Cyclic Redundancy Checking (CRC) Accelerator for Embedded Processor Datapaths

Abdul Rehman Buzdar*, Liguo Sun*, Rao Kashif†, Muhammad Waqar Azhar‡, Muhammad Imran Khan†§
*Department of Electronic Engineering and Information Science
†Micro/Nano Electronic System Integration R & D Center (MESIC)
University of Science and Technology of China (USTC), Hefei, China
‡Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
§Department of Electronics Engineering, University of Engineering and Technology Taxila, Pakistan

Abstract—**We present the integration of a multimode Cyclic Redundancy Checking (CRC) accelerator unit with an embedded processor datapath to enhance the processor performance in terms of execution time and energy efficiency. We investigate the performance of CRC accelerated embedded processor datapath in terms of execution time and energy efficiency. Our evaluation shows that the CRC accelerated Microblaze SoftCore embedded processor datapath is 153 times more cycle and energy efficient than a datapath lacking a CRC accelerator unit. This acceleration is achieved at the cost of some area overhead.**

Keywords—CRC; Accelerator; Codesign; FPGA; MicroBlaze; Embedded Processor

I. Introduction

For reliable data communication Cyclic Redundancy Checking (CRC) is a well-known technique for error detection. The CRC calculations requires limited hardware resources and can be implemented easily. This is the reason that CRC is being used in industry for three decades, in spite the fact that more advance techniques for error detection and correction have been developed e.g. Viterbi decoder, low-density parity-check (LDPC), Reed Solomon and Turbo codes [1], [2], [3], [4].

Generally CRC can be implemented in software and executed on an embedded processor which requires a lot of clock cycles for the computation of CRC. The CRC can be implemented more efficiently in dedicated hardware which will require few clock cycles for the computation of CRC with some area overhead. The high speed communication systems today requires fast data rates which can only be delivered using dedicated hardware solutions.

Different hardware modules like USB, Ethernet, TCP/IP and CAN protocol are included in modern embedded processors to speedup certain parts of application in areas like signal processing, communication and control systems. All these protocols uses CRC for error detection. The addition of CRC accelerator into the embedded processor datapath will help to improve the overall performance. The commercially available off the shelf microcontrollers [5] and DSPs [6], [7] contain CRC hardware accelerator blocks.

II. CRC Computation Techniques

The computation of CRC is remainder of long modulo-2 division of input polynomial with a key polynomial. The CRC operation is performed in hardware using exclusive-OR and shift operations. The hardware implementation of CRC operation is composed of Exclusive-OR gates computational network which gives remainder and the registers for storage and shifting of current state, shown in Fig. 1. The key polynomial decides the width of state register e.g. for 16 bit key polynomial the width of state register will be 16 bits. The exclusive-OR gate network size depends on the input width and the technique used for the computation of CRC function.

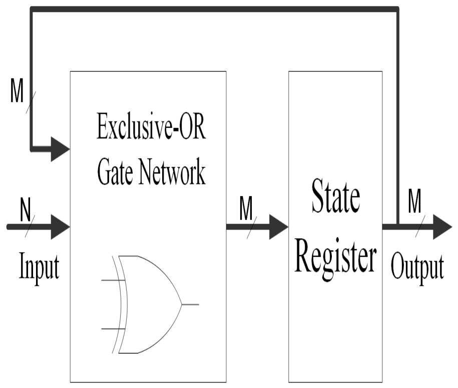

Figure 1: General architecture of a CRC computation circuit.

In serial implementation of CRC function Serial Linear Feedback Shift Registers (LFSR) are used which take only one input bit at a time. The serial implementation of CRC results into smaller exclusive-OR gate network but they are very slow as only one bit is shifted into LFSR circuitry in each cycle. Now a days parallel implementations of CRC are mostly used as they deliver faster speed. Fig. 2 shows the serial LFSR circuit implementation of key polynomial Equation (1). Every exponent of key polynomial is converted into an exclusive-OR gate between input and feedback path. This implementation is very slow as it accepts one bit at each cycle.

