Reconfigurable Concurrent VLSI (FPGA) Design Architecture of CRC-32 for high-speed data communication

Jubin Mitra* and Tapan K. Nayak
Variable Energy Cyclotron Center
Kolkata, India
Email: *jm61288@gmail.com

***Abstract*—CRC (Cyclic Redundancy Check) is a simple and an elegant method for error detection. It finds application in most of the high-speed data communication protocol. In High Energy Physics experiment often CRC is used for control and data frame communication with detectors placed at radiation zone. Reliability of CRC error detection capability alters with generator polynomial chosen. The most popular choice is to use a 32-bit checksum. However, it again comes with many standards. So, in our work we have proposed a reconfigurable VLSI architecture of CRC-32 to meet the problem statement. Our approach can meet all the existing CRC-32 standards. The novelty of our design lies in the ability of CRC engine to generate checksum within a single clock cycle of information presented. The usability of our design is justified based on power, latency and resource utilization variations, with bus-width and technology changes.**

Keywords – Reconfigurable Logic; High Throughput Encoder; High Energy Physics; Error detection; Low latency; Power optimized; Variable bus width; CRC; checksum; FPGA

I. Introduction

In this age of information revolution, data integrity is everything. Protecting data is of primary importance. Even with most reliable high-speed communications the data errors do exist. So, to handle such cases often protective measures are taken by applying some error detection and corrective measures. To match the modern requirement of high-speed data transmission the algorithm of the error detection needs to be revised for minimum latency. Throughput intensive applications require low latency architectures. The programmer can hide the latency by pipelining and parallelism. FPGAs are an appealing alternative to traditional CPUs, GPUs and other specialized architectures, in this regard. Their interconnect fabrics are optimized for wide pipelining architecture, which gives them an advantage for data-level parallelism. These reconfigurable boards are easy to use for prototyping designs before final ASIC production.

The hardware based CRC concurrent architecture happens to be faster than the most efficient software implemented CRC algorithm. The concurrent VLSI implementation of byte-wise CRC generator proposed by Sait[1] requires 10 clock cycles for transmission of 8 byte message, which is just 2 more than the time required for it calculation.

CRC-32 finds common application in 10 gigabit Ethernet. There the CRC-32 generation is very time critical. In the Henriksson [2] work in 2001, it is mentioned that a speed of 8.7Gb/s is achieved with clock frequency of 1.09 GHz using 8bit parallelism. Increased clock frequency speed-up method is used in this process. But it still falls short from 10Gbps throughput.

With the advent of re-configurable boards like FPGA, it became attractive to explore design that can be configured run-time. Toal [3] in 2009 explores programmable CRC architecture supporting 5Gb/s line rate for variable number of input octets.

The FPGA technology has evolved tremendously since then, in terms of size, power and speed. Our paper discusses how the CRC-32 architecture can be reconfigured to suit high speed multi-gigabit communications in High Energy Physics Experiments [4]. The uniqueness of this design is its ability to calculate CRC in the same clock cycle in which the data is loaded. Only the final result is delayed by 1 clock cycle for latching the output result to prevent jitter in the CRC generated.

Our work is organized in following sections: Section II gives a short survey of existing CRC-32 standards; Section III discusses the operational functionality with timing diagram for CRC-32 with 64-bit bus width arbitrarily chosen; Section IV gives the detailed implementation methodology for reconfigurable CRC-32 logic; Section V highlights the theory behind trade-off between power, latency and throughput; Section VI presents the test results for various bus width and its relation with power, throughput and latency.

