5x4Gbps 0.35 Micron CMOS CRC Generator Designed With Standard Cells

José María Nadal Serrano
José María Nadal Serrano is an undergraduate student at
ETSI Telecomunicación, Technical University of Madrid (UPM), Spain.
E-mail: chema@scouts-es.org

Abstract

Design highlights for a 32-bit parallel and highly pipelined Cyclic Redundancy Code (CRC) generator are presented. The design can handle 5 different channels at an input rate of 2Gbps each (the total output throughput is 5x4Gbps.) The generated CRCs are compatible with the 32-bit Ethernet standards. The circuit has been implemented with standard cells in a $0.35\mu m$ standard CMOS process using the properties of Galois Fields and has been conceived as a "free" IP.

Keywords Cyclic Redundancy Code (CRC), Galois Fields (GF), 10Gbps Ethernet, Free hardware, Intellectual Property (IP).

1. INTRODUCTION

Nowadays, digital IC designers are somehow enforced to use blocks of a higher level of abstraction when facing systems with higher complexity. This is the case of the so-called "Systems-on-chip," which use even complete subcircuits to build the final system. In this context new concepts such as Intellectual Property (IP) have arisen. The Cyclic Redundancy Code (CRC) presented here has been conceived as one of those IP blocks, not as a standalone circuit. Its VHDL code is released under the General Public License (GPL), enlarging the emerging group of freely available "IP-less" IC cores and therefore allowing anyone its use and free improvement. VHDL code and further documentation are available on email request.

Several suggestions to accelerate the generation of CRCs can be found in the literature (see [1], [2]) , but the only one that takes into account the advantages of binary finite fields such as Galois Fields in the generation of CRCs is [1] . Galois Fields have some properties which make them very advantageous to implement hardware solutions for arithmetic problems.

Further improvements to the previous work in the references are described in this paper. Parallelism and pipelining plus a wider word length are introduced, making our circuit achieve higher data throughputs as well as a five independent channel feature. The existence of different channels makes the circuit suitable for TDMA designs in which the Ethernet polynomial is used for the generation of the CRCs (ATM, 10Gbps Ethernet,...) Outputs are updated as data arrive.

Although just briefly commented in this paper, a low-level, "layout oriented" VHDL description style has been used as the most suitable way to achieve very high throughputs and avoid at the same time a full-custom design.

2. THE ALGORITHM

The CRCs have been traditionally calculated by shifting the incoming message into the MSB of a Linear Feedback Shift Register. This LFSR carries out a bit by bit multiplication in the Galois Field modulo the polynomial that generates the field. Divisions are then performed through shifting and feeding back into the LSFR, so that the result (the CRC) is the value of the register once the whole message has been processed. This approach is not suitable for high speed applications. The use of Galois Fields properties makes it possible to implement CRC generators for high-speed applications.

The polynomial used is the standard one for ATM and Ethernet:

$$G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8 +x^7+x^5+x^4+x^2+x+1$$

and can be seen as a vector of the $GF(2^{32})$:

$$B'10000010011000001000111011011 0111$$

This polynomial has several advantages: it is irreducible and has all the useful properties of the GF.

Using the main property of the Galois Fields, $\alpha^i \otimes \alpha^j = \alpha^{i+jmod(2^{32}-1)}$, word shifts (16 bits at a time) and the fact that the incoming data are also elements of the GF, the computation of the CRC will be reduced to (see [1] for a more detailed explanation:)

$$CRC(N+1) = CRC(N)\otimes\alpha^{16}\oplus Word(N+1) \quad [\mathbf{2.1}]$$

Where $CRC(N)$ is the output of the generator at a given moment, α^{16} is a vector of the Galois Field $GF(2^{32})$,

and $Word$ is the input word (16 bits.) The '$\otimes$' sign denotes a multiplication modulo the polynomial within the Galois Field.

3. HARDWARE IMPLEMENTATION

The main problem is the Galois Field multiplier which is the subcircuit that implements the '$CRC(N) \otimes \alpha^{16}$' operation. The so-called "H-matrix" (depicted in figure 1) is the mathematical representation of the core of the GF multiplier, and is the main part of the generator itself. The advantage of using the properties of the GFs is that the multiplier can be implemented with combinational logic (additions in GF are just XORs.) The H-matrix makes it possible to implement the multiplication modulo the polynomial as an addition of a certain depth, so a matrix of XOR gates will do the multiplication. We have converted the problem of generating CRCs from calculating the remainder of a division to some combinational logic using the properties of Galois Fields. The elements of the matrix are 1's and 0's as shown in figure 1. The places where a '1' is placed correspond to an XOR gate, while for the places occupied by 0's, a wire will be placed.