$$p(x) = x^5 + x^3 + x + 1 \tag{1}$$

The unfolding methodology [8], [9] can be used to implement parallel CRC circuitry which also uses LFSR as basic building block. Equation (1) can be converted into parallel CRC circuit using these unfolding techniques, as shown in Fig. 3. This parallel implementation of CRC gives twice speedup

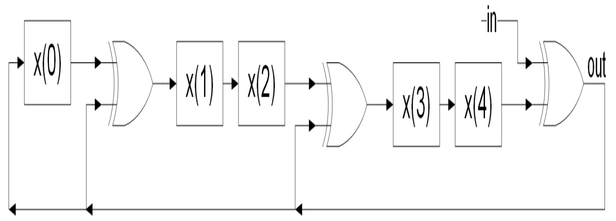

Figure 2: CRC circuit based on LFSR, implementing the key in Eq. 1.

compared to serial LFSR. This unfolding technique can be used to implement higher order parallelism which gives more speedup. But this technique has a drawback of higher fan-out as we increase the order of parallelism [10].

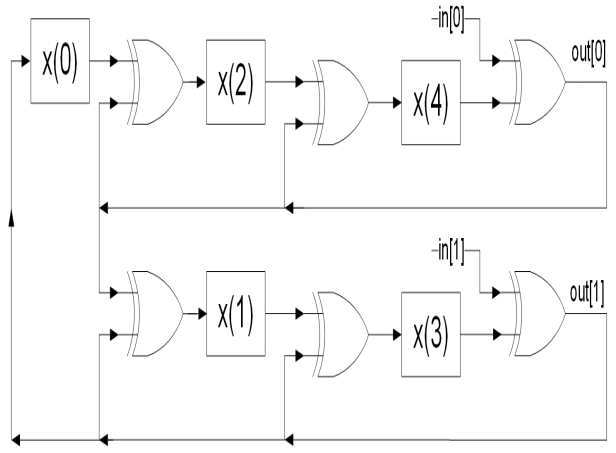

Figure 3: 2-level, unfolded CRC circuit that accepts two input bits each cycle.

A more efficient technique called state-space transformation can be used to implement parallel CRC circuits [11]. We implemented the parallel CRC circuits used in this work by following a technique invented by Campobello et al [12], shown in Fig. 4.

III. CRC Accelerator Implementation

We have implemented a 32-bit accelerator unit by including commonly used CRC circuits i.e. CRC5, CRC8, CRC16 and CRC32 inside a CRC accelerator main block. As we want to integrate this CRC accelerator unit with an embedded processor datapath, so it should be able to perform commonly used CRC operations. The required CRC operation can be selected using a 2-bit control signal. This configurable CRC accelerator unit is depicted in Fig. 5 and it can perform the following CRC operations [13], [14]:

1) **00**: CRC5 for USB interface.

$$p(x) = x^5 + x^2 + 1 \tag{2}$$

2) **01**: CRC8 for ATM protocols, etc.

