Australian Journal of Basic and Applied Sciences, 9(15) Special 2015, Pages: 172-178

ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Double output hard Multiple Generator (HMG) for modulo 2^n+1 Multiplier

A. Rosi, P. Arun Kumar, R. Seshasayanan

Department of Electronics and Communication, college of engineering, Guindy, Chennai, India

ARTICLE INFO

Article history:
Received 10 March 2015
Received in revised from 20 March
Accepted 25 March 2015
Available online 10 April 2015

Keywords:

ABSTRACT

In this paper a novel double output Hard Multiple Generator (HMG) architecture has been proposed for modulo 2n+1. Hard Multiple Generator (HMG) is used to generate |3X|m in modulo multiplication that are based on radix8 booth encoding. Radix 8 booth encoding consists of +3 called as hard multiple. In modulo 2n+1 we need to generate |3X+2|. Generating this |3X+2| hard multiple is tedious because the multiplicand has to be added thrice and to be added with 2. The proposed modulo 2n+1 double output HMG generates double output i.e 2*n/2 or n bit output in n bit HMG architecture, where n is the bit width of the modulo multiplier. To develop double output modulo 2n+1 multiplier, the first and foremost is to develop twin output HMG. Considering this we have developed twin output HMG. Normally in ASIC environment to get n and 2*n/2 HMG output we need n and n/2 bit HMG architecture. But in proposed double output HMG n bit architecture is sufficient to do it. This leads to area and power efficient HMG architecture. Area reduction of 33% is achieved for 2n+1 double output HMG. Power reduction of 36% is achieved for modulo 2n-1 double output HMG when compared with 2n+1 HMG in Normal Twin Logic(NTL). All our evaluations are made under cadence environment of 180nm.

To Cite This Article: A. Rosi, P. Arun Kumar, R. Seshasayanan, Double output hard Multiple Generator (HMG) for modulo 2n+1 Multiplier. ***Aust. J. Basic & Appl. Sci.,*** *9(15): 172-178, 2015*

INTRODUCTION

Modulo arithmetic such as modulo addition, modulo multiplication are widely used in digital computing systems. Arithmetic modulo 2^n+1 is commonly referred in Residue Number Systems (RNS) (N.Szabo *et al*., 1967), (M.A Soderstrand *et al*., 1986),(P.V Ananda mohan *et al*., 2002),(A.Omandi *et al*., 2007). It has been used for the design of RNS processors, FIR filters, cryptography etc. High speed modulo multipliers using Booth encoding which reduces the number of partial products have been proposed in (R. Zimmermann *et al*., 1999), (C. Efstathiou *et al*., 2004). In booth encoding technique the numbers of partial product rows are reduced which reduces the hardware. By increasing the radix beyond four even greater reduction in power and area are met (B. S. Cherkauer *et al*., 1997), (P. M. Siedel *et al*., 2005).

The bottleneck found in radix8 booth encoding technique is the generation of hard multiple $|3X|_m$ involves in modulo addition of X and 2X which results in long carry propagation delay. Basically RNS applications consider three moduli sets {2^n, 2^n-1, 2^n+1} and the execution delay here is calculated by 2^n+1 channel because it has to deal with (n+1) bit wide operands. This problem is overwhelmed by diminished -1 representation introduced by (Leibowitz 1976). In diminished-1 number system each number X is represented by X* = X-1 and representation of 0 is treated in special way.

Diminished-1 representation of modulo adders for conventional CLA adders and parallel prefix are presented in (H.T. Vergos et..al., 2002). Efficient parallel prefix modulo 2^n+1 adder in diminished-1 representation is presented in(R. Muralidharan *et al*, 2010). Here carry equations are reformulated by considering modulo 2^n+1 addition of X and 2X independently. This is done by number of theoretical properties of modulo arithmetic. These parallel prefix adders to generates hard multiple in modulo 2^n+1multiplication consists of three stages pre processing, prefix computation and post processing operations.

In (Ramya Muralidharan *et al*., 2012) modified prefix operators are used in HMG architecture of modulo 2^n+1 adder which reduces the number of gates and thereby even reduction in hardware of parallel prefix adders is met. We aim at obtaining double output HMG in these parallel prefix modulo 2^n+1 adders. In this paper novel twin $2^n + 1$ hard multiple generator for computing $|3X + 2|_m$ for modulo $2^n + 1$are proposed. The addition of X and 2X are computed by using the customized parallel

Corresponding Author: A Rosi, Research scholar, Department of Electronics and Communication, college of engineering, Guindy, Chennai, India.
E-mail: arosyme@gmail.com

prefix binary adder as proposed in (Ramya Muralidharan *et al*., 2012). The proposed double output HMG yields n and n/2 bit output in n bit architecture. This leads to hardware reusability thereby reduction in area and power is achieved in proposed double output HMG.

