SUST Journal of Engineering and Computer Sciences (JECS), Vol. 19, No.2, 2018

High Speed Low Power Cyclic Redundancy Check-32 using FPGA

Mohamed S. Abdulnabi, Hisham Ahmed
Computer and Network Engineering, Sudan University of Science and Technology
mohd.salah181@gmail.com, hisham@ieee.org

Received : 15/12/2018
Accepted : 22/01/2019

ABSTRACT- Cyclic Redundancy Check (CRC) is a method used for error detection technique and data integrity. CRC take a block of a message's bits and divide it by a binary number called polynomial, the result of this division is the checksum that will be added to the message. On the receiver side, the same division will be performed to get the remainder which could be compared with the transmitted checksum if there are no differences that are mean there are no errors. This paper aims to design CRC32 that applied in the Ethernet frame by using Field Programmable Gate Array (FPGA) Virtex-7. Lookup tables and slicing-by-16 algorithm are used together to calculate the CRC32 in parallel. Xilinx ISE used as IDE and synthesis tool and I-Sim used for simulation purposes. The result of this design is 1.250 ns which is the processing time and 102.4 Gbps which is the throughput, furthermore the power consumption is very low as well as the device utilization.

المستخلص – إختبار التكرار الدوري هو عبارة عن تقنية تستخدم لضمان وصول البيانات بصورة صحيحة. في هذه التقنية يتم أخذ مجموعة من البيانات وهي في صورتها الثنائية ومن ثم قسمتها على عدد محدد وثابت ، المتبقي من عملية القسمة يسمى المختبر ، هذا المختبر يتم اضافته الى نهاية الرسالة المراد ارسالها. من ناحية المستقبل يتم إجراء نفس الإختبار وذلك من أجل الحصول على باقي القسمة أيضاً من أجل الحصول على قيمة المختبر والتي سيتم مقارنتها مع قيمة المختبر المرسلة من ناحية جهاز الإرسال فاذا كانت القيم متساوية فان ذلك يعني خلو هذه الرسالة من الأخطاء. هذه الورقة توضح طريقة تصميم أختبار التكرار الدوري من النوع 32 والذي يستخدم في شبكات الإيثرنت وذلك عن طريق تطبيقها على مصفوفة البوابات المنطقية القابلة للبرمجة من النوع فيرتكس 7. التصميم مبني على جداول البحث و خوارزمية التشريح الى 16 واللذان يستطيعان تنفيذ حسابات إختبار التكرار الدوري بطريقة متوازية. تم استخدام بيئة التصميم المتكاملة التابعة لزايلنكس والتي تسمى أي إس إي كما تم استخدام المحاكي أي سيم. زمن المعالجة الناتج من هذه الإجراءات هو 1.250 نانو ثانية بينما كان معدل البيانات يساوي 102.4 جيجابت لكل ثانية مع ملاحظة إنخفاض القدرة المستهلكة و كذلك إنخفاض استهلاك الدوائر المتاحة بمصفوفة البوابات المنطقية القابلة للبرمجة.

Keywords: *CRC32, FPGA, Xilinx*

INTRODUCTION

These and the following days, data communication and the internet considered one of the very important aspects of human life. The internet already impacting the growth of science, business, government, humanity, and communication. The advances in communications technologies and services add a great value for the exchange of data between the connected devices. diverse applications are widely found that span across fields as never before.

Nevertheless, the Internet of Things (IoT) is a recent communication field that will add great value in the near future, in which the sensors and devices will be equipped with microcontrollers, senders and transmitter for digital communication, and suitable protocol stacks that will allow them able to communicate with each another and with users, so the IoT is becoming a vital part of the Internet [1].