$$p(x) = x^8 + x^2 + x + 1 \tag{3}$$

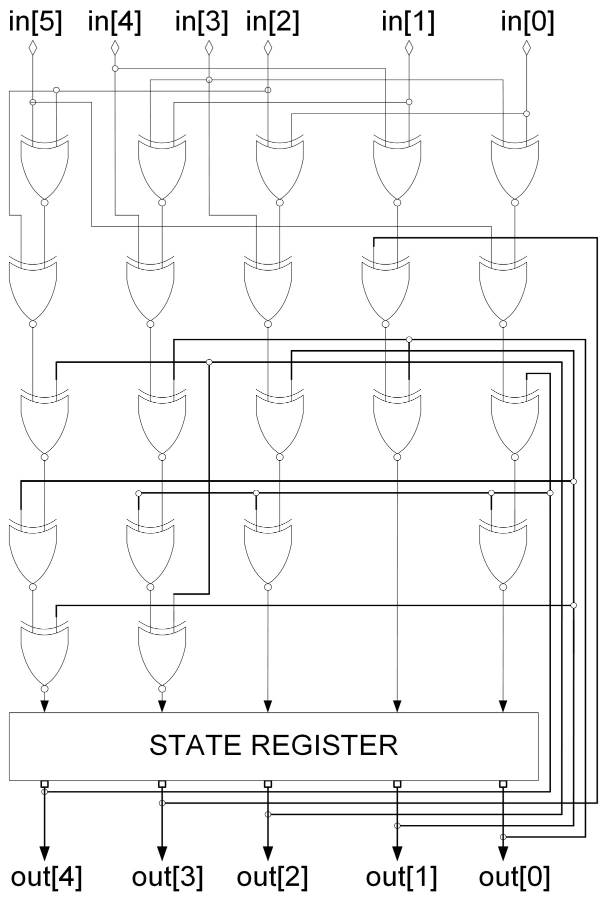

Figure 4: 6-bit input parallel CRC circuit for the key in Eq. 2.

3) **10**: CRC16 for XMODEM, X25 protocols, etc.

$$p(x) = x^{16} + x^{12} + x^5 + 1 \tag{4}$$

4) **11**: CRC32 for IEEE 802.3 standard.

$$\begin{aligned} p(x) =& x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + \\ & x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \end{aligned} \tag{5}$$

After performing the power analysis of CRC accelerator unit, shown in Fig. 5. We found that this design is not power efficient. The reason of this power inefficiency is the unnecessary switching taking place in all the CRC blocks. Because only one CRC sub-unit is required for the computation of desired CRC operation. So we decided to design a more power efficient CRC accelerator unit by disabling the CRC blocks which are not in use by employing power gating technique. We also used distributed multiplexer at the output of CRC accelerator unit and clock gating for disabling the registers which are not in use to save the switching power. The power efficient CRC accelerator unit is shown in Fig. 6. Both the CRC Accelerator units were verified and synthesized using Xilinx ISE Design Suit [15]. The Initial CRC unit and Low Power CRC hardware accelerator block were synthesized on 7vx485tffg1157-3 Virtex-7 FPGA device which is based on a 28nm technology. It gives a critical path delay of 1.868ns and 1.986ns respectively. The synthesis results are shown in

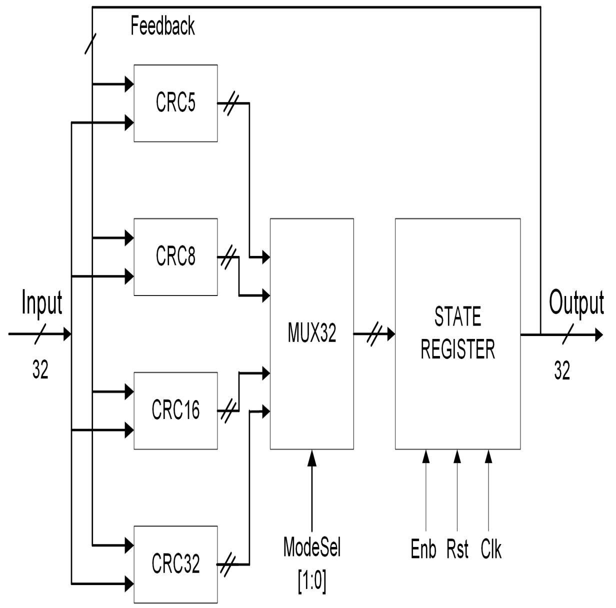

Figure 5: First CRC accelerator unit.

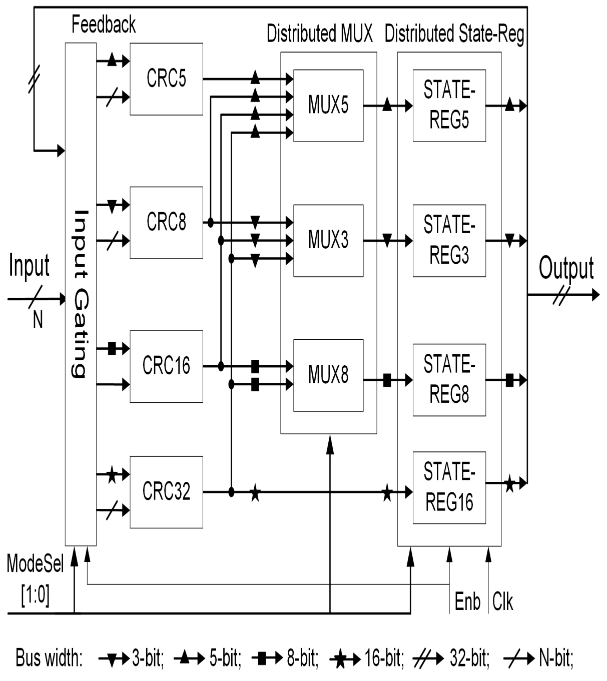

Figure 6: Low-power CRC accelerator unit.

