National Conference on Innovative Trends in Science and Engineering (NC-ITSE'16)
Volume: 4 Issue: 7
ISSN: 2321-8169
205 - 210

Multi-Modulus Adder Implementation and its Applications

Aditya Urkude,
M. Tech. VLSI Design,
VIT University, Vellore, TN.
gardesh8@gmail.com

Gargi Deshmukh,
M. Tech. VLSI Design
VIT University, Vellore, TN
adisurkude92@gmail.com,

Minal Shinde
M. Tech. VLSI Design
VIT University, Vellore, TN
minalvshinde92@gmail.com

Anjali Deshmukh
Associate Prof. Dept. of Electronics,
Shri Shivaji College, Akola, MS
ajaydesh@rediffmail.com

***Abstract*—** The parallel prefix adders has wide applications in operation of arithmetic computations specially in multi-modulus applications in order to achieve high speed, compact area and time efficiency. Two architectures of multi-modulus adders that support the most common module cases in RNS channels, that is, modulo 2^n -1, 2^n and 2^n + 1 are implemented. The architectures that performs parallel prefix carry computation units are composed of log2n levels.The experimental results shows that the implemented adders are significantly faster and/or lesser in area. The new Multi-modulus subtractor and multiplier is proposed that rely on the use of the implemented multi-modulus adders. The aim of this project is to implement proposed work experimentally and compare their performance with existing module in respect to area, power and delay.

Keywords- *Kogge Stone Adder, Ladner Fischer, Residue number system(RNS), Parallel prefix computation, Parallel-prefix adder(PPA).*

*****

I. INTRODUCTION

The Arithmetic Operation is very important in our day to day life. All the arithmetic work is carried on addition, subtraction, multiplication and division. Basically addition is used as a reference and using addition technique all other techniques of subtraction, division and multiplication can be carried out. As, subtraction can be computed by the addition of a number and two's complement of the other number, similarly is the multiplication and division is carried out. So adder contributes to the fundamental model.

It depends upon the adder that is how efficiently adder has been developed to carry addition effectively with minimum power, delay calculations. The parallel prefix adder plays a very important role in developing such adders with all its constraints meet in power, delay calculations. Basically two adders has been implemented in this paper Kogge stone adder[2] and Lander-Fisher[4] adder. Kogge stone adder[2] uses its fewest logic levels for implementation and lander-fischer [4]is used for high performance addition operation. The introduction to the RNS (Residue number System)[3]has also been included in this paper.RNS system consists of three channels 2^n, 2^n-1,2^n+1[1]. In this modulo 2^n arithmetic can be straight forwardly applied from the given arithmetic results upto n bits but the design of 2^n+1 and 2^n-1 plays very important role as they are of special importance and has a wide application in designing any multi-modulo adder .while the design of 2^n-1 is performed in the regular way, modulo 2^n+1 operation is complex operation ,therefore it is performed using diminished-1[1] operation in modulo 2^n+1 .Use of modulo 2^n+1 in adder[5] plays important role in adder while addition of a number in zero operand. The objective in developing is reduction in carry-chain path for the adder computation, Area reduction for N-bit parallel adder design with efficient outcome, Time delay constrain in carry propagation minimization for high speed computation. Power can be reduced by minimizing the computational block.

II. RELATED WORK

2.1 Architecture for Modulo 2^n+1 arithmetic:

Modulo of 2^n+1 arithmetic has found very wide applications in various fields specially in cryptography. These modulo is a part of the residue number system which consists of three channels 2^n, 2^n+1, 2^n-1. Basically RNS [1]is an arithmetic number system which divides a number into residues (parts) and performs arithmetic operations concurrently for each residue (parts), and it does not need any carry propagation among them. Because of these it has increased the speed over various binary operations. Among the three channels in RNS arithmetic systems, modulo 2^n is obtained in straight forward way limiting upto n bits and 2^n-1is complicated then, 2^n but not as 2^n+1, in 2^n-1 zero values are taken into consideration while in 2^n+1, this has to deal with operands one bit wider than the other two, to avoid this difficulty diminished-1[1] representation is used, so that it can be used for consideration of zero operands in 2^n+1. In diminished-1 level operand is represented as a_zA, where a_z is a single bit, which is often used for indicating the zero value, and A is an n-bit vector and called as the number part. A diminished -1 representation allows to limit the operation upto n bits, where A_{nor} is normal weighted representation which belongs in the range from 0 to 2^n, and it is in diminished -1 notation represented by a_z and A= A_{nor}-1, when A_{nor} is not equal to zero else if we have to