The IoT applications such as smart home, wearables, smart city, smart grids, connected car, and smart farming already started and working and it is traffic expected to increased every day. Internet traffic increased continuously with dramatic growth. in 1992, the internet networks traffic reached approximately 100 GB per day. Then after ten years, in 2002, internet traffic raised up to 100 GB per seconds (GBps). In 2015, internet traffic exceeded 20000 GBps. By 2020 global IP traffic expected to reach 2.3 Zettabyte

per year (Zettabyte is 10^{21} bytes), or 194 Exabyte (Exabytes is 10^{18} or 1 billion Gigabytes) per month. Furthermore, the fixed broadband speed will reach up to 47.7 Mbps by 2020. In addition to that, by 2020 the expected video traffic will be about million minutes of video per each minute in addition to that, the traffic that generated from the video on demand (VoD) will be equivalent to 7.2 Million DVD per month. Mobile users will generate traffic can reach up to 30.6 Exabytes per month by 2020. In general, IP traffic can reach 511 terabits per seconds (Tbps) in 2020, this is equal to 142 million people watching high-definition (HD) video in the same time, all day, every day.

The processing time for the Ethernet CRC32 (Cyclic Redundancy Check 32) will be affected when the speed of data traffic exceeded network devices processing capability. However, the processing time that used currently for the CRC32 in the core and backbone devices does not handle the expected increment in the traffic, according to that the processing time should be decreased and throughput of the CRC32 must be increased as much as possible. This paper talks about the CRC32 design in the Ethernet frame using Xilinx Virtex-7 Field Programmable Gate Array (FPGA). The design depends on the slicing-by-16 algorithm that can perform the needed CRC32 calculations in parallel.

The CRC computation using lookup tables explained by Sarwate, he proved the calculation of checksum value through lookup table method is highly efficient when it compared to the traditional methods such as Linear Feedback Shift Register (LFSR) which process the data in a serial manner [2]. Anand and Bajarangbali in [3] used slicing by 16 algorithm and reduced lookup tables to implement the CRC algorithm on the FPGA and the achieved throughput was 40 Gbps. In [4], [5] and [6] there are other studies discussed the implementation of the CRC in parallel.

Algorithm and types of CRC

The function of Cyclic Redundancy Check (CRCs) deployed in most of the digital communications and storage systems, that is to detect accidental errors in the data. The binary data that handled are subjected to a CRC which will get binary check sequence with a fixed length. This fixed length of the check sequence is attached to the data that will be transmitted and act to find out the correctness. In the receiver side, CRC performed on the data one more time and the result is compared to the attached check sequence. If it is matched, that is mean the data does not corrupted. [7]. The bits in the CRC treated as a binary polynomial using specific width, after that it finds out the remainder using the division of data bits with a polynomial. the resulted remainder of this calculation known as the checksum that must be added to the data bits. Also at the receiver side, the data and the checksum divided by the polynomial, if the result was zero it means the data was received correctly [8].

There are various types of CRC, its length and its polynomial depends on the application that works on. Table 1 illustrates some of these types.

Table1: various types of CRC

Name	Uses	Polynomial
CRC-1	most hardware	0x1
CRC-3	mobile networks	0x3
CRC-4	ITU-T G.704	0x3
CRC-5	Gen 2 RFID	0x09
CRC-6	mobile networks	0x27
CRC-8	DVB-S2	0xD5
CRC-10	ATM	0x233
CRC-11	FlexRay	0x385
CRC-14	Radio Channel	0x0805
CRC-15	CAN	0x4599
CRC-16	USB	0x8005
CRC-24	OS-9 RTOS.	0x800063
CRC-32	Ethernet	0x04C11DB7

Mathematically, a k-bit message can be considered as the coefficients of a polynomial $B(x) = bk-1xk-1 + \dots + b1x1 + b0x0$. The most significant bit starts the data stream. Furthermore, an (m+1) bit generator polynomial $P(x) = xm + pm-1xm-1 + \dots + p1x1 + p0x0$ of order m is selected. Calculations are performed in modulo-2 arithmetic. The CRC is the remainder of the division of xm B(x) by P(x) and will be appended to the message [4].