II. Common CRC standards in use

Varieties of cyclic redundancy checks have been incorporated in technical standards. Koopman and Chakravarty recommends that the polynomial should be chosen according to the suitability of the application requirement and the distribution of message length [5]. There is often a confusion among developers about which CRC to use and the number of CRCs that already exists in literature. Like there exist six types of CRC-32 polynomial [6]. The polynomial chosen may not be the efficient one in terms of the Hamming distance

978-1-4673-9692-9/15 $31.00 © 2015 IEEE
DOI 10.1109/iNIS.2015.66

for the given message length. Sometimes the difference between two protocols can be as little as pre-inversion, post-inversion and reversed bit ordering, like CRC-32/Ethernet vs CRC32/BZIP2. Then it requires re-configuration of the entire CRC generator design. To address this issue, we made our design reconfigurable on user demand basis. In this paper we have proposed Run-time design reconfiguration with minimal user intervention. The table I lists only some of the popular CRC standards in use along with its parameter settings.

TABLE I: CRC TYPES

Name	Width	Polynomial	Initialize	Input Reflection	Output Reflection	Xorout
CRC-32/Ethernet	32	0x04C11DB7	0xFFFFFFFF	TRUE	TRUE	0xFFFFFFFF
CRC-32/MPEG-2	32	0x04C11DB7	0xFFFFFFFF	FALSE	FALSE	0x00000000
CRC-32/BZIP2	32	0x04C11DB7	0xFFFFFFFF	FALSE	FALSE	0xFFFFFFFF
CRC-32C	32	0x1EDC6F41	0xFFFFFFFF	TRUE	TRUE	0xFFFFFFFF
CRC-32D	32	0xA833982B	0xFFFFFFFF	TRUE	TRUE	0xFFFFFFFF
CRC-32/POSIX	32	0x04C11DB7	0x00000000	FALSE	FALSE	0xFFFFFFFF
CRC-32Q	32	0x814141AB	0x00000000	FALSE	FALSE	0x00000000
JAMCRC	32	0x04C11DB7	0xFFFFFFFF	TRUE	TRUE	0x00000000
XFER	32	0x000000AF	0x00000000	FALSE	FALSE	0x00000000

III. CRC Functional Description

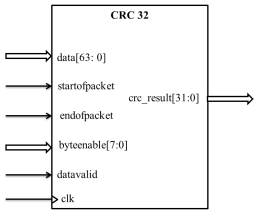

Fig. 1: CRC-32 block diagram

The fig. 1 shows a signal flow block diagram of CRC-32. The design of CRC-32 checksum calculations includes a data line of 64 bit chosen arbitrarily, but it can be extended to any needed data bus line width in the integral power of 2. The CRC-32/Ethernet design specification is used in illustration of this functional block. It is upon the user's discretion to modify or reconfigure the design according to the electronic system requirement.

The CRC functional block receives a message of bit size 64 bit in data line and generates checksum of 32bit in crc_result. It receives message from the data input interface on each clock cycle during the assertion of $datavalid$ signal. The $datavalid$ signal can be asserted or deasserted arbitrarily during or between packets as shown in fig. 2. When $datavalid$ is asserted, all data lines must contain valid data. To mark the byte size of the valid data lines $byteenable$ signal is asserted. For full bus width data line valid, $byteenable$ is loaded with x"FF" value and likewise for half bus width data line valid, $byteenable$ is loaded with x" F0" value. Here, it is important to note that MSB is in the 64th bit postion and it follows Big-endian format of data representation. The beginning of data packet transaction is marked by startofpacket signal, which is asserted synchronously with the data loaded and kept high for 1 clock cycle. This signal resets all internal states to its default state. The last data packet transmission is marked by endofpacket signal. It is kept high for 1 clock cycle and asserted synchronously with the data loaded. It loads the output register with computed checksum value. If $datavalid$ is deasserted, then $data$, $startofpacket$, $endofpacket$, and $byteenable$ are ignored. The output result is reflected in the crc_result line after a delay of 1 clock cycle. One of the major advantage

