COMPUTATION OF FIELD PROGRAMMABLE CYCLIC REDUNDANCY CHECKS CIRCUIT ARCHITECTURE

M.Anto Bennet[1*], Lakshmi Ravali [2] ,T.R.Sughapriya[2] ,J.Jenitta[2] , K.Vaishnavi [2] ,Priyanka Paree Alphonse[2]

[1] Faculty of Electronics and Communication Department, Vel tech, Chennai, India.

[2] UG Students of Electronics and Communication Department, Vel tech, Chennai, India.

Email: bennetmab@gmail.com

Submitted: May 27, 2017 ***Accepted: June 15, 2017*** ***Published: Sep 1, 2017***

Abstract- In this work we are going to simulate a field programmable cyclic redundancy check circuit architecture. The transmitted data or stored data must be free from error. The increased use of error correction techniques by digital communications designers has created a demand for tools to evaluate and exercise error correction coding approaches before they are committed to expensive ASICs or firmware. Cyclic redundancy check is an error detection method but it can be used only for a specific application. A field programmable circuit is one which enables a wide range of polynomial width and input port width to be used with in the same circuit. The parameters are reprogrammable and it is fully flexible. The circuit also consists of an embedded configuration controller that reduces the programming time and complexity. The hardware cost is reduced and the line speed is increased. The primary tool used is modelsim 6.1a.

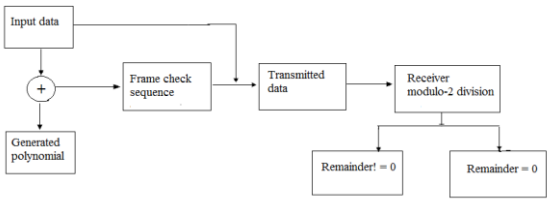
Index terms*:* Cyclic Redundancy Check (CRC), Application Specific Integrated Circuit. ASIC ..

I. INTRODUCTION

A cyclic redundancy check (CRC) or polynomial code checksum is a non-secure hash function designed to detect accidental changes to raw computer data, and is commonly used in digital networks and storage devices such as hard disk drives. A CRC-enabled device calculates a short, fixed-length binary sequence, known as the CRC code or just CRC, for each block of data and sends or stores them both together[1,3].. When a block is read or received the device repeats the calculation; if the new CRC does not match the one calculated earlier, then the block contains a data error and the device may take corrective action such as rereading or requesting the block be sent again, otherwise the data is assumed to be error free. CRCs are so called because the check (data verification) code is a redundancy (it adds zero information) and the algorithm is based on cyclic codes[2,4]. The term CRC may refer to the check code or to the function that calculates it, which accepts data streams of any length as input but always outputs a fixed-length code. CRCs are popular because they are simple to implement in binary hardware, are easy to analyse mathematically, and are particularly good at detecting common errors caused by noise in transmission channels. A CRC is an error-detecting code. Its computation resembles a polynomial long division operation in which the quotient is discarded and the remainder becomes the result, with the important distinction that the polynomial coefficients are calculated according to the carry-less arithmetic of a finite field[5,6]. The length of the remainder is always less than the length of the divisor (called the generator polynomial), which therefore determines how long the result can be. The definition of a particular CRC specifies the divisor to be used, among other things. Although CRCs can be constructed using any finite field, all commonly used CRCs employ the finite field GF. This is the field of two elements, usually called 0 and 1, comfortably matching computer architecture. The rest of this article will discuss only these binary CRCs, but the principles are more general. An important reason for the popularity of crcs for detecting the accidental alteration of data is their efficiency guarantee. typically, an n-bit crc, applied to a data block of arbitrary length, will detect any single error burst not longer than n bits (in other words, any single alteration that spans no more than n bits of the data), and will detect a fraction $1 - 2^{-n}$ of all longer error bursts. errors in both data transmission channels and magnetic storage media tend to be distributed non-randomly (i.e. are "bursty"), making crcs' properties more useful than alternative schemes such as multiple parity checks[7,8,9].