CRC is designed for the detection of errors which is common in the communication channels. The n-bit CRC called CRCn when it is value is n bits long. for example, when one bit used it is called CRC-1, this type called parity check as well and in this type only one bit added to the original message, another example in CRC-4 the number of bits that attached to the original message is 4 and so on in the other CRC-n. this paper focuses

on the Ethernet frame CRC which is the CRC32. This type has a fixed polynomial value equal to 0x04C11DB7. The transmitted message contains the original message and the reversed-byte swapped CRC for example if the message has the hex value (31 32 33 34 35) the Ethernet CRC32 will be 0xCBF53A1C, and the transmitted message will be (31 32 33 34 35 1C 3A F5 CB). the FCS (Frame Check Sequence) field in the Ethernet frame is the CRC. Behind that, The Ethernet frame consists of the source and destination MAC, Ethernet frame length, in addition to the data, the Ethernet frame illustrated in Figure 1.

Preamble	SFD	Destination	Source	Length	Data and Pad	FCS
7	1	6	6	2	46 – 1500	4

Figure 1: Ethernet Frame format

The System Design

The purpose of this paper is to decrease the needed time to perform the processing of the CRC32 algorithm by performing parallel operations using the FPGA. the CRC32 computation performed by the lookup table and slicing by 16 methods. By using this method the lookup table possible values of the CRC32 precomputed and saved. The pre-computed values will be used in the calculation of the final checksum value.

In the calculation of the CRC Each byte will catch its own CRC32 value which will be used as an argument in the operations that can find the result. The algorithm could be implemented based on one lookup table, but for efficiency enhancements of the system multiple lookup tables implemented simultaneously and work with another technique called slicing by N. FPGA Virtex-7 used as a platform for the design. the code is written through Very High Speed Integrated Circuit Description Language (VHDL).

Behind that, the code synthesized by Xilinx synthesis tool. then the simulation executed on ISim which is Xilinx simulation tool, VHDL used to write the simulation code as well as the implementation code. However, during the initialization of the system the precomputed values prepared. Figure 2 shows that a FPGA connected to PC, the purpose of the PC in the design is to generate the pre-computed values of the lookup tables using Java program.

Later when these values generated it will be transferred to the FPGA and then saved inside RAMs, these RAMs designed inside the FPGA to reduce the transfer time, the design area and the power. This PC can be replaced by a microprocessor located outside the FPGA or inside it in the real systems, but for simulation purposes the pre-computed values generated through software written in Java.

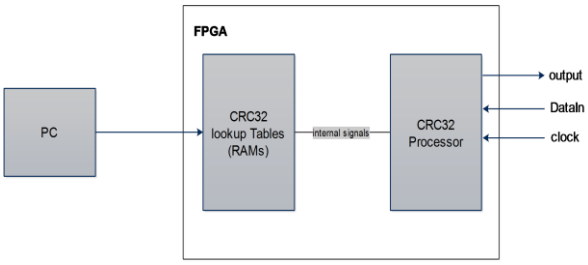


Figure 2: block diagram

Generation of Lookup Tables

The purpose of lookup tables is to decrease the number of the needed operations to calculate the CRC32 for the Ethernet frame. as long as the divisor (polynomial) is always fixed we can pre-compute the division for all bytes and save the remainder in a lookup table. This could be performed through the below code

```
for (int dividend = 0; dividend <=
255; dividend ++)
{
int temp = dividend;
  for (int bits=0; bits<8; bits++)
// generate the CRC
temp = (temp >>1) ^ ((temp &1) *
0x04C11DB7);
crcTable[dividend] = temp;}
```

In this code, in each repetition the dividend (beginnig from 0x00 to 0xFF) saved in a temporary variable named temp, after that the temporary variable treated in term of bits in the second for loop. the equation behind the second for loop checkup the LSB of the temp, if it is '1' then the shifted right value of the temp will perform xor with the polynomial, this operation remains for the eight bits to find out the remainder of (dividend mod polynomial).