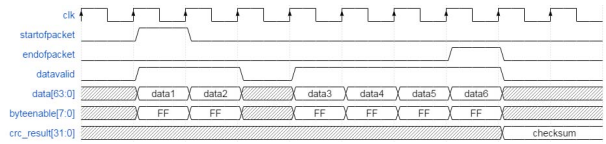

Fig. 2: CRC-32 Generator Timing diagram

of CRC engine that it can be shared both by the encoding block and the decoding block. The engine can start computing the checksum of a new packet while it is completing the calculation for the previous packet. Figure 2 shows the CRC engine to be used as CRC-32/Ethernet generator. At each clock cycle with $datavalid$ signal asserted 64bit data is loaded and operated concurrently. The $byteenable$ signal is set to indicate full bus data valid. The packet transmission starts on assertion of the $startofpacket$ signal and stops on assertion of the $endofpacket$ signal. In the immediate clock cycle following the $endofpacket$ signal, the CRC checksum is latched in the crc_result bus signal.

IV. Reconfigurable Architecture

FPGA logic must be flexible enough and future proof for CRC polynomial adaptation. Our design discusses reconfiguration architecture that allows run-time custom modification of polynomial and parameter settings by running a simple C-code remotely in the RISC processor. In fig. 3, the main buildings blocks are presented.

The syndrome matrix $\mathbb{S}$ is a run $(n \times r)$ matrix, that defines a linear map from processed input data vector $(\mathbb{P}^{2r})$ to cyclic code $(\mathbb{C}^r)$. It is a linear mapping for cyclic division. Equations (2) to (5) gives the expanded syndrome matrix for most commonly used polynomial, "0x04C11DB7".

$$\mathbb{S} = \begin{bmatrix} S_7 \\ S_6 \\ S_5 \\ S_4 \\ S_3 \\ S_2 \\ S_1 \\ S_0 \end{bmatrix} \tag{1}$$

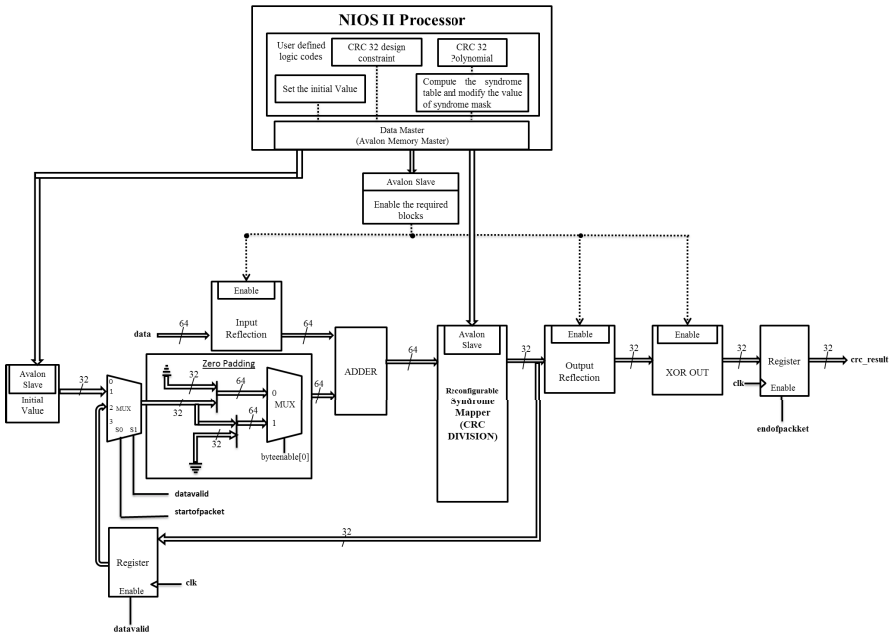

Fig. 3: CRC-32 run time reconfigurable architecture

$$S_7 = \begin{bmatrix} \texttt{79005533} \\ \texttt{BEE0A442} \\ \texttt{5F705221} \\ \texttt{ADD8A7CB} \\ \texttt{D48CDD3E} \\ \texttt{6A466E9F} \\ \texttt{B743B994} \\ \texttt{5BA1DCCA} \end{bmatrix} \quad S_6 = \begin{bmatrix} \texttt{2DD0EE65} \\ \texttt{9488F9E9} \\ \texttt{C824F22F} \\ \texttt{E672F7CC} \\ \texttt{73397BE6} \\ \texttt{399CBDF3} \\ \texttt{9EAED022} \\ \texttt{4F576811} \end{bmatrix} \tag{2}$$