Table I. As can be seen the low power CRC accelerator is power efficient compared to the initial CRC unit.

Table I: Synthesis Results of CRC Accelerator Units

	First CRC Unit	**Low Power CRC Unit**
Power	523mW	241mW
Max Freq	535.217MHz	503.499MHz
Latency	1.868ns	1.986ns
Slice Registers	32	32
Slice LUTs	229	340
Occupied Slices	98	166

IV. Integration of CRC Accelerator Unit with MicroBlaze Processor

We have implemented the CRC accelerator unit in VHDL hardware description language and verified it using Xilinx ISE design suit [15]. We used Xilinx Spartan-6 FPGA SP605 Evaluation Kit [17] and Xilinx Embedded Development Kit (EDK) [15] for the implementation. The Hardware/Software co-design is a well established technique which improves the performance of the system [16-19]. Xilinx Microblaze soft core processor [16] was used to run the software implementation of CRC. There are two ways to integrate a hardware accelerator core into a MicroBlaze-based embedded soft processor system. One way is to connect the accelerator through the Processor Local Bus (PLB). The second way is to connect it using MicroBlaze dedicated Fast Simplex Link (FSL) bus system [18]. First PLB was tried but it was taking a lot of cycles. Because it is a traditional memory mapped transaction bus. Then it was decided to integrate our CRC accelerator unit using a dedicated FIFO style FSL Bus with the MicroBlaze processor system, shown in Fig. 7.

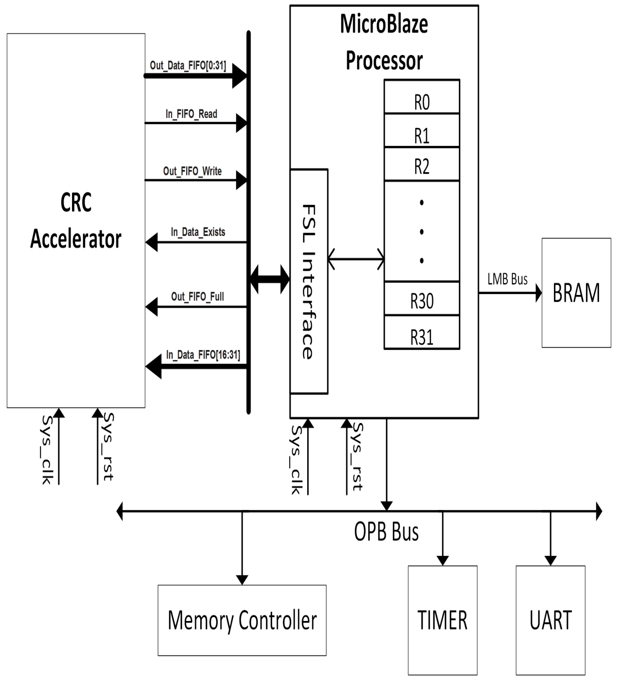

Figure 7: CRC Accelerator Unit with MicroBlaze Processor System

The software only C code for CRC5, CRC8, CRC16, CRC32 was implemented and verified. Later these C codes were executed on the MicroBlaze processor using Xilinx Software Development Kit (SDK) [15]. The cycle count for the complete software implementations of CRC was measured using the XPS hardware timer block, shown in Table II. Fig. 8 and 9 shows the cycle count and energy dissipation of different architectures, respectively.

Table II: Cycle Count and Energy Dissipation at Clock Period 20ns

Architecture	#Cycles	Power (mW)	Energy* (μJ)
CRC5 SW	1086	178	3.8661
CRC8 SW	2652	178	9.4411
CRC16 SW	5200	178	18.512
CRC32 SW	5373	178	19.1278
CRC HW	35	184	0.13

*: Energy = #cycles $\times$ clock period $\times$ power.