The value of this remainder saved in the lookup table Array that indexed by the dividend. The flowchart in Figure 3 shows the generation of the lookup table. Now using this lookup table, we can

find out the value of CRC32 for only one byte in the period. but in order to achieve parallel processing another 15 tables will be generated, they are generated based on the values on this table. This point will be explained more in the next section.

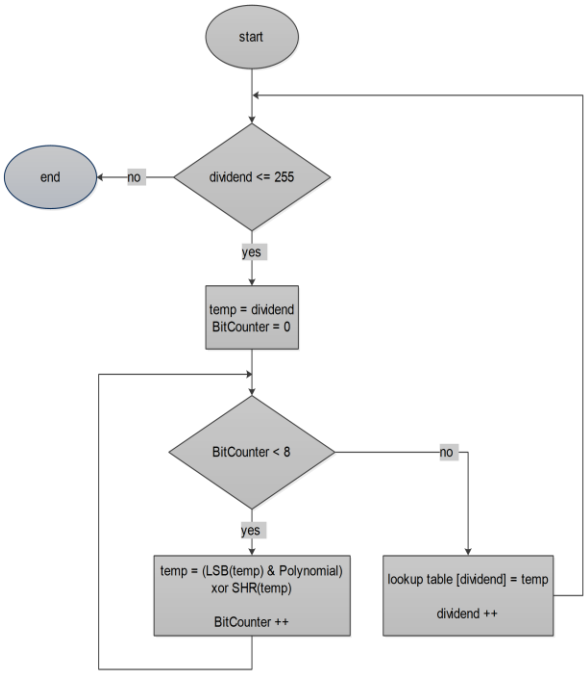


Figure 3: Flow chart for lookup table generation

Slicing by 16 algorithm

The design going to find out the CRC32 for 16 bytes (128 bit) in one operation which means another 15 additional lookup tables must be generated as well. The process that used to generate these tables depends on the first lookup table that already generated, the below code responsible for generating the CRC32 values for these lookup tables. any value for each byte on each lookup table depends on the prior lookup table, this preceding value shifted 8 times to the right and then xor with lookup table 0 that indexed by the LSB byte from the prior lookup table.

```
for(inti=0;i<=255;i++)
{
crcTable1[i] =
(crcTable0[i]>>8) ^
crcTable0[(int)(crcTable0[i]&0xFF)
];
crcTable2[i] =
(crcTable1[i]>>8) ^
crcTable0[(int)(crcTable1[i] &
0xFF)];
…
crcTable15[i]=
(crcTable14[i]>>8) ^
crcTable0[(int)(crcTable14[i]&
0xFF)];}
```

Lookup table inside the FPGA

A lookup table could be considered as RAMs, it could be implemented outside the FPGA or inside it. But if it is implemented outside the FPGA the cost will be increased and the design area as well, not yet the power consumption may be increaseded too. So in order to optimize the design as much as possible the lookup table implement inside the FPGA. One of the aspects to implement the lookup tables inside the FPGA is the transferring speed, as the transfer speed between the outside RAMs and the FPGA is slower than the transfer rate if the RAMs are implemented inside the FPGA. By using VHDL The sixteen RAMs created inside the FPGA. A new type named lookup Table defined as array consist of 256 locations, each one has 28 bit. After that sixteen signal of this type has defined. Then these memory locations loaded with the precomputed values during the initialization.

Slicing-by-16 CRC32

The entity in VHDL describes the ports of the circuit. In this project, the system entity consists of 6 ports illustrated in Figure 4. As illustrated in Figure 5, the stream of the data saved in a buffer. Splitting circuit slices the data into 12 bytes and 4 bytes. Each byte of the twelve will be used to address the RAMs (from RAM0 to RAM11), the remaining 32 bits XORed with the previous CRC32 value and then sliced into four bytes, each one used to address the RAMs (from RAM12 to RAM15).

The addressed values in the RAMs XORed together to get the new value for the CRC32. There is no considerable time to get the output from the xor gates, so the sixteen RAMs could be accessed at the same time. The implementation of the code in the VHDL performed concurrently, that is will support to achieve the operation of this algorithm in parallel.

