MATEC Web of Conferences **48**, 04001 (2016)
DOI: 10.1051/matecconf/20164804001

A study of hardware implementations of the CRC computation algorithms

Evgeniy Mytsko[1,a], Andrey Malchukov[1], Svetlana Ryzova[1] and Valeriy Kim[1]

[1]*National Research Tomsk Polytechnic University, 634050 Tomsk, Russia*

Abstract. The paper is about hardware implementations of the CRC computation algorithms. Combinational circuits of CRC8 and CRC32 computation devices, which can be embedded in satellites for error checking in configuration memory and data transmission module, were considered. The conclusion about the advantages of matrix-driven algorithm hardware implementation, which are a simple diagram is built using only logic «exclusive OR» was done. The examples of CRC8 and CRC32 devices working were presented according to parametric model with different input data.

1 Introduction

For the aerospace and defense electronics, telecommunications and control systems are very important to be able to make sure that the configuration data in the Field-Programmable Gate Arrays (FPGAs) on board does not contain any errors. Ionizing radiation in space can cause unwanted switching of memory cells in the FPGA, which contains the configuration memory, user memory and registers. Such switching may cause equipment failure, which is critical in outer space. CRC (Cyclic redundancy code) is algorithm of checksum computation, which is applied in different data transmission standards, compress algorithms, coding standards and bitmaps [2, 3]. CRC checksums can be used to detect errors and failures in the configuration memory and data transmission modules There are different types of CRC such as CRC8, CRC16, CRC32, which are differed by generator polynomial length [1] and accordingly by the checksum length. In the papers [4, 5] software implementation of table-driven and matrix-driven algorithms were described. The advantages of matrix-driven algorithm implementation, which are concluded in the low requirements of memory, were showed. The asynchronous hardware implementations of CRC8 and CRC32 for FPGA, which can be embedded in satellites, were offered in the paper. The implementation allows to simplify device circuit and to apply features of matrix-driven algorithm in hardware implementation.

2 Hardware implementations of the CRC computation algorithms

2.1 Hardware implementation of the table-driven algorithm

[a] Corresponding author : evgenvt@tpu.ru

Functional diagram of the asynchronous hardware implementation was developed using block-based design in Quartus II for FPGA Cyclone. Figure 1 shows functional diagram of CRC8 computation device. A byte of data which is summed by modulo 2 with *init* [7..0] according to parametric model is supplied to *data* [7..0] input. A byte of CRC is supplied to *crc* [7..0] input.

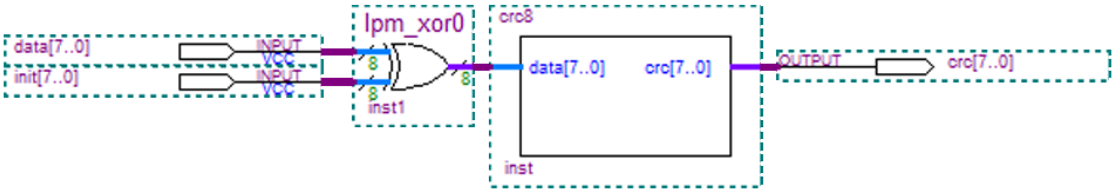

Figure 1. Functional diagram of CRC8 computation device.

