Design of modulo 2^n-1 multiplier Based on Radix-8 Booth Algorithm using Residue Number System

K.RAMAMOHAN REDDY
M.Tech Student, Dept. of ECE
Vaagdevi Institute of Technology & Science,Proddatur, Kadapa (Dt.), A.P.

V.RAMESH
Assistant Professor, Dept. of ECE
Vagdevi Institute of Technology &Science,Proddatur,Kadapa(Dt) A.P.

C.Md.ASLAM
HOD, Dept. of ECE
Vagdevi Institute of Technology & Science,Proddatur,Kadapa(Dt) A.P.

ABSTRACT

Modular arithmetic operations (inversion, multiplication and exponentiation) are used in several cryptography applications. A special moduli set of forms $\{2^n-1, 2^n, 2^n+1\}$ are preferred over the generic moduli due to the ease of hardware implementation of modulo arithmetic functions as well as system-level inter-modulo operations, such as RNS-to-binary conversion and sign detections. With this precept, a family of radix-8 Booth encoded modulo 2^n-1 multipliers, with delay adaptable to the RNS multiplier delay, is proposed.

The first-ever family of low-area and low-power radix-8 Booth encoded modulo 2^n-1 multiplier whose delay can be tuned to match the RNS delay closely has been proposed in this paper. A CSA tree with end-around-carry addition for accumulation of redundant partial products and a Sklansky parallel-prefix structure has also been implemented.

Index Terms— Public Key Cryptographic (PKC),Booth algorithm, modulo arithmetic, multiplier, residue number system (RNS)

1. INTRODUCTION

RIVEST, Shamir, and Adleman (RSA) and elliptic curve cryptography (ECC) are two of the most well established and widely used public key cryptographic (PKC) algorithms.

The encryption and decryption of these PKC algorithms are performed by repeated modulo multiplications [1]–[3].

These multiplications differ from those encountered in signal processing and general computing applications in their sheer operand size. Key sizes in the range of 512~1024 bits and 160~512 bits are typical in RSA and ECC, respectively [4]–[7]. Hence, the long carry propagation of large integer multiplication is the bottleneck in hardware implementation of PKC. The residue number system (RNS) has emerged as a promising alternative number representation for the design of faster and low power multipliers owing to its merit to distribute a long integer multiplication into several shorter and independent modulo multiplications

Modular Multiplication in Public Key Cryptosystems

Modulo 2^n+1, 2^n, 2^n-1 addition and multiplication are the crucial operations in the IDEA algorithm and also modulo 2^n+1 arithmetic operations are used in Fermat number transform computation. Moduli choices of the forms $\{2^n+1, 2^n, 2^n-1\}$ have received significant attention because they offer very efficient circuits when considering the area * time^2 product and efficient converters from and to the binary system. Therefore, designing efficient modulo 2^n-1 multipliers is an interesting issue. Modulo 2^n-1 multiplication is used extensively in Residue Number System (RNS) based Digital Signal Processing (DSP) and cryptography units.

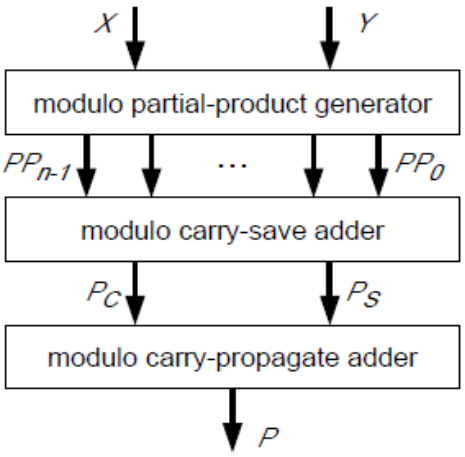


Figure 1: Modulo (2^n-1) multiplier architecture

The modulo $2^n - 1$ multiplication of two numbers n-bit each follows 3 steps: production of n^2 partial products modulo $2^n - 1$ reduction of this n^2 partial products $2^n - 1$ into two numbers of n bits addition of these two numbers modulo $2^n - 1$ with the preceding adder.