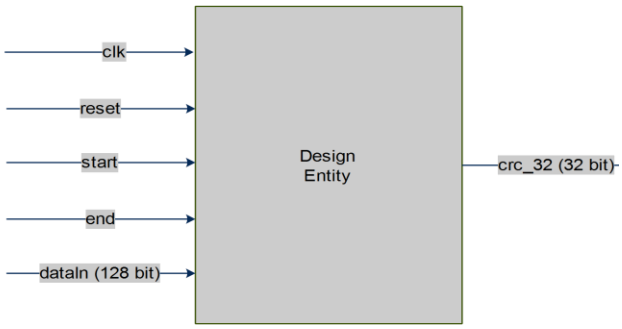


Figure 4: Design Ports

Results and Discusions

The simulation written by VHDL and Isim version 14.5 simulator is used to perform it. Isim has the capability to create a component to act like the tested device so that one component created to do that. this component has the same ports in the design entity. After that, the Isim can generate the needed clock pulses and attach it to the simulated component. The generated clock pulses are 800 MHz which is equal to the proposed clock source.

The simulation has two processes working concurrently, the first process which is already discussed which can generate the clock pulses and the second process responsible for the simulation stimulus. Inside the process of the stimulus 128 bits randomly assigned to the DataIn signal in a period specified in simulation code, behind that the control signals such as start, end and reset also considered in the simulation code. Figure 6 illustrates how the control signals and the dataIn stimulated in the simulation.

The throughput of this design is 102.4 Gbps which means the processing time is 1.250 ns. The design simulated based on virtex-7 FPGA specifications. The oscillator frequency is 800 MHz. In this work the resulted throughput considered very high, however it is also higher than the throughputs that mentioned in the previous works. Behind that, the consumed power is 0.158W and the dynamic power consumption is only 0.015W. so most of the power consumption generated from the leakage which is around 0.143W. According to that the power consumption is low and could be considered very low when compared to the maximum power value that generated from virtex-7 FPGA.

In Figure 5 the function of the highlighted combinational circuit is to perform xor operation between the prior result and the dataIn LSB. This circuit is a combinational circuit it does not depends on the clock. On another hand, the buffer sends the data to the entity dataIn port when the clock falling edge received.

The output of the combinational circuit could be found in time between the falling edge and the rising edge. Behind that, the none-highlighted circuit is responsible for finding out the final result, this circuit triggered by the rising edge of the clock pulses.

Accordingly, the none-highlighted circuit added to process with the related sensitivity list in VHDL code in order to perform its work during the high pulse. Figure 6 shows the connection between the dataIn, the output of the CRC32 and the related pulse. the utilized resources in the FPGA which could be considered very low, can allow the system designer to combine it inside any system. The number of slices register that used is 27 from 408000, the number of slice LUTs is 550 from 204000. Table 3 illustrates the device utilization summary generated by Xilinx ISE.

Table 3: device utilization summary

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	27	408000	0%
Number of Slice LUTs	550	204000	0%
Number of fully used LUT-FF pairs	27	550	4%
Number of bonded IOBs	164	600	27%
Number of BUFG/BUFGCTRL/BUFHCEs	2	200	1%

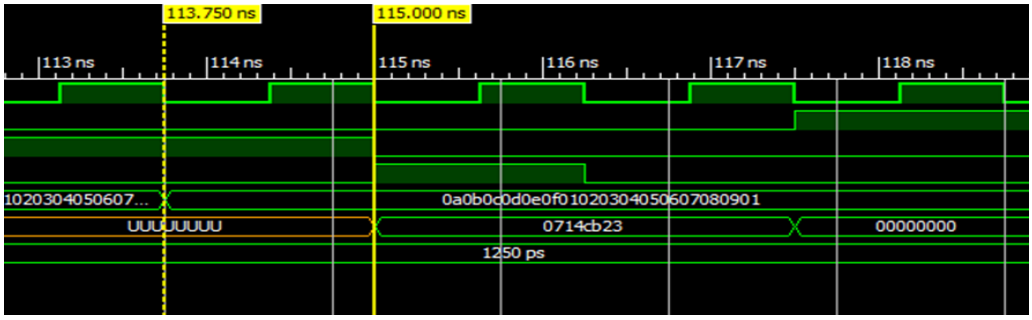


Figure 6: Clock frequency, dataIn the 4th row and the CRC32 output in the 6th row