II. Proposed system

2.1 Cyclic Redundancy Check



A. FIGURE 1 Block diagram of CRC

B. 2.2 Computation of CRC

To compute an n-bit binary CRC, line the bits representing the input in a row, and position the (n+1)-bit pattern representing the CRC's divisor (called a "polynomial") underneath the left-hand end of the row. Here is the first calculation for computing a 3-bit CRC

```
11010011101100 <--- input
    1011       <--- divisor (4 bits)
----------------------
01100011101100 <--- result
```

If the input bit above the leftmost divisor bit is 0, do nothing and move the divisor to the right by one bit. If the input bit above the leftmost divisor bit is 1, the divisor is exclusive-ORed into the input (in other words, the input bit above each 1-bit in the divisor is toggled). The divisor is then shifted one bit to the right, and the process is repeated until the divisor reaches the right-hand end of the input row. Here is the last calculation:

```
00000000001110  <--- result of previous
                1011  <--- divisor
----------------------
00000000000101 <--- remainder (3 bits)
```

Since the leftmost divisor bit zeroed every input bit it touched, when this process ends the only bits in the input row that can be nonzero are the n bits at the right-hand end

of the row. These n bits are the remainder of the division step, and will also be the value of the CRC function (unless the chosen CRC specification calls for some post processing) shown in fig 1.

2.3 Designing CRC polynomials

The selection of generator polynomial is the most important part of implementing the CRC algorithm. The polynomial must be chosen to maximize the error detecting capabilities while minimizing overall collision probabilities. The most important attribute of the polynomial is its length (the number of the highest nonzero coefficient), because of its direct influence of the length of the computed checksum.

III. Field programmable CRC

3.1 Exclusive OR Tree

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission. 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. The technique is also sometimes applied to data storage devices, such as adisk drive. In this situation each block on the disk would have check bits, and the hardware might automatically initiate a reread of the block when an error is detected, or it might report the error to software. It consists of a single exclusive or gate together with some control circuitry. For bit parallel transmission, an exclusive or tree may be used, to compute the parity bit in softwareshown in fig 2.

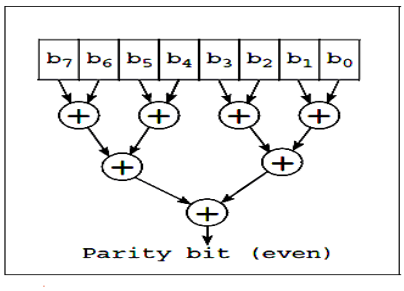


Figure 2. Exclusive or tree

Other techniques for computing a checksum are to form the exclusive or of all the bytes in the message, or to compute a sum with end-around carry of all the bytes. In the latter method the carry from each 8-bit sum is added into the least significant bit of the accumulator. It is believed that this is more likely to detect errors than the simple exclusive or, or the sum of the bytes with carry discarded. A technique that is believed to be quite good in terms of error detection, and which is easy to implement in hardware, is the cyclic redundancy check. This is another way to compute a checksum, usually eight, 16, or 32 bits in length, that is appended to the message. The following facts about generator polynomials are proved in [PeBr] and/or [Tanen]: If Gcontains two or more terms, all single-bit errors are detected. If is a factor of G, all errors consisting of an odd number of bits are detected. An r-bit CRC checksum detects all burst errors of length (A burst error of length ris a string of rbits in which the first and last are in error, and the intermediate bits may or may not be in error.). The generator polynomials used by some common CRC standards. The "Hex" column shows the hexadecimal representation of the generator polynomial shown in table 1.