represent zero values of $A_{nor,}$ we have to use the combination a_z=1 and A=0 .The following cases is used to determine the sum of the diminished-1 adder :

1. Diminished-1 adder module is used when both the inputs a_z and b_z are not equal zero, their number parts A and B are added.
2. Result is equal to non-zero operand when any one of both the inputs is zero.
3. Result is zero when both operands are zero.

Taking care of diminished-1 representation for 2^n+1 addition, an extra carry look ahead adder block[2] is added along with integer adder. This carry look ahead block used to compute carry. After inverting, obtained carry is given as carry input to integer adder. In implemented IEAC along with parallel prefix adder some extra logic levels are included in order to avoid race around conditions. Although these adders avoids race around conditions, use of an extra logic level makes them slower as compared to their corresponding parallel prefix adder. It is shown that recirculation of carry of is computed in same parallel prefix adder so that we can avoid use of extra prefix level [1] in order to make addition faster. But it may increase area as several carry computation required separate parallel prefix adder for carry computation unit.

Although these adders are complex in area but are advantageous over other for speed and hence are widely used in many applications like multiplication, subtraction, squarer, excess-3[4] code converter etc. It is shown that we can use two adders simultaneously in order to compute carry and addition separately to increase the speed of operation but results in more area. So it is shown that hybrid multiplier can be used in order to reduce area.

III. BACKGROUND

1. Parallel prefix computation

The most basic block of any digital logic circuit is adder. In most of the case performance of adder decides the overall performance of the system however the performance of adders becomes critical as number of bits increases. There are many straight forward linear adders like ripple carry adder but the major disadvantage of these adders is that they are very slow. Thus to improve the speed of operation slight changes in algorithm of carry look ahead adder[5] as well as carry select adder for carry propagation are done in order to improve speed but results in increases area as well as complexity.

a. Parallel prefix adders:-

Parallel prefix adders [1] uses simple cell implementation and maintain uniformity in logic connectivity.

Frequent use of adder in arithmetic operations results in high power consumption which affects the reliability of circuit. High power dissipation results in the form of excessive heat generation in circuit. In order to solve this cooling circuit is required which increases the cost of the circuit. In order to maintain reliability low power and high speed computational circuit is required .and these specifications are achieved using parallel prefix adders.

Performance of adder in perspective with area and power can be improved using parallel prefix adder and hence its use is always preferable for high speed applications. Parallel prefix computation is performed in three necessary steps which are shown below

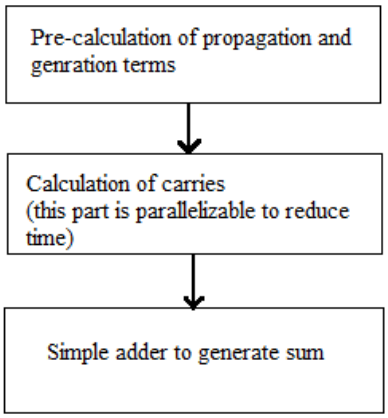

Fig.1. Structural flow diagram of Parallel prefix Adder

Parallel prefix adder [1] is similar carry look ahead adder. The rotation of the carries in parallel prefix adder can be designed in different ways to perform addition in more optimized way to get minimized power and delay. Parallel prefix adder reduces delay and used for high performance arithmetic structures in industries. The parallel prefix addition is done in three steps as mentioned below.

