ISSN:0975-9646

On the Design of High Speed Parallel CRC Circuits using DSP Algorithams

[1] B.Naresh Reddy, [2] B.Kiran Kumar, [3] K.Mohini sirisha

[1]*Dept.of ECE ,Kodada institute of Technology & Science for women,kodada,india*
[2,3] *Dept.of ECE ,GIST,Jagayyapeta,india* *

***Abstract* — Error correction codes provide a mean to detect and correct errors introduced by the transmission channel. Basically there are two categories of codes a).Block codes and b).convolution codes. Both the codes introduce redundancy by adding parity symbols to the message data. Cyclic redundancy check (CRC) codes are the subset of the cyclic codes.**
The hardware implementation of a CRC is a simple linear feedback shift register. LFSR circuit is simple and can runs at very high clock speeds, but it suffers from the limitation that the stream must be of bit-serial. CRC architectures for the generator polynomial are developed using DSP algorithms such as pipelining, unfolding and retiming. CRC architectures are first pipelined to reduce the iteration bound by using novel look-ahead pipelining methods and then unfolded and retimed to design high-speed parallel circuits. High-Speed parallel CRC increases the speed or throughput rate up to 25% when compared to the other techniques and reduce the hardware cost.

***Keywords*—** Cyclic redundancy check (CRC), linear feedback shift register (LFSR), pipelining, retiming, unfolding.

I. INTRODUCTION

Error correction codes provide a mean to detect and correct errors introduced by the transmission channel. CRC is a very powerful and easily implemented technique to obtain data reliability. Even if error correcting codes exists, their use is limited, like when the channel is simplex, where retransmissions cannot be requested. Most often error detection followed by retransmission is preferred because it is more efficient. The CRC technique is used to verify the integrity of blocks of data called Frames. Using this technique, the transmitter appends an extra n bit sequence to every frame called Frame Check Sequence (FCS). FCS holds redundant information about the frame that helps the receiver detect errors in the frame.

Cyclic redundancy check (CRC) is widely used to detect errors in data communication and storage devices. When high-speed data transmission is required, the general serial implementation cannot meet the speed requirement. Since parallel processing is a very efficient way to increase the throughput rate, parallel CRC implementations have been discussed extensively in the past decade [1], [2]. Although parallel processing increases the number of message bits that can be processed in one clock cycle, it can also lead to a long critical path (CP); thus, the increase of throughput rate that is achieved by parallel processing will be reduced by the decrease of circuit speed. Another issue is the increase of hardware cost caused by parallel processing, which needs to be controlled. This brief addresses these two issues of parallel CRC implementations.CRC architectures for the generator polynomial are developed using DSP algorithms such as pipelining, unfolding and retiming. The architectures are first pipelined to reduce the iteration bound by using novel look-ahead techniques and then unfolded and retimed to design high speed parallel circuits.

A. *Existing System*

In the past recursive formulas have been developed for parallel CRC hardware computation based on mathematical deduction. They have identical CPs. The parallel CRC algorithm in [2] processes an m-bit message in (m+k)/L clock cycles, where k is the order of the generator polynomial and L is the level of parallelism. However, in [1], m message bits can be processed in m/L clock cycles. High-speed architectures for parallel long Bose–Chaudhuri– Hocquenghen (BCH) encoders in [3] and [4], which are based on the multiplication and division computations on generator polynomial, are efficient in terms of speeding up the parallel linear feedback shift register (LFSR) structures. They can also be generally used for the LFSR of any generator polynomial. However, their hardware cost is high.

B.Proposed System

The proposed design starts from LFSR, which is generally used for serial CRC. An unfolding algorithm is used to realize parallel processing. However, direct application of unfolding may lead to a parallel CRC circuit with long iteration bound, which is the lowest achievable CP. Two novel look-ahead pipelining methods are developed to reduce the iteration bound of the original serial LFSR CRC structures; then, unfolding algorithm is applied to obtain a parallel CRC structure with low iteration bound. The retiming algorithm is then applied to obtain the achievable lowest CP.