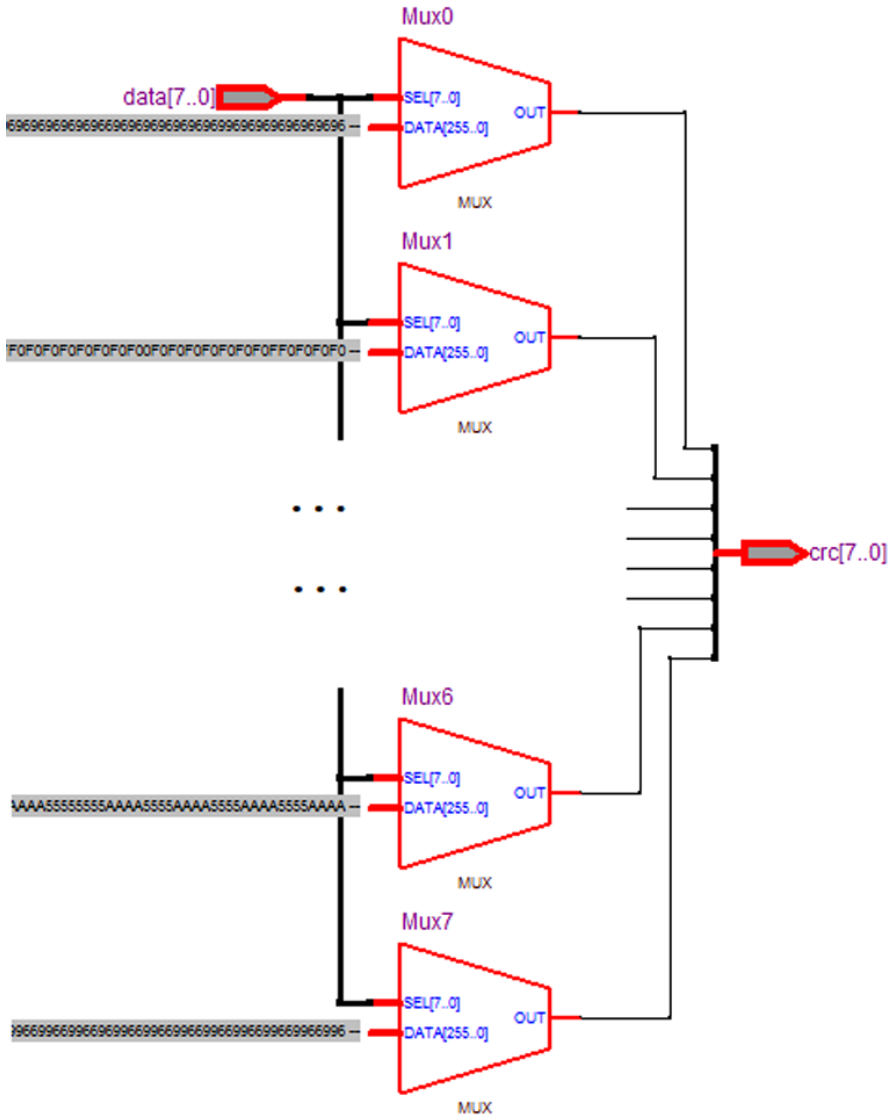

Figure 2. The combinational circuit of hardware implementation of CRC8 table-driven algorithm.

Figure 2 presents the combinational circuit of hardware implementation of CRC8 table-driven algorithm with storage elements (for constant storing) and multiplexers.

According to the table-driven algorithm of CRC computation [4] a byte of data is summed by modulo 2 with checksum byte (the initial value of checksum is chosen by parametric model of algorithm [4]). After that the result of addition is used to index of the table which is filled by reminders of all byte combinations by generator polynomial. The value from the table is a CRC8 checksum. CRC32 computation algorithm is similar to CRC8 with the exception of some features. In this case the length of CRC, the length of generator polynomial and the length of table elements increases to 32 bits.

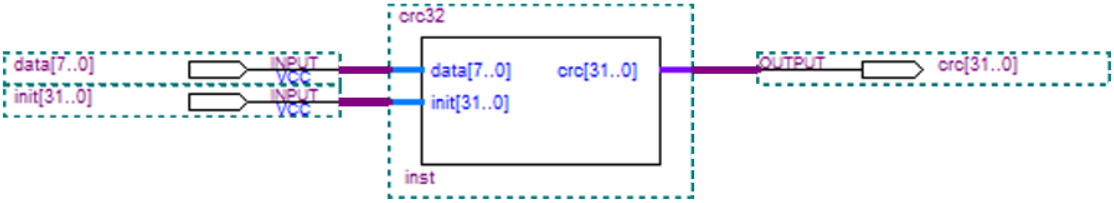

Figure 3. Functional diagram of CRC32 computation device.