Existing 2^n+1 HMG:

A modulo $2^n + 1$ addition of two diminished-1 represented operands, A and B, equivalent to an n-bit addition of A and B with $\overline{c_{out}}$, i.e.,

$$|S + 1|_m = |A + B + \overline{C_{out}}|_{2^n} \quad (1)$$

Where $\overline{c_{out}}$ is the carry output from the addition of A and B . As $\overline{c_{out}}$ is added to the sum of A and B at the LSB position, modulo $2^n + 1$ addition is commonly referred to as complementary end around carry (CEAC) addition.

This $2^n + 1$ modulo adder can be implemented by a parallel-prefix structure with three operator stages, namely pre-processing, prefix-computation and post-processing as illustrated in fig1, 2 & 3.

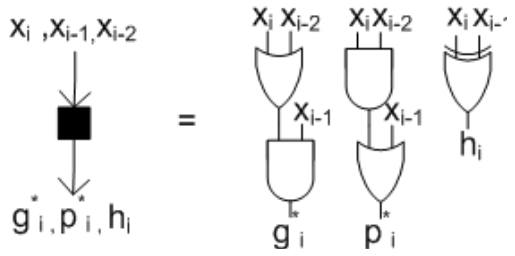

Fig.1: Preprocessing operator of HMG.

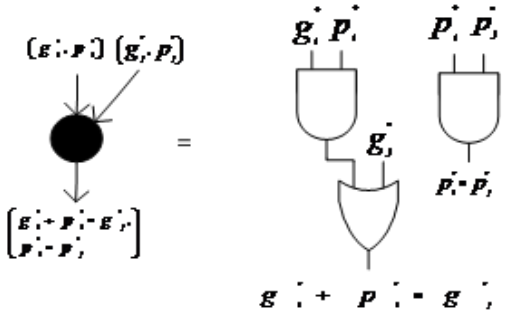

Fig. 2: Prefix operator of HMG.

The pre-processing stage computes generate (g_i), propagate $(p_i)$and half-sum(h_i) bits using the eq (2) for $i = 0\ to\ n - 1$.

$$g_i = a_i \,.\, b_i$$
$$p_i = a_i + b_i$$
$$h_i = a_i \oplus b_i \quad (2)$$

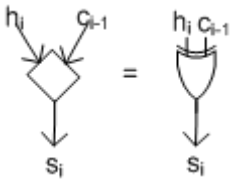

Fig. 3: Post processing operator of HMG.

In the prefix computation stage the carry bits c_i for CEAC addition, is computed by using prefix operator $(\cdot)$ on $(g_i \cdot p_i)$ for i $= 0\ to\ n - 1$ as given by eq(3) and explained in (H. T. Vergos, C. Efstathiou *et al*., 2002)(H.T.Vergosand ,C.Efstathiou *et al*., 2009)

$$c_i = (g_i, p_i) \cdot \ldots \cdot (g_0 \cdot p_0) \cdot \overline{(g_{n-1} \cdot p_{n-1}) \cdot (g_{i+1} \cdot p_{i+1})} \quad (3)$$

For modulo $2^n + 1$ HMG, the addends are expressed as X $= \sum_{i=0}^{n-1} x_i \cdot 2^i$and CCLS(X, 1) $= \sum_{i=0}^{n-2} x_i \cdot 2^{i+1} + \bar{x}_{n-1} \cdot 2^0$ in order to utilize the bit correlation between them.

By Property 3 of (Ramya Muralidharan *et al*., 2012),

$$|2.X|_m = CCLS(X, 1) - 1 \quad (4)$$

As per the definition of modulo $2^n + 1$ addition described in equation (1)& (4), the sum of X and CCLS(X, 1) is given by

$$|X + 2 \cdot X + 1 + 1|_{\boldsymbol{m}} = |3X + 2|_{\boldsymbol{m}} \quad (5)$$

The hard multiple is generated along with a constant bias of 2. Instead of removing this constant bias from the sum, which will increase the critical path delay of the HMG, it is merged into the modified partial products, as detailed in the following section.