1. Pre-processing stage
2. Carry generation stage
3. Post processing stage

1. Pre-processing stage

In this stage carry generation bits g_i, carry propagation bits p_i, and half sum bits h_i for every n bit is calculated.

$$g_i = a_i \cdot b_i \quad \text{.....(i)}$$

$$p_i = a_i \oplus b_i \quad \text{.....(ii)}$$

Where, $\oplus$ denote logical AND, exclusive-OR respectively.

2. Carry generation stage

This stage computes the carry signal. Execution is done in parallel form. After the computation of carries in parallel they are divided into smaller pieces. Carry operator contains two AND gates, one OR gate. It uses propagate and generate as intermediate signals.

$$p_{(i:k)} = p_{(i:j)} \cdot p_{(j-1:k)} \quad \text{.....(iii)}$$

$$g_{(i:k)} = g_{(i:j)} + (g_{(j-1:k)} \cdot p_{(i:j)}) \quad \text{.....(iv)}$$

Carry computation stage can be transformed into parallel prefix problem using o operator, which is associated with pairs of generation and propagation and is defined as

 ISSN: 2321-8169
205 - 210

$(g_{(k:j)},p_{(k:j)})= (gk.pk)o\ (g_{k-1},p_{k-1})\ o\ .....o(g_j,p_j)$…….(v)

Since every carry Ci=gi:0, three algorithms have been implemented to compute carry using only o operator.

3. Post processing stage

This final stage is used to compute summation of input bits. It is same as all adder and final sum is computed using following equations

$S_i = P_i \oplus C_{i-1}$…….(vi)

$C_{i-1}=(P_i . C_0) + g_i$……(vii)

Some of the parallel prefix adders are Kong-Stone parallel prefix adder, Lander-Fisher adder, Brent-Kung adder, spares-4 parallel prefix adder, Spanning carry look ahead adder.

The basic cells those are used to implement all parallel prefix adders basic blocks are shown below-

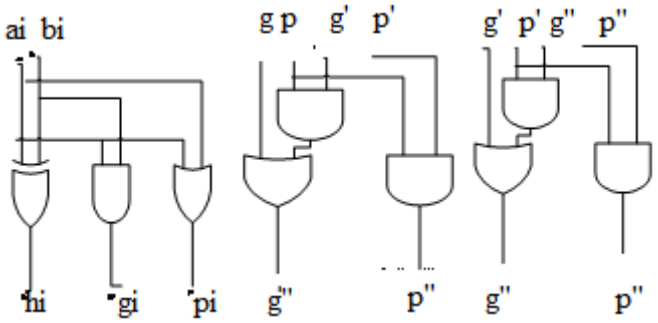

Fig. 2.The logic-level basic cells in parallel-prefix adders

b. Kogge –Stone Parallel prefix adder:-

This type of parallel prefix adder uses least logic levels among all parallel adders. KS adder[2] is fast adder design as it generates carry in $O(\log_2 N)$ time but has large area and minimum fan-out. Basic architecture of KS adder is as shown below

No of carry stages:- $\log_2 N$
Total number of cells:- $n\log_2 n$
Maximum fan-out :- 2

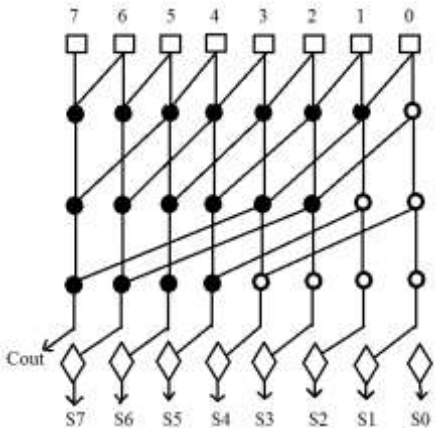

Fig.3. 8-bit Kogge-stone parallel prefix structure for integer adder

The ◊ symbol in the diagram indicates simple integer addition and ● indicates carry propagation which is used as input to the next block of carry calculation terms while ○ indicates final terms of middle stage which is directly given as input to final adder stage.

c. Lander-Fischer Adder:-

This architecture sits between Brent-Kung adder[8] and spanning tree[9]. This uses less number of logic block but has large fan-out. In this adder case instead of carry propagation carry overlap operation is performed. Delay for this adder is given by $\log_2 n+1$.