For the hardware implementation of CRC32 an additional input *init* [31..0] for circuit is required. 32-bit initial value of CRC32 is supplied to the *init* input according to parametric model. A Low byte of *init* (7 – 0 bits) is summed by modulo 2 with data byte. Then similarly to CRC8 the element of the table is selected and summed by modulo 2 with high three bytes of *init* (31 – 8 bits), which are shifted to the right 8 times. Figure 3 shows functional diagram of CRC32 computation device.

In the CRC32 implementation for table-driven algorithm logic «exclusive OR» will be added to combination circuit and the length of constant will be increased. Herewith the hardware implementation of matrix-driven algorithm of CRC computation allows to avoid multiplexors, storage usage and build combinational circuit using only logic «exclusive OR».

2.2 Hardware implementation of the matrix-driven algorithm

Feature of matrix-driven algorithm of CRC computation is multiplication by modulo 2 of vector (input byte) by matrix [6] which is calculated based on generator polynomial. The feature allows implement matrix-driven algorithm by combinational circuit without storage using only logic «exclusive OR». Figure 4 represents multiplication by modulo 2 of vector by matrix.

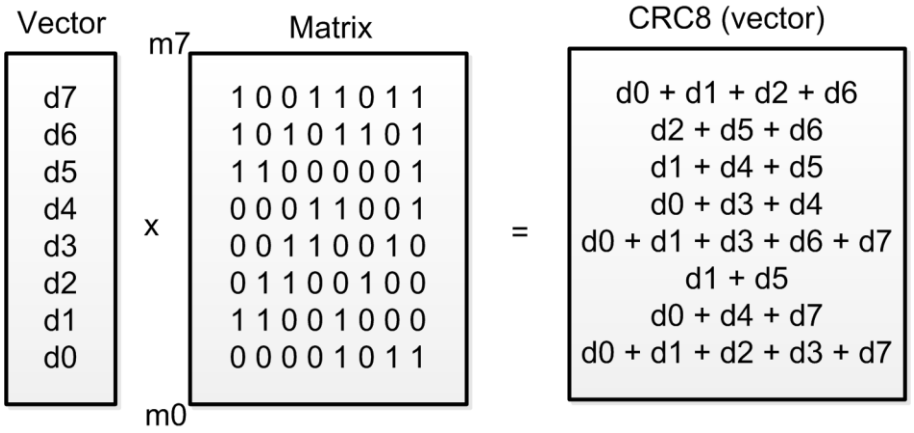

Figure 4. Multiplication by modulo 2 of vector by matrix.

Thus, figure 5 represents the combinational circuit of CRC8 computation by matrix algorithm. There are no multiplexors and constants in the circuit. CRC computation is produced without storage elements using only logic «exclusive OR», which is a feature of matrix algorithm hardware implementation.

