International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 10, December- 2012

RNS And Adders Based Radix-8 2^n-1 Modulo Multiplier

Reshma Janardhan
M.Tech Student, LIET, Hyderabad, AP ,India

Abstract

The modulo 2^n-1 multiplier is usually the non critical data path among all modulo multipliers in such high-DR RNS multiplier. This timing slack can be exploited to reduce the system area and power consumption without compromising the system performance. Several commercial processors have selected the radix-8 multiplier architecture to increase their speed, thereby reducing the number of partial products. Radix-8 encoding reduces the digit number length in a signed digit representation. Its performance bottleneck is the generation of the term 3X, also referred to as hard multiple. The modified booth encoder will reduce the number of partial products generated by a factor of 2 using radix-4 but by using radix-8 the partial products reduces by a factor of n/3.

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. To facilitate design of high-speed full-adder based modulo arithmetic units, it is worthwhile to keep the moduli of a high-DR RNS in forms of {2^n-1, 2^n, 2^n +1}.The modulo 2^n-1 multiplier is usually the non critical data path among all modulo multipliers in such high-DR RNS multiplier. 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 modulo 2^n-1 multiplier delay is made scalable by controlling the word-length of the ripple carry adder, employed for radix-8 hard multiple generation.

Keywords- *Residue number system(RNS), RSA, Radix 8 booth encoding, moduli set, prefix structure.*

1. Introduction

Two Algorithms that are mostly established in public Key cryptography are RSA (Rivest and Shamir) and ECC (Elliptical curve cryptography)[1]-[3]. For these algorithms Encryption and decryptions are performed by modulo multiplications. During encryption and decryption process of RSA ECC key sizes ranges from 512 1024 bits and 106 512 bits[3]-[6]. Due to this it becomes difficult to implement hardware because of Long carry propagation of large inter multiplies. To make it easy for hardware implementation Residue Number system has come into existence to design low power and faster multipliers.

RNS (Residue Number System) is based on generic moduli. But we are going to prepare special moduli form of 2n-1 over generic moduli for the hardware implementation. There is triple moduli set {2^n-1, 2^n, 2^n+1} And the speed of RNS processor of 2^n-1 is increasingly dominated by the residue arithmetic operation rather than one time forward and one time back ward or reverse conversion .[7]

1.1. Modulo 2^n-1 Multiplier Architecture

Modulo multiplier are used effectively for the area and time purpose. The design of 2^n-1 modulo multiplier consists of three as shown in figure 1, steps one is modulo particial product generation then second slip for addition of those partial products and the third step for the addition of carry and propagate bits of the adder. The fig. 1 represents the modulo multiplier architecture. The modulo multiplication is extensively used in Residue number system (RNS) based digital signal processing and cryptography units.

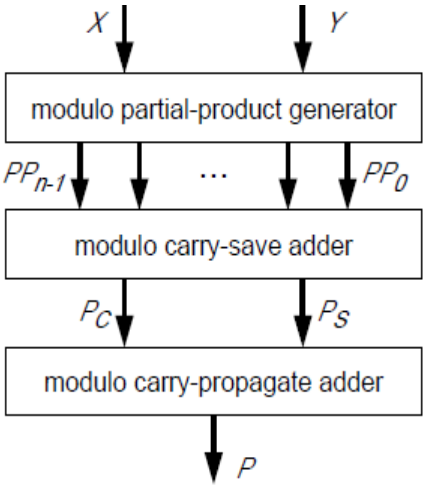


Figure 1: Modulo (2n-1) Multiplier Architecture.

2. Review

Radix-8 is the advanced technique than that of Radix4. In Radix 4 the multiplier the partial product generation the multiplicand is taken and is divided into digits denoted by D_i where D_i is one of {-2,-1,0,1,2)

Radix 4 are used for generation of soft multiples shown in table 1, where as Radix 8 is used for generation of both hard and soft multiples as shown in table 4.