Table 1 Generator polynomials of some CRC codes

Common Name	r	Generator	
		Polynomial	Hex
CRC-12	12	$x^{12}+x^{11}+x^3+x^2+x+1$	80F
CRC-16	16	$x^{16}+x^{15}+x^2+1$	8005
CRC-CCITT	16	$x^{16}+x^{12}+x^5+1$	1021
CRC-32	32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+$ $x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	04C11DB7

The CRC standards differ in ways other than the choice of generating polynomial. Most initialize by assuming that the message has been preceded by certain nonzero bits, others do no such initialization. Most transmit the bits within a byte least significant bit first, some most significant bit first. Most append the checksum least significant byte first, others most significant byte first. Some complement the checksum.

The usual solution to this problem is for the sender to complement the checksum before appending it. Because this makes the remainder calculated by the receiver nonzero (usually), the

remainder will change if trailing 0's are inserted or deleted. Using the "mod" notation for remainder, we know that

$(Mxr + R) \bmod G = 0.$

$(Mxr + R) \bmod G = (Mxr + (xr-1 + xr-2 + \ldots + 1 - R)) \bmod G$

$= ((Mxr + R) + xr - 1 + xr - 2 + \ldots + 1) \bmod G$

$= (xr - 1 + xr - 2 + \ldots + 1) \bmod G.$

Thus the checksum calculated by the receiver for an error-free transmission should be

$(xr - 1 + xr - 2 + \ldots\ldots\ldots + 1) \bmod G.$

This is a constant (for a given G). For CRC-32 this polynomial, called the residualor residue, is

$x31+x30+ x26+ x25+ x24+ x18+ x15+ x14+ x12+$

$x11+ x10+ x8+ x6+ x5+ x4+ x3+ x\ 1$,or hex C704DD7B [Black]

To develop a hardware circuit for computing the CRC checksum, we reduce the polynomial division process to its essentials. The process employs a shift register, which we denote by CRC. This is of length r(the degree of G) bits, not as you might expect. When the subtractions(exclusive or's) are done, it is not necessary to represent the high-order bit,because the high-order bits of Gand the quantity it is being subtracted from are both 1. The division process might be described informally as follows:

Initialize the CRC register to all 0-bits.

Get first/next message bit m.

If the high-order bit of CRC is 1,

Shift CRC and mtogether left 1 position, and XOR the result with the low-order rbits of G. Otherwise, Just shift CRC and mleft 1 position.If there are more message bits, go back to get the next one. It might seem that the subtraction should be done first, and then the shift. It would be done that way if the CRC register held the entire generator polynomial, which in bit form is bits. Instead, the CRC register holds only the low-order rbits of G, so the shift is done first, to align things properly.

IV. Polynomial division circuit

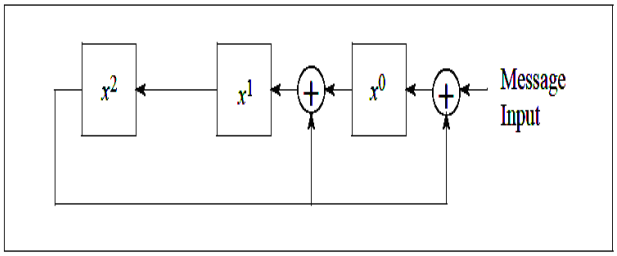


Figure 3. Polynomial division circuit forG=X3+X+1.

The three boxes in the figure represent the three bits of the CRC register. When a message bit comes in, if the high-order bit (x2 box) is 0, simultaneously the message bit is shifted into the x0 box, the bit in x0 is shifted to x1, the bit in x1 is shifted to x2, and the bit in x2 is discarded. If the high-order bit of the CRC register is 1, then a 1 is present at the lower input of each of the two exclusive orgates. When a message bit comes in, the same shifting takes place but the three bits that wind up in the CRC register have been exclusive or'ed with binary 011. When all the message bits have been processed, the CRC holds Mmod G. If the circuit of Figure 3 were used for the CRC calculation, then after processing the message, r(in this case 3) 0-bits would have to be fed in. Then the CRC register would have the desired checksum,Mxrmod G. But, there is a way to avoid this step with a simple rearrangement of the circuit. Instead of feeding the message in at the right end, feed it in at the left end, rsteps away, as shown in Figure 4. This has the effect of premultiplying the input message Mby xr. But premultiplying and postmultiplying are the same for polynomials. Therefore, as each message bit comes in, the CRC register contents are the remainder for the portion of the message processed, as if that portion had r0-bits appended shown in fig 5.