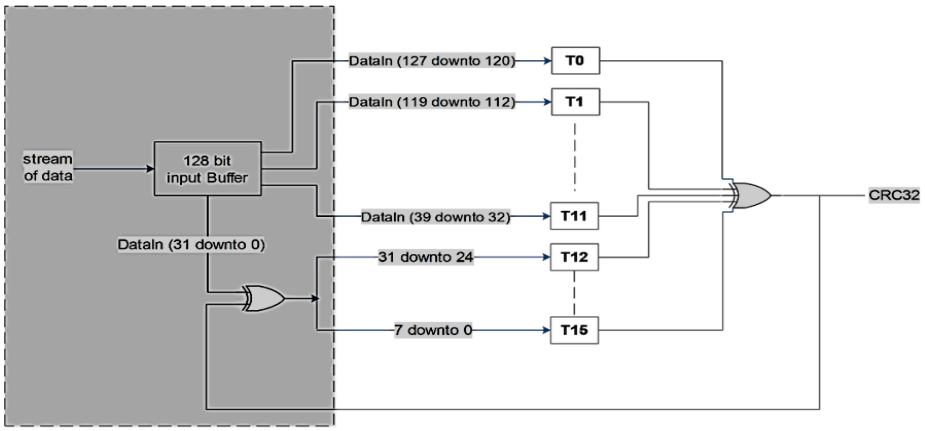


Figure 5: CRC32 using slicing-by-16

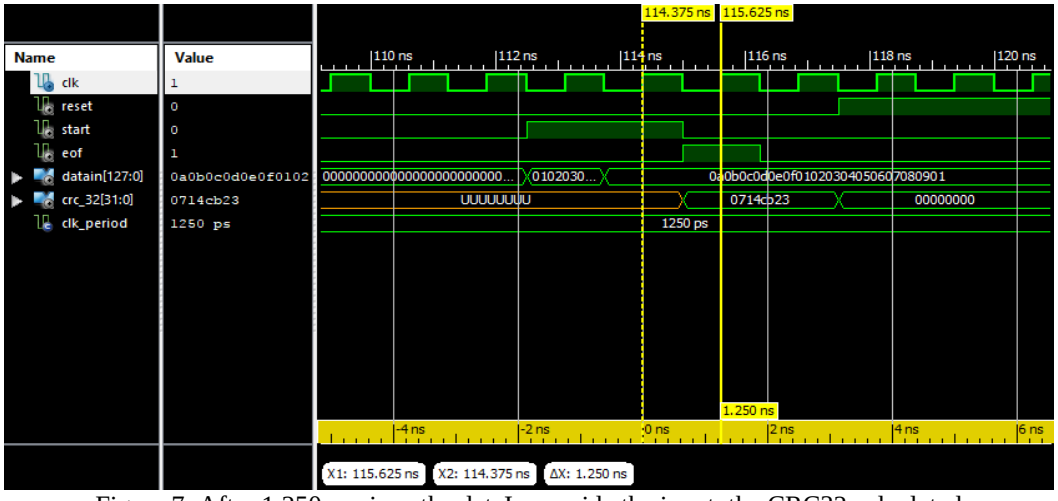
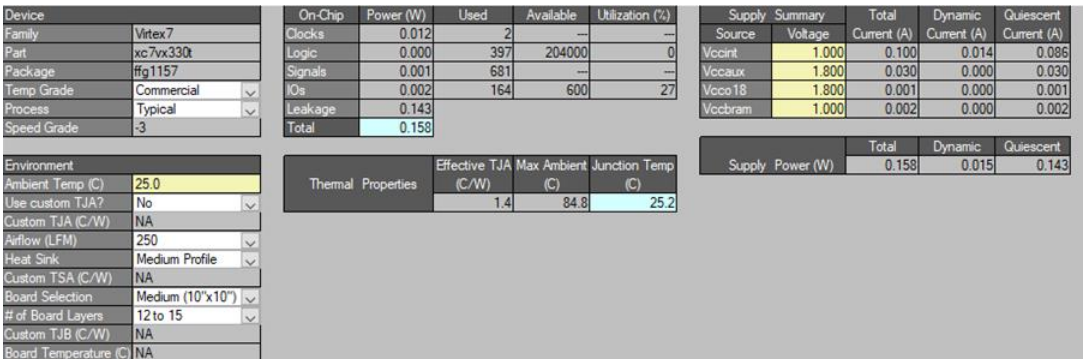