2. LITERAUTRE SURVEY

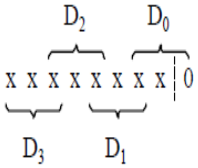
The radix-8 Booth encoding reduces the number of partial products to $\lceil n/3 \rceil + 1$ which is more aggressive than the radix-4 Booth encoding. However, in the radix-8 Booth encoded modulo 2^n-1 multiplication, not all modulo-reduced partial products can be generated using the bitwise circular-left-shift operation and bitwise inversion. Particularly, the hard multiple $|+3X|_{2^n-1}$ is to be generated by an n -bit end-around-carry addition of X and 2X.

Radix-4 and radix-8 multiplication

Recoding of binary numbers was first hinted at by Booth four decades ago. MacSorley proposed a modification of Booth's algorithm a decade after. The modified Booth's algorithm (radix-4 recoding) starts by appending a zero to the right of x_0 (multiplier LSB). Triplets are taken beginning at position x_{-1} and continuing to the MSB with one bit overlapping between adjacent triplets. If the number of bits in X (excluding x_{-1}) is odd, the sign (MSB) is extended one position to ensure that the last triplet contains 3 bits. In every step we will get a signed digit that will multiply the multiplicand to generate a partial product entering the Wallace reduction tree. The meaning of each triplet can be seen in table I:

Table I: Radix-4 encoding

x_{i+2} x_{i+1} x_i	Partial product
0 0 0	0Y
0 0 1	+1Y
0 1 0	+1Y
0 1 1	+2Y
1 0 0	-2Y
1 0 1	-1Y
1 1 0	-1Y
1 1 1	0Y



This recoding scheme applied to a parallel multiplier halves the number of partial products so the multiplication time and the hardware requirements decrease. This gain is possible at the expense of somewhat more complex operations in every step. However, that the required multiples of Y {0, Y, 2Y} are available by merely shifting Y to the left. Although the algorithms and operations specified above seem rather arbitrary at the first sight, they are based on meaningful number systems. If one focuses on what modifications are being done to X, then one may arrive at a different representation for the 2s-complement number X as shown in figure 2:

$$X = \sum_{i=0}^{\frac{n}{2}-1} D_i \cdot 4^i$$

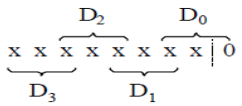


Figure 2: Signed-digit representation

where digits Di are one of -2, -1, 0, 1, 2 found in the table of figure 1, based on the value of triplets in the form (xi+2 xi+1 xi). Here we have a signed digit representation of

X in radix-4. Signed-digit number representation allows redundancy to exist. Thanks to this we can make a parallel recodification that is, all triplets are recoded at the same time, and the value of each triplet is independent from the adjacent triplets.

Radix-8 recoding applies the same algorithm as radix-4, but now we take quartets of bits instead of triplets. Each quartet is codified as a signed-digit using the table II:

TableII:Radix-8 recoding

Quartet value	*Signed-digit value*
0000	0
0001	+1
0010	+1
0011	+2
0100	+2
0101	+3
0110	+3
0111	+4
1000	-4
1001	-3
1010	-3
1011	-2
1100	-2
1101	-1
1110	-1
1111	0

Here we have an odd multiple of the multiplicand, 3Y, which is not immediately available. To generate it we need to perform this previous add: 2Y+Y=3Y. But we are designing a multiplier for specific purpose and thereby the multiplicand belongs to a previously known set of numbers which are stored in a memory chip. We have tried to take advantage of this fact, to ease the bottleneck of the radix-8 architecture, that is, the generation of 3Y. In this manner we try to attain a better overall multiplication time, or at least comparable to the time we could obtain using radix-4 architecture (with the additional advantage of using a less number of transistors). To generate 3Y with 21-bit words we only have to add 2Y+Y, that is, to add the number with the same number shifted one position to the left, getting in this way a new 23-bit word, as shown in figure 3:

$$\begin{array}{cccccccccc|cc} y_{20} & y_{19} & y_{18} & \cdots & y_3 & y_2 & y_1 & y_0 & 0 & 2\cdot Y \\ y_{20} & y_{20} & y_{19} & \cdots & y_4 & y_3 & y_2 & y_1 & y_0 & Y \\ \hline z_{22} & z_{21} & z_{20} & z_{19} & \cdots & z_4 & z_3 & z_2 & z_1 & z_0 \end{array}$$