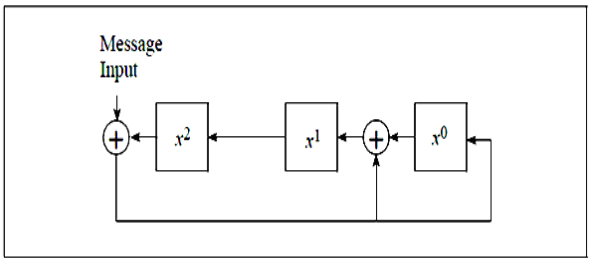


Figure 4. CRC circuit for G=X3+X+1.

Figure 5 shows the circuit for the CRC-32 polynomial.

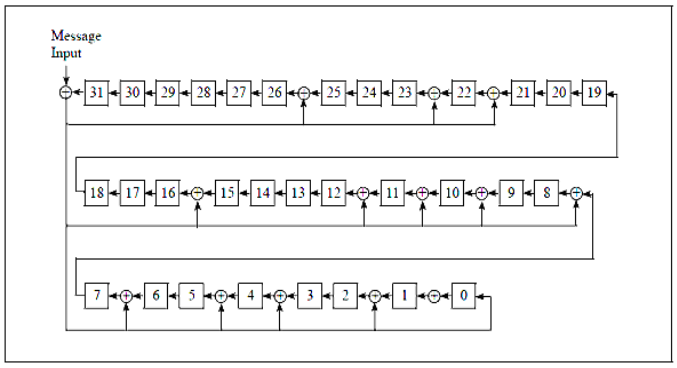


Figure 5. CRC circuit for CRC-32

V. Design of field programmable redundancy circuit

5.1. Frame check sequence

The CRC is a more complicated, but robust, error checking algorithm. The main idea behind the CRC algorithm is to treat a message as a binary bit stream and divide it by a fixed binary number. The remainder from this division is considered as the checksum. Like in division, the CRC calculation is also an iterative process. The only difference is that these operations are done on modulo arithmetic based on mod2. The checksum is a unique number associated with a message or a particular block of data containing several bytes. Whether it is a data packet for communication or a block of data stored in memory, a piece of information like the checksum helps to validate it before processing. The simplest way to calculate a checksum is to add all of the data bytes present in the message. However, this method of checksum calculation fails when the message is modified by inverting or swapping groups of bytes. The generator polynomial is given by G(x) and the message polynomial is given by M(x). To perform the CRC calculation, a suitable divisor is first selected. This divisor is called the generator polynomial. Since the CRC is used to detect errors, a suitable generator polynomial of suitable length should be chosen for a given application, as each polynomial has different error detection capabilities.

VI. Programmable cyclic redundancy check circuit

6.1 Circuit Diagram

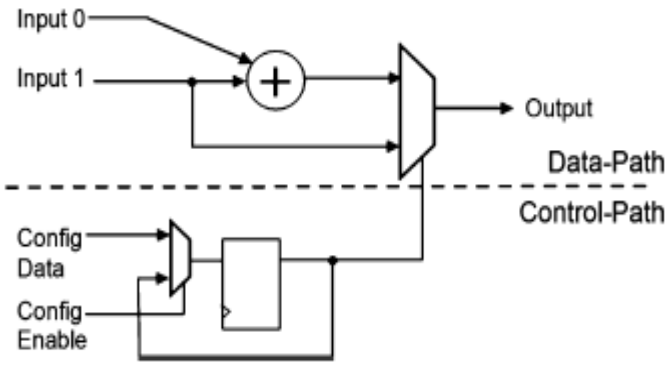
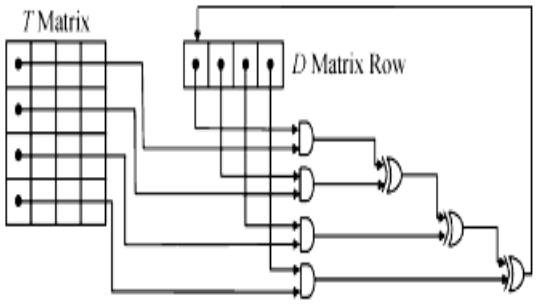