Table 1:Modulo Reduced Multiples For The Radix 4 Booth Encoding

D_i	$\lvert d_i.X\rvert_{2^n-1}$
+0	$\underbrace{0.....0}_{n}$
+1	X
+2	CLS(X,1)
-2	CLS($\overline{X}$,1)
-1	$\overline{X}$
-0	$\underbrace{1.....1}_{n}$

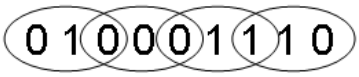
Table 2: Radix 4 Recoding.

d_i	**Signed values**
X_{i+2} X_{i+1} X_i	
0 0 0	0X
0 0 1	1X
0 1 0	1X
0 1 1	2X
1 0 0	-2X
1 0 1	-1X
1 1 0	-1X
1 1 1	0

In Radix 4 booth algorithm it starts by appending a zero to the right of X_0 and three bits are take once as an input and they are denoted by Di.

Example: **01000111-multiplicand**

01001011-multiplier



The number 1X represent the same number, 0X represents all digits are zero and 2x represents circular left shift by one position. The negative symbol for ix i.e –1x means 2's complement & -2x means 2's complement and circular left shift by one position.

Whereas for Radix 8 for the particle product generation the 8 bit multiplicand is considered and a zero bit is added at MSB LSB of it.

Unlike the Radix 4 in Radix 8 we are grouping 4 bits each as one group and hence we obtain three partial products. Thus reduci8ng one of the partial product when compared with Radix4.

Table 3: Modulo Reduced Multiples For Radix 8 Booth Encoding.

d_i	$\lvert d_i \cdot X \rvert_{2^n-1}$	d_i	$\lvert d_i \cdot X \rvert_{2^n-1}$
+0	$\underbrace{0\cdots0}_{n}$	-0	$\underbrace{1\cdots1}_{n}$
+1	X	-1	$\bar{X}$
+2	$CLS(X,1)$	-2	$CLS(\bar{X},1)$
+3	$\lvert +3X \rvert_{2^n-1}$	-3	$\lvert -3X \rvert_{2^n-1}$
+4	$CLS(X,2)$	-4	$CLS(\bar{X},2)$

Tables 4: 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

In Radix 8 we are having hard multiple generation i.e 3x. and this 3x is obtained by 2x +x i.e. circular left shift of the multiplicand added with the multiplier. Here, the Di ranges between {-4, -3, -2, -1, 0, 1, 2, 3, 4}

The radix8 booth encoding reduces the partial products to (n/3)+1 from that of radix 4.

After obtaining the partial products from the multiplier and the multiplicand they are given it to adder. the result of 2^n-1 multiplier is the reduced output i.e we obtain $|x.y|_{2n-1}$. And in this project the maximum range of the output is of only 255 because its designed for only 8 bit number.

3. Proposed Radix-8 Multiplier

3.1. Generation Of Hard Multiple

In RNS multiplier the carry propagation length in the hard multiple generation should not exceed 'n' bits so the carry propagation through the half adders as shown in figure 2 can be eliminated by making end around carry bit c_7 a partially product to be accumulated in the CSA (carry save accumulator) tree in figure 3. so this technique reduces the carry propagation length to n bits as shown in figure below.