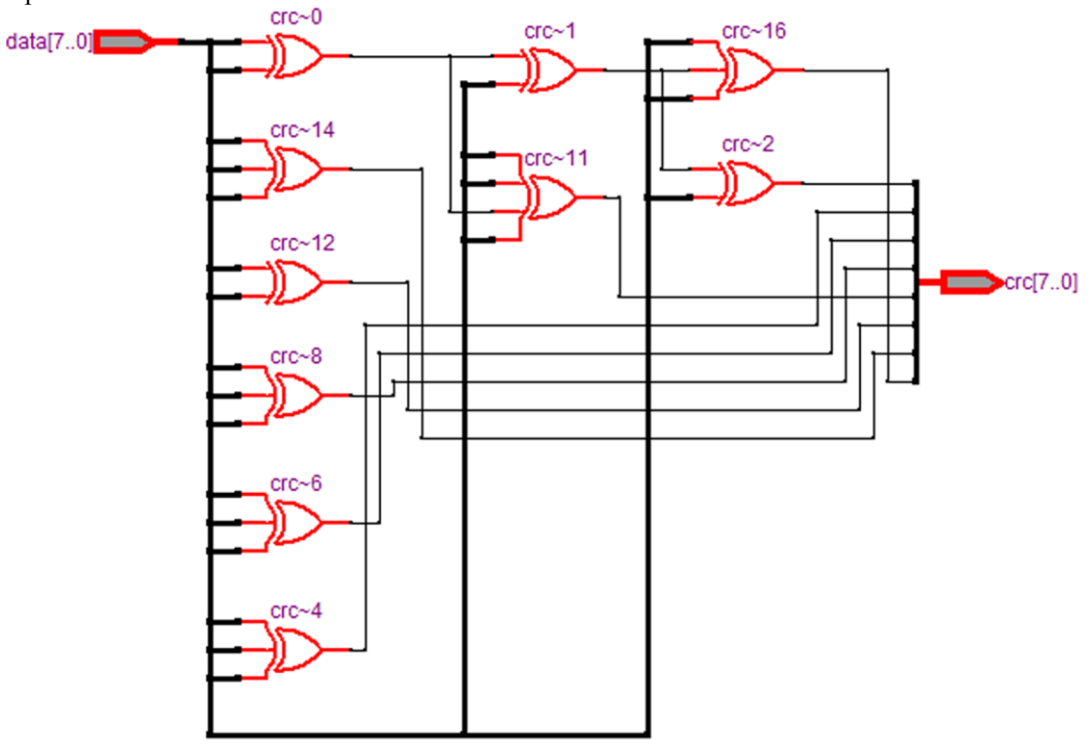

Figure 5. Hardware implementation of CRC8 matrix algorithm using «exclusive OR».

The combinational circuit for CRC32 matrix algorithm is built similarly, but the length of matrix values is increased to 32, which increases the number of logic elements «exclusive OR».

Developed combinational circuits are represented in functional blocks (figures 1, 3) and allow compute CRC for one data byte. CRC computation for data packet is produced by serial communication of functional blocks. Figure 6 shows the diagram of CRC8 matrix hardware implementation such as part of emergency protection system. Each next byte is summed by modulo 2 with computed CRC and result is supplied to next CRC8 block input.

Thus, asynchronous hardware implementation of matrix algorithm has an advantage over table-driven which is reflected in the simplicity of circuit, the absence in the circuit multiplexers and storage elements.

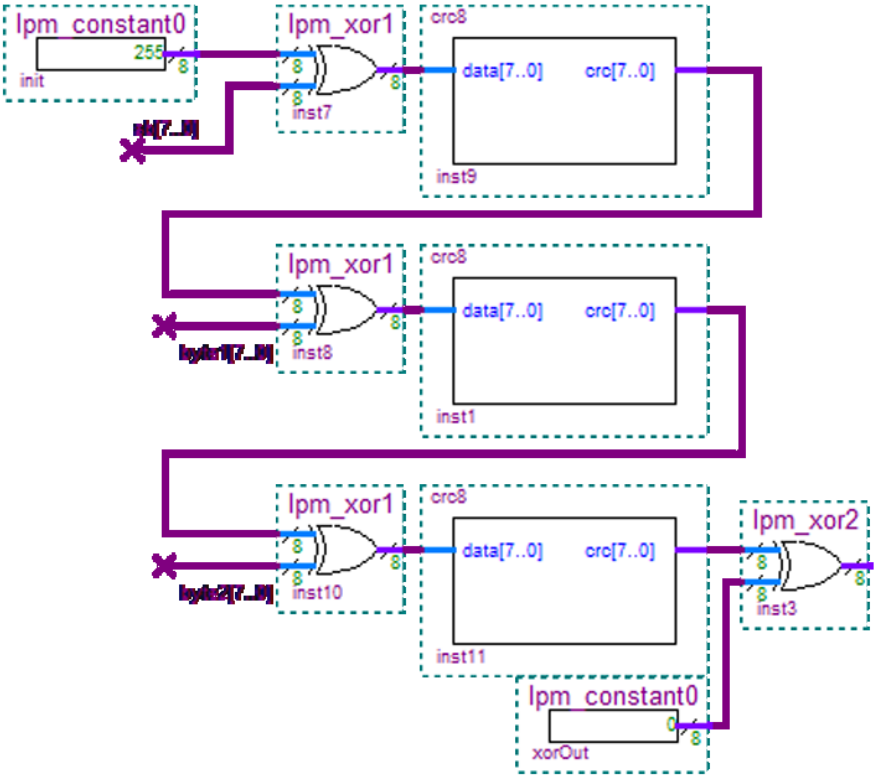

Figure 6. Functional diagram of CRC8 computation for 3 data bytes.