Figure 6. Programmable CRC cell array

The apparatus includes at least a first device for calculating a cyclical redundancy check value on a full set of bits of input data and producing a first value and a second device for calculating a cyclical redundancy check value on a first subset of the full set of bits of input data and producing a second value. A multiplexer, coupled to the first and second devices, receives the first and second values and a selection input to the multiplexer selects one of the values for further transmission. The apparatus also includes a register having an input coupled to the output of the multiplexer. A second multiplexer can be coupled between the first multiplexer and the register for selecting the selected value from the first multiplexer or the output from the register for input to the register. The output from the register may be fed back to the first and second devices through a logic gate which selects an initialization input for the initial input bits and thereafter selects the output value from the registershown in fig 6.

VII. Matrix computation



Figrue 7. D Matrix calculation

The D matrix effectively forms the logic array of cyclic redundancy check calculator circuit. The configurable XOR array is comprised of interconnected cells, corresponding to the D matrix. Each cell can be configured as an XOR gate (a "1" in D), or as a basic input to output connection (a "0" in D). The data-path can be configured so that the output will XOR the two inputs, or simply output Input1. The control-path contains a configuration register which selects the data-path function and is programmed via the Config Datainput when the Config Enableinput is set high. The computation of the D matrix is an iterative process, where the computation of each row is based on the result from the previous row.Each value in G is multiplied (ANDed) with the corresponding value in T. The results are XORed together, producing a D matrix of 0's or 1's. The position of the 1's in D determines the position of XOR gates within the logic array while is the width of the input port in bits. Enabling programmability for parallel CRC computation requires the D matrix to be configurable for all known generator polynomials G(x). Furthermore, configuration logic for the configurable XOR array is required to adjust the input and CRC sizesshown in fig 7..