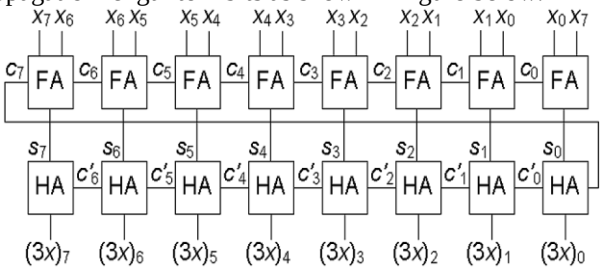


Figure 2: Generation Of Hard Multiple Architecture $|+3X|_{2^n-1}$

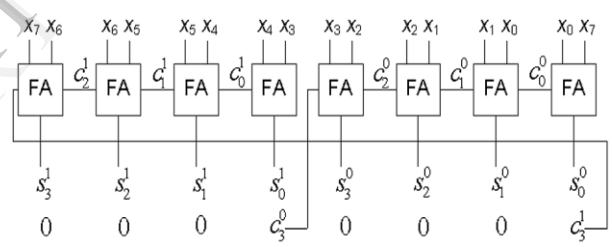


Figure 3: Modified Architecture Of Hard Multiplier Using K Bit RCA's

The carry out of RCA0 i.e. C_3^0 is not propagated to the input of RCA 1 but is taken as one of the partial product bit to be accumulated in CSA tree. The output of the RCA1 i.e. C_3^1 is been reduced before being accumulated. Therefore the partially redundant for of $|+3X|_{2n-1}$ is given by

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

And the partially redundant form of $|-3X|_{2n-1}$ is given by

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

To avoid many long strings of ones in $\overline{C}$ an appropriate bias B is added to hard multiple so that C and $\overline{C}$ are sparse.

$$B = \sum_{j=0}^{M-1} 2^{K.j} = \overbrace{0....01....0....01}^{n} \underbrace{\qquad}_{k} \underbrace{\qquad}_{k} \quad \text{.....(3)}$$

The block diagram for $|B+3X|_2 n_{-1}$ is as shown in figure 4 and its generated by simple XNOR, OR operations on the bit positions 2^{kj}.

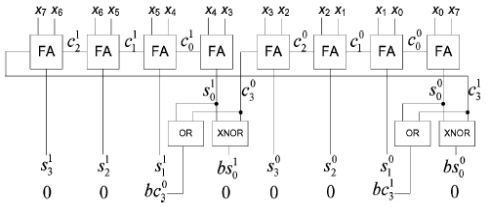


Figure 4: Generation Of Partial Redundant Of $|B+3X|_2^n{}_{-1}$

3.2. Generation Of Soft Multiples

Since the hard multiple can't be predicted at design time, all the multiples must be similarly represented. So the bias is added to the soft multiples also. It is represented as $|B+0|_2 n_{-1}$, $|B+X|_2 n_{-1}$, $|B+2X|_2 n_{-1}$, $|B+4X|_2 n_{-1}$ in the partial redundant form for 8-bit. the figure 5 represents the simple multiples.

0	0	0	1	0	0	0	1	B+0
		0				0		
x_7	x_6	x_5	$\bar{x}_4$	x_3	x_2	x_1	$\bar{x}_0$	B+X
		x_4				x_0		
x_6	x_5	x_4	$\bar{x}_3$	x_2	x_1	x_0	$\bar{x}_7$	B+2X
		x_3				x_7		
x_5	x_4	x_3	$\bar{x}_2$	x_1	x_0	x_7	$\bar{x}_6$	B+4X
		x_2				x_6		

Figure 5: Generation Of Partially Redundant Of Simple Multiples

3.3. Generation Of Partial Product

To generate partial product it requires booth selector and booth encoder. to generate each partial product it requires one booth encoder and ten booth selector. in Radix 8 we can generate three partial products pp0 , pp1, pp2. the booth encoder produces signed one hot encoded digit from adjacent overlapping multiplier bits. The signed one hot encoded digit is then used to select the correct multiple to generate pp_i. The block diagram of booth encoder and booth selector is as shown in figure 6.

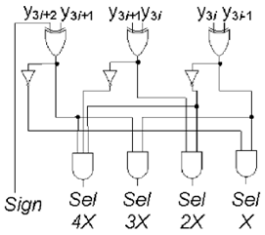


Figure 6.a: Booth Encoder

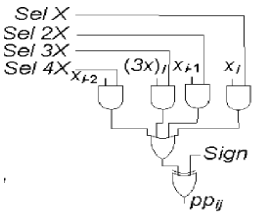


Figure 6.b: Booth Selector

In booth encoder the selected inputs i.e D_i is given and we obtain any one as output from X,2X,3X,4X with respect to its sign value and the outputs are then again given as input to booth selector.where as to booth selector we give inputs as output of booth encoder and also the D_i bits and we obtain the output as the partial product pp_{ij}.In partial product generation we obtain PP_{ij} and q_{ij} values. Each PP_{ij} consists of an n-bit vector, pp_{i7}.....$pp_{i1}pp_{10}$ and a vector of n/k = 2 and redundancy carry bits $q_{i1}q_{i0}$ are the carryout bitsas in figure 7 and 8.After we obtain the partial product and redundancy carry bits we add compensation constant for the bias introduced in partial redundant representation