For addition of X and CCLS(X, 1) as per property 6 of (Ramya Muralidharan *et al*., 2012)

$$p_0 \cdot \bar{p}_{n-1} = (x_0 + \bar{x}_{n-1})(\overline{x_{n-1} + x_{n-2}}) = \bar{p}_{n-1} \quad (6)$$

$$p_i \cdot g_{i-1} = \begin{cases} (x_1 + x_0) \cdot (x_0 \cdot \bar{x}_{n-1}) = (x_0 \cdot \bar{x}_{n-1}) = g_0 \\ (x_i + x_{i-1}) \cdot (x_{i-1} \cdot x_{i-2}) = (x_{i-1} \cdot x_{i-2}) = g_{i-1} \end{cases} \quad (7)$$

Using Property 6(Ramya Muralidharan *et al*., 2012),the number of modified generate-propagate bit pairs(g_i^*, p_i^*) explained in eq(8) of prefix computation in modulo $2^n + 1$ HMG is reduced to only n/2

$$c_i = \{(\bar{g}_i^*, \bar{p}_i^*) \cdot (\bar{g}_{i-2}^*, \bar{p}_{i-2}^*) \cdot \ldots \cdot (\bar{g}_0^*, \bar{p}_0^*) \cdot (\bar{g}_{n-2}^*, \bar{p}_{n-2}^*) \cdot (\bar{g}_{i+2}^*, \bar{p}_{i+2}^*)$$
$$for\ even\ i$$
$$c_i = \{(\bar{g}_i^*, \bar{p}_i^*) \cdot (\bar{g}_{i-2}^*, \bar{p}_{i-2}^*) \cdot \ldots \cdot (\bar{g}_1^*, \bar{p}_1^*) \cdot (\bar{g}_{n-1}^*, \bar{p}_{n-1}^*) \ldots \cdot (\bar{g}_{i+2}^*, \bar{p}_{i+2}^*)$$
$$for\ odd\ i \quad (8)$$

Eq(8) is implemented with two identical prefix operators to generate (g_i^*, p_i^*) and $(\bar{p}_i^*, \bar{g}_i^*)$ which is explained in (Ramya Muralidharan *et al*., 2010).The number of such dual prefix operators required in each prefix level l ,$1 \leq l \leq [\log_2 n] - 1$, given by $n - 2^i + 1$. In order to eliminate the implementation area and interconnects of the dual prefix operators, a modified prefix operator that computes $(g_i^*, p_i^*) \cdot (g_j^*, p_j^*)$ as well as $(\bar{g}_i^*, \bar{p}_i^*) \cdot (\bar{g}_j^*, \bar{p}_j^*)$ from the inputs, $(g_i^*, p_i^*) and (g_j^*, p_j^*)$, using fewer number of logic gates is proposed in (Ramya Muralidharan *et al*., 2012)which is portrait in fig.4.

$$g_i^*.p_i^* = g_i^* \text{ and } \bar{p}_i^*.\bar{g}_i^* = \bar{p}_i^* \quad (9)$$

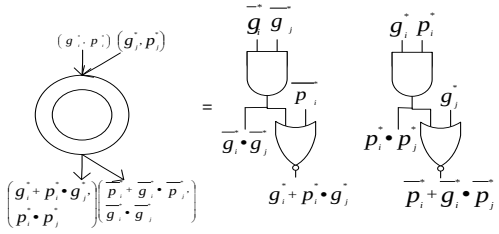

Fig. 4: Modified Prefix operator of HMG.