VIII. Field programmable crc architecture

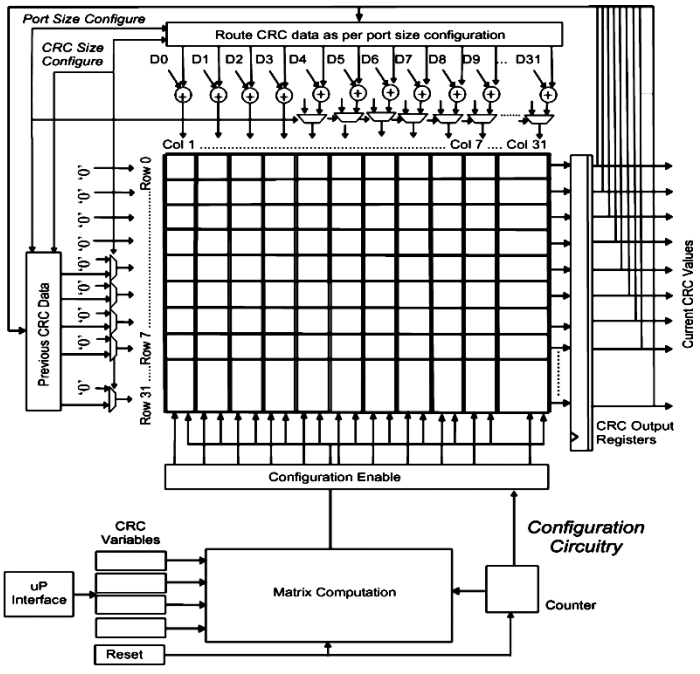


Figure 8. Field programmable CRC architecture

The circuit is composed of six main components, the programmable input and feedback multiplexers, the configurable XOR array, the array configuration circuit and the CRC configuration processor. The top of the diagram shows the logic associated with the CRC cell array. The input data enters the array down the columns and the outputs are formed along the rows. The current CRC value is held in a register at the array output, which is fed back and XORed with the input data of the next clock cycle as part of the CRC computation process. The outputs are then stored in the registers for the next clock cycle.The CRC configuration parameters are passed via a microprocessor interface. The desired CRC polynomial G(x) and the input port size are stored in registers. By selecting a signal Generate Matrix, a process is initiated that computes the configuration data and configures the CRC cell array with the required datashown in fig 8.