$$S_5 = \begin{bmatrix} \texttt{A5CB3AD3} \\ \texttt{D08513B2} \\ \texttt{684289D9} \\ \texttt{B641CA37} \\ \texttt{D9406BC0} \\ \texttt{6CA035E0} \\ \texttt{36501AF0} \\ \texttt{1B280D78} \end{bmatrix} \quad S_4 = \begin{bmatrix} \texttt{0D9406BC} \\ \texttt{06CA035E} \\ \texttt{036501AF} \\ \texttt{83D20E0C} \\ \texttt{41E90706} \\ \texttt{20F48383} \\ \texttt{921ACF1A} \\ \texttt{490D678D} \end{bmatrix} \tag{3}$$

$$S_3 = \begin{bmatrix} \texttt{A6E63D1D} \\ \texttt{D1139055} \\ \texttt{EAE946F1} \\ \texttt{F7142DA3} \\ \texttt{F9EA980A} \\ \texttt{7CF54C05} \\ \texttt{BC1A28D9} \\ \texttt{DC6D9AB7} \end{bmatrix} \quad S_2 = \begin{bmatrix} \texttt{EC564380} \\ \texttt{762B21C0} \\ \texttt{3B1590E0} \\ \texttt{1D8AC870} \\ \texttt{0EC56438} \\ \texttt{0762B21C} \\ \texttt{03B1590E} \\ \texttt{01D8AC87} \end{bmatrix} \tag{4}$$

$$S_1 = \begin{bmatrix} \texttt{828CD898} \\ \texttt{41466C4C} \\ \texttt{20A33626} \\ \texttt{10519B13} \\ \texttt{8A484352} \\ \texttt{452421A9} \\ \texttt{A0F29E0F} \\ \texttt{D219C1DC} \end{bmatrix} \quad S_0 = \begin{bmatrix} \texttt{690CE0EE} \\ \texttt{34867077} \\ \texttt{9823B6E0} \\ \texttt{4C11DB70} \\ \texttt{2608EDB8} \\ \texttt{130476DC} \\ \texttt{09823B6E} \\ \texttt{04C11DB7} \end{bmatrix} \tag{5}$$

A. Syndrome Table Synthesis

The computation of syndrome table from the CRC polynomial involves three steps, namely:

1) Generation of each row of the *Non-Systematic Generator matrix* from the shifted replica of the CRC polynomial. The row size of the matrix depends upon the data bus width over which the CRC need to be calculated.
2) Application of Gaussian elimination method to generate its Systematic form. The time complexity of this algorithm is $\frac{n(n+1)}{2}$
3) In reconfiguration architecture, the objective is to load the Syndrome Mapper with Syndrome Matrix value .
   a) For *run time reconfiguration* loading of the Syndrome Matrix value to the design, requires modification of the *syndrome mask*. The time complexity to reprogram the mask is $\frac{(n \times r)}{write\ data\ bus\ width}$.

B. Run-time reconfiguration

The design is generated for data bus of 64 bits wide. The purpose of choosing 64 bit to meet the MAC layer requirement of IEEE 10-gigabit ethernet standards, which operates at 125MHz. The design includes all pre and post processing blocks of CRC-32 ethernet standards. Each block is then wrapped with an interface block, such that it could able to communicate with the control unit. In our design the role of control unit is served by NIOS-II processor, which is a soft-core RISC processor provided by Altera. The initial value block, enabler block and Reconfigurable Syndrome Mapper uses Avalon Memory Mapper interface [7] for communication

with the data master of the NIOS-II processor. The enabler block then supervises the activation of the Input Reflection, Output Reflection and XOROUT block.