Figure 3: 21-bit previous add

In fact, only a 21-bit adder is needed to generate the bit positions from z1 to z21. Bits z0 and z22 are directly known because z0=y0 and z22=y20 (sign bit of the 2s-complement number; 3Y and Y have the same sign). If in the memory from where we take the numbers just two additional bits are stored together with each value of the set of numbers, we can decompose the previous add in three shorter adds that can be done in parallel. In this way, the delay is the same of a 7-bit adder:

$$\begin{array}{ccccccc} y_6 & y_5 & y_4 & y_3 & y_2 & y_1 & y_0 \\ y_7 & y_6 & y_5 & y_4 & y_3 & y_2 & y_1 \\ \hline z_7 & z_6 & z_5 & z_4 & z_3 & z_2 & z_1 \end{array}$$

$$\begin{array}{ccccccc} y_{13} & y_{12} & y_{11} & y_{10} & y_9 & y_8 & y_7 \\ y_{14} & y_{13} & y_{12} & y_{11} & y_{10} & y_9 & y_8 \\ \hline z_{14} & z_{13} & z_{12} & z_{11} & z_{10} & z_9 & z_8 \end{array} \leftarrow c_8$$

$$\begin{array}{ccccccc} y_{20} & y_{19} & y_{18} & y_{17} & y_{16} & y_{15} & y_{14} \\ y_{20} & y_{20} & y_{19} & y_{18} & y_{17} & y_{16} & y_{15} \\ \hline z_{21} & z_{20} & z_{19} & z_{18} & z_{17} & z_{16} & z_{15} \end{array} \leftarrow c_{15}$$

Fig. 4: Modified previous add.

Bits which are going to be stored are the two intermediate carry signals c8 and c15. Before each word of the set of numbers is stored in the memory, the value of its intermediate carries has to be obtained and stored beside it. In this way, they are immediately available when it is required to perform the previous add to get the multiple 3Y of one of the numbers that belongs to the set.

The radix-4 Booth encoding technique is most prevalent as all required modulo reduced partial products can be generated by circular-left-shift operation and bit-wise complementation, thereby minimizing the hardware complexity. The reduction in the number of partial products is determined by the radix of the Booth encoding technique employed. Reduction of partial products by more than half is possible with higher radix Booth encoding. Similar to the radix-4 algorithm, the radix-8 Booth encoding algorithm can be considered as a digit set conversion of four consecutive multiplier bits y3i-1,y3i,y3i+1,y3i+2 , yi $\in$ {0, 1}from *Y*, to d_i, $d_i \in [-4, 4]$, for $i = 0, 1, \ldots N/3$.

The digit set conversion is given by

$$d_i = y_{3i-1} + y_{3i} + 2y_{3i+1} - 4y_{3i+2} \quad (1)$$

where y_{-1}, y_n, y_{n+1} and y_{n+2} are zero. For the radix-8 Booth encoded modulo 2^n-1 multiplier, the required modulo-reduced partial products are shown in Table III.

From Table 3, the necessary modulo-reduced partial products except ±3*X* can be generated by circular-left-shift operat-ion and/or bit-wise complemen-tation of the multiplicand, *X*. The generation of±3X requires a large word-length adder which increases the critical path delay of the multiplier significantly.

TABLE III: MODULO-REDUCED PARTIAL PRODUCTS FOR RADIX-8 BOOTH ENCODING

d_i	PP_i
0	00···00
+1	$x_{n-1-3i}x_{n-2-3i}\cdots x_0 x_{n-1}\cdots x_{n-3i}$
+2	$x_{n-2-3i}x_{n-3-3i}\cdots x_0 x_{n-1}\cdots x_{n-1-3i}$
+3	+3*X*
+4	$x_{n-3-3i}x_{n-4-3i}\cdots x_0 x_{n-1}\cdots x_{n-2-3i}$
–4	$\overline{x}_{n-3-3i}\overline{x}_{n-4-3i}\cdots \overline{x}_0\overline{x}_{n-1}\cdots \overline{x}_{n-2-3i}$
–3	-3*X*
–2	$\overline{x}_{n-2-3i}\overline{x}_{n-3-3i}\cdots \overline{x}_0\overline{x}_{n-1}\cdots \overline{x}_{n-1-3i}$
–1	$\overline{x}_{n-1-3i}\overline{x}_{n-2-3i}\cdots \overline{x}_0\overline{x}_{n-1}\cdots \overline{x}_{n-3i}$
–0	11···11

results also confirm that the proposed method helps pathologists distinguish exact lesion sizes and regions