IX. Experimental results

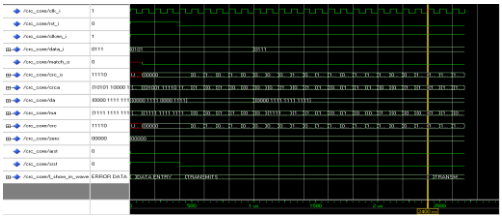


Fig 9: 4 Bit Data with Error

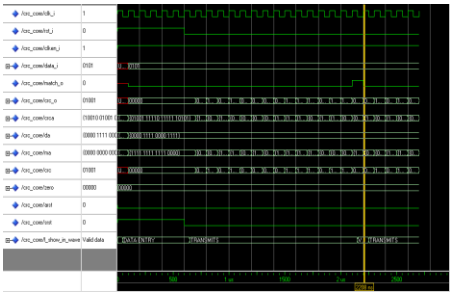


Fig 10. 4 Bit Data without Error

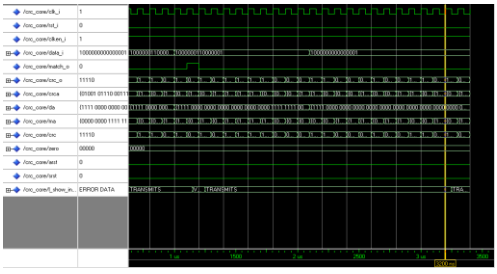


Fig 11: 16 Bit Data with Error

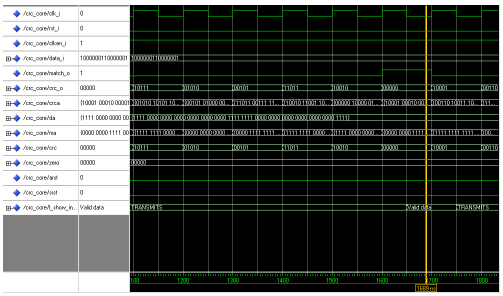


Fig 12: 16 Bit Data without Error

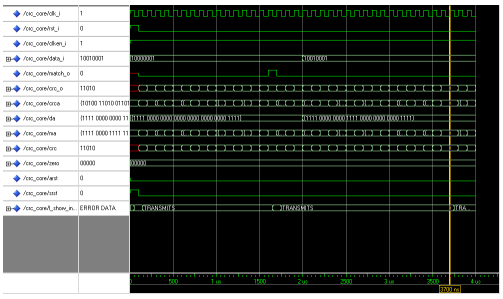


Fig 13: 8 Bit Data with Error

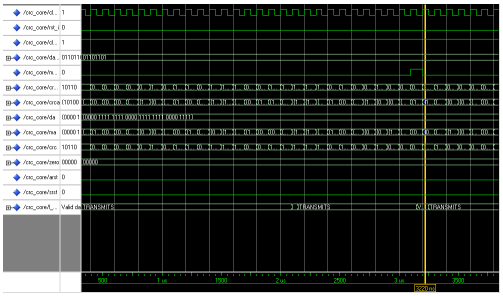


Fig 14:8 Bit Data without Error

Most of the modern communication protocols use some error detection algorithms. Cyclic Redundancy Check, or CRC, is the most popular one among these. CRC properties are defined by the generator polynomial length and coefficients. The protocol specification usually defines CRC in hex or polynomial notation. For example, CRC-8 used in ATM HEC field is represented as 0x07 in hex notation or as G(X)=X^8 + X^2 + X^1 +1. in the polynomial notation.The code given below is capable of computing ,CRC-8 for 32 bit input.The module need 32 clock cycles for the computationThe simulation result for CRC using single execution unit and parallel execution units are obtained by Xilinx ISE 9.1i.Parallel execution units are formed by 2,4 and 8 execution units. For 32-bit CRC, remainder is A4568879 and according to xecution unit got the remainder at clock cycle.The simulation result are as follows in fig 9-14.

X. Conclusion

The aim of this work to construct an architecture where CRC parameters were fully programmable. The matrix is based on computation technique. The circuit uses an embedded configuration controller where the programming task of the logic array is reduced and its enables different CRC polynomials and I/O port. The performance and flexibility is high when compared to other technologies. The architecture is highly reconfigurable. Run time programmability enables configuration time to be reduced and the hardware cost is low. The throughput is 4.6 Gbps which is increased compared to previous method. FPGAs have been slower, less energy efficient and generally achieved less functionality than their fixed ASIC counterparts. FPGA architectures, are dominated by interconnect. This makes them far more flexible but also far more complex to design for. The inherent parallelism of the logic resources on an FPGA allows for considerable computational throughput even at a low MHz clock rates. The flexibility of the FPGA allows for even higher performance by trading off precision and range in the number format for an increased number of parallel arithmetic units. In future we are going to increase the throughput by reducing clock rate.

REFERENCES

[1] Aizat Azmi, Ahmad Amsyar Azman, Sallehuddin Ibrahim, and Mohd Amri Md Yunus, “Techniques In Advancing The Capabilities Of Various Nitrate Detection Methods: A Review”, International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 2, June 2017, pp. 223-261.

[2] Tsugunosuke Sakai, Haruya Tamaki, Yosuke Ota, Ryohei Egusa, Shigenori Inagaki, Fusako Kusunoki, Masanori Sugimoto, Hiroshi Mizoguchi, “Eda-Based Estimation Of Visual Attention By Observation Of Eye Blink Frequency”, International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 2, June 2017, pp. 296-307.

[3] Ismail Ben Abdallah, Yassine Bouteraa, and Chokri Rekik , “Design And Development Of 3d Printed Myoelctric Robotic Exoskeleton For Hand Rehabilitation”, International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 2, June 2017, pp. 341-366.

[4] S. H. Teay, C. Batunlu and A. Albarbar, “Smart Sensing System For Enhanceing The Reliability Of Power Electronic Devices Used In Wind Turbines”, International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 2, June 2017, pp. 407- 424

[5] SCihan Gercek, Djilali Kourtiche, Mustapha Nadi, Isabelle Magne, Pierre Schmitt, Martine Souques and Patrice Roth, "An In Vitro Cost-Effective Test Bench For Active Cardiac Implants, Reproducing Human Exposure To Electric Fields 50/60 Hz", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 1- 17

[6] P. Visconti, P. Primiceri, R. de Fazio and A. Lay Ekuakille, "A Solar-Powered White Led-Based Uv-Vis Spectrophotometric System Managed By Pc For Air Pollution Detection In Faraway And Unfriendly Locations", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 18- 49

[7] Samarendra Nath Sur, Rabindranath Bera and Bansibadan Maji, "Feedback Equalizer For Vehicular Channel", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 50- 68

[8] Yen-Hong A. Chen, Kai-Jan Lin and Yu-Chu M. Li, "Assessment To Effectiveness Of The New Early Streamer Emission Lightning Protection System", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 108- 123