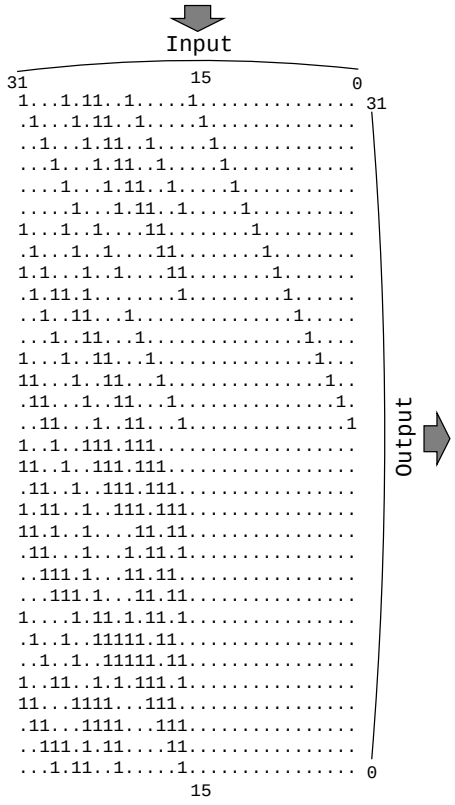


Fig. 3.1. H-matrix for 16 bit input and the Ethernet polynomial

If the result of a multiplication is expressed as a vector in the form '$Vector[0..31]$,' the structure of the multiplier must be obtained from the matrix as follows:

CRC[T+1][31] = CRC[T][31] $\oplus$ CRC[T][27] $\oplus$ CRC[T][25] $\oplus$ CRC[T][24] $\oplus$ CRC[T][21] $\oplus$ CRC[T][15].

CRC[T+1][30] = CRC[T][30] $\oplus$ CRC[T][26] $\oplus$ CRC[T][24] $\oplus$ CRC[T][23] $\oplus$ CRC[T][20] $\oplus$ CRC[T][14].

CRC[T+1][29] = ...

where '$\oplus$' denotes an addition within the GF and the indexes ('$[25], [26], ...$')can be read directly from the matrix (as shown before.) Once the multiplication is done, there is just the '$\oplus Word(N+1)$' part left. This consists of an array of XOR gates. The result is stored in an output register so that the feedback can be completed. The register is initially set to $H'46AF6449$ to be compliant with the Ethernet standard.

A sketch of the general circuit will give the reader an overall view of the circuit (depicted in figure 2.)

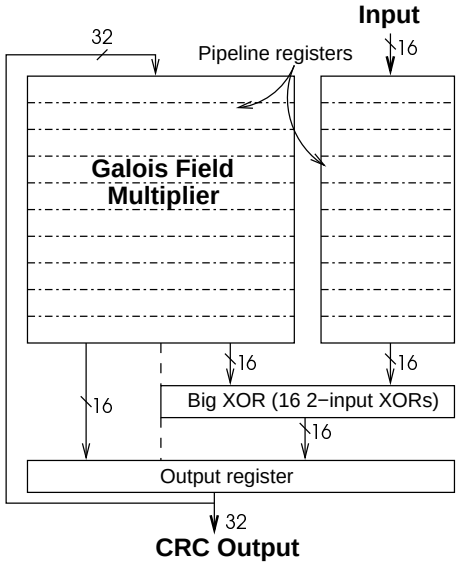


Fig. 3.2. General scheme of the CRC generator

Upon completion of the equations that lead to a straight forward implementation of the GF multiplier, we will see that the maximum depth of the logic is 10XOR gates (see $CRC[T+1][12]$.) The critical path is too long to achieve the 10Gbps aimed. The solution is the use of a highly pipelined architecture: the critical path will be reduced to 1XOR (each pipelining stage will consist on just one gate.) This way, race conditions are minimized, glitches are not a serious problem and the speed can be increased in theory up to 10Gbps ($16bits * 625MHz$, maximum safe clock speed with this architecture and a $0.35\mu m$ CMOS process.) The area is also increased because there will be a pipelining register for every layer of gates.

Examining the pipelined multiplier and the input we conclude that there will have to be some pipelininig in

the input as well. This will consist on the addition of registers which will just delay the input, but that are necessary in order to calculate the CRC correctly.

3.1. Timing: Two Phase Logic

Two-phase logic is used in order to be able to use pipelining, and therefore the parts of the circuit that will be active with φ_1 and φ_2 have to be carefully designed in order to avoid problems with the arrival and the delivery of the data (see figure 3.)