II. DESIGN OF ARCHITECTURES USING DSP TECHNIQUES

A.Algorithm for unfolding:

1. For each node U in the original DFG, draw J node U0, U1, U2,…. UJ-1.
2. For each edge U →V with w delays in the original DFG, draw the J edges

$$U_i \rightarrow V(i+w) \,\%\, J \qquad \text{with} \quad (\; floor(i+w)/J \;)$$

delays for i = 0, 1, …, J-1.

Consider the CRC architecture with the generator polynomial , which is shown in Fig. 1. After applying an unfolding algorithm [5] with unfolding factors and , we obtain the two-parallel and three-parallel architectures shown in Figs. 3 and 4, respectively.

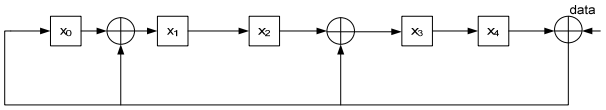


Fig 1. CRC Architecture For $G(y)=1+y+y^3+y^5$

Assume the Xor gates as 1, 2 and 3 nodes

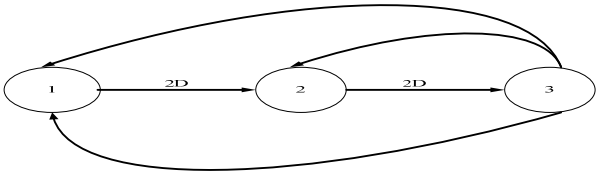


Fig 2.Data flow graph of Figure 1

After applying the unfolding technique with unfolding factor J=2 two-parallel architecture are obtained. To unfold the above data flow graph by an unfolding factor 2, the 6 nodes1, 2, 3, 4, 5, 6 are first drawn according to the first step of the unfolding algorithm. After these nodes have been drawn the 2nd step of the unfolding algorithm needs to be performed. For an edge U→V with no delays this step reduces to drawing the J edges $U_i \rightarrow V_i$ with no delays for i=0,1,…J-1.For example, edge with 3→1 with no delay in figure 3 results 3→1 and 6→4 with no delays in the 2-unfolded data flow graph in figure 3.3. For the edge 1→2 with w=2 delays in figure 3.2 draw the edges $1\rightarrow 2_{(2+0)\%2}$ with floor((2+0)/2) delays and $1\rightarrow 2_{(2+1)\%2}$ with floor((2+1)/2) delays which correspond to the edges 1→2 with 1 delay and 4→5 with 1 delay respectively in figure 3.Iteration bound is defined as the maximum of all the loop bounds. Loop bound is defined as t/w, where t is the computation time of the loop and w is the number of delay elements in the loop [5].

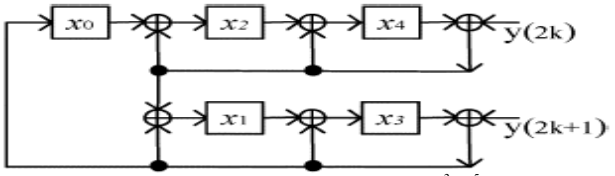


Fig 3. Two parallel CRC Architecture For $G(y)=1+y+y^3+y^5$

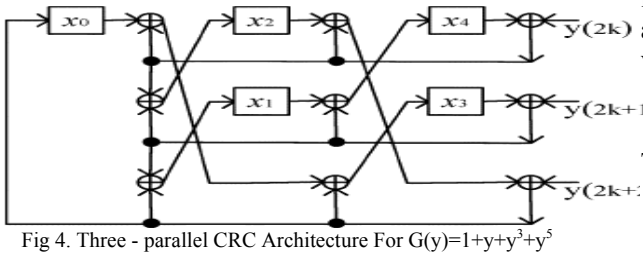


Fig 4. Three - parallel CRC Architecture For $G(y)=1+y+y^3+y^5$

It is obvious that the iteration bounds for the CRC architectures in Figs. 1–4 are T_{XOR}, $2T_{XOR}$ and , $3T_{XOR}$ respectively, where T_{XOR} is the computation time of an XOR gate. In this case, the iteration bound of a J-parallel CRC architecture for $G(y)=1+y+y^3+y^5$ is JT_{XOR}. Although retiming is used to reduce the critical path of a circuit, but cannot achieve a CP with a computation time that is less than the iteration bound of this circuit. In other words, the CP of a parallel CRC architecture cannot be less than J.T∞, where J is the level of parallelism and T∞ is the iteration period bound of the original data flow graph. Therefore, it is very important to reduce the iteration bound before the unfolding algorithm is applied.