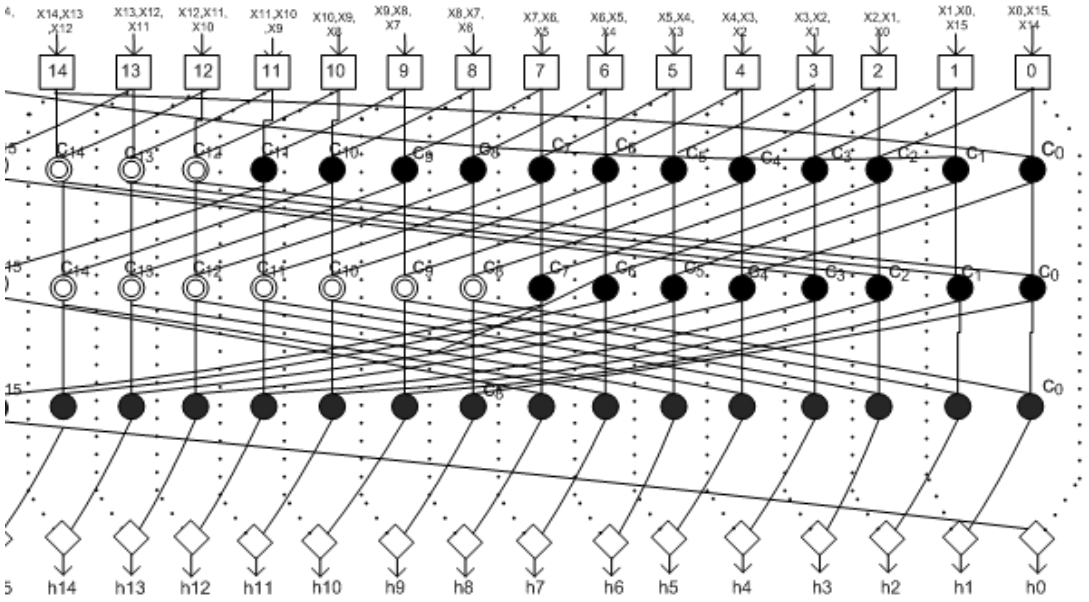

Fig. 5: Modulo 2^n+1 HMG for n=16.

Finally in the post processing stage each operator ◇ computes a sum bitss_i by the XOR of h_i from the pre-processing stage and c_{i-1} from the prefix-computation stage. All the above changes in carry equation are implemented in fig.5.

Proposed Double Output HMG (DO-HMG):

In designing modulo 2^n+1 DO-HMG 2^n+1 multiplier, it is necessary to get HMG output for n bit and n/2 bit multiplication. To perform this we have included multiplexers in the HMG architecture proposed in (Ramya Muralidharan *et al*., 2012) to select appropriate signals for n and n/2 bit operation. Simple logic only involves multiplexers in the HMG architecture which selects the signals for n and n/2 bit operation. We have proposed double output HMG architecture to get hard multiple (+/- 3x +2) for n bit and two (LSB and MSB) n/2 bit operation as portrait in fig 6. To obtain n/2 bit output in HMG, the n bit inputs of HMG(X_0 to X_N) are divided equally i.e 0 to $X_{(n/2-1)}$ and $X_{(n/2)}$ to $X_{(N-1)}$ called as HMG LSB part and MSB part and hard multiple is generated for each part . In fig.6 dual HMG output has been derived for n=16. HMG uses the parallel adder (Ramya Muralidharan *et al*., 2012) structure to generate 3x+2 for 2^n+1. Signals from h0 to h15 in the architecture (fig.6) is the HMG output for n bit multiplication. Whereas for n/2 bit multiplication, the two 8 bit output of HMG are taken from h0 to h7 and h8 to h15. HMG has three stages pre processing(rectangle-■), prefix computation(●) and post processing(diagonal- ◊).In the following explanation we have referred the pre processing as rectangle, prefix computation as circle and post processing as diagonal. The changes to be made in existing HMG architecture (fig.5) for n=16 to obtain n/2 output are as follows. And all the changes described below are shown in fig.6.

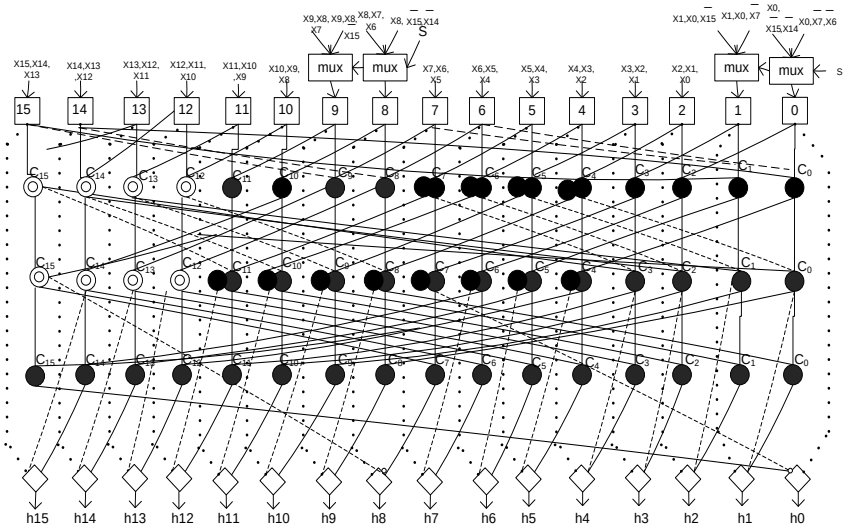

Fig. 6: Proposed modulo 2^n+1 DO-HMG for n=16.