For run-time modification of the CRC polynomial or its parameter setting, user simply needs to modify the software parameters in the application program running over the NIOS-II processor. As the application program detects any change in the design input parameters, it re-computes the entire syndrome table and accordingly it modifies the syndrome mask value. It communicates with Avalon standard bus to set the initial value block, enabler block and loads the syndrome mask matrix. During this design modification the CRC computation is disrupted, which is only for a few millisecond. The advantage of this design is its ability to reconfigure run-time, without reloading the entire firmware. But, for bus width modification the entire code need to be regenerated. So, users need to make choice about bus width from the beginning of firmware design and remains fixed throughout.

V. Physical details

The latency of any error detection algorithm is a combination of several delays. Gaining of high speed computation involves losing in area, bandwidth, reliability and power aspect. To increase throughput using parallelism of serialized logic involves more area, hence there is a trade-off between hardware costs versus detection latency [8]. More area utilization means more resources involved. Subsequently, more power consumption [9]. During high clock frequency operation, data reliability is a big issue as there is a chance of jitter introduction due to improper clock routing, hence there is a tradeoff between reliability and latency [10]. If the channel is dedicated for single data line, then full bandwidth can be used. However, if it is to be shared among multiple clients, subsequently probability of data contention increases with decrease in latency. This latter requires placing of flow control; buffering and congestion control mechanisms. So, there is a tradeoff between latency and bandwidth for multiple users [11]. The goal is to optimize CRC-32 computation latency with respect to the discussed aspects.

Understanding the data dependencies is fundamental part of parallelizing hardware implementation. Bernstein's conditions [12] states when two program segments are independent then it can be executed in parallel. The architecture of CRC-32 can be parallised, as the input data channel has one to one mapping with output response data-channel for syndrome mapper.

The syndrome mapper of the architecture can be designed concurrently, but it involves the longest time delay part, hence forms the most time critical portion. Its response lines are independent of each other. But, the entire response vector together forms a meaningful CRC value, hence it need to be hold for 1 clock cycle to gather the value of all the response lines. The *duration of clock cycle* has to be greater than the *maximum delay path*.

VI. Test Results

The performance study of the CRC core with variation in data bus width over different FPGA boards and its dependence on latency, power and resources is plotted. For homogeneity in test result, all through CRC32/Ethernet specification is followed.

This setup is intended for performance study of CRC32/Ethernet design specification with variation in data bus width over different FPGA boards. The results tabulated in the Tables II to IV lists the comparative analysis of the design for 64 bit data bus CRC32/Ethernet engine over three kinds of Altera FPGA boards on the basis of timing, power and logic resource analysis. Refer to graphs for results of data bus width variation with the same CRC32/Ethernet specification.

In the comparative analysis of the designs at various FPGA boards with varying data bus, involves the same data vectors with same clock frequency. This is done to make sure a standardization in the evaluation of the performance of the algorithm in various test boards with data bus width variation. The results will vary with modification in the test vectors. Here, it is important to note that the graphs are indicative of relative qualitative measurement of the performance, for absolute measurement it is completely dependent upon the users test vectors.

A. Timing Analysis

Minimization of latency is done with removal of pipeline registers to operate within 1 clock cycle, but it involves increase in combinatorial delay between registers. The maximum frequency of operation is limited by the maximum delay involved in the longest concurrent path. It is very important to abide by the time critical portion, for stable output response.