3 Testing of the hardware implementations

Simulation of CRC computation for FPGA Cyclone for different input data was conducted to ensure efficiency of developed devices. Table 1 presents parametric model for CRC8 computation [4]. Figure 7 shows results of CRC8 computation for table-driven and matrix-driven algorithm.

Table 1. Parametric model of CRC8.

Parameter	Value
Name	"CRC 8"
Width	8
Poly	0x9B
Init	0x00
RefIn	False
RefOut	False
XorOut	0x00
Check	0x25

In hardware implementation of CRC algorithms such as in software implementation the parametric model can be changed by changing *Poly, Init, RefIn, RefInOut, XorOut* parameters. However unlike a software implementation in hardware each parameter requires I/O FPGA pins which prevent parametric model control. Usually for a hardware implementation determined parametric model is selected for the device. In the developed devices only the *Init* parameter can be changed.

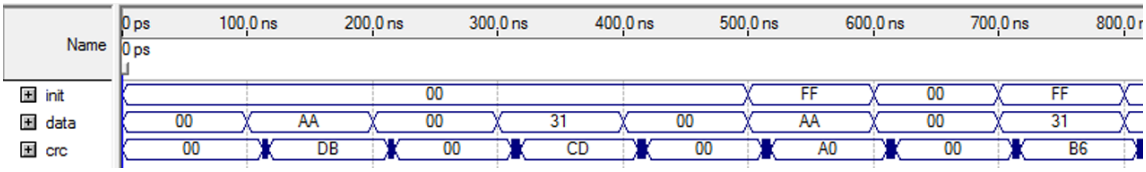

Figure 7. The results of CRC8 device testing.

Thus, values 0xAA and 0x31 are supplied to input *data* for testing device; values 0x00 and 0xFF are supplied to *init* input. Other values of parametric model do not change. Figure 8 shows the result of CRC8 computation by software for the same parametric model.

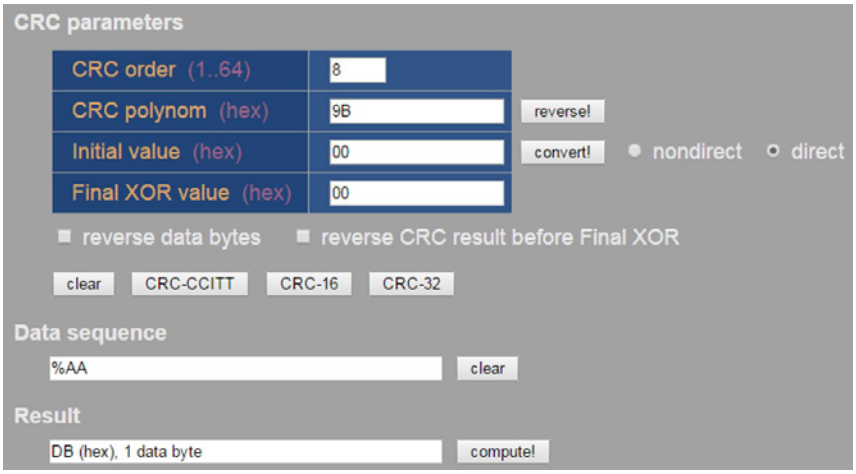

Figure 8. The result of CRC8 computation by software with the same parametric model.

Table 2 presents parametric model for CRC32 computation [4]. Figure 9 shows the results of CRC32 computation for table-driven and matrix-driven algorithms using the parametric model. Figure 10 shows the results of CRC32 computation by software with the same parametric model.

Table 2. Parametric model of CRC32.