3. PROPOSED RADIX-8 BOOTH ENCODED MODULO 2^n-1 MULTIPLIER DESIGN

To ensure that the radix-8 Booth encoded modulo 2^n-1multiplier does not constitute the system critical path of a high-DR moduli set based RNS multiplier, the carry propagation length in the hard multiple generation should not exceed *n* bits. To this end, the carry propagation through the HAs in Fig. 1 can be eliminated by making the end-around-carry bit c_7 a partial product bit to be accumulated in the CSA tree. This technique reduces the carry propagation length to *n* bits by representing the hard multiple as a sum and a redundant end-around-carry bit pair. The resultant [*n*/3] +1 end-around-carry bits in the partial product matrix may lead to a marginal increase in the CSA tree depth and consequently, may aggravate the delay of the CSA tree. In which case, it is not sufficient to reduce the carry propagation length to merely *n* bits using the above technique.

Since the absolute difference between the noncritical modulo 2^n-1 multiplier delay and the system critical path delay depends on the degree of imbalance in the moduli word-length of a RNS, the delays cannot be equalized by arbitrarily fixing the carry propagation length to *n* bits. Instead, we propose to accomplish the adaptive delay equalization by representing the hard multiple in a partially-redundant form [48].

A. Generation of Partially-Redundant Hard Multiple

Let $X\ |_{2^n-1}$ and $2X\ |_{2^n-1}$ be added by a group of *M (=n/k) k*-bit RCAs such that there is no carry propagation between the adders. Fig. 2 shows this addition for *n*=8 and *k*=4,where the sum and carry-out bits from the RCA block *j* are represented as S_i^j andc_i^j for $i \in [0,k-1]$ and $j \in [0,M-1]$,respectively. In Fig. 2, the carry-out of RCA 0,C_3^0 , is not propagated to the carry input of RCA 1 but preserved as one of the partial product bits to be accumulated in the CSA tree. The binary weight of the carry-out $C_3^1$of RCA 1 has,however, exceeded the maximum range of the modulus and has to be modulo reduced before it can be accumulated by the CSA tree.

By *Property 2*, the binary weight of C_3^1 can be reduced from 2^8 to 2^0 . Thus, C_3^1 is inserted at the least significant bit (lsb)position in Fig. 6. It should be stressed that the carry-out C_3^1 is a partial carry propagated through only *k* most significant FAs and hence, is different from the end-around-carry bit in the modulo 2^n-1 addition of *X* and 2*X* , *i.e.*, c_7 of Fig. 5.From Fig. 6, the partially-redundant form of $|+3X|_{2^n-1}$ is given by the partial-sum and partial-carry pair*(S,C)*

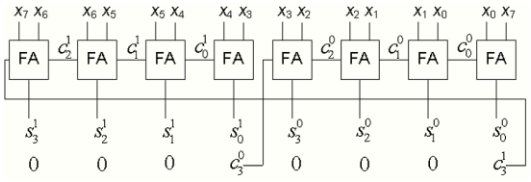


Fig. 5. Generation of partially-redundant $|+3X|_{2^n-1}$ using k-bit RCAs

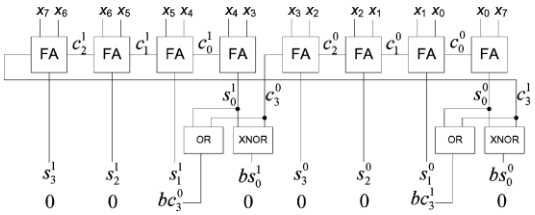


Fig.6. Generation of partially-redundant $|B+3X|_{2^n-1}$

where

$$S = s_{k-1}^{M-1} s_{k-2}^{M-1} \cdots s_0^{M-1} \cdots s_{k-1}^0 s_{k-2}^0 \cdots s_0^0$$
$$C = \underbrace{0\cdots0}_{k-1} c_{k-1}^{M-2} \cdots \underbrace{0\cdots0}_{k-1} c_{k-1}^0 \underbrace{0\cdots0}_{k-1} c_{k-1}^{M-1}. \quad (5)$$