Total combinatorial logic (T_{logic}) is theoretically a summation of individual logic blocks

$$T_{logic} = T_{Input\ Reflection} + T_{Adder} \tag{6}$$

$$+ T_{Syndrome\ Mapper} + T_{Output\ Reflection} + T_{Xorout} \tag{7}$$

Among all the logical blocks, Syndrome Mapper involves the lengthiest combinatorial logic. It can safely be approximated that

$$T_{logic} \approx T_{Syndrome\ Mapper} \tag{8}$$

The maximum allowable frequency[13] can be defined by eq. (9)

$$F_{max} = \frac{1}{T_{clk-q} + T_{logic} + T_{timing} + T_{setup} + T_{skew}} \tag{9}$$

where F_{max} is maximum allowable frequency for clock; T_{clk-q} is time from clock arrival until data arrives; T_{logic} is propagation delay through logic between flip-flops; $T_{routing}$ is routing delay between flip-flops; T_{setup} is minimum time data must arrive

For guaranteed silicon performance in FPGA, different Timing Models [14] are used for analysis. FPGA able to

TABLE II: Timing Analysis of CRC-32 logic core with 64 bit bus

Board No.	Fast Timing Model Delay	F_{max} = 1/(Minimum Delay)
CYCLONE III (EP3C120F780I7)	4.502 ns	181.62 MHz
CYCLONE IV E (EP4CE115F29I8L)	6.639 ns	194.29 MHz
STRATIX V (5SGSED6N1F45I2)	2.374 ns	216.78 MHz

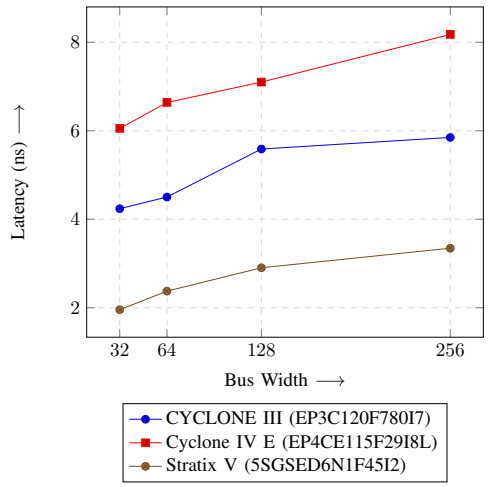

Fig. 4: Latency vs CRC bus Variation

operate at wide range of temperature variations. Commercial temperature varies from 0^oC to 85^oC. Fast timing model is used for latency calculation as it gives the minimum threshold. The timing analysis is conducted using post place and route simulation/Gate level simulation for Cyclone series FPGA and TimeQuest timing analyzer tool for Stratix series.

B. Power Analysis

FPGA is a CMOS based technology, where dynamic power consumption is related to charging and discharging of parasitic capacitances on gates and metallic contacts. The energy dissipation is linked with current dissipation as voltage is fixed in a FPGA design[13]. So,

$$I = V \times C \times f \tag{10}$$

where I is total current, V is voltage, C is capacitance, and f is frequency. In it we can manipulate C and f. The capacitances C is directly related to the togging rate of the gates and the distance of the routes connecting the gates. The frequency f is related with the clock frequency.

The power estimation tool provided by Altera is used for calculation of power consumption. Figure 5 shows at constant frequency of 100MHz, with increase in data width power consumption increases steeply. It is obvious as number of logic gate involved also increases with bus width. While fig. 6 shows when operating frequency increases power consumption also increases, which clearly tally with the nature of eq. (10).

C. Logic resource utilization

Device densities and corresponding design sizes have become very large (over millions of gates), and newer methodologies have been developed to assist the placement tools. Place and routing tools helps in laying out logic elements from a sea of gates. Floorplanning is used by designers who are having difficulty in setting the critical paths. The proposed design includes many time critical path for operation at very high frequency. Proper physical constraints must annotated into the fitter tool to concentrate the logic resources and optimizing the overall routing resources and improving timing performance.