The CRC accelerator unit was attached with the Microblaze processor system via FSL bus using Xilinx Platform Studio (XPS) [15]. The software part of CRC accelerator unit was implemented in C programming with Xilinx SDK. The predefined C functions of SDK were used to communicate with hardware part of CRC accelerator unit via FSL bus. Our evaluation shows that an accelerated MicroBlaze processor datapath is 153 times more cycle and energy efficient than a datapath lacking CRC accelerator.

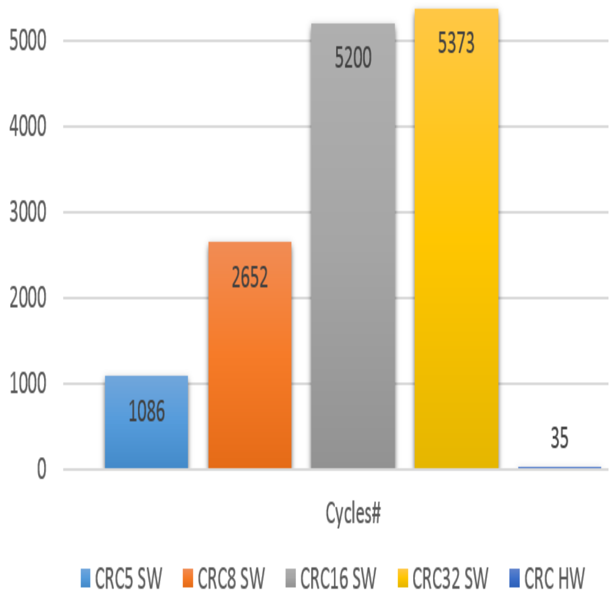

Figure 8: Cycle count of various CRC implementations.

V. Conclusion

In this paper, we have designed a flexible CRC accelerator unit using VHDL. We have integrated the CRC accelerator unit with the Microblaze Softcore processor system using FSL Bus to enhance the processor performance. We used Xilinx Spartan-6 FPGA Evaluation Kit and Xilinx Embedded Development Kit (EDK) for the implementation. We have shown that a CRC accelerated Microblaze embedded processor datapath is 153 times more cycle and energy efficient that a datapath lacking a CRC accelerator with some area overhead.

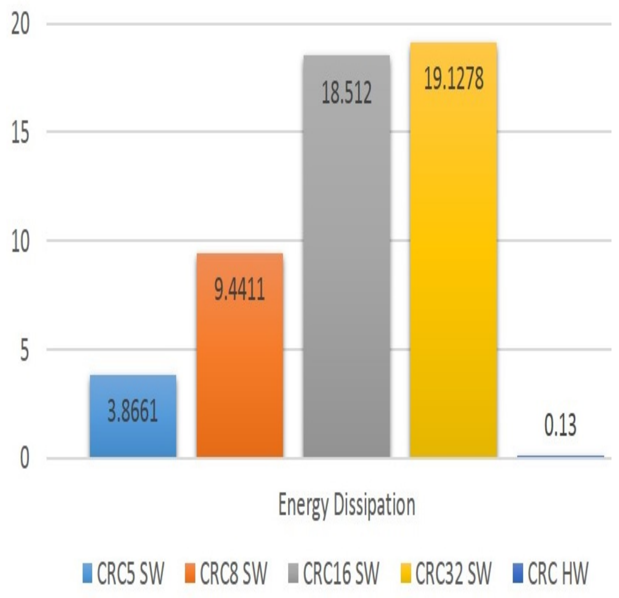

Figure 9: Energy dissipation of various CRC implementations.

Acknowledgment

This work is partially supported by the Chinese Academy of Sciences and The World Academy of Sciences CAS-TWAS President's Fellowship 2013-2017.

References

[1] M. F. Brejza, L. Li, R. G. Maunder, B. Al-Hashimi, C. Berrou, L. Hanzo, "20 years of turbo coding and energy-aware design guidelines for energy-constrained wireless applications", IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 8-28, 1st Quart. 2016.