Since modulo 2^n-1 negation is equivalent to bitwise complementation by *Property 1*, the negative hard multiple in a partially-redundant form, $|-3X|_{2^n-1}=(\bar{S},\bar{C})$, is computed as follows:

$$\bar{S} = \bar{s}_{k-1}^{M-1} \bar{s}_{k-2}^{M-1} \cdots \bar{s}_0^{M-1} \cdots \bar{s}_{k-1}^0 \bar{s}_{k-2}^0 \cdots \bar{s}_0^0$$
$$\bar{C} = \underbrace{1\cdots1}_{k-1} \bar{c}_{k-1}^{M-2} \cdots \underbrace{1\cdots1}_{k-1} \bar{c}_{k-1}^0 \underbrace{1\cdots1}_{k-1} \bar{c}_{k-1}^{M-1}. \quad (6)$$

To avoid having many long strings of ones in $\bar{C}$, an appropriate bias *B*, , is added to the hard multiple such that both *C* and $\bar{C}$ are sparse [48]. The value of *B* is chosen as

$$B = \sum_{j=0}^{M-1} 2^{k\cdot j} = \overbrace{\underbrace{0\cdots01}_{k}\cdots\underbrace{0\cdots01}_{k}}^{n}. \quad (7)$$

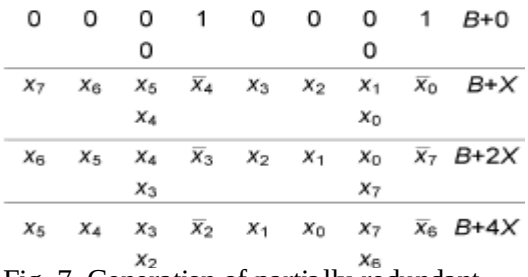


Fig. 7. Generation of partially-redundant simple multiples.

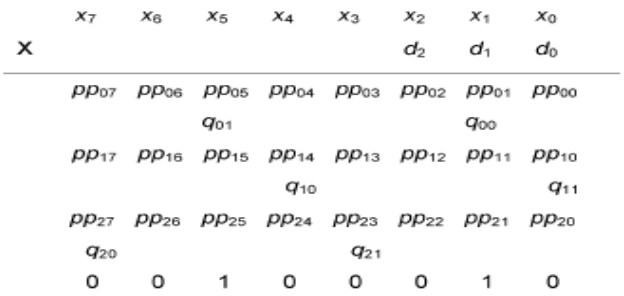


Fig. 8. Modulo-reduced partial products and *CC* for $|X\cdot Y|_{2^8-1}$

The addends for the computation of the biased hard multiple, $|B+3X|_{2^n-1}$ in a partially-redundant form are $|X|_{2^n-1}$, $|2X|_{2^n-1}$ and B or equivalently S,C and B. Since is chosen to be a binary word that has logic ones at bit positions 2^{kj} and logic zeros at other bit positions, $|B+3X|_{2^n-1}$ can be generated by simple XNOR and OR operations on the bits of S and C at bit positions 2^{kj}. Fig. 6 illustrates how these bits in the sum and the carry outputs of RCA 0 and RCA 1 are modified.

In general $|B+3X|_{2^n-1}$, is given by the partial-sum and partial-carry pair (*BS*,*BC*)such that

$$BS = s_{k-1}^{M-1} s_{k-2}^{M-1} \cdots bs_0^{M-1} \cdots s_{k-1}^0 s_{k-2}^0 \cdots bs_0^0$$
$$BC = \underbrace{0\cdots0}_{k-2} bc_{k-1}^{M-2} 0 \cdots \underbrace{0\cdots0}_{k-2} bc_{k-1}^0 0 \underbrace{0\cdots0}_{k-2} bc_{k-1}^{M-1} 0 \quad (8)$$

where

$$bs_0^j = \begin{cases} \overline{s_0^j \oplus c_{k-1}^{j-1}} & \text{when } j \neq 0 \\ \overline{s_0^0 \oplus c_{k-1}^{M-1}} & \text{when } j = 0 \end{cases} \quad (9)$$

and

$$bc_{k-1}^j = \begin{cases} s_0^{j+1} + c_{k-1}^j & \text{when } j \neq M-1 \\ s_0^0 + c_{k-1}^{M-1} & \text{when } j = M-1 \end{cases} \quad (10)$$

For $j=0, 1 \ldots M-1$.

