International Journal of Electrical and Electronic Engineering & Telecommunications

ISSN 2319 – 2518 www.ijeetc.com
Vol. 3, No. 1, January 2014

Research Paper

DESIGN OF BOOTH ENCODED MODULO 2^n-1 MULTIPLIER USING RADIX-8 WITH HIGH DYNAMIC RANGE RESIDUE NUMBER SYSTEM

N Niranjan[1]* and P Sreenivasa Rao[1]

*Corresponding Author: N Niranjan, ✉ niranjan.n38@gmail.com

A special moduli set Residue Number System (RNS) of high Dynamic Range (DR) can speed up the execution of very-large word-length repetitive multiplications found in applications like public key cryptography. The modulo 2^n-1 multiplier is usually the noncritical datapath 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. 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, *k* employed for radix-8 hard multiple generation. Formal criteria for the selection of the adder word-length are established by analyzing the effect of varying *k* on the timing of multiplier components. It is proven that for a given *n*, there exist a number of feasible values of *k* such that the total bias incurred from the partially-redundant partial products can be counteracted by only a single constant binary string. This compensation constant for different valid combinations of n and *k* can be precomputed at design time using number theoretic properties of modulo 2^n-1 arithmetic and hardwired as a partial product to be accumulated in the carry save adder tree. The adaptive delay of the proposed family of multipliers is corroborated by CMOS implementations. In an RNS multiplier, when the critical modulo multiplier delay is significantly greater than the noncritical modulo 2^n-1 multiplier delay, $k = n$ and $k = n/3$ are recommended for *n* not divisible by three and divisible by three, respectively. Conversely, when this difference diminishes, *k* is better selected as $n/4$ and $n/6$ for *n* not divisible by three and divisib le by three, respectively. Our synthesis results show that the proposed radix-8 Booth encoded modulo 2^n-1 multiplier saves substantial area and power consumption over the radix-4 Booth encoded multiplier in medium to large word-length RNS multiplication.

Keywords: Residue Number System (RNS), Booth algorithm, Multiplier, Radix-8

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

[1] Department of ECE, SRTS, Kadapa, Andhra Pradesh, India.

algorithms are performed by repeated modulo multiplications. 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. 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. RNS has also been successfully employed to design fault tolerant digital circuits.

A RNS is defined by a set of pair-wise co-prime moduli $\{L_1, L_2, \ldots, L_n\}$ such that any integer within the Dynamic Range (DR), i.e., L_1 is represented as an-tuple, where is the rmuesidue of modulo. RNS multipliers based on generic moduli have been reported in. However, special moduli of forms 2^n or $2^n \pm 1$ 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 detection. The most popular of these special moduli sets is the triple moduli set $\{2^n - 1, 2^n, 2^n \pm 1\}$ which however has a DR of only bits. It is obvious that the DR of an existing moduli set can be extended by appending many small word-length moduli or a few large word-length moduli.

As the time complexity of partial product summation by a Carry Save Adder (CSA) tree and a two-operand parallel-prefix adder is a logarithmic function of, the critical path delay can be modeled as, but the delays of the modulo and modulo multipliers are only. This speedup of around by modulo and modulo multipliers over the critical path delay is of no consequence.

As encryption and decryption in PKC involves repeated multiplications, the cumulative difference in the critical and noncritical modulo multiplier delays will increase with the number of multiplications involved. For lightweight cryptographic applications, such as smartcards and Radio Frequency Identification (RFID) tags, the considerations of power, size and cost are of paramount importance.

The complexity of implementing reliable cryptographic hardware can be reduced by an ingenious exploitation of this timing headroom in the design of RNS multiplier. The noncritical modulo multipliers can be made to operate at a slower speed that nearly matches the delay of the critical modulo multiplier. In doing so, the timing slack freed up from the modulo and modulo multipliers can be effectively explored for more area and power efficient architectures without compromising the overall system performance. This approach to reduce the overall area and power consumption of a RNS multiplier is based on architectural modification and can be implemented with any standard cell library. It does not require multiple supply voltages, multiple threshold voltages, or control circuitries for the generation and scaling of voltage and frequency in order to exploit the timing surplus in the noncritical paths for power saving.

