International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 8 Issue 1 –APRIL 2014.

Design of Field Reprogrammable Cyclic Redundancy Check technique using Multibit Flipflops

SATHYA.R[1]
Asst Professor

Department of Electronics and Communication Engineering[1],
Adithya Institute of Technology, Coimbatore, Tamil Nadu, India.

Abstract:* In modern integrated circuits, the power consumed by clocking gradually takes a dominant part. Given a design of Field Reprogrammable CRC which is used to detect and correct the errors in system on chips(SOCs), its power consumption can be reduced by replacing some flip-flops with fewer multi bit flip-flops. However, this procedure may affect the performance of the original circuit. Hence, the flip-flop replacement without timing and placement capacity constraints violation becomes a quite complex problem. Deal with the difficulty efficiently, several techniques are proposed. In this paper a co-ordinate transformation is performed to identify those flip flops that can be merged and their legal regions. The architecture has been designed to be field programmable so that it is fully flexible in terms of polynomial deployed and input port width. In comparison with previous works, the clock cycles and area will be reduced drastically because of removing precomputation matrix and this technique have been modified for N number of polynomials with multi bit flipflops.

***Index terms-* Cyclic redundancy check (CRC), error detection, field programmable, multi bit flipflops.**

I. INTRODUCTION

As technology advances, a systems-on-chip (SOC) design can contain more and more components that lead to a higher power density. This makes power dissipation reach the limits of what packaging, cooling or other infra structure can support. Reducing the power consumption not only can enhance battery life but also can avoid the overheating problem, which would increase the difficulty of packaging or cooling . Therefore, the consideration of power consumption in complex SOCs has become a big challenge to designers. In that Cyclic redundancy check method is used to detect and correct the errors.

In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability whether or not an error occurred in transmission. If an error occurred, the receiver sends a “negative acknowledgement” (NAK) back to the sender, requesting that the message be retransmitted.

Given a design that the locations of the cells have been determined, the power consumed by clocking can be reduced further by replacing several flip-flops with multi-bit flip-flops. During clock tree synthesis, less number of flip-flops means less number of clock sinks. Thus, the resulting clock network would have smaller power consumption and uses less routing resource.

This paper explores the architecture and functions of the field programmable CRC computation circuit and analyses its performance when implemented using standard ASIC technology with multibit flipflops.

II. CYCLIC REDUNDANCY CHECK

The CRC method treats the message as a polynomial in GF(2). The sender and receiver agree on a certain fixed polynomial called the generator poly- nomial. To compute an r-bit CRC checksum, the generator polynomial must be of degree r.

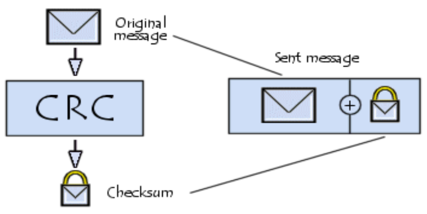


Fig 1 .CRC Process

The sender appends r 0-bits to the m-bit message and divides the resulting polynomial of degree m + r – 1 by the generator polynomial. This produces a remainder polynomial of degree r – 1 (or less) [3]. The remainder polynomial has r coefficients, which are the checksum. The quotient polynomial is discarded. The data transmitted (the code vector) is the original m-bit message followed by the r- bit checksum. In this shifting operation can be done by using Linear Feedback Shift register LFSR. For example,

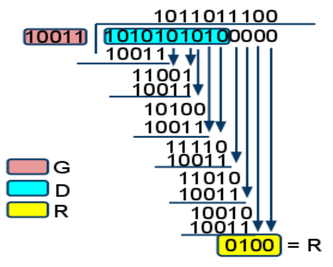


Fig 2.Calculation of CRC

III. MULTI BIT FLIPFLOPS IN FIELD PROGRAMMABLE ARCHITECTURE

In Flip flops,The driving capability of a clock buffer can be evaluated by the number of minimum-sized inverters that it can drive on a given rising or falling time .Because of this phenomenon, several flip-flops can share a common clock buffer to avoid unnecessary power waste.