No of carry stages:- $\log_2 N$
Total number of cells:- $(n/2)\log_2 n$
Maximum fan-out :- n/2

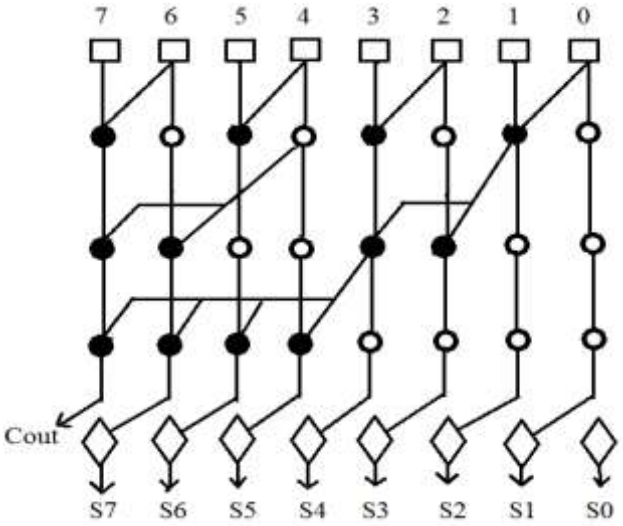

Fig.4. 8-bit Ladner-Fischer PPA structure

d. Sparse Adder:-

As number of bits increases above adders becomes more complicated and hence these adders can be replace by sparse adder as it significantly reduces area and complexity of logic without compromising delay. Sparse adder[8] design mainly consists of parallel prefix adder along with carry select blocks which are used to compute carry. As carries are calculated only at the boundary of blocks it significantly saves the area required for carry computation. Figure shows the 16-bit sparse adder with bit carry select block. As carry select block computes carry for 4-4 bits above adder is called as sparse-4 parallel prefix adder. Depending upon the required constraint we can select the cs-blocks[8] with more inputs. This block consists of multiplexer which is used to select carry depending upon carry input of previous stage which is given as select line to the multiplexer used. Combinational logic diagram of carry select block is as shown below.

In this adder we are using carry save block which avoids race around conditions. In carry save blocks two outputs are generated and are given to multiplexer which selects the

 ISSN: 2321-8169
205 - 210

required the output depending upon carry which is acting like select line for multiplexer. The block of CSA is shown in fig 5

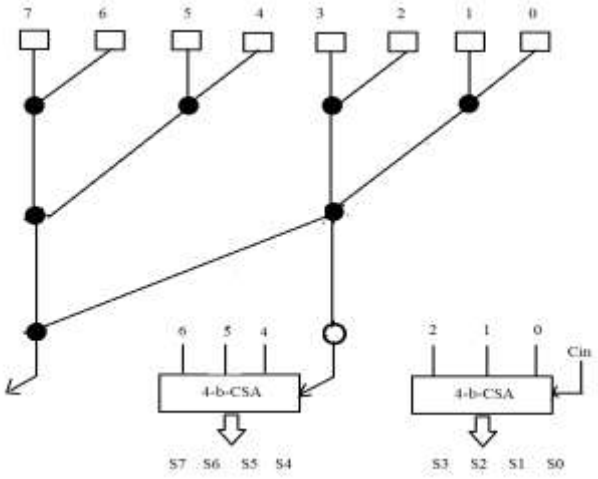

Fig.5. Structure of 8-bit SPP adder

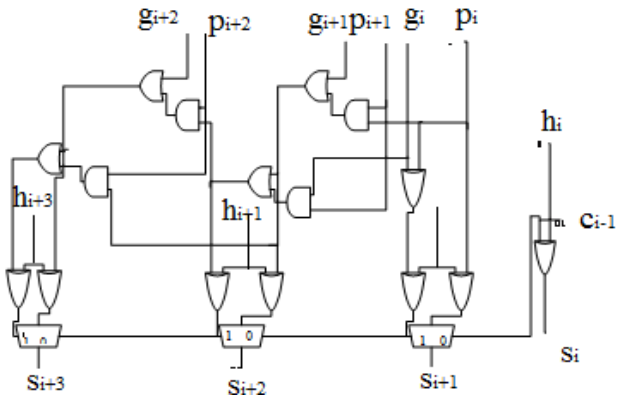

Fig.6. Logic Levels of Carry save Block

IV. METHODOLOGY & IMPLIMENTATION

1. Multi-modulus Subtractor

To perform multi-modulus subtraction[1], it needs to be divided into three different cases as follows:

a. 2^n arithmetic: R= $|B-A|_{2^n}$ = $|B+\bar{A}+1|_{2^n}$ where $\bar{A}$ is bitwise complement of A.

b. 2^n-1 arithmetic: R= $|B-A|_{2^n-1}$= $|B+(2^n-1)-A|_{2^n-1}$

c. 2^n+1 arithmetic: R=|(B+1)-(A+1)-1|=$|B+(2^n-1)-A+1|_{2^n+1}$= $|B+\bar{A}+1|_{2^n+1}$

So from above three cases, multi-modulus subtracter can be implemented using all the above adders simply by complementing A. As described above modulo 2^n case required to add one along with complementation of A. so by modifying equation used for calculating generation and propagation term.

$g_0= (a_0 \wedge b_0)\&\sim(n/m)|a_0\&b_0$……
$p_0= a_0 \wedge b_0 \wedge \sim(n/m)$……..
where (n/m)=0 for modulo 2^n

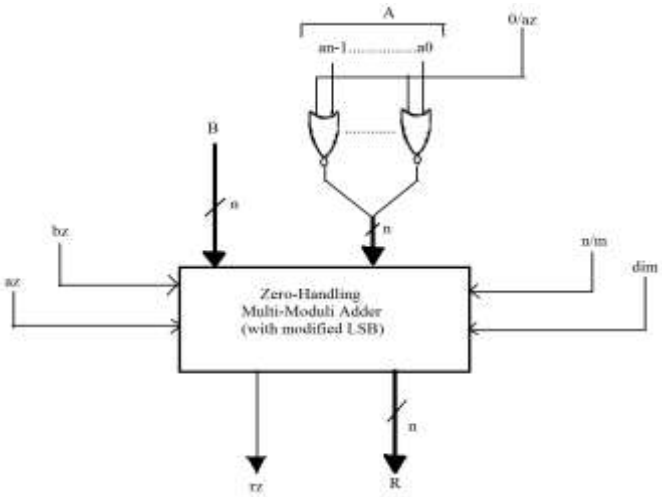

Fig. 6. Structure of Multi-modulo subtractor

2. Multi-modulus multiplier

Multiplication is one of the important intensive and core function in arithmetic operations. Multiplier serves as basic building block of many DSP Processors and also efficiently used in Image Processing. It computes the significant product result but it also highly contribute to time delay in systems. Hence the overall performance of system is depends on the multiplier computation. Multiplication and squarer functions are used significantly in applications like DSP, multimedia and image processing [3]-[4].

Generally multiplication is carried out as the repetitive addition of multiplier upto multiplicand while shifting the bits in every 'n' number of cycles. This is what the conventional number systems are multiplied with adding and shifting process, consumes time delay in computation.

Parallel multipliers are highly efficient for the design constrains of chip area, power dissipation and the speed of operation. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. Hence to reduce number of partial products to be added and to achieve speed of operation different parallel multipliers architectures were designed like Booth multiplier algorithm, Wallace tree multiplier[6], series-parallel architecture, Braun array multiplier.

The proposed architecture in this project for multiplication is subjected to the 8 bit parallel multiplication, which is carried out with repetitive addition of operand over the multiplicand value. The 8 bit multiplication gives the resultant product of 16 bit output. In this architecture the 16 bits are split up into two 8 bits. The lower 8 bits are computed with the repetitive addition 8 bit operand using the parallel prefix adder over and over till the last computation stage. And remaining 8 bits appended as the computation of carry addition in the last product bits. The carry occurred in every stage is appended with the rest of zero bits and is carried ahead to add with the next carry occurred bit in the following stage.