This paper focuses on the design space exploration of arithmetic operation in one of

the two special moduli, i.e., $2^n - 1$, the modulo multiplier design. The Montgomery modulo multiplication, while computing the modular product without trial division, is modulus-independent and incapable of exploiting number theoretic properties of modulo $2^n - 1$ arithmetic for combinational circuit simplification. The properties of modulo $2^n - 1$ arithmetic were most effectively exploited for the full adder based implementation of modulo multiplier in. In the multiplier bits were not encoded, which lead to higher implementation area and longer partial product accumulation time.

In and, the radix-4 Booth encoding algorithm was employed to reduce the number of partial products to $n/2 + 1$ and $n/2$, respectively. The shorthand notations [a]and [a]denote the smallest integer greater than or equal to[a] and the largest integer smaller than or equal to respectively. With higher radix Booth encoding the number of partial products is reduced by more than half and consequently, significant reduction in silicon area and power dissipation is feasible. The radix-8 Booth encoding reduces the number of partial products to $n/3+1$, which is more aggressive than the radix-4 Booth encoding.

However, in the radix-8 Booth encoded modulo multiplication, not all modulo-reduced partial products can be generated using the bitwise circular-left-shift operation and bitwise inversion.

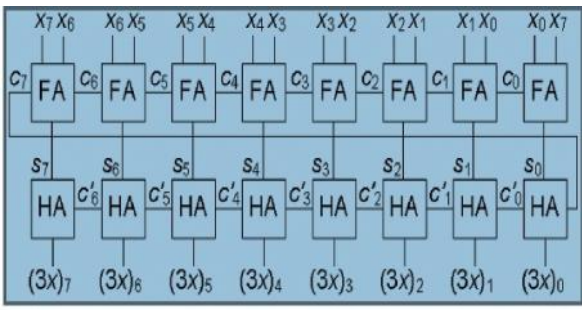
The performance overhead due to the end-around-carry addition is by no means trivial and hence, the use of Booth encoding for modulo multipliers have been restricted to only radix-4 in literature.

In this paper, we propose the first-ever family of low-area and low-power radix-8 Booth encoded modulo multipliers whose delay can be tuned to match the RNS delay closely. In the proposed multiplier, the hard multiple is generated using small word-length Ripple Carry Adders (RCAs) operating in parallel. The Appendix provides the derivation of the predetermined compensation constant for different valid combinations of the multiplier and RCA word-lengths.

RADIX-8 BOOTH ENCODED MODULO MULTIPLICATION $2^n - 1$ ALGORITHM

By using above technique booth multiplication can be performed. But this technique $+3X2^n - 1$ for computation involves two -bit carry-propagate additions in series such that the carry propagation length is twice the operand length In the worst case, the late arrival of $+3X 2^n - 1$ the may considerably delay all subsequent stages of the modulo $2^n - 1$ multiplier. Hence, this approach for hard multiple generation can no longer categorically ensure that the multiplication in the modulo channel still falls in the noncritical path of a RNS multiplier.

Figure 1: Generation of $+3X2^n - 1$ Using Two Bit RCAs



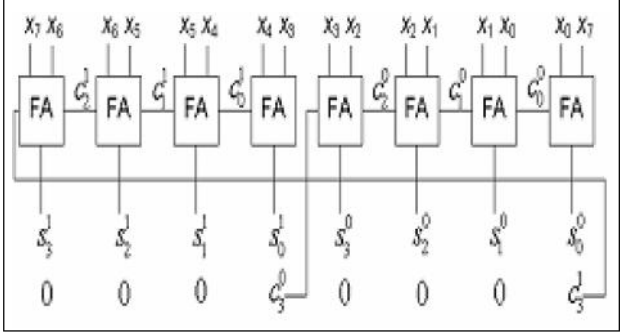
In what follows, we propose a family of low-power and low area modulo $2^n - 1$ multipliers based on the radix-8. Booth encoding, which allows for an adaptive control of the delay to match the delay of the critical modulo channel of a RNS multiplier.

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

To ensure that the radix-8 Booth encoded modulo $2^n - 1$ multiplier 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 Figure 2 can be eliminated by making the end-around-carry bit c7 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 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.

So above technique used in our project to reduce the carry length and increase the speed of the system.