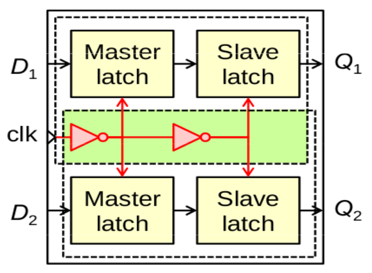


Fig. 3 Merged 2 Bit Flipflop

Fig. 3 shows the block diagram of 2-bit flip-flop. If we replace the two 1-bit flip-flops, the total power consumption can be reduced as shown in Table.1 because the two 1-bit flip-flops can share the same clock buffer [7].

Table.1 Comparison of Various bits

Bit number	**1**	**2**	**4**
Normalized power per bit	1	0.86	0.780
Normalized area per bit	1	0.96	0.713

IV. TRANSFORMATION AND PLACEMENT

The pseudo code to reduce the no of Flipflops is shown in algorithm.T is Existing flip-flops can be transformed and placed to form the several types of new flipflops according to rules of an algorithm[5]. Suppose we want to form a flip-flop in n4, which needs two 1-bit flip-flops according to the combination table. Each pair of flip-flops in n1 are selected and checked to see if they can be combined. If there are several possible choices, the pair with the smallest cost value is chosen to break the tie. In f1 and f2 are chosen because their combination gains the smallest cost.

```
Algorithm :
// Initialization
1. lexicographically sort the MBFF library
2. collapse MBFFs
3. X' <--sort {sx'(i), ex'(i): i = 1..n}, j □ 1, Q □ □
// Main body
4. while (X' is not empty) do
5. find a decision point in X'
6. Q <--Q + essential flip-flops and related flip-flops
7. Y' <-- sort {sy'(i), ey'(i): i □ Q}
8. foreach essential flip-flop k do
// Flip-flop clustering
9. Kmax <--max_clique(Y', k)
10. find the appropriate MBFF cell of bit number B for |Kmax|
11. Kmax <--sort {ex'(i): i □ Kmax - {k}}
12. Kj <--flip-flop k and the first (B-1) flip-flops in Kmax
// Flip-flop placement  p pp
13. find bounding box Bb for Kj
14. project Bb's corner and center points to Fr(Kj)
15. find the projected point with min distance between Bb and Fr(Kj)
16. legalize this point and assign it to MBFF Kj
17.if legalization fails then go to line 9
18.Q <--Q - Kj, X' | X' - Kj
19. j++
```

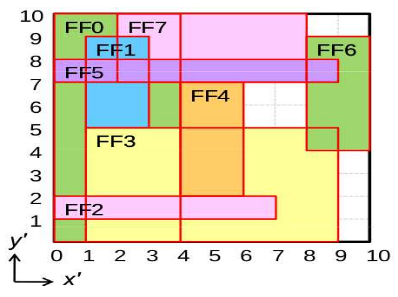
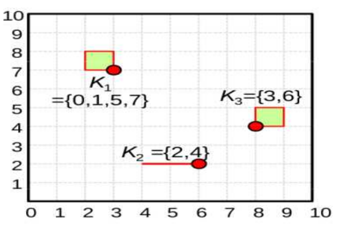


Fig. 4 Overlapping of Flipflops

Thus, add a new node k1 in the figure below with a combination of the regions ff0,ff1,ff5,ff7 which gives 4 bit mergable flipflop. Similarly, ff3 and ff6 are combined to obtain a new flip- flop k3 to produce 2 bit flip flop.After all flip-flops in the combinations of 1-level trees (n4 and n5) are obtained. Start to form the flip-flops in the combinations of 2-level trees (n6, and n7)[8]. Suppose there is no overlap region between the couple of flip-flops it fails to form a 4-bit flip-flop in k1. if the 2-bit flip-flops k2 and k3 are mergeable, they can combine them to obtain a 4-bit flip-flop. Finally, because there exists no couple of flip- flops that can be combined further, the procedure finishes.