Parameter	Value
Name	"CRC 32"
Width	32
Poly	0x04C11DB7
Init	0x00000000
RefIn	False
RefOut	False
XorOut	0x00000000
Check	0x89A1897F

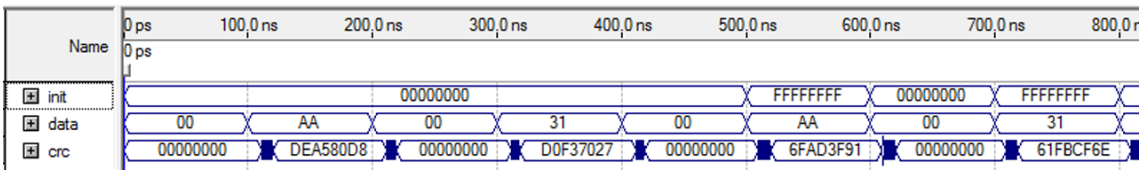

Figure 9. The results of CRC32 device testing.

Values 0xAA and 0x31 are supplied to input *data* for testing device; values 0x00000000 and 0xFFFFFFFF are supplied to *init* input. Other values of parametric model do not change.

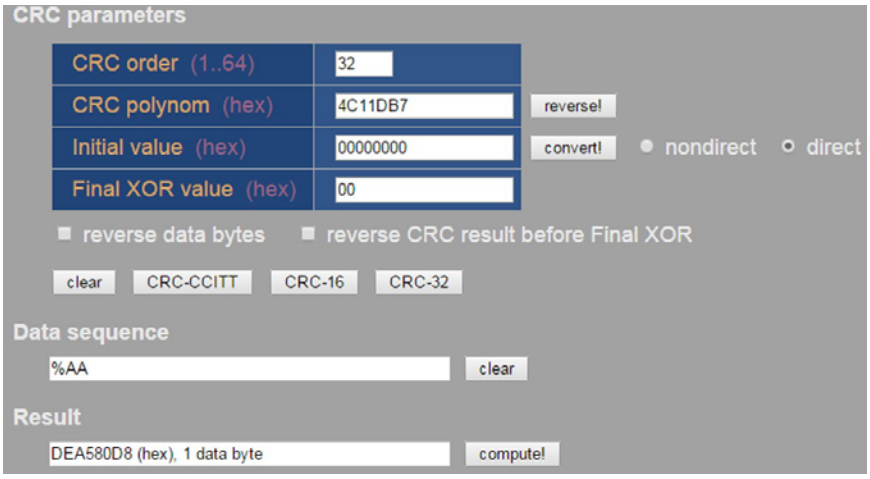

Figure 10. The result of CRC32 computation by software with the same parametric model.

4 Conclusion

The paper discussed hardware implementations of CRC computation algorithms, such as a table-driven which is widely used in data transmission protocols, also a matrix-driven, which allows significantly simplify combinational circuit of CRC computing device. CRC computing device based on FPGA Cyclone which can be used for the satellites configuration memory control and data transmissions modules was developed and tested following parametric model described by R. Williams [4]. Combinational circuit of hardware implementation of matrix-driven algorithm is simpler than table-driven and does not require any complex combinational devices such as multiplexor and storage elements for constant storing. CRC computation circuit in this case is built using only logic «exclusive OR» which gives advantage in hardware implementation. This fact allows embedding CRC computation device based on FPGA in satellites data transmission and configuration modules saving hardware resources for other important modules.

Acknowledgement

The work was performed under the Federal Program «Research and development in priority areas of science and technology Complex of Russia in 2014 - 2020 years» State contract № 14.578.21.0032.

References

1. M. Evgeniy, M. Andrey, *Proceedings of the 9th International Forum on Strategic Technology* (IFOST), 5 (2014)
2. H. Payal, K. Mankar, Intern. J. of Innov. Res. in Adv. Eng. (IJIRAE) **2**, 254 (2015)
3. S. Cherian, N. George, S. Enosh, S. Pillai, Intern. J. of Scien. and Res. Pub. (IJSRP) **4**, 1 (2014)
4. W. Ross, *A Painless Guide to CRC Error Detection Algorithms* (1993)
5. E. Mytsko, *Modern Techniques and Technologies: Proceedings of the 18th International Scientific and Practical Conference of Students, Post-graduates and Young Scientists*, 124 (2012)
6. A. Malchukov, A. Osokin, Y. Bourkatovskaya, *7th Korea/Russia International Symposium on Science and Technology* (KORUS), 189 (2003)