It is advantageous over existing multipliers as it involves parallel multiplication. In existing module requires adders with different bits and hence there is increase in area as well as power. While in proposed multiplier same adder is repeated

 ISSN: 2321-8169
205 - 210

again and again which reduces the area as well as power at the cost of slight increase of delay.

In proposed multiplier we are comparing the multiplier and multiplicand and if multiplier is less than multiplicand is added multiplier times to get the resultant eight bits in LSB while remaining eight bits are calculated simply by adding carry if exist. Algorithm for proposed multiplier is as shown below-

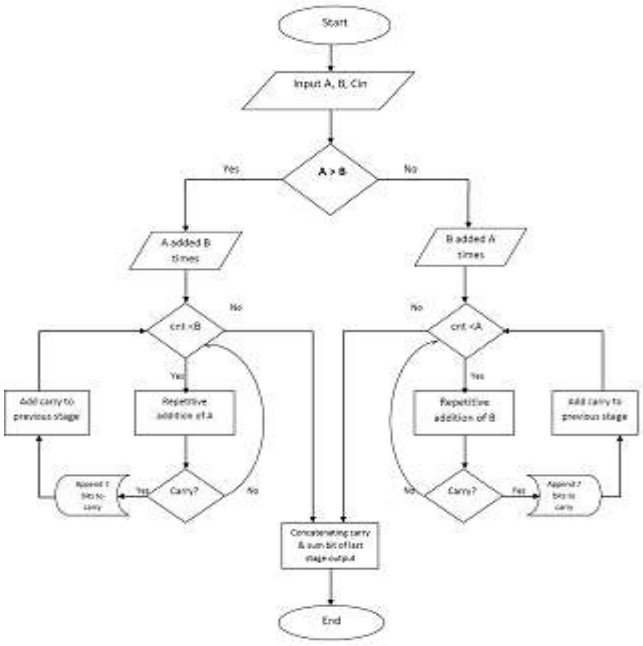

Fig.7. Data flow diagram of proposed Multiplier

V. RESULT

All the computations of the parallel prefix adders are simulated and analyzed on ModelSim-Altera 10.1d (Quartus II 13.1) tool. The generated verilog outputs implemented with the tool. The resultant power of the Adders and subjected multilplier using parallel prefix adder is complied and anlysed with Powerplay powe analyzer. To study detail area and power calculation we have implemented above adders in cadence and simulation is performed.

The computation results of all mentioned parallel prefix adders for 8-bit computation is found as-

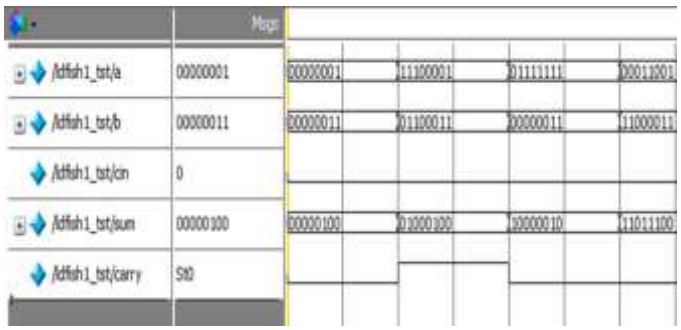

Fig. 8. Simulation result of Adder

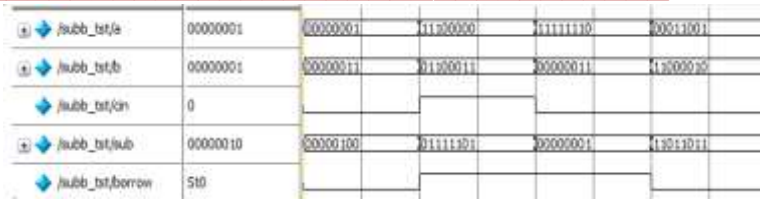

Fig. 9. Simulation result of Subtractor

Fig. 10. Simulation results of Multiplier

n= 8 bit	Area	Delay	Power
KS Adder	**1544**	**372**	**0.528**
L-F Adder	**1376**	**364**	**0.217**
SPP	**1201**	**337**	**0.446**

Power is calculated in mW, delay psec while area is in um^2.