Fig. 5 Placement of Multibit Flipflops



IV. PROPOSED TECHNIQUE AND ARCHITECTURE

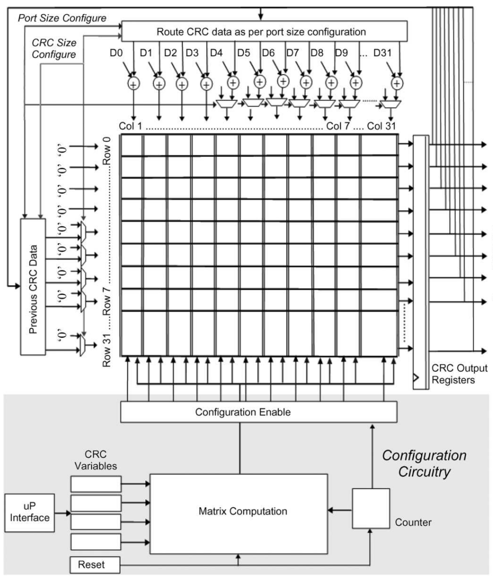


Fig. 6. Field programmable CRC architecture.

When the CRC cells are fully configured, the configuration processor is not used. The port size configure and CRC size configure signals control a set of multiplexers that enable/disable input and CRC feedback data to cater for the size of CRC polynomial and size of the input port[4]. The port size configure is also responsible for reconfiguring the circuit to process various input word sizes, e.g., if the port size is configured as 32-bit and the last cycle of the payload data contains only 16-bits to be processed, the port size configure signal; switches input bits 16 to 31 to the “0” input of each multiplexer; switches the bottom 16 multiplexers to the previous CRC data input at the left-hand side of the array and finally routes bits 16 to 31 of previous CRC data to rows 16 to 31 of the array.

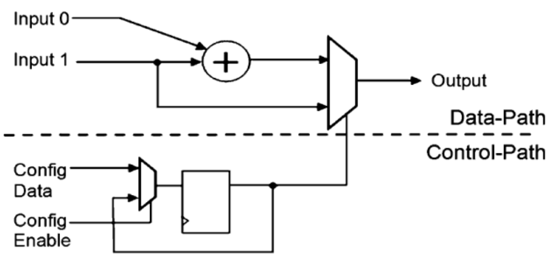


Fig. 7 Programmable CRC array cell.

The configurable XOR array is comprised of interconnected cells, corresponding to the D matrix. Each cell can be configured as an XOR gate or as a basic input to output connection.

The Fig.6 shows the diagram of one of the field programmable elements that facilitate this function in the array. The data path contains a configuration register which selects the data-path function and is programmed via the Config Data input when the Config Enable input is set high. The computation of D-matrix is an iterative process, where the computation of each row is based on the result from the previous row. Therefore, one row is computed every clock cycle and configuring the whole matrix requires 33 clock cycles. This is the minimum time possible due to the feedback required. The Fig.8 shows the logic used to compute a D matrix row in one clock cycle for an example 4-bit CRC polynomial, using the T matrix data and the current D matrix row signal**.**

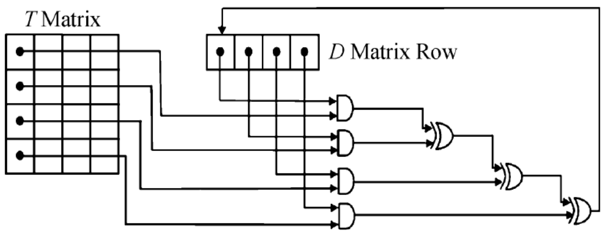


Fig. 8. D matrix row calculation first loop.

The diagram shows how a single bit of D matrix is computed on the first loop using the rows and columns shaded in the D and T matrices. For CRC-64 or higher, two or more of the data-path circuits are combined in a diagonal cascade with additional back wiring, so that column and row 0 of the second circuit become column and row 32 of the CRC-64.

A) Modified Linear feedback shift register:

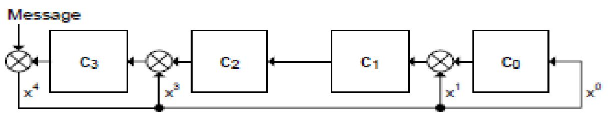


Fig.9.Modified Linear feedback shift register.

The circuit can be modified according to Fig. 9, where the message is combined with the most significant register bit to form the feedback. An advantage arises as no zeros need to be shifted in at the end, and thus the CRC can be obtained m clock cycles earlier. Following Campobello et al. [6] this circuit is referred to as LFSR2 in contrast to the first version, which is referred to simply as LFSR.

B) Merged flip-flop:

By using updated technique, the clock power and area is reduced when compare to the existing technique where the clock pulses are sepersted and given to each flipflop.The capacitance for the entire circuit is also reduced in the given diagram is 3C by avoding the switching operations and wire length is also drastically reduced .

.

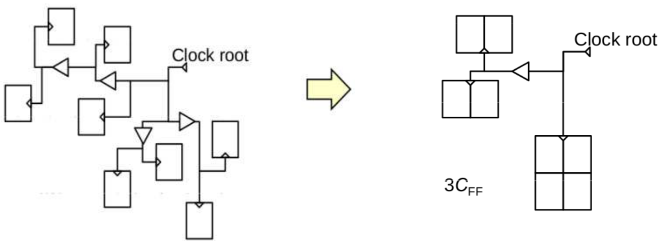
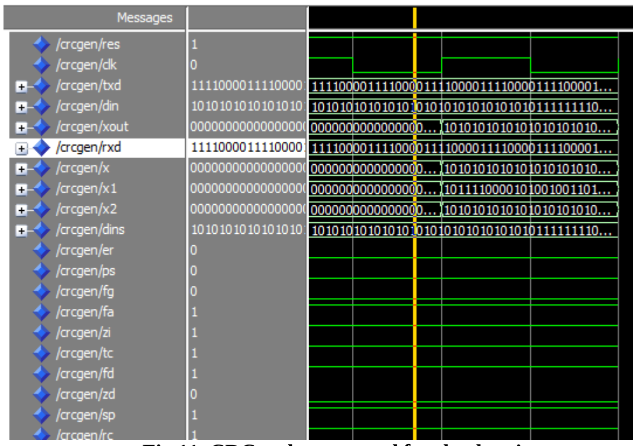


Fig.10. Merged clock path for the data input

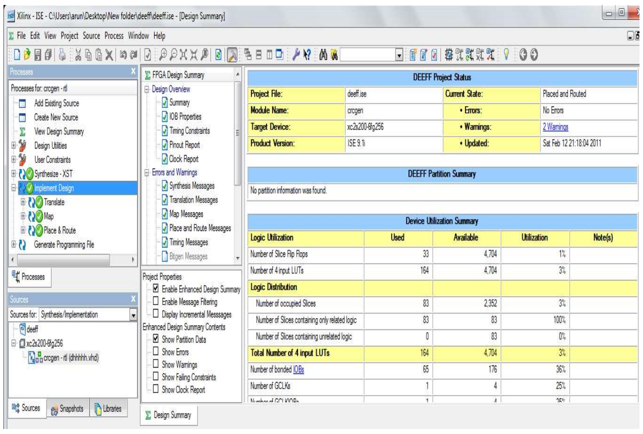
V. SIMULATION AND SYNTHESIS RESULTS

The architecture has been described in VHDL and it is simulated using modelsim and the design is synthesized for 32x32 cell array using Xilinx tool.