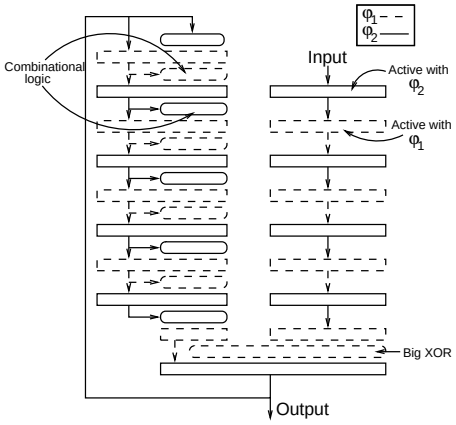


Fig. 3.3. Two phase structure of the circuit

Of particularly importance is the input and the output of the output register, because the output of the block of input registers and the output of the block of the GF multiplier have to be active within the same phase of the clock. This is important to make the CRC generator a "black box" that could be included in a larger circuit where it could be used without full understanding of how the IP works or the aid of external circuitry. The same timing constraint exists for the output of the 'output register' and the input of the 'input registers,' since the input from the feedback and the number of idle registers have to be delayed the same number of clock cycles in order to maintain synchronization of the data (the have to arrive at the "Big XOR" at the same time.) There was no need to add more idle registers to the input block than those shown in figure 3 and therefore the latency was not increased.

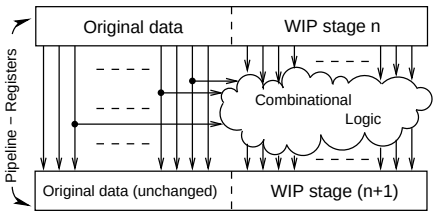


Fig. 3.4. General scheme of a pipelining stage of the GF multiplier

It is also interesting to have a look at the design of the pipelining registers: they are 16 bits long in the input block, 32 bits for the output register and 64 bits long in the GF multiplier block. This last length is needed due to the way the result of the multiplication is generated: both the WIP (the work in progress, this is, the result 'in progress') and the 'original data' -the factor that will be multiplied by α^{16}- are needed (see figure 4.) Since the 'original data' have to be available in all stages of the computation of the multiplication, 32 extra pipelining flip-flops have to be provided in each layer. The last register of the GF multiplier block is 32 bits wide because we only need the 32 bit WIP part (there are no more layers of XOR gates, so we no longer have to keep the 'original data' for the next stage.) The output of this last WIP register is the result of the whole multiplication.

The pipelining solution has some drawbacks, such as latency and 'idle' clock cycles. Those clock cycles come from the pipelined feedback: four clock cycles are needed to perform a multiplication, and the result is then XORed with the input word. This intermediate result is fedback and after four more clock cycles, the new result from the multiplication is ready to be XORed with the next input. Through the addition of pipelining we have lowered the input throughput from 10Gbps to $10/5 = 2Gbps$. However, this dataflow forms an 'independent channel,' this is, the data in a certain stage of the computation never get mixed with the data in the adjacent stages. Hence, it is possible to convert the 2Gbps CRC generator into a 5channelx2Gbps=10Gbps input throughput CRC generator again.

In order to have 5 independent channels the value of the output register must be equal for the first input word for each of the five channels because we have to make the CRC generator 'transparent' to the TDMA CRC generation -each channel has to see a CRCgenerator-. This problem can be solved easily by simply loading the registers of the multiplier with appropriate values other than zero.

Since the pipelining registers of the input will be set to '0' during reset and the '0' is the neutral element of the XOR operation, the output XOR can be thought of as a dummy operation for this case. Then, for the output register to be correct, the initial (reset) values are calculated as follows: the 32 MSB of the first pipelining register of the GF multiplier will be H'E3ED5B2A. This value is the 'GF division' of the initial value of the output register (H'46AF6449) by α^{16}. The 32 LSB of the first pipelining register are calculated applying the 32 MSB already calculated (H'E3ED5B2A) to the the first combinational layer. The result is H'68B932F5. The rest of the setup values are found by applying the combinational logic layer by layer to the first one (the results for each stage are in the table labeled as figure 5.)

Registers	Values 63..32 (HEX)	Values 31..0 (HEX)
GF2	68B932F5	E3ED5B2A
GF3	68B932F5	CEAD1918
GF4	68B932F5	90903DD8
GF5	68B932F5	74EBF27F
GF6	68B932F5	462A4987
GF7	68B932F5	46AFBDFF
GF8	68B932F5	46AF747D
GF9	68B932F5	46AF7449
GF10	68B932F5	46AF6449

Fig. 3.5. Table of initial values of the pipelining registers of the GF multiplicator

4. TECHNOLOGY ISSUES

The design shows a meet-in-the-middle design flow. A full custom aproximation was not desired because of time constraints and the lack of flexibility that such solutions have. On the other hand, the required specifications (the throughput of the overall system and the compatibility with the 32 bit Ethernet standards) pointed to a low-level or full custom design. The third design constraint was the technology, a standard $0.35\mu m$ CMOS process made the problem even more difficult.