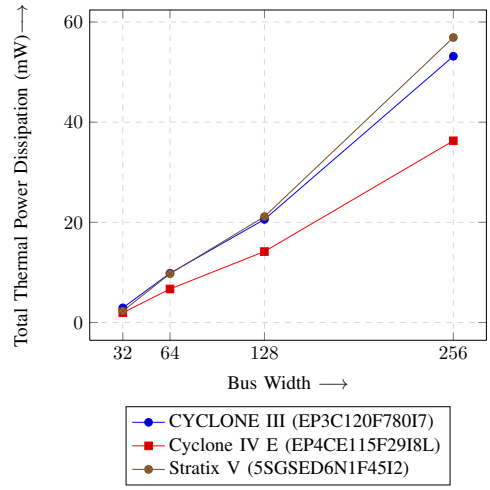

Fig. 5: Power vs CRC bus Width Variation

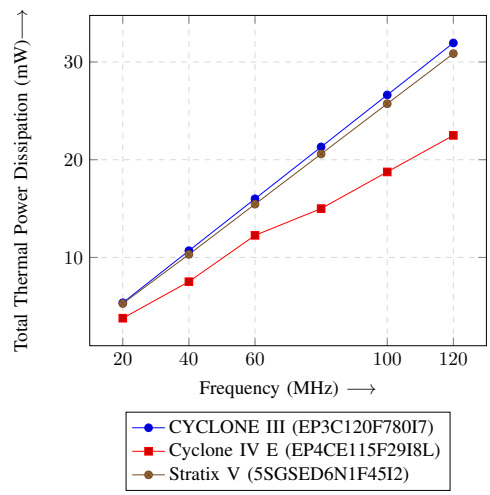

Fig. 6: Power vs Frequency for 32 bit data bus

TABLE IV: Resource Utilization of CRC-32 logic core with 64 bit bus

Board	Combinatorial with no register	Combinatorial with a register	Register only	Total Logic Resources
CYCLONE III (EP3C120F780I7)	367	5	59	431
CYCLONE IV E (EP4CE115F29I8L)	374	12	52	425
STRATIX V (5SGSED6N1F45I2)	172	23	10	205

TABLE III: Power Analysis of CRC-32 logic core with 64 bit bus

Board	State	Block Type	Total Thermal Power	Dynamic Power	Routing Thermal Power	Block Average Toggle Rate (millions of transitions/sec)
CYCLONE III(EP3C120F780I7)	Fully ACTIVE	Combinatorial Cell	8.35 mW	3.23 mW	5.12 mW	91.720
		Clock Control Block	0.84 mW	0.00 mW	0.84 mW	10.123
		Register Cell	0.65 mW	0.39 mW	0.26 mW	5.347
CYCLONE IV E(EP4CE115F29I8L)	Fully ACTIVE	Combinatorial Cell	5.76 mW	2.21 mW	3.55 mW	95.927
		Clock Control Block	0.44 mW	0.00 mW	0.44 mW	10.123
		Register Cell	0.50 mW	0.24 mW	0.26 mW	5.347
STRATIX V(5SGSED6N1F45I2)	Fully ACTIVE	Combinatorial Cell	8.83 mW	4.06 mW	4.78 mW	122.498
		Clock Control Block	0.52 mW	0.00 mW	0.52 mW	29.491
		Register Cell	0.40 mW	0.12 mW	0.28 mW	5.601