Figure 2: Generation of Partially-Redundant $+3X2^n - 1$ Using Bit RCAs



Radix-8 Booth Encoded Modulo $2^n - 1$ Multipliers Diagram

Figure 3: Generation of Partially-Redundant Bits

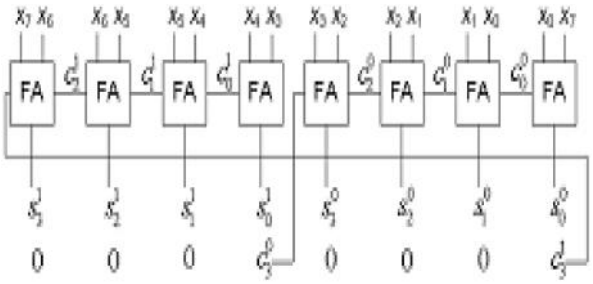


Figure 4: Partial Product Generator

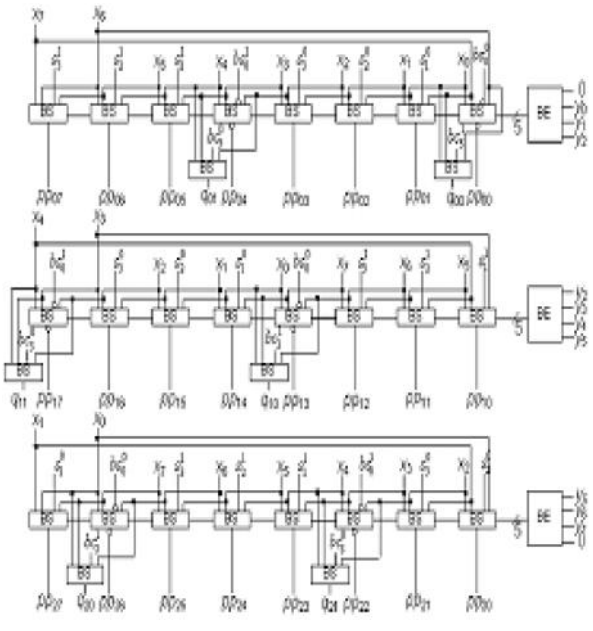


Figure 5: Booth Encoder, Booth Selector

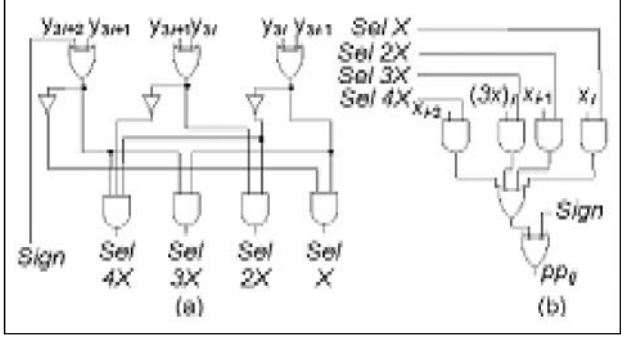
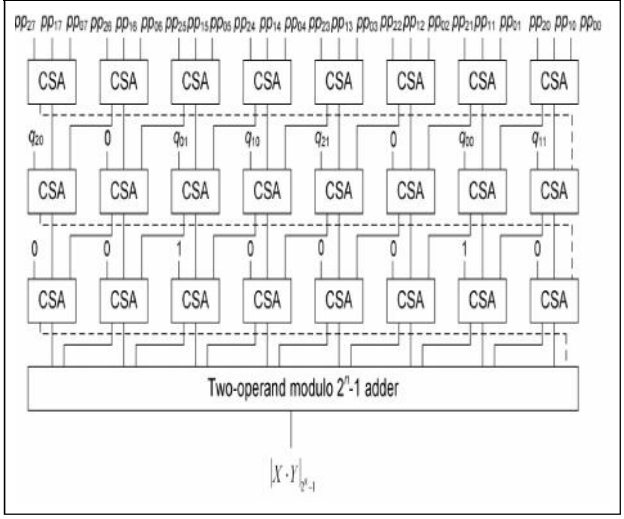


Figure 6: Carry Save Adder



BOOTH'S MULTIPLICATIO ALGORITHM OF RESIDUE

Number System

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950 while doing research on crystallography at Birkbeck College in Bloomsbury, London. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.