Changes in 2^n+1 HMG:

First step in obtaining double output HMG is to make changes in the input pattern of pre-processing stage. For n/2 bit operation the complementary circular left shift operation has to be made between multiplicand x0 to x7(LSB part) and x8 to x15(MSB part). So the changes in the inputs to pre-processing stage for first n/2 bit operation (LSB part) are to be made at first two columns i.e 0^{th} and 1^{st} rectangle. In 0^{th} rectangle for n bit operation (n=16) the inputs are x0,(not x15), (not x14) whereas for n/2 bit operation(LSB part) inputs will be x0,(not x7),(not x6) because n=8 for n/2 bit operation which is illustrated in fig.6. Likewise in 1^{st} rectangle inputs for n bit operation are x1, x0,(not x15) whereas for n/2 bit operation x1, x0, (not x7) are its input.

Now for second n/2 bit operation (MSB part), the input changes to be made in the corresponding rectangle are given by the following equation.

{Rec =(n-(n/2)) , Rec =(n-((n/2)- 1))} (10)

Eq(10) shows that in 8^{th} and 9^{th} rectangle the changes are to be made in its input. In 8^{th} rectangle the inputs are x8,x7,x6 for n bit and this has to be altered as x8,(not x15),(not x14) for n/2 bit operation in RHS side.

Usually carry is complemented added in the LSB position for modulo 2^n+1addition called as CEAC addition. While performing n bit operation in HMG, the inputs of the first two circle(c0,c1) of first level (row) is g14, p14 and g15, p15 which are obtained from n-1(15^{th}) and n-2(14^{th}) rectangle. This input will change if the HMG is performing n/2 operation because CEAC addition will vary for n and n/2 bit operation.

In the first row of the circle we use the twin circle (c4, c5, c6, c7) as shown in fig.6 . Because for n/2 bit operation in LHS side of modulo 2^n+1 HMG we need to generate the pair wise swapped and complemented generate, propagate signals $(\bar{p}_i^*, \bar{g}_i^*)$ as per eq(8). Here modified prefix operator can not be used in twin circle places because for n bit operation the signal generation will be different i.e HMG has to generate only (g_i, p_i). End around carry addition is performed with these signals called as complementary end around carry addition (CEAC). In circles such as c12,c13,c14,c15 we use modified circle(◎) as explained in [12] which is same as twin circle but it has less number of gates and it always produce normal circle , pair wise swapped and complemented output. In second row, the twin Circles c4 to c11 and modified circle c12 to c15 is used to generate CEAC for n/2 and n bit operation.

Eq(8) explain the circle operation and with this as base first n/2 output (h0 to h7) i.e LSB part, the inputs for first two circle(c0,c1) of first row will be generate (g_0,g_1) and propagate (p_{n-2},p_{n-1}) which is from pre-processing stage(rectangle) 0^{th} , n-1(15^{th}) and n-2(14^{th}) rectangle. But for n/2 bit LSB operation where n=8 c0,c1 circle inputs should be from (n/2 -2) (6^{th}) and (n/2-1) (7^{th}) rectangle and these signals act as CEAC which is represented as dotted line in fig.6.The dotted line denotes the signal for n/2 bit operation. Therefore first two circles of first row is employed with multiplexers to select corresponding g,p for n and n/2 bit operation.

Likewise the changes are made in the first row of circle in the RHS side to obtain second n/2 output (h8 to h15). Here HMG is performed between the multiplicand x8 to x15. C7 and c8 will be considered as first two circles for obtaining second n/2 bit output. Usually propagate and generate signals for c7 and c8 in first level of prefix computation for n bit operation is from (n/2 -1) (7^{th}) and (n/2 -2) (6^{th})rectangle. But these signals will be from n-2 (14^{th}) and n-1 (15^{th})rectangle and CEAC is performed.

$[Log_2 n]$-1 levels of prefix computation is performed. So three levels (three rows) of computation is performed for n=16 whereas only two

levels is needed for n=8. In second level of prefix computation four CEAC of first row is added to the LSB part of second level. For n bit operation i.e. when n=16 c12, c13,c14 and c15 output of first row is fed as inputs to second row circles of c0,c1,c2 and c3. For LSB part n/2 operation c4,c5,c6,c7 are CEAC of first level will be the input to c0,c1,c2,c3 of second level. The selection of inputs for n and n/2 are made by multiplexers. Likewise for MSB part n/2 operation CEAC such as c12,c13,c14 and c15 of first level will be the input to c8,c9,c10 and c11 of second level. Now the outputs of second level is directly passed to the post processing(diagonal) stage bypassing the third level which is only necessary for n bit operation.

Diagonal will generate the sum bit(s) by performing XOR of hi and ci-1 as portrait in fig.4. For n bit operation ci-1 will be the third level outputs of prefix computation. Whereas for n/2 bit operation it will be the second level outputs.

For n bit operation the n^{th} bit is complemented and is XOR with lsb of half sum for CEAC. Likewise for n/2 bit operation (n/2 -1)bit and n bit carry is complemented and is xor with corresponding lsb and n/2 bit of half sum bit for CEAC.

The dotted line in the diagram represents the routing for n/2 operation. For n/2 bit operation h0 to h7 will be the HMG output of LSB part and h8 to h15 will be the HMG output for MSB part.

RESULTS AND DISCUSSIONS

The generation of $|3X+2|_m$ called Hard Multiple Generation is the bottleneck operation in the radix-8 booth encoding $2^n + 1$ modulo multipliers.In this paper , we have proposed double output HMG which yields n and n/2 bit output in n bit architecture.

Table 1: NTL vs PTL for modulo 2n + 1 modulo HMG

2n+1	Normal Twin Logic(NTL)				Proposed Twin Logic(PTL)			
	Area (μm^2)	Power(nW)	Delay (ps)	Power Delay (PJ)	Area (μm^2)	Power (nW)	Delay (ps)	Power delay (PJ)
16_8	628	2087160.356	215	.45873952	467	1639332.625	239	.391800497
64_32	1355	4643486.435	567	2.6328568	1013	3567218.893	635	1.6135927
256_128	2049	9416974.21	1082	9.2956	1427	6903523.322	1203	8.30493

Our proposed double output hard multiple generator for modulo 2n +1 multiplier architecture is capable to produce double output (N and 2xN/2 bit output) and it is represented as PTL (Proposed Twin Logic). Normally to obtain N and N/2 bit output in HMG we need N bit and N/2 bit modulo multiplier architecture (Ramya Muralidharan, 2012) and we represent it as NTL. But we have achieved N and 2xN/2 output in N bit architecture at the cost of N bit architecture. Due to this, a drastic reduction in area and power has been achieved. The area, power for normal twin logic is calculated according to (11) and is common for area, delay and power.

$$N\_2\ (n/2) \xrightarrow{NTL(Area,Power)} (n/2) + n \qquad (11)$$

According to (11) total area required for normal twin logic will be addition of area required for two n/2 bit architecture and n bit architecture. Likewise power calculations are made but the delay will be N bit delay. We have framed the equation (11) in this manner because we cannot obtain a single n/2 bit output in n bit architecture by making MSB part zero as we do in normal multiplication. .

Table 2: NTL vs PTL for modulo 2n + 1 modulo HMG

S.NO	Bit Width	2n+1 HMG							
		Normal Twin Logic(NTL)				Proposed Twin Logic (PTL)			
		■	●	◎	◊	■	●	◎	◊
1	16(one 16 + one 8)	24	48	16	24	16	52	8	16
2	64(one 64 + one 32)	96	360	98	96	64	336	44	64
3	256(one 256 + one 128)	384	2184	376	384	256	1856	188	256

Area:

The total area for proposed twin and normal twin HMG is tabulated in table.1. According to (11) the area has been calculated for NTL HMG. The total area occupied for the double(twin) output modulo 2^n+1 HMG depends on the pre processing operator, prefix computation operator, modified prefix computation operator and post processing operator . Proposed DO-HMG in PTL produces hard multiple for n and two n/2 bit operation with lesser area than NTL. Total pre-processing operator (■), prefix computation operator(●) and post processing operator (◊) required for NTL will be the total operators required for n/2 bit HMG architecture and n bit HMG architecture. Compare to this our proposed DO-HMG yields less area and is tabulated in table.1. For n=16, 2^n+1 HMG in NTL requires32 pre processing operator, 60 prefix computation operator,20 modified prefix computation operator and 32 post processing operation. Whereas our proposed DO-HMG requires 16 pre and post processing operators. And 52 prefix computation

operators and 8 modified prefix computation operators as shown in table.2. XOR gate is involved in the pre and post processing operator. By reducing these operators to half in DO-HMG the overall area for HMG in PTL is reduced. Though muxes are involved in DO-HMG for routing in n and n/2 bit operation its area is still lesser than HMG in NTL. The Number of operators are reduced in the PTL leads to Area reduction in TWIN HMG .The area reduction of about 33% is achieved in our PTL for 2n+1 HMG which is tabulated in table.1. fig.7 shows the area improvement for our PTL compare to NTL.

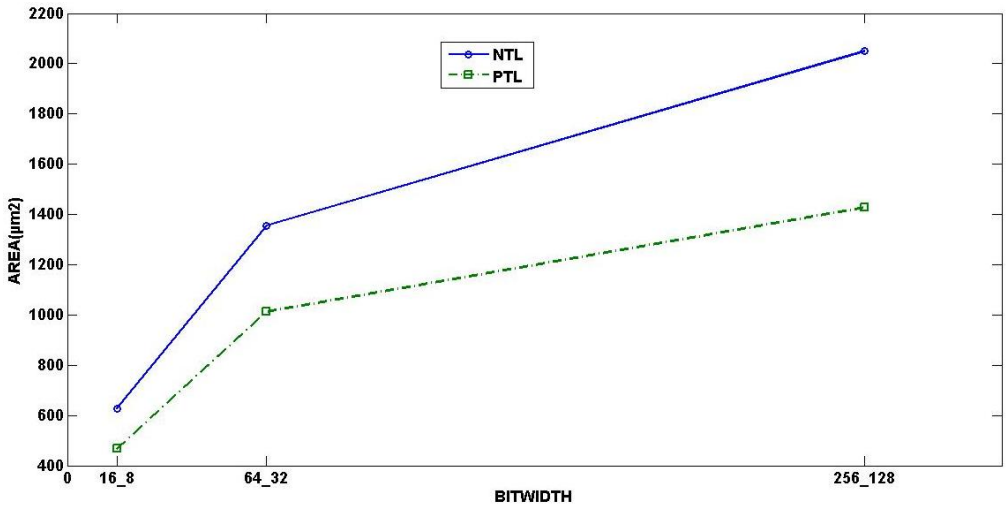

Fig. 7: Area comparision between NTL and PTL.

Power:

NTL power tabulated in table.1 is taken from individual modulo 2^n+1 HMG architecture of n and n/2 bit. Eq(11) explains this concept . The individual power of n and n/2 bit modulo 2n+1 HMG is added and we consider this power as NTL power. We have considered the power in this manner because there is no single architecture to perform n and n/2 bit operation. In contrast to this our PTL modulo 2^n+1 HMG offers n and n/2 bit operation in single n bit architecture. Compared to NTL reduction in power in PTL modulo 2n + 1 HMG is about 36 % this has been validated for n=16, 64,256 as portrait in fig.8.

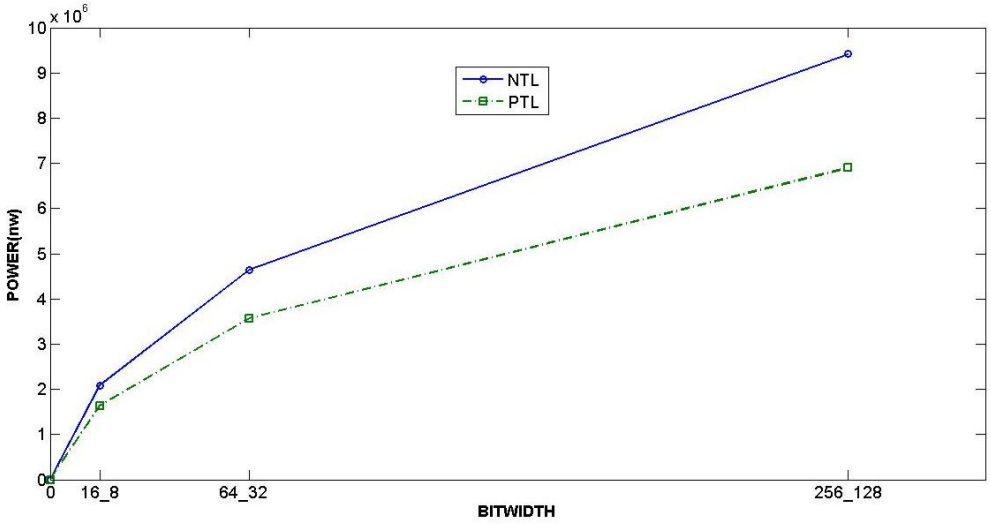

Fig. 8: Power comparision between NTL and PTL.

Delay:

Another important parameter to be considered is the delay. The area and power calculation for normal twin logic is calculated by addition of area and power utilised for two n/2 bit modulo HMG and a n bit modulo HMG. Since there is no single modulo HMG architecture to obtain n and n/2 bit output we have calculated area and power in this manner for NTL. Whereas the delay value for NTL will be the n bit delay . In our PTL the n and n/2 bit HMG are achieved in single architecture. Though we have achieved better area and power in PTL, the n/2 bit computation is achieved in n bit delay. But the power delay product for PTL is less compare to the NTL a

tabulated in table.1. Graphical analysis of power-delay product is shown in Fig.9. So our proposed design is very much efficient compare to the NTL .

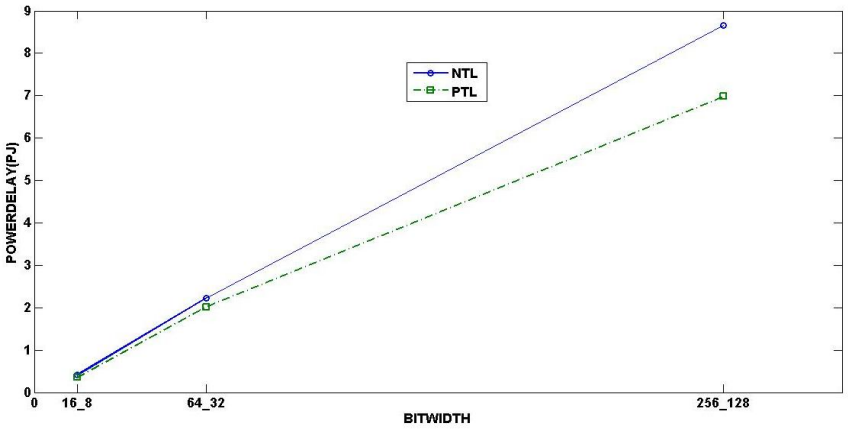

Fig. 9: Power-Delay comparision between NTL and PTL.

Conclusion:

We have designed modulo 2^n+1 DO-HMG architecture which yields n and n/2 bit HMG output in n bit architecture. And this proposed architecture is obtained by making wise changes in the existing HMG architecture reported in (Ramya Muralidharan, 2012). Since hardware reusability is met in our proposed DO-HMG , the reduction in area of about 33% is achieved compare to NTL logic. Power reduction of about 36% is achieved. Delay will be the n bit architecture delay.

REFERENCES

Ananda mohan, P.V., 2002. Residue number systems, Algorithms and Architectures, Kluwer Academic Publishers, Dordrecht.

Cherkauer, B.S. and E.G. Friedman, 1997. "A hybrid radix-4/radix-8 low power signed multiplier architecture," IEEE Trans. on Circuits and Syst. – II, 44(8): 656-659.

Efstathiou, C., H.T. Vergos and D. Nikolos, 2004. "Modified Booth modulo 2^n-1 multipliers," IEEE Trans. on Computers, 53(3): 370-374.

Leibowitz, L.M., 1976. A simplified binary arithmetic for fermat number transform, IEEE Trans.Acoust.Speech signal processing.

Muralidharan, R. and C.H. Chang, 2010. "Fast hard multiple generators for radix-8 Booth encoded modulo 2^n-1 and 2^n+1 multipliers," in Proc. 2010 IEEE Int. Symp. Circuits Syst., Paris, France, pp: 717–720.

Omandi, A., B. Premkumar, 2007. Residue number systems, Theory and implemenatatins, World Scientific, Singapore.

Ramya Muralidharan and Chip-Hong Chang, 2012. Area-Power Efficient modulo 2^n -1 and modulo 2^n + 1 Multipliers For$\{2^n$ -1 , 2^n , 2^n + 1$\}$ based RNS ", IEEE Transactions on circuits and systems – 1: Regular Papers, vol.59,No. 10.

Siedel, P.M., L.D. McFearin and D.W. Matula, 2005. "Secondary radix recodings for higher radix multipliers," IEEE Trans. on Computers, 54(2): 111-123.

Soderstrand, M.A, *1986.* Residue Number System Arithmetic: Modern application in digital signal processing, IEEE press, New York.

Szabo, N.. R. Tanaka, 1967. Residue arithmetic and its application to computer technology, McGraw-Hill, New York.

Vergos, H.T., C. Efstathiou and D. Nikolos, 2002. "Diminished-one modulo 2^n+1 adder design," IEEE Trans. Comput, 51(12): 1389–1399.

Vergos, H.T., C. Efstathiou and D. Nikolos, 2002. "Diminished-one modulo adder design," IEEE Trans. Comput, 51(12): 1389–1399.

Vergosand, H.T., C. Efstathiou, 2009. "Efficientmodulo adderarchitec- tures," VLSI J. Integr, 42(2): 149–157.

Zimmermann, R., 1999. "Efficient VLSI implementation of modulo (2^n ±1) addition and multiplication," in Proc. IEEE Symp. on Computer Arithmetic, Adelaide, Australia, pp: 158-167.