Figure 7: After 1.250 ns since the dataIn provide the input, the CRC32 calculated



Device	
Family	Virtex7
Part	xc7vx330t
Package	ffg1157
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.012	2	--	--
Logic	0.000	397	204000	0
Signals	0.001	681	--	--
IOs	0.002	164	600	27
Leakage	0.143			
Total	0.158			

Supply Summary	Total	Dynamic	Quiescent
Source Voltage	Current (A)	Current (A)	Current (A)
Vccint 1.000	0.100	0.014	0.086
Vccaux 1.800	0.030	0.000	0.030
Vcco18 1.800	0.001	0.000	0.001
Vccbram 1.000	0.002	0.000	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.158	0.015	0.143

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.4	84.8	25.2

Figure 8: Power consumption

Concolusion

the computation of the CRC32 is extremely affected by the increase of the data rates as the current processor cannot handle it. The purpose of this paper is to improve the processing speed of the CRC32 in order to encounter the new requirements and speed. The needed throughput is 100 Gbps so that in order to achieve it, the FPGA used in the design. The main reason to select FPGA is its concurrent behavior, it can execute more than one operation in parallel. Behind that, more than one algorithm has studied and the slicing by 16 algorithms selected, this algorithm reduces the computation of the CRC32 by creating lookup tables hold all the possible values of the CRC32. The code of the design and the simulation written by VHDL. The simulation test shows that; the achieved processing time is equal to 1.250 ns and the throughput is equal to 10.24 Gbps behind that, the consumed power and the device utilization are very low.

References

[1] Cisco, "The Zettabyte Era: Trends and Analysis," 2016.

[2] Sarwate, D.V."Computation of Cyclic Redundancy Checks via Table Lookup," Comm. ACM, vol. 31, no. 8, pp. 1008-1013, Aug. 1988.

[3] P. A. Anand and D. Bajarangbali, "Design of High Speed CRC Algorithm for Ethernet on FPGA Using Reduced Lookup Table Algorithm," presented at the IEEE Annual India Conference (INDICON), India 2016.

[4] M. Braun, J. Friedrich, T. Grün, and J. Lembert, "Parallel CRC computation in FPGAs," in Proc. 6th Int. Workshop Field-Program. Logic, Smart Appl., New Paradigms Compilers (FPL), London, U.K., 1996, pp. 156–165, Springer-Verlag.

[5] C. Cheng and K. Parhi, "High-speed parallel CRC implementation based on unfolding, pipelining, and retiming," IEEE Trans. Circuits Syst. II, Expr. Briefs, vol. 53, no. 10, pp. 1017–1021, Oct. 2006.

[6] C. Kennedy and A. Reyhani-Masoleh, "High-speed parallel CRC circuits," in Proc. 42nd Asilomar Conf. Signals, Syst. Comput., Oct. 2008, pp. 1823–1829.

[7] Tomas Zavodnik and Lukas Kekely, "CRC Based Hashing in FPGA Using DSP Blocks," presented at the IEEE 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, 2014.

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