	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
X						d_2	d_1	d_0
	pp_{07}	pp_{06}	pp_{05}	pp_{04}	pp_{03}	pp_{02}	pp_{01}	pp_{00}
			q_{01}				q_{00}	
	pp_{17}	pp_{16}	pp_{15}	pp_{14}	pp_{13}	pp_{12}	pp_{11}	pp_{10}
				q_{10}				q_{11}
	pp_{27}	pp_{26}	pp_{25}	pp_{24}	pp_{23}	pp_{22}	pp_{21}	pp_{20}
	q_{20}				q_{21}			
	0	0	1	0	0	0	1	0

Figure 7: Reduced Partial Product Generation

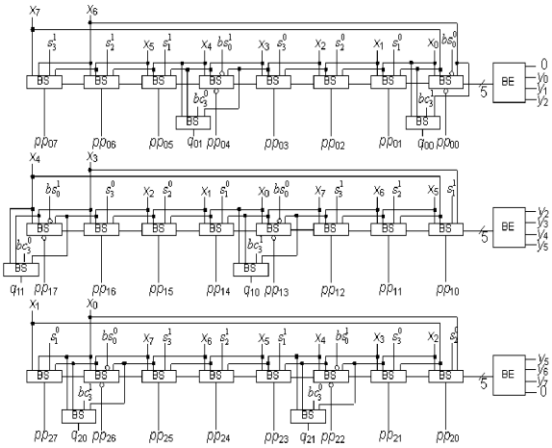


Figure 8: Modulo Reduced Partial Product And CC For $|X.Y|_{2^n-1}$

3.4. Addition Of Partial Products

The obtained partial product are accumulated in the carry save adder and also the partial carry redundant bit along with bias is given to the parallel prefix adder as in figure 9. Thus the output obtained is $|X.Y|_{2^n-1}$

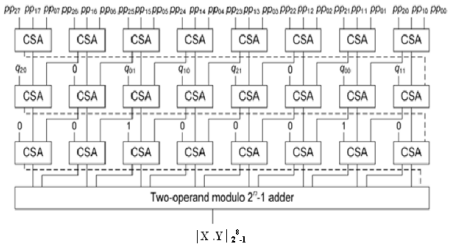


Figure 9: modulo reduced partial product accumulation

3.5. Modification Of Adder

Now instead of using parallel prefix adder after accumulation we are using carry look ahead adder. So let us consider the case where the underlying adder is a one-level carry -look ahead adder The logic equation for such an adder are

$G_i = A_i B_i$

$P_i = \overline{A}_i \; B_i + A_i \overline{B}_i = A_i \oplus B_i$

$C_i = G_i + P_i C_{i-1}$

$S_i = P_i \oplus C_{i-1}$

$$C_i = G_i + P_{iGi-1} + P_i P_{i-1} G_{i-2} + ... P_i P_{i-1} .... P_1 G_0 G_{n-1} P_i P_{i-1} ... P_0 + P_{n-1} G_{n-2} P_i P_{i-1} ..... P_0 + G_{i+1} P_i P_{i-1} ..... P_0 P_{n-1} .... P_{i+2}$$

when the output is obtained by parallel prefix adder the number of the gate count is less than that of CLA. where as instead of parallel prefix adder if we give input to the carry look ahead adder the no of gate count is less when compared to previous. The implies that the area is reduced in CLA adder than that of parallel prefix adder.

