs (658µs – 306 µs).

From the final simulation waveform, it shows that the output data of the CRC block match the theoretical expectation. The timing for each data bit is 4 µs.

Measurement Result

From Figure 9, it shows that within M1 and M2, where only "shift_enable" is enabled, output data is shifted serially to the "data_out". The range of "data_out" is approximately 352 µs and the logic value is exactly similar to the output data in simulation waveform.

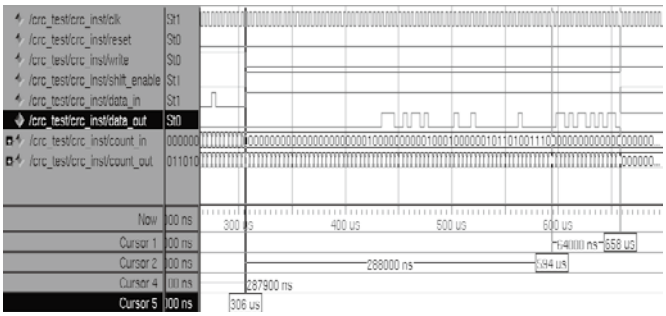


FIGURE 8. SIMULATION WAVEFORM WHEN "SHIFT ENABLE" IS ENABLED

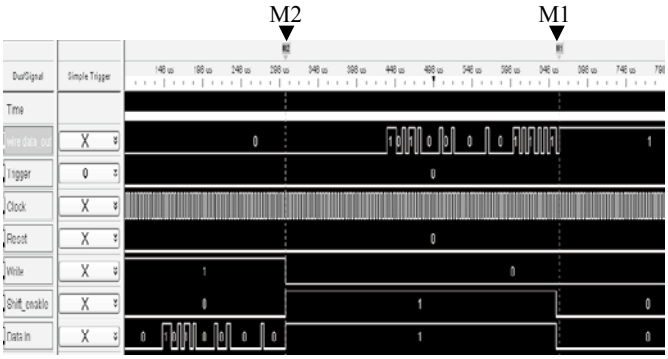


FIGURE 9. MEASUREMENT WAVEFORM OF THE OUTPUT DATA

From the measurement result, a small delay occurred at the output data after the rising edge of a clock, which is most probably caused by pin and net delays on Spartan3E. Since the CRC block was implemented on Spartan3E with speed grade of 5, the following occupation was obtained: the CRC configuration required about 102 slices of 4 656 (2%). The minimum input arrival time before clock is 5.627 ns and the maximum output required time after clock is 4.364 ns. All signals were completely routed within 16 s with 165 MB CPU memory usage.

CONCLUSIONS

This paper describes the development of CRC block for 2.4 GHz band Zigbee Standard on FPGA. This block is used to detect errors in packet data during transmission. Verilog code has been used to characterize the CRC block behavior which is then simulated, synthesized and successfully implemented on Spartan3E. From the measurement result, the functionality of CRC block matches the theoretical expectation. The design has been verified with the frequency of 250 kHz.

ACKNOWLEDGMENT

The authors would like to acknowledge the Universiti Sains Malaysia Short Term Grant for financial support of this project.

REFERENCES

[1] *EDA 390-Computer Communication and Distributed Systems: A Paper on Zigbee*, http://www.cs.chalmers.se/~tsigas/Courses/DCDSeminar/Files/diovoad%20Zigbee.pdf .

[2] J. S. Lee, Y. W. Su and C. C. Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, Zigbee and WiFi," *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pp. 46-51, Nov. 2007.

[3] Y. Zhou, X. Yang, X. Gou, M. Zhou and L. Wang, "A Design of Greenhouse Monitoring & Control System Based on Zigbee Wireless Sensor Network," *International Conference on Wireless Communications, Networking & Mobile Computing*, pp. 2563-2567, 2007.

[4] *MSP430 Goes Zigbee/802.15.4*, http://focus.ti.com/lit/ml/slap129/slap129.pdf .

[5] C. C. William, *Zigbee: Wireless Control That Simply Works*, ZMD America, Inc.

[6] T. L. Khanh, "Zigbee System on Chip (SoC) Design," *High Frequency Electronics*, pp. 16-25, January 2006.

[7] *Introduction*, www.tutorial-reports.com/book/print/153 .

[8] Xilinx, *Spartan3E FPGA Family: Complete Data Sheet*, 2006.

[9] A. E. Oualkadi, L. V. Andendorpe and D. Flandre, "System-level Analysis of O-QPSK Transceiver for 2.4 GHz band IEEE 802.15.4 Zigbee Standard," *14th International Conference on Mixed Design*, pp. 469-474, June 2007.

[10] *IEEE 802.15.4-2003 Standard*, http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf .

[11] *A Low Power 2.45 GHz Zigbee Transceiver for Wearable Personal Medical Devices in WPAN*, http://ieeexplore.ieee.org/iel5/4145986/4099325/04146217.pdf .

[12] S. Khaled, A. Maryam, B. Mohamed, J. Imad, S. Farag and L. Abderrahmane, "Performance Evaluation of IEEE 802.15.4: Experimental and Simulation Results," *Journal of Communications*, Vol. 2, No. 4, pp. 29-37, June 2007.

[13] N. John, C. Anthony and L. Gary, "CMOS RFIC Architectures for IEEE 802.15.4 Networks," *Cadence Design Systems,* Columbia, USA, 2003.

[14] S. Shukla and N. W. Bergmann, "Single bit error correction implementation in CRC-16 on FPGA," *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*, pp. 319-322, 2004.

[15] D. C. Feldmeier, "Fast Software Implementation of Error Detection Codes," *IEEE/ACM Transactions on Networking*, vol. 3, No. 6, pp. 640-651, December 1995.

[16] N. Matloff, *Cyclic Redundancy Checking*, Department of Computer Science, University of California, 2001.

[17] X. Deng, M. Rong, T. Liu, Y. Yuan and D. Yu, "Segmented Cyclic Redundancy Check: a Data Protection Scheme for Fast Reading RFID Tag's Memory," *Proceedings of WCNC*, pp. 1576-1581, 2008.

[18] O. O. Khalifa, M. D. R. Islam and S. Khan, "Cyclic Redundancy Encoder for Error Detection in Communication Channels," *RF and Microwave Conference*, pp.224-226,October 2004.