VI. CONCLUSION

Here we compare SPP, KS adder and lander fischer adder and proposed multiplier with existing. The main parameters to compare any circuits are delay, power, and area obtained from architecture. There are two types of power dissipation in the circuit which are static and dynamic. Out off these two dynamic power plays an important role as it depends on switching action taking place due to change in input.

Kogge stone adder and lander ficsher adder offers almost similar execution speed.fanout offered by ks-adder is 2. While that of ld- adder is (n/2). But this increase in fanout results in increased number of prefixed operator and which results in more routing i.e wiring complexity. From timing analysis we can comment that SPP can save delay and we can say that, this reduction in delay is obtained due to implemented carry is getting overlapped instead of carry propagation from one stage to next stage. As in SPP [2] replace carry computational unit by multiplexer. But it should be noted that SPP is also a type of parallel prefix adder but removal of critical path for carry save delay for large word-length number. We can perform many combinations for carry propagation path in order to reduce delay. In implemented SPP fan-out is limited to 2 as well as size of carry save block is limited to 4.

Now we can go for finding area efficiency of different adders. It mostly depends on number of input bits. We can comment that lander fischer adder is preferable over kogge stone adder. While implemented SPP provides lowest area of all for increasing number of bits. It means SPP decreases the number of calculations of prefix operators. Similarly we can draw conclusion about various multi-modulo about power dissipation in architecture and it is noted that power dissipation in circuit is directly proportion area required to

 ISSN: 2321-8169
205 - 210

implement architecture assuming same operating frequency for all circuit under comparison.

In above paper we have proposed a new multiplier in which we are performing multiplication by repeating addition for n number of time and carry and multiplication is computed simultaneously. Which significantly reduces area, As it has been proven that area and power are highly correlated, we can conclude that power is reduced significantly.

VII. REFERENCES

[1] H.T. Vergos, D. Bakalis, "Area-time efficient multi-modulus adders and their applications",Microprocessror and Microsystem j.micpro.2012.02.004.

[2] H.T. Vergos, G. Dimitrakopoulus, "On Modulo 2^n+1 Adder Design" IEEE trans. On computers, vol. 61, No.2, feb.2012.

[3] D. Milford, A. Wrzyszez, "A new modulo 2^n+1 multiplier", IEE Int. Conf. on Computer Design(ICCD-2003), pp.614-617' oct.1992.

[4] H. Vergos, Dimitris Nikolos, G. Dimitrakopulos, C. Efstahiou, "Efficient diminished-1 modulo 2^n+1 multipliers", IEEE Trans. Computers, vol. 54, no.4, pp 491-496, Aprl. 2005.

[5] Kogge P., Stone H.,"A parallel algorithm for efficient solution of general class Recurrence relations", IEEE Trans. Computer, vol. C-22, no. 8, pp 786- 793, Aug. 1973.

[6] H. T. Vergos, D. Adamidis, "RNS multiplication/sum-of-squares", IET Proc.Computer Digital Tech. 1(1) (2007) 38-48.

[7] R.E. Ladener, M.J. Fishcher, "Parallel Prefix computation", J. ACM27 (4) (1980) 831-838.

[8] G. Dimitrakopoulous, D. Nikolos, "High-speed and Reduced-Area Modular Adder structures for RNS", IEEE Trans. Computer, vol. 51, no. 1,pp. 84-89,jan 2002.

[9] G. Jullien, M. Bayoumi, W. Miller, "AVLSI Implimentation of residue Adders", IEEE Tras. Circuits and System, vol. CAS-34, no. 3, pp. 284-288, Mar. 1987.

[10] S.Mathw,m.Anderse,S.Borkar A 4-GHz generation unit with 32-bit sparse adder core,J.solid statecirciuts38(5)(2003)689-167

[11] R.P.Lander,M.J.Fischer parallel prefix computation, ACM27(4)(1980)831 -383