4. Result

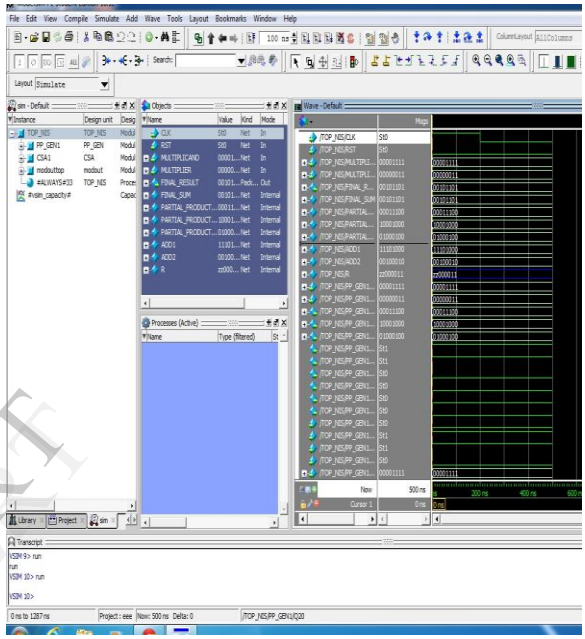


Figure 10 : Multiplier Simulation Results

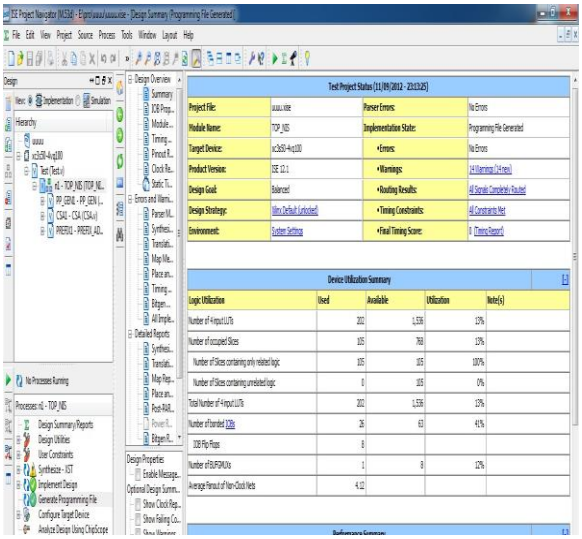


Figure 11:Radix-8 Synthesis Report (Gate Count) using parallel prefix adder

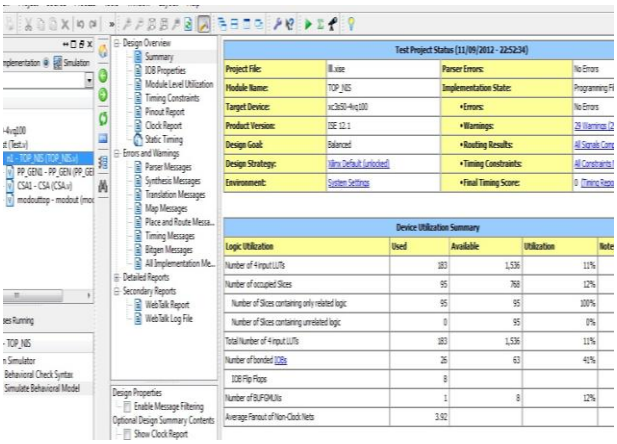


Figure 12 :Radix-8 Synthesis Report (Gate Count) using carry look ahead adder

5. Conclusion

A new approach for modulo multiplication of 2^n-1 has been approached and the partial products, hard and soft multiples are accomplished by the AND XNOR and OR gates. Thus a family of low area and low power modulo 2^n-1 multiplies was proposed The decay of the proposed multiplies is controlled by word length of the small parallel RAS which is used to compute hard multiples From synthesis result its Clearly shown that the area is used by CLA is less when compare to parallel prefix And also its useful for dynamic range for RNS multiplication.

6. Reference

[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] D. J. Soudris, V. Paliouras, T. Stouraitis, and C. E. Goutis, "A VLSI design methodology for RNS full adder-based inner product architectures," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol.44, no. 4, pp. 315–318, Apr. 1997.