[2] Mehran Mozaffari Kermani, Vineeta Singh, Reza Azarderakhsh, "Reliable Low-Latency Viterbi Algorithm Architectures Benchmarked on ASIC and FPGA," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 1, pp. 208-216, 2017.

[3] Linjia Chang, Avhishek Chatterjee, Lav R. Varshney, "Performance of LDPC Decoders With Missing Connections," IEEE Transactions on Communications, vol. 65, no. 2, pp. 511-524, 2017.

[4] Salvatore Pontarelli, Pedro Reviriego, Marco Ottavi, Juan Antonio Maestro, "Low Delay Single Symbol Error Correction Codes Based on Reed Solomon Codes," IEEE Transactions on Computers, vol. 64, no. 5, pp. 1497-1501, 2015.

[5] Atmel, "Secure microcontroller for smart cards." [Online]. Available: http://www.atmel.com

[6] Freescale, "MAPLE hardware accelerator and SC3850 DSP core." [Online]. Available: http://www.freescale.com

[7] Microchip, "PIC32mx775f512l datasheet." [Online]. Available: http://www.microchip.com

[8] K. K. Parhi, VLSI Digital Signal Processing Systems - Design and Implementation. Wiley-Interscience Publishers Inc., 1999.

[9] C. Cheng and K. K. Parhi, "High-Speed Parallel CRC Implementation Based on Unfolding, Pipelining, and Retiming," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 53, no. 10, pp. 1017-1021, 2006.

[10] T.-B. Pei and C. Zukowski, “High-Speed Parallel CRC Circuits in VLSI,” IEEE Transactions on Communications, vol. 40, no. 4, pp. 653-657, Apr. 1992.

[11] J. H. Derby, “High-Speed CRC Computation Using State-Space Transformations,” in IEEE International Global Telecommunications Conference, Nov. 2001, pp. 166-170.

[12] G. Campobello, G. Patane, and M. Russo, “Parallel CRC Realization,” IEEE Transactions on Computers, vol. 52, no. 10, pp. 1312-1319, Oct. 2003.

[13] E. Stavinov, “A practical parallel CRC generation method.” [Online]. Available: www.OutputLogic.com

[14] Muhammad Waqar Azhar, Tung Thanh Hoang, and Per Larsson-Edefors, “Cyclic Redundancy Checking (CRC) Accelerator for the Flex-Core Processor,” in Proc. of EUROMICRO Conf. on Digital System Design, 2010, pp. 675-680.

[15] Xilinx Inc. FPGA Design Tools. Silicon Devices. [Online]. Available: http://www.xilinx.com

[16] Xilinx MicroBlaze [Online] www.xilinx.com/tools/microblaze.htm

[17] Xilinx Spartan-6 FPGA SP605 Evaluation Kit. [Online] Available: www.xilinx.com/products/boards-and-kits/ek-s6-sp605-g.html

[18] Xilinx Fast Simplex Link (FSL). [Online] Available: http://www.xilinx.com/products/intellectual-property/fsl.html

[19] Abdul Rehman Buzdar, Liguo Sun, Azhar Latif and Abdullah Buzdar, “Distance and Speed Measurements using FPGA and ASIC on a high data rate system” International Journal of Advanced Computer Science and Applications(IJACSA), 6(10), 2015, pp.273-282.

[20] Abdul Rehman Buzdar, Liguo Sun, Azhar Latif and Abdullah Buzdar, “Instruction Decompressor Design for a VLIW Processor”, Informacije MIDEM-Journal of Microelectronics, Electronic Components and Materials Vol. 45, No. 4 (2015), pp.225-236.

[21] Abdul Rehman Buzdar, Azhar Latif, Liguo Sun and Abdullah Buzdar, “FPGA Prototype Implementation of Digital Hearing Aid from Software to Complete Hardware Design” International Journal of Advanced Computer Science and Applications(IJACSA), 7(1), 2016, pp.649-658.

[22] Abdul Rehman Buzdar, Liguo Sun, Shoab Ahmed Khan, Abdullah Buzdar, “Area and Energy efficient CORDIC Accelerator for Embedded Processor Datapaths” Informacije MIDEM-Journal of Microelectronics, Electronic Components and Materials Vol. 46, No. 4(2016), pp.197-208