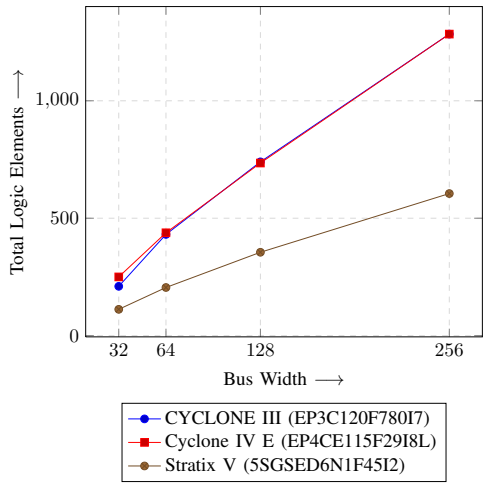

Fig. 7: Resources vs CRC bus Variation

VII. Discussion

The presented CRC-32 architecture has constant latency and high throughput, which suits the requirement of protocol standards in High Energy Physics experiments [15]. The plotted graphs in figs. 4, 5 and 7 shows latency, power and resource variation vs bus width modification. The slope of the graph is always positive. It means that the performance of the module is consistently compromised with increase in data bus width. Now it is upto user's discretion to choose the particular bus width after optimizing the design against latency, power and resource variation. Like, there are certain design requirements which requires extra wide data bus, but it will operate at low clock frequency. Now this trade-off is clearly reflected in fig. 4. But as the bus width increases the number of resource utilization also increases as shown in fig. 5. Corresponding requiring more power as plotted in fig. 5. But power consumption might also be decreased with lowering in operating clock frequency.

VIII. Conclusion

The main objective of this work was to develop a reconfigurable very high throughput CRC-32 design architecture. This resulted in the development of this novel CRC-32 architecture which is independent of the standards chosen and bus width applied. In test setup it is clearly seen that the CRC-32 design of particular bus width depends on the parameters preferred by the user's choice of area, power and latency. It is a multi-optimization problem to satisfy all the design constraints set by the user. Future research can be done in this direction to develop an optimization algorithm to choose ideal design parameters, and automatically suggesting the user with the best choice.

References

[1] S. M. Sait and W. Hasan, "Hardware design and VLSI implementation of a byte-wise CRC generator chip," *IEEE Transactions on Consumer Electronics*, vol. 41, no. 1, pp. 195–200, 1995.

[2] T. Henriksson, H. Eriksson, U. Nordqvist, P. Larsson-Edefors, and D. Liu, "VLSI implementation of CRC-32 for 10 Gigabit Ethernet," in *The 8th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, vol. 3. IEEE, 2001, pp. 1215–1218.

[3] C. Toal, K. McLaughlin, S. Sezer, and X. Yang, "Design and implementation of a field programmable CRC circuit architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 8, pp. 1142–1147, 2009.

[4] ALICE Collaboration, "Upgrade of the ALICE Readout & Trigger System," Tech. Rep., 2014.

[5] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *International Conference on Dependable Systems and Networks*. IEEE, 2004, pp. 145–154.

[6] G. Cook, "Catalogue of parametrised crc algorithms," 2010.

[7] Altera Corporation, "Quartus II Handbook Version 13.1," Tech. Rep., 2013.

[8] S. G. Pestana, E. Rijpkema, A. Radulescu, K. Goossens, and O. P. Gangwal, "Cost-performance trade-offs in networks on chip: A simulation-based approach," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 2. IEEE, 2004, pp. 764–769.

[9] I. Kuon and J. Rose, "Area and delay trade-offs in the circuit and architecture design of FPGAs," in *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*. ACM, 2008, pp. 149–158.

[10] A. Singh, A. Mukherjee, L. Macchiarulo, and M. Marek-Sadowska, "PITIA: an FPGA for throughput-intensive applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 3, pp. 354–363, 2003.

[11] L. Kleinrock, "The latency/bandwidth tradeoff in gigabit networks," *IEEE Communications Magazine*, vol. 30, no. 4, pp. 36–40, 1992.

[12] A. J. Bernstein, "Analysis of programs for parallel processing," *IEEE Transactions on Electronic Computers*, no. 5, pp. 757–763, 1966.

[13] S. Kilts, *Advanced FPGA design: architecture, implementation, and optimization*. John Wiley & Sons, 2007.

[14] "Guaranteeing Silicon Performance with FPGA Timing Models," *White Paper*, no. WP-01139-1.0, 2010.

[15] P. Moreira, A. Marchioro, and K. Kloukinas, "The gbt: A proposed architecture for multi-gb/s data transmission in high energy physics," *Published in Prague*, 2007.