Let

$$\overline{BS} = \overline{s_{k-1}^{M-1} s_{k-2}^{M-1}} \cdots \overline{bs_0^{M-1}} \cdots \overline{s_{k-1}^0 s_{k-2}^0} \cdots \overline{bs_0^0}$$

$$\overline{BC} = \underbrace{0 \cdots 0}_{k-2} \overline{bc_{k-1}^{M-2}} 0 \cdots \underbrace{0 \cdots 0}_{k-2} \overline{bc_{k-1}^0} 0 \underbrace{0 \cdots 0}_{k-2} \overline{bc_{k-1}^{M-1}} 0. \quad (11)$$

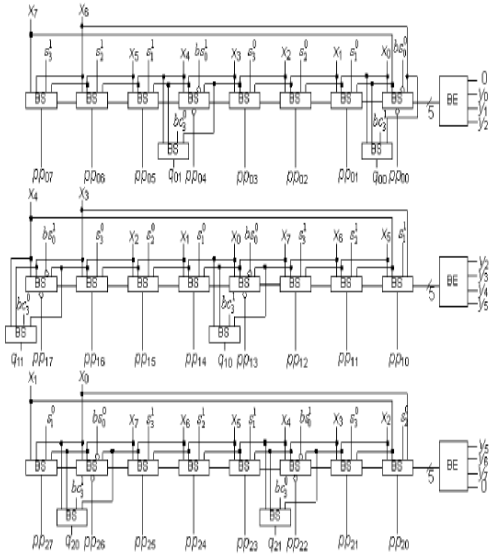


Fig. 9. Modulo-reduced partial product generation.

It can be easily verified that the sum of (*BS*, *BC*) and() modulo 2^n-1 is $|2B|_{2^n-1}$. Therefore,()represents the partially-redundant form of $B-3X|_{2^n-1}$.

B. Generation of Partially-Redundant Simple Multiples

The proposed technique represents the hard multiple in a biased partially-redundant form. Since the occurrences of the hard multiple cannot be predicted at design time, all multiples must be uniformly represented. Similar to the hard multiple, all other Booth encoded multiples listed in Table I must also be biased and generated in a partially-redundant form. Fig. 7 shows the biased simple multiples, $|B+0|_{2^n-1}$, $|B+X|_{2^n-1}$, $|B+2X|_{2^n-1}$, and $|B+4X|_{2^n-1}$ represented in a partially-redundant form for n=8. From Fig. 10, it can be seen that the generation of these biased multiples involves only shift and selective complementation of the multiplicand bits without additional hardware overhead.

C. Radix-8 Booth Encoded Modulo 2^n-1 Multiplication with Partially-Redundant Partial Products

The i-th partial product of a radix-8 Booth encoded modulo 2^n-1 multiplier is given by

$$PP_i = |2^{3i} \cdot d_i \cdot X|_{2^n-1} \quad (12)$$

To include the bias B necessary for partially-redundant representation of PP_i, (12) is modified to

$$PP_i = |2^{3i} (B+d_i \cdot X)|_{2^n-1} \quad (13)$$

Using *Property 3*, the modulo 2^n-1 multiplication by 2^{3i} in (13) is efficiently implemented as bitwise circular-left-shift of the biased multiple, $(B+d_i \cdot X)$. For n=8 and k=4 ,Fig. 8 illustrates the partial product matrix of $X \cdot Y)_{2^8-1}$with

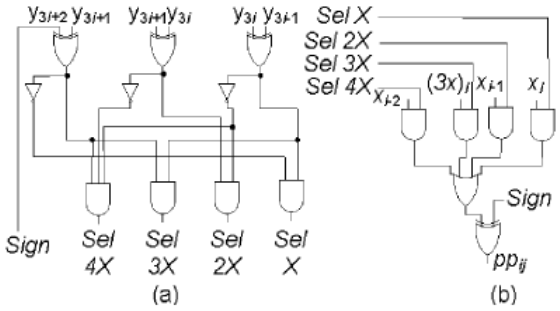


Fig.10. (a) Bit-slice of Booth Encoder (BE). (b) Bit-slice of Booth Selector (BS).