The solution tried to balance all these problems and get a result that had both the benefits of a "physical design" and the convenience of the design flow of a VHDL description (easier to simulate and debug, automatic place and route...) A low-level, layout oriented description of the circuit appeared as the best solution. This "layout oriented" description tried to enforce the final layout by using a description style that seemed to be almost a netlist.

If other technologies are used, big improvements can be achieved, but the chip will have to be redesigned. In particular, the maximum clock speed available and the critical paths, determined by the properties (propagation delay and sensitivity to variations in the skew) of the "XOR gate + Flip-flop" chain will determine the number of XORs per pipelining stage and thus the latency and number of simultaneous channels.

5. RESULTS AND CONCLUSIONS

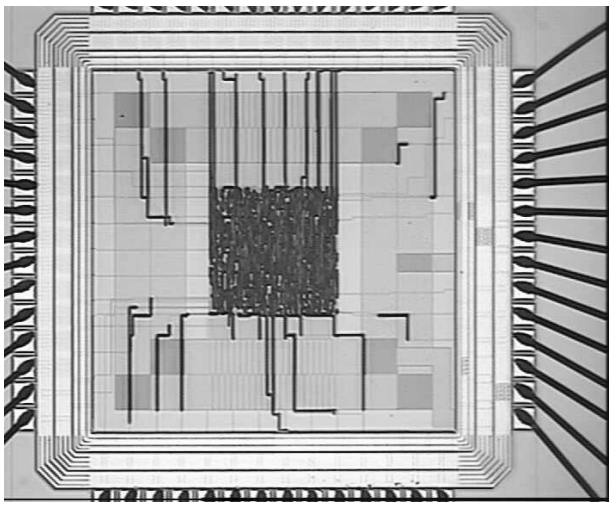
- Even if the circuit could not be tested due to the lack of equipment that could handle such high I/O rates -the probes and the pins themselves make it impossible-, the IP is designed to be used as part of electroptical devices and the I/O rate is therefore to be achieved in on-chip operation.) However, post-synthesis (post-layout simulations were not available) simulations report good performance even beyond the rates discussed above.

- Changes in the design for better technologies are relatively straightforward, while big improvements in the order of 2-3 times in area and speed could be achieved. The use of latches instead of flip-flops, for example, can reduce the amount of area the circuit needs. It could also be possible to use a 32-bit input instead of 16. However, the gain will be clearer for better process technologies since both area and clock speed can be reduced. With this technology, the area would have been larger while no increase in speed would have been achieved.

In any case, some changes will be needed and data from the foundry (XOR and flip-flop propagation times, skews,...) will be indispensable to match critical path, clock speed and number of channels. The concepts and the design flow will remain the same, nevertheless.

- The "low-level" VHDL description has proven to be an interesting option when trying to achive the best out off the "design time vs. high speed" trade-off; however, it may be impractical for big or non-critical designs.

Die Photo



Die photo of the circuit (pad-limited). The size of the core is about $500\mu m$x$500\mu m$

Acknowledgments

The author would like to acknowledge:

Prof. Peter Nilsson, Thomas Olsson and all the people in the ASIC design department at Lunds Tekniska Högskola, Lund, Sweden, for his support and patience during all the design process.

Prof. Mats Cederwall at Lunds Tekniska Högskola, Lund, Sweden for telling me about Galois Fields and their properties.

Mr. René J. Glaise at IBM CER labs, La Gaude, France for his helpful hints and clarifications.

Prof. Javier Macías-Guarasa, at ETSI Telecomunicación, UPM, Madrid, Spain for the opportunity he has given to me.

References

[1] R. J. Glaise, X. Jacquart, "Fast CRC Calculation", *Proc. IEEE*, ©IEEE, 1993.

[2] Peterson and Weldon, *Error correcting codes*, The MIT Press, 2nd edition, 1972.

[3] R.F.Hobson, K.L.Cheung, "A High Performance CMOS 32-bit Parallel CRC Engine", *IEEE Journal of Solid-State circuits*, vol. **40**, no. 2, Feb 1999.

[4] J.M. Rabaey, *Digital Integrated Circuits - A Design Perspective*, Prentice Hall, 1996.

[5] Peter J. Ashenden, *The Designer's Guide to VHDL* Morgan Kaufmann Publishers.