[9] Iman Heidarpour Shahrezaei, Morteza Kazerooni and Mohsen Fallah, "A Total Quality Assessment Solution For Synthetic Aperture Radar Nlfm Waveform Generation And Evaluation In A Complex Random Media", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 174- 198

[10] P. Visconti ,R.Ferri, M.Pucciarelli and E.Venere, "Development And Characterization Of A Solar-Based Energy Harvesting And Power Management System For A Wsn Node Applied To Optimized Goods Transport And Storage", International Journal on Smart Sensing and Intelligent Systems., VOL. 9, NO. 4, December 2016 , pp. 1637- 1667

[11] YoumeiSong,Jianbo Li, Chenglong Li, Fushu Wang, "Social Popularity Based Routing In Delay Tolerant Networks", International Journal on Smart Sensing and Intelligent Systems., VOL. 9, NO. 4, December 2016 , pp. 1687- 1709

[12] Seifeddine Ben Warrad and OlfaBoubaker, "Full Order Unknown Inputs Observer For Multiple Time-Delay Systems", International Journal on Smart Sensing and Intelligent Systems., VOL. 9, NO. 4, December 2016 , pp. 1750- 1775

[13] Rajesh, M., and J. M. Gnanasekar. "Path observation-based physical routing protocol for wireless ad hoc networks." International Journal of Wireless and Mobile Computing 11.3 (2016): 244-257.

[14]. Rajesh, M., and J. M. Gnanasekar. "Congestion control in heterogeneous wireless ad hoc network using FRCC." Australian Journal of Basic and Applied Sciences 9.7 (2015): 698-702.

[15]. Rajesh, M., and J. M. Gnanasekar. "GCCover Heterogeneous Wireless Ad hoc Networks." Journal of Chemical and Pharmaceutical Sciences (2015): 195-200.

[16]. Rajesh, M., and J. M. Gnanasekar. "CONGESTION CONTROL USING AODV PROTOCOL SCHEME FOR WIRELESS AD-HOC NETWORK." Advances in Computer Science and Engineering 16.1/2 (2016): 19.

[17]. Rajesh, M., and J. M. Gnanasekar. "An optimized congestion control and error management system for OCCEM." International Journal of Advanced Research in IT and Engineering 4.4 (2015): 1-10.

[18]. Rajesh, M., and J. M. Gnanasekar. "Constructing Well-Organized Wireless Sensor Networks with Low-Level Identification." World Engineering & Applied Sciences Journal 7.1 (2016).

[19] L. Jamal, M. Shamsujjoha, and H. M. Hasan Babu, “Design of optimal reversible carry look-ahead adder with optimal garbage and quantum cost,” International Journal of Engineering and Technology, vol. 2, pp. 44–50, 2012.

[20] S. N. Mahammad and K. Veezhinathan, “Constructing online testable circuits using reversible logic,” IEEE Transactions on Instrumentation and Measurement, vol. 59, pp. 101–109, 2010.

[21] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. A. Perkowski, “Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis,” IEEE Trans. on CAD of Integrated Circuits and Systems, vol. 25, no. 9, pp. 1652–1663, 2006.

[22] F. Sharmin, M. M. A. Polash, M. Shamsujjoha, L. Jamal, and H. M. Hasan Babu, “Design of a compact reversible random access memory,” in 4th IEEE International Conference on Computer Science and Information Technology, vol. 10, june 2011, pp. 103–107.

[23] Dr. AntoBennet, M, Sankar Babu G, Suresh R, Mohammed Sulaiman S, Sheriff M, Janakiraman G ,Natarajan S, “Design & Testing of Tcam Faults Using T_H Algorithm”, Middle-East Journal of Scientific Research 23(08): 1921-1929, August 2015 .

[24] Dr. AntoBennet, M "Power Optimization Techniques for sequential elements using pulse triggered flipflops", International Journal of Computer & Modern Technology , Issue 01 ,Volume01 ,pp 29-40, June 2015.