(n/3+1) partial products in partially-redundant representation.Each PP_i consists of an n-bit vector, pp_{i7}--pp_{i1} pp_{i0} and a vector of n/k=2 redundant carry bits, q_{i1} and q_{i0} . Since q_{i0} and q_{i1} are the carry-out bits of the RCAs, they are displaced by k-bit positions for a given PP_i. The bits, q_{ij} is displaced circularly to the left of $q_{(i-1)\,j}$ by 3 bits, *i.e.*, q_{20} and q_{21} are displaced circularly to the left of q_{10} and q_{11} by 3 bits, respectively and q_{10} and q_{11} are in turn displaced to the left of q_{00} and q_{01} by 3 bits, respectively. The last partial product in

Fig. 8 is the Compensation Constant (*CC*) for the bias introduced in the partially-redundant representation.

The generation of the modulo-reduced partial products, PP_0, PP_1 and PP_2, in a partially-redundant representation using Booth Encoder (BE) and Booth Selector (BS) blocks are illustrated in Fig. 8. The BE block produces a signed one-hot encoded digit from adjacent overlapping multiplier bits as illustrated in Fig. 10(a). The signed one-hot encoded digit is then used to select the correct multiple to generate PP_i. A bit-slice of the radix-8 BS for the partial product bit, PP_{ij} is shown in Fig. 10(b).

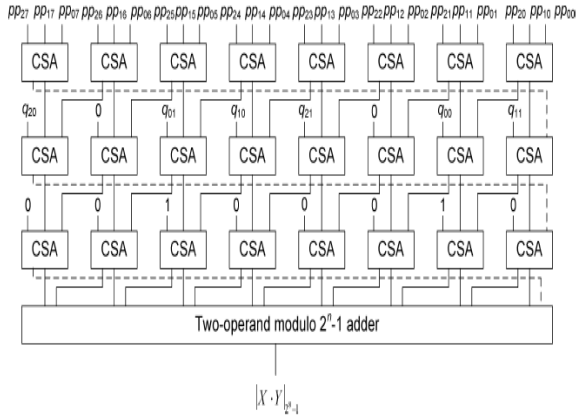


Fig. 11. Modulo-reduced partial product accumulation.

As the bit positions of q_{ij} do not overlap, as shown in Fig. 8, they can be merged into a single partial product for accumu-lation.The merged partial products , PP_i and the constant *CC* are Accumulated using a CSA tree with end-around-carry addition at each CSA level and a final two-operand modulo 2^n-1 adder as shown in Fig. 11.

4. RESULTS

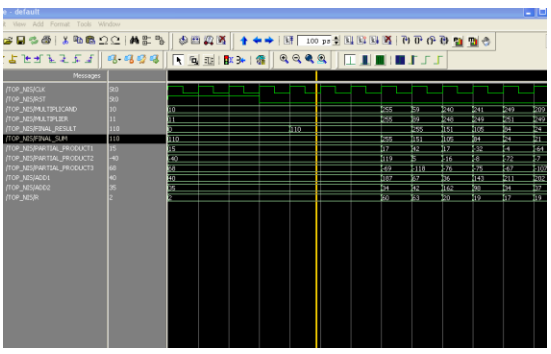


Fig. 12.Multiplier Simulation Results

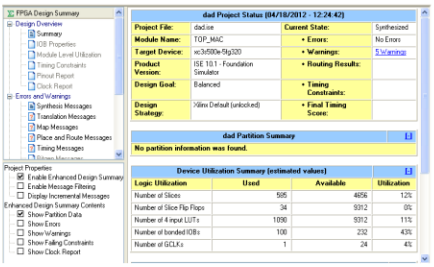


Fig 13.Radix-4 Synthesis Report(Gate Count)

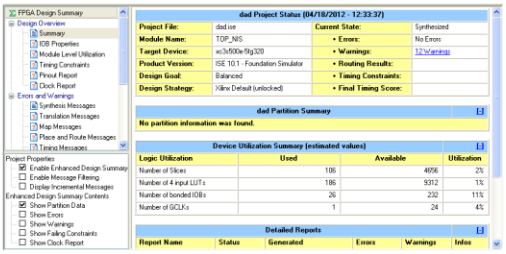


Fig.14.Radix-8 Synthesis Report (Gate Count)