BOOTH ALGORITHM

Booth's algorithm examines adjacent pairs of bits of the *N*-bit multiplier *Y* in signed two's complement representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for *i* running from 0 to *N*–1, the bits yi and y_{i-1} are considered. Where these two bits are equal, the product accumulator *P* remains unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to *P*; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from *P*. The final value of *P* is the signed product.

The representation of the multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. As stated here, the order of the steps is not determined.

Typically, it proceeds from LSB to MSB, starting at $i = 0$; the multiplication by 2^i is then typically replaced by incremental shifting of the *P* accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest *N* bits of *P*. There are many variations and optimizations on these details.

The algorithm is often described as converting strings of 1's in the multiplier to a high-order +1 and a low-order –1 at the ends of the string. When a string runs through the MSB, there is no high-order +1, and the net effect is interpretation as a negative of the appropriate value.

A TYPICAL IMPLEMENTATION

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned

binary addition) one of two predetermined values *A* and *S* to a product *P*, then performing a rightward arithmetic shift on *P*. Let m and *r* be the multiplicand and multiplier, respectively; and let *x* and *y* represent the number of bits in *m* and *r*.

1. Determine the values of *A* and *S*, and the initial value of *P*. All of these numbers should have a length equal to (*x* + *y* + 1).
   - *A*: Fill the most significant (leftmost) bits with the value of *m*. Fill the remaining (*y* + 1) bits with zeros.
   - *S*: Fill the most significant bits with the value of (⁻*m*) in two's complement notation. Fill the remaining (*y* + 1) bits with zeros.
   - P: Fill the most significant *x* bits with zeros. To the right of this, append the value of *r*. Fill the least significant (rightmost) bit with a zero.
2. Determine the two least significant (rightmost) bits of *P*.
   - If they are 01, find the value of *P* + *A*. Ignore any overflow.
   - If they are 10, find the value of *P* + *S*. Ignore any overflow.
   - If they are 00, do nothing. Use *P* directly in the next step.
   - If they are 11, do nothing. Use *P* directly in the next step.
3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.
4. Repeat steps 2 and 3 until they have been done y times.

Drop the least significant (rightmost) bit from *P*. This is the product of m and *r*.

BOOTH RECODING

Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is the standard technique used in chip design, and provides significant improvements over the "long multiplication" technique.

SHIFT AND ADD

A standard approach that might be taken by a novice to perform multiplication is to "shift and add", or normal "long multiplication". That is, for each column in the multiplier, shift the multiplicand the appropriate number of columns and multiply it by the value of the digit in that column of the multiplier, to obtain a partial product. The partial products are then added to obtain the final *r*

0 0 1 0 1 1

0 1 0 0 1 1

0 0 1 0 1 1

0 0 1 0 1 1

0 0 0 0 0 0

0 0 0 0 0 0

0 0 1 0 1 1

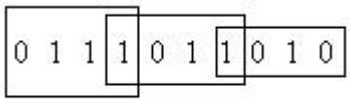
0 0 1 1 0 1 0 0 0 1

with this system, the number of partial products is exactly the number of columns in the multiplier.

Radix-8 Booth Recoding

To Booth recode the multiplier term, we consider the bits in blocks of four, such that each block overlaps the previous block by one

bit. Grouping starts from the LSB, and the first block only uses three bits of the multiplier



Grouping of bits from the multiplier term, for use in Booth recoding. The least significant block uses only three bits of the multiplier, and assumes a zero for the fourth bit.

The overlap is necessary so that we know what happened in the last block, as the MSB of the block acts like a sign bit. We then consult the table to decide what the encoding will be.

Table 1: Booth Encoding

Block	Partial Product
0000	0
0001	1* Multiplicand
0010	1* Multiplicand
0011	2* Multiplicand
0100	2* Multiplicand
0101	3* Multiplicand
0110	3* Multiplicand
0111	4* Multiplicand
1000	–4* Multiplicand
1001	–3* Multiplicand
1010	–3* Multiplicand
1011	–2* Multiplicand
1100	–2* Multiplicand
1101	–1* Multiplicand
1110	–1* Multiplicand
1111	0* Multiplicand

Note: 0 = same as zero; 1 = same as multiplicand; –1 = compliment of multiplicand; 2 = circular left shif by 1 bit position; –2 = circular left shift (compliment of multiplicand) one position; 3 = $x + 2x$ (it means x is the multiplicand, $2x$ is circular shift of x; –3 = compliment of $(x + 2x)$; 4 = circular shift of multiplicand by 2 bits position; –4 = complement of (4).