B.Algorithm for Pipelining:

Effective critical path is reduced by introducing pipelining latches along the critical data path either to increase the clock frequency or sample speed or to reduce power consumption at the same speed. It is done using a look-ahead pipelining algorithm to reduce the iteration bound of the CRC architecture

Consider the CRC circuit with generator polynomial $1+y+y^8+y^9$ .Its iteration bound is 2Txor which corresponds to the section in the dashed squre and the term y^8+y^9 in the generator polynomial.

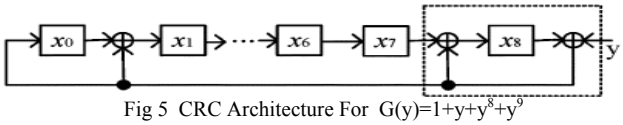


Fig 5 CRC Architecture For $G(y)=1+y+y^8+y^9$

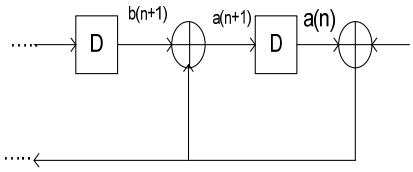


Fig 6. Loop bound of 2Txor

The largest iteration bound of a general serial architecture is also 2Txor. For example, the serial architectures of commonly used generator polynomials CRC-16 and CRC-12 have the iteration bound of 2Txor because they have terms $y^{15}+y^{16}$ and $y^{11}+y^{12}$ in their generator polynomials respectively. The critical Loop with the delayed input redrawn in figure 6 is described by

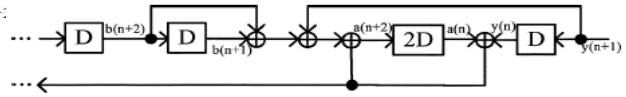
$$a(n+1) = a(n) + y(n) + b(n+1) \qquad (1)$$

There are two types of pipelining. 1) Proposed look ahead pipelining 2). Improved look ahead pipelining.

1).Proposed look ahead pipelining: In this section, we propose a look-ahead pipelining algorithm to reduce the iteration bound of the CRC architecture. If we apply Look ahead pipelining to (1) we can obtain a two level pipelined version given by

$$a(n+2) = a(n+1) + y(n+1) + b(n+2)$$
$$a(n+2) = a(n) + y(n) + b(n+1) + b(n+2) + y(n+1) \qquad (2)$$

The architecture for (2) is shown in figure 7



Fig(7).Pipelined Loop with a Loop bound T_{XOR}

In Fig. 7, we can see that the loop bound in Fig. 5 has been reduced from $2T_{XOR}$ to T_{XOR} at the cost of two XOR gates and two flip-flops. The CRC architecture in Fig. 5 can now be pipelined as shown in Fig. 8.

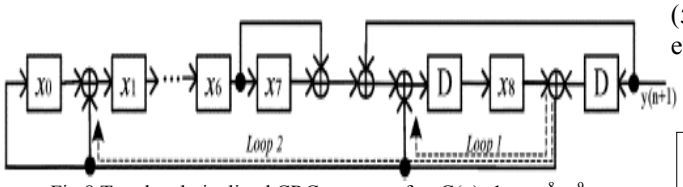


Fig 8.Two level pipelined CRC structure for $G(y)=1+y+y^8+y^9$

The above figure 8 shows two level pipelined CRC circuit with generator polynomial $1+y+y^8+y^9$. The loop bound in figure 7 is reduced from 2Txor to Txor at the cost of two XOR gates and two flip flops. In figure 8 the loop bounds of loop1 and loop2 are Txor and 5/8 Txor respectively. So, the iteration bound of the two level pipelined CRC architecture is Txor.

2).Improved look ahead pipelining: Consider the CRC architecture for polynomial $G(y)=1+y+y^7+y^9$.

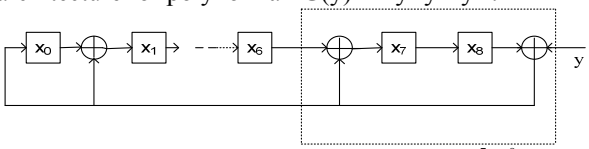


Fig 9. CRC Architecture for $G(y)=1+y+y^7+y^9$

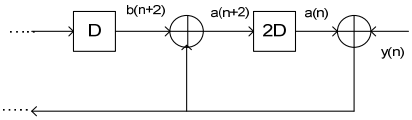


Fig 10. Loop bound of Txor

Loop 1 in Fig. 8 can be represented as shown in Fig.10 Consider the inner loop marked by the dashed square in Fig. 9. This loop can be redrawn as shown in Fig. 10 and can be represented by

$$a(n+2) = a(n) + y(n) + b(n+2) \qquad (3)$$

If we Apply improved look ahead pipelining technique to (3) we can obtain the four level pipelined structures.

$$a(n+4) = a(n+2) + y(n+2) + b(n+4)$$

$$a(n+4) = a(n) + y(n) + b(n+2) + y(n+2) + b(n+4)$$

(4) The architecture for (4) is shown in figure 11.In figure 11 we can see that four level pipelining can be achieved at the cost of two Xor gates and two flip-flops if the there are two delay elements in the initial loop.However, in the proposed pipelining method 1,four level pipelining can be achieved at the cost of four Xor gates and four flip-flops.

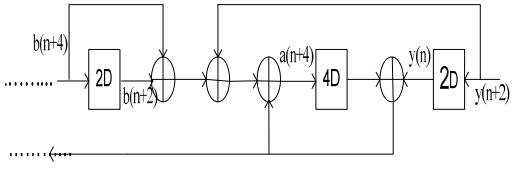


Fig 11. Pipelined Loop wih a loop bound of ($\frac{1}{2}$) T_{XOR}

The CRC architecture in figure 9 can be pipelined as shown in figure 12.In figure 12 we can see that the loop bounds of loop 1 and loop 2 are (1/2) T_{XOR} and (5/8) T_{XOR} , respectively So the iteration bound of four-level pipelined CRC architecture is (5/8)T_{XOR}. Use of four level pipelining reduces the iteration bound of figure 9 from T_{XOR} to (5/8)T_{XOR} at the cost of two XOR gates and two delay elements.

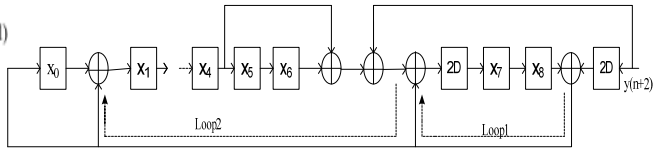


Fig 12. Four level pipelined CRC Architecture for $G(y)=1+y+y^7+y^9$

Consider the LFSR representation of four level pipelined structure. Let's assume that the Xor gates as 1,2,3,4 & 5 nodes as shown in figure 3.13, After apply the Unfolding algorithm we can obtain the figure 3.14.

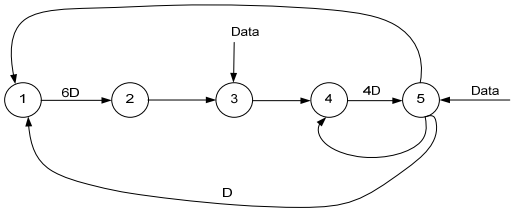


Fig 13. Data flow graph of four level pipelined structure

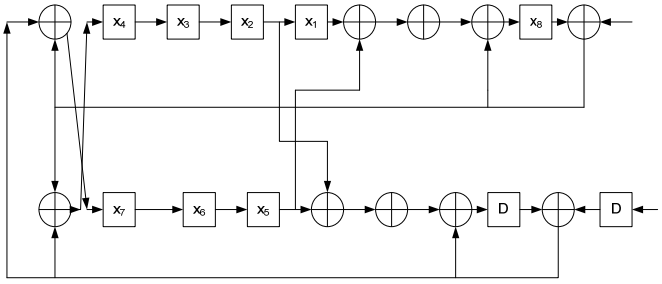


Fig 14.LFSR representation of unfolded architecture for four-level pipelined circuit.

If we perform four-level pipelining to Fig. 5 by the proposed pipelining method 2, as shown in Fig. 15, we need four extra XOR gates in loop 2, instead of six, as needed by method 1. Thus, the loop bound of loop 2 is reduced from $\left(\frac{6}{8}\right)T_{XOR}$ to $(^7/_8)T_{XOR}$ and the iteration bound is also reduced from $\left(\frac{6}{8}\right)T_{XOR}$ to $(^7/_8)T_{XOR}$ two XOR gates are saved.

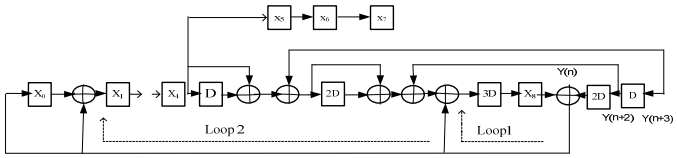


Fig 15. Improved four-level pipelined CRC architecture for G(y)) =

$$1 + y + y8 + y9$$

For the CRC circuit in figure 15 let the message sequence be 101011010. Table I shows the data flow at the marked points of this architecture at different time slots. In Table I, we can see that the achieved four-level pipeline introduces a latency of three clock cycles. However, this is not a drawback, when the message bits are long.

C.Retiming Algorithm

It is a technique used to change the locations of delay elements in a circuit without affecting the input/output characteristics of the circuit. Retiming has many applications in synchronous circuit design. These applications include reducing the clock period of the circuit, reducing the number of registers in the circuit, reducing the power consumption of the circuit and logic synthesis. It can be used to increase the clock rate of a circuit by reducing the computation time of the critical path. Let $M = t_{\max} \times n_{\max}$ where $t_{\max}$ is the maximum computation time of the nodes in G and n is the no. of nodes in G. Form a new graph G' which is the same as G except the edge weights are replaced by $W'(e) = Mw(e) - t(u)$ for all edges $U \to V$ Solve the all-pairs shortest path problem on G'. This solution can be found using the Floyd-Warshall algorithm Let S'_{uv} be the shortest path from $U \to V$ If *U! = V* then $W(U,V) = ceil(S'_{UV} / M)$ and $D(u,v) = MW(U,V) - S'_{UV} + t(V)$

If U=V then $W(U,V) = 0$ and $D(U,V) = t(u)$.

The values of $W(U,V)$ and $D(U,V)$ are used to determine if there is a retiming solution that can achieve a desired clock period. Given a desired clock period *c*, there is a feasible retiming solution *r* such that $\varphi(G_r) \le c$ if the following constraints hold.After we apply pipelining to the original serial CRC architecture, the minimum achievable CP (iteration bound) of the unfolded CRC architecture is reduced. In this section, we carry out retiming for minimum CP to obtain fast parallel CRC architectures by using the example in Fig.15. Applying three-parallel unfolding to Fig. 15, we obtain the design in Fig. 16, where all the numbered nodes (1),(2),…….represent XOR gates. If the input sequence is 101011010, we can get the data flow of Fig. 16, as shown in Table II. We can see that four clock cycles are needed to encode a 9-bit message. It is obvious that the CP of Fig. 16 is $5T_{XOR}$. After we apply retiming to it, its CP can be reduced to $3T_{XOR}$, which is shown in Fig. 17. If the input sequence is still 101011010, we can get the data flow of Fig. 17, as shown in Table III.

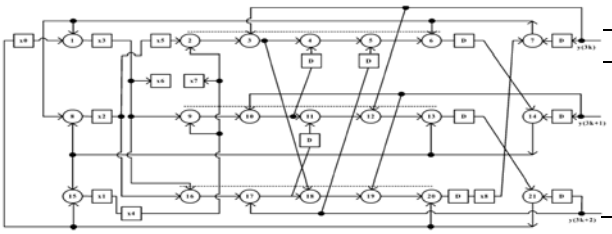


Fig 16. Three-parallel CRC architecture for the Figure 15 after unfolding

Note that as shown in Fig. 12, when the pipelining level increases,the loop bound of loop 2 will become larger than that of loop 1. Increasing the pipelining level will increase the loop bound of loop 2 and thus increase the iteration bound. The iteration bound of Fig. 15 can be further reduced if we move the XOR gate of y(n+2) from loop 2 to loop 1.

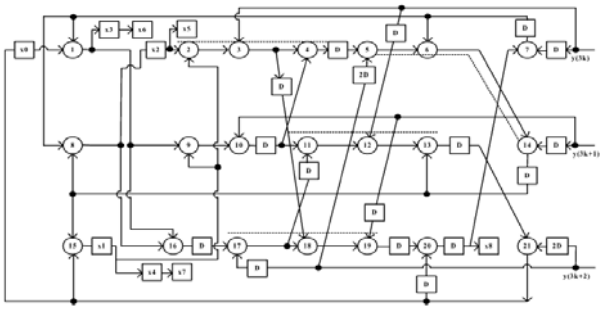


Fig 17. Retimed three-parallel CRC architecture for fig 15.

In Fig 17, we can see that the given solution requires 21 XOR gates to achieve a CP of 3Txor by four-level pipelining. However, two-level pipelining would be enough to get a CP of 3Txor, which leads to a three-parallel solution requiring only 15 XOR gates. The preceding example is only used to illustrate the complete procedure of how to apply proposed design based on method 2.

TABLE I
DATA FLOW OF FIG. 15 WHEN THE INPUT MESSAGE IS 101011010

clock	y(3k)	y(3k+1)	(3k+2)	x0	x1	x2	x3	x4	x5	x6	x7	x8
1	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
4	0	1	1	1	1	0	0	0	0	0	0	1
5	1	0	0	1	0	1	0	0	0	0	0	1
6	1	1	1	0	1	0	1	0	0	0	0	0
7	0	1	0	0	0	1	0	1	0	0	0	0
8	1	0	1	1	1	0	1	0	1	0	0	1
9	0	1	1	0	1	1	0	1	0	1	0	0
10	0	0	0	0	0	1	1	0	1	0	1	0
11	0	0	1	1	1	0	1	1	0	1	0	0
12	0	0	0	0	1	1	0	1	1	0	1	0

TABLE II
DATA FLOW OF FIG. 16 WHEN THE INPUT MESSAGE IS 101011010

clock	y(3k)	y(3k+1)	(3k+2)	x0	x1	x2	x3	x4	x5	x6	x7	x8
1	1	0	1	0	0	0	0	0	0	0	0	0
2	0	1	1	0	1	0	1	0	0	0	0	0
3	0	1	0	0	1	1	0	1	0	1	0	0
4	0	0	0	0	1	1	0	1	1	0	1	0

TABLE IIII
DATA FLOW OF FIG. 17 WHEN THE INPUT MESSAGE IS 101011010

clock	y(3k)	y(3k+1)	(3k+2)	x0	x1	x2	x3	x4	x5	x6	x7	x8
1	1	0	1	0	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0	0	0	0	0
3	0	1	0	0	1	0	1	0	0	0	0	0
4	0	0	0	0	1	1	0	1	0	1	0	0
5	0	0	0	0	1	1	0	1	1	0	1	0

In Table I, we can see that the achieved four-level pipeline introduces a latency of three clock cycles. However, this is not

a drawback, when the message bits are long. Compared with Table II, Table III shows a latency of one more clock cycle caused by retiming. One may be led to believe that three-parallel design does not process 3 bits of the message efficiently because encoding a 9-bit message requires five clock cycles. In real applications, the message length will be much longer than 9 bits. For example, if the message is 90 bits long, the CRC architecture in Fig. 3.22 will take

30+2=32 clock cycles to encode it. It is obvious that 90/32 is very close to 3.

III. RESULTS AND ANALYSIS

Each architecture is coded in Verilog and simulated. The simulation results and the netlist simulation are verified for each architecture. For the message bits: 101011010 and for the generator polynomial $1+y+y^8+y^9$

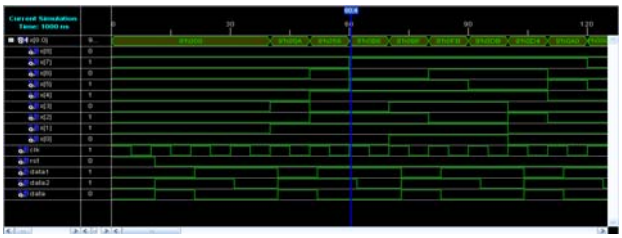


Fig 18.Retimed three parallel CRC Architecture for improved four level pipelined CRC architecture.

Fig.18 shows the simulated results of the retimed CRC circuit with generator polynomial $1+y+y^8+y^9$. It is the retimed version for unfolded and further improved pipelined implementation. There is a latency of two clock cycles. For every positive edge of the clock one input has given. Input is applied through "data" port of the CRC design. Clock is applied through "clk" port. Reset signal is applied through "rst". Initially upon the reset status the contents of the CRC circuit are zeroes. So initial output is Zero. After reset is removed for every clock the input data bit is processed to obtain the out put. But for very first two clock cycles after reset is removed output is not obtained because of the latency. If we observe the waveform after 5 clock cycle for the applied input data "10101101" the out put is "0B6". The clock period has decreased considerably with this retimed implementation.Table IV ,V&VI shows the no.of clock cycles,critical paths & iteration bounds of the different CRC circuits.

Table IV: Clock cycles of CRC circuits

Architecture	No.of clk cycles
Original architecture	9
2-level pipelined	10
4-level pipelined	12
Retiming after pipelining	12
Unfolding the 4-level Pipelined	4
Retiming the unfolded architecture	5

Table V: Critical path of CRC circuits

Architecture	Critical path
Unfolding the 4-level Pipelined	5 T_Δ
Retiming the unfolded architecture	3 T_Δ

TABLE VI: CLOCK CYCLES OF CRC CIRCUITS

Architecture	Iteration bound
Original architecture	$2T_{XOR}$
2-level pipelined	T_{XOR}
4-level pipelined & Retiming	$7/8\,T_{XOR}$

V. CONCLUSION

Generally when high-speed data transmission is required serial implementation is not preferred because of slow throughput. So parallel implementation is preferred which takes less time.This project explains the unfolding, pipelining and retiming methods used in high-speed parallel CRC implementation. Basically, serial CRC circuits have high iteration bound. In order to decrease the iteration bound of serial CRC novel look-ahead pipelining technique is used, then apply the unfolding algorithm to obtain the parallel CRC circuit. Converting serial CRC to parallel CRC increases the critical path. To reduce the critical path apply the retiming algoritham.By applying pipelining, unfolding and retiming to serial CRC throughput rate or speed can be increased.This project is implemented using Verilog HDL language on the Xilinx tool.

VI. FUTURE WORK

The future scope of this project involves analyzing the effects of pipelining depth on the hardware increase and the timing optimizations we get. This can be done by applying different levels of pipelining for the standard crc polynomials and concluding the results respect to hardware and timing issues. Also the design can be analyzed for different levels of unfolding factors to discuss the hardware overhead involving in different parallelism levels.

REFERENCES

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

[2]. K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. Hoboken, NJ: Wiley, 1999.

[3]. T. V. Ramabadran and S. S Gaitonde, "A tutorial on CRC computations," IEEE . Micro, vol. 8, no. 4, pp. 62–75, Aug. 1988

[4] Athenaeum, Andrew S. Computer Networks, Second Edition. Prentice Hall, 1988.

[5] . K. K. Parhi, "Eliminating the fan-out bottleneck in parallel long BCH encoders," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 51, no. 3, pp. 512–516, Mar. 2004.

[6]. X. Zhang and K. K. Parhi, "High-speed architectures for parallel long BCH encoders," in Proc. ACM Great Lakes Symp. VLSI, Boston, MA, Apr. 2004, pp. 1–6.

[7] W.W.Peterson, D.T.Brown, "Cyclic Codes for Error Detection," Proc. IRE, Jan. 1961.

[8] A.S.Tanenbaum, Computer Networks. Prentice Hall, 1981.

[9] W.Stallings, Data and Computer Communications. Prentice Hall, 2000.

[10] T. D. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," IEEE J. Solid-State Circuits, vol.35, no. 11, pp. 1571–1580, Nov. 2000.

1. *Mr.B.Naresh Reddy is presently working as Asst.Prof in KITS Kodada, A.P. His area of intrest are digital signal Processing and communication*
2. *Mr.B.Kiran Kumar is presently working as Asst.Prof in GITS ,Jagayapeta ,A.P. His area of intrest are digital signal Processing and communication*
3. *Ms.K.Mohini Sirisha is presently working as Asst.Prof in GITS ,Jagayapeta ,A.P. Her area of intrest are digital signal Processing and communication*