5. CONCLUSION

A family of low-area and low-power modulo 2^n-1 multipliers with variable delay to achieve delay balance amongst individual modulo channels in a high-DR RNS multiplier was proposed. The delay of the proposed multiplier is controlled by the word-length of the small parallel RCAs that are used to compute the requisite hard multiple of the radix-8 Booth encoded multiplication in a partially-redundant form. From synthesis results constrained by the critical channel delay of the RNS, it was shown that the proposed multiplier simultaneously reduces the area as well as the power dissipation of the radix-4 Booth encoded multiplier for $n \geq 28$, which is the useful dynamic range of RNS multiplication to meet the minimum key-size requirements of ECC and RSA algorithms.

REFERENCES

[1] R. Rivest, A. Shamir, and L.Adle-man, "A method for obtaining digital signatures and public key cryptosy-stems," Commun. ACM, vol. 21, no.2, pp. 120–126, Feb. 1978.[2] V. Miller, "Use of elliptic curves in cryptography," in Proc. Advancesin Cryptology-CRYPTO'85, Lecture Notes in Computer Science, 1986,vol. 218, pp. 417–426.
[3] N. Koblitz, "Elliptic curve cryptosys-tems," Mathemat. of Comput., vol.48, no. 177, pp. 203–209, Jan. 1987.
[4] National Institute of Standards and Technology. Available: http:// csrc.nist.gov/publications/PubsSPs.html
[5] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes,"J. Cryptol., vol. 14, no. 4, pp. 255–293, Aug. 2001.
[6] C. McIvor, M. McLoone, and J. V. McCanny, "Modified Montgomery modular multiplication and RSA exponentiation techniques," IEE Proc. Comput. and Dig. Techniq., vol. 151, no. 6, pp. 402–408, Nov.2004.
[7] C. McIvor,M.McLoone, and J. V. McCanny, "Hardware elliptic curve cryptographic processors over ," IEEE Trans. Circuits Syst. I,Reg. Papers, vol. 53, no. 9, pp. 1946–1957, Sep. 2006.
[8] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas,and T. Stouraitis, "An RNS implementation of an Elliptic curve point multiplier," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no.6, pp. 1202–1213, Jun. 2009.
[9] J. C. Bajard and L. Imbert, "A full RNS implementation of RSA," IEEE Trans. Comput. – Brief Contributions, vol. 53, no. 6, pp. 769–774, Jun.2004.
[10] H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, "Implementation of RSA algorithm based on RNS Montgomery multiplication," in Proc. Workshop on Cryptographic Hardware and Embedded Systems,Paris, France, May 2001, pp. 364–376.
[11] T. Stouraitis and V. Paliouras, "Considering the alternatives in lowpower design," IEEE Circuits Devices Mag., vol. 17, no. 4, pp. 22–29,Jul. 2001.
[12] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in Proc. 14th IEEE Int. On-Line Testing Symp., Rhodes, Greece, Jul. 2008, pp. 192–194.

[13]

Mr.K.Ramamohan Reddy,is currently doing post graduation in Vagdevi institute of Tech & Science,Proddatur,Kadapa(Dt) A.P with the specialization of VLSI

[14]

Mr.V.Ramesh,is an Assistant Professor in ECE dept. at Vagdevi institute of Tech&Science,Proddatur.He obtained his B.Tech degree in ECE from MeRITS,JNTUA,Udayagiri in 2007,M.Tech degree in Electronic Instrumentation and Communication Systems from S.V.University,Tirupati in 2009.He has 3years of teaching experience and his area of interest includes Electronics instrumentation and Antennas.He has published 4 papers in referred international journals and also 2papers at national level conferences.

[14]

Mr.C.Mahammed Aslam is the HOD,Dept.of ECE at Vagdevi institute of Tech&Science,Proddatur.He obtained his B.Tech degree in ECE from Dr.Babasaheb Ambedkar Maratwada University,Aurangabad in 1997,M.Tech degree in Digital Electronics and Computer Science from JNTU,Hyderabad in 2007.He ahs 10 years of teaching experience.He registered his Ph.D from JNTU,Anantapur in digital Image Processing.His area of interest is Microprocessor and EDC circuits. He has published 2 papers in international journals.