Messages		
/crcgen/res	1	
/crcgen/clk	0	
/crcgen/txd	1111000011110000	11110000111100001111000011110000111100001...
/crcgen/din	1010101010101010	1010101010101010101010101010101011111110...
/crcgen/xout	0000000000000000	0000000000000000... 1010101010101010101010...
/crcgen/rxd	1111000011110000	11110000111100001111000011110000111100001...
/crcgen/x	0000000000000000	0000000000000000... 1010101010101010101010...
/crcgen/x1	0000000000000000	0000000000000000... 1011110000101001001101...
/crcgen/x2	0000000000000000	0000000000000000... 1010101010101010101010...
/crcgen/dins	1010101010101010	1010101010101010101010101010101011111110...
/crcgen/er	0	
/crcgen/ps	0	
/crcgen/fg	0	
/crcgen/fa	1	
/crcgen/zi	1	
/crcgen/tc	1	
/crcgen/fd	1	
/crcgen/zd	0	
/crcgen/sp	1	
/crcgen/rc	1	

Fig.11. CRC code generated for the data input.



Xilinx - ISE - C:\Users\arun\Desktop\New folder\deeff\deeff.ise - [Design Summary]

DEEFF Project Status			
Project File:	deeff.ise	Current State:	Placed and Routed
Module Name:	crcgen	• Errors:	No Errors
Target Device:	xc2s200-6fg256	• Warnings:	2 Warnings
Product Version:	ISE 9.1i	• Updated:	Sat Feb 12 21:18:04 2011

DEEFF Partition Summary

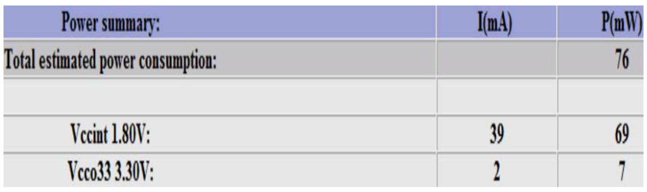
No partition information was found.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	33	4,704	1%	
Number of 4 input LUTs	164	4,704	3%	
Logic Distribution				
Number of occupied Slices	83	2,352	3%	
Number of Slices containing only related logic	83	83	100%	
Number of Slices containing unrelated logic	0	83	0%	
Total Number of 4 input LUTs	164	4,704	3%	
Number of bonded IOBs	65	176	36%	
Number of GCLKs	1	4	25%	
Number of GCLKIOBs	1	4	25%	

Fig.12.synthesize report of CRC architecture

The Fig.11 shows the simulation result for the 32-bit CRC code generated for the given data input. It is obtained by performing the modulo 2 division of data input by the 32-bit generator polynomial. [4]The Fig.12 shows the synthesize report of the CRC architecture which reveals that the CRC architecture design has been designed and implemented successfully.

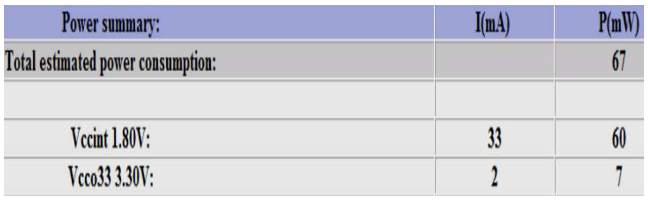
Table.2 Power analysis of existing technique



Power summary:	I(mA)	P(mW)
Total estimated power consumption:		76
Vccint 1.80V:	39	69
Vcco33 3.30V:	2	7

Table.2 shows the Power analysis of existing technique which reveals the amount of power consumed by the design of existing CRC architecture.

Table.3 Power analysis of proposed technique



Power summary:	I(mA)	P(mW)
Total estimated power consumption:		67
Vccint 1.80V:	33	60
Vcco33 3.30V:	2	7

Table.3 reveals the amount of power consumed by the design of proposed CRC architecture. Further, the CRC architecture has to be designed and implemented by using merged flip-flop in the CRC array cell design and has to compare their power analysis results which is shown in Table 4 and using chart.

Table.4 Comparison of Power

PARAMETERS	EXISTINGFLIPFLOP	MERGED FLIP-FLOP
POWER	76	67

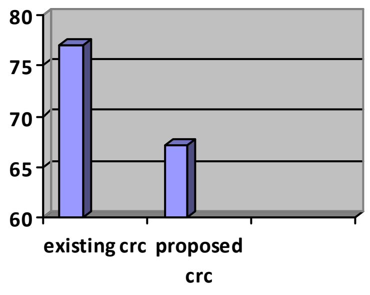


Fig.13.graphical comparison of power in milli watt

IV.CONCLUSION

Thus the result shows the design and implementation of CRC architecture by using the Merged multibit flip flop in CRC array cell and the power consumed by that design. It is expected that the power might consumption have been reduced by using multibit flip flop since it reduces its operating frequency to half and the number of clock cycles required for data transitions gets reduced with area minimization . The circuit uses an embedded configuration controller which enables the different CRC polynomials and I/O port and processing data widths to be deployed.

REFERENCES

[1] Martin Grymel and Steve B.Furber,"A Novel Programmable Parallel CRC Circuit" IEEE transactions on very large scale integration (vlsi) systems, vol. 19, no. 10, october 2011.

[2] W. Hou, D. Liu, and P.-H. Ho, "Automatic register banking for low-power clock trees," in Proc. Quality Electron. Design, San Jose, CA, Mar. 2009, pp. 647–652.

[3]F. Monteiro, A. Dandache, A. M'sir, and B. Lepley, "A fast CRC implementation on FPGA using a pipelined architecture for the polynomial division," in Proc. ICECS, 2001, vol. 3, pp. 1231-1234.

[4]Ciaran Toa and XinYang ,"Design and implementation of field programmable crc circuit architecture"IEEE transactipns on very large scale integration systems,vol.17,no.8,aug 2009.

[5] Y. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang, "Power-aware placement," in Proc. Design Autom. Conf., Jun. 2005, pp. 795–800.

[6] P. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R. L. Allmon, "High-performance microprocessor design," IEEE J. Solid-State Circuits, vol. 33, no. 5, pp. 676–686, May 1998.

[7] D. Duarte, V. Narayanan, and M. J. Irwin, "Impact of technology scaling in the clock power," in Proc. IEEE VLSI Comput. Soc. Annu. Symp., Pittsburgh, PA, Apr. 2002, pp. 52–57.

[8] H. Kawagachi and T. Sakurai, "A reduced clock-swing flip-flop (RCSFF) for 63% clock power reduction," in VLSI Circuits Dig. Tech. Papers Symp., Jun. 1997, pp. 97–98.

[9] P. Koopman, "32-bit cyclic redundancy codes for internet applications," in proc. DSN, pp. 459-472.

[10] T. Bi-Pei and C. Zukowski, "High-speed parallel CRC circuits in VLSI," IEEE Trans. Commun., vol. 40, no. 4, pp. 653-657, Apr. 1992.

[11] Weidong Lu, "Designing TCP/IP functions in FPGA," from MSc THESIS, computer engineering, Mekelweg 4, 2003.

[12] G. Campobello, M. Russo, and G. Patane, "Parallel CRC realization," IEEE Trans. Comput., vol. 52, no. 10, pp. 1312-1319, Oct. 2003.

[13] M. Sprachmann, "Automatic generation of parallel CRC circuits," IEEE Des. Test Comput., vol. 18, no. 3, pp. 108-114, May/Jun. 2001.

[14] P. Koopman and T. Charkravarty, "Cyclic Redundancy Code (CRC) polynomial selection for embedded networks," in Proc. DSN, pp. 145-154.

Sathya.R received the B.E. degree in electronics and communication engineering from Thanthai Periyar Govt Institute of Technology, Vellore in 2009. She completed M.E. under the specialization of VLSI Design in Govt College of Technology, Coimbatore, India in the year 2006. Currently working as Assistant Professor in the department of ECE in Adithya Institute of Technology. Her research interest includes VLSI Design and Embedded Systems.