RESIDUE NUMBER SYSTEM

Residue Number System (RNS) represents a large integer using a set of smaller integers, so that computation may be performed more efficiently. It relies on the Chinese remainder theorem of modular arithmetic for its operation, a mathematical idea from Sun Tsu Suan-Ching (Master Sun's Arithmetic Manual) in the 4th century AD.

Defining A Residue Number System

A residue number system is defined by a set of N integer constants,

$$\{m_1, m_2, m_3, ..., m_N\},$$

referred to as the moduli. Let M be the least common multiple of all the m_i.

Any arbitrary integer X smaller than M can be represented in the defined residue number system as a set of N smaller integers

$$\{x_1, x_2, x_3, ..., x_N\}$$

with

$$x_i = X \text{ modulo } m_i$$

representing the residue class of X to that modulus.

Note that for maximum representational efficiency it is imperative that all the moduli are coprime; that is, no modulus may have a common factor with any other. M is then the product of all the m_i.

For example RNS(4|2) has non-coprime moduli, resulting in the same representation for different values.

SELECTION OF k

The guidelines for choosing the RCA word-length, k, to achieve the desired performance are presented in this section.

Firstly, irrespective of the targeted delay, the choice of *k* must satisfy the following two criteria.

Criterion 1: As the residues of modulus 2^n–1 are represented using only n bits, it is imperative that k divides *n*. *k* = 1 is a trivial case and is excluded from this consideration. This criterion is expressed as *k*/*n*, *kf*\.

Criterion 2: Since each partial product in radix-8 Booth encoding is shifted by three bits relative to the previous partial product, *k* must not be a multiple of three to ensure that the *qy* bits are nonoverlapping. Therefore, 3*fk*.

In the proposed modulo 2^n–1 multiplier, each partial product PPi is incremented by a bias of 2^{3i}**B* as expressed in (13). To negate the effect of the bias, a constant CC is added and the value of CC is given bywhere *B* is an *n*-bit binary word consisting of logic one at bit position 2*k*, *j* € [0, *M*-1] and logic zero at all other positions as defined is (7). It is evident that the value of CC depends only on *n* and *k* As CC is considered as one or more partial products to be summed in the CSA tree, the choice of *k* indirectly determines the regularity of the multiplier design and consequently its efficiency in VLSI implementation. A detailed analysis on the computation of CC for various combinations of *n* and *k* is presented in the Appendix. For any *k* that satisfies Criteria 1 and 2, it is shown that CC can be simplified by the properties of modulo 2^n–1 arithmetic and precomputed at design time.

The resultant CC is shown to be a single binary word with a specific repetitive pattern of logic ones and zeros. As the generation of CC involves merely the assignment of logic.

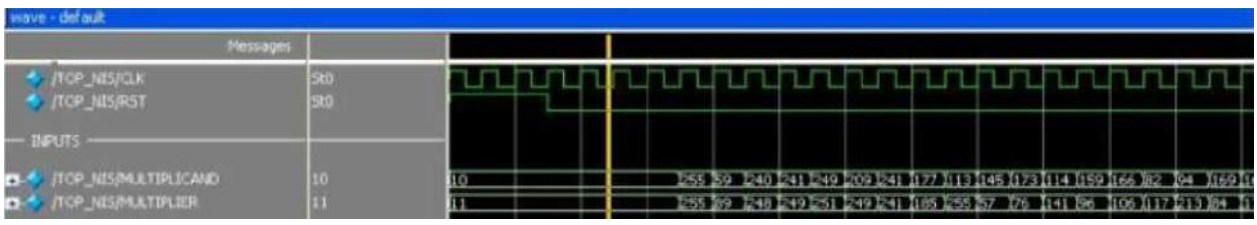
SIMULATION RESULT

Figure 7 shows the output waveforms of Radix-8 booth Encoded Modulo 2^n–1 multiplier for various inputs. If the number of bits for multiplier and multiplicand are 8, i.e., *n* = 8 that means modulo 255 multiplier. The modulo result is the least positive remainder when the decimal multiplication result of the inputs is divided by the modulus 255. Hence if the decimal multiplication result is less than 255, the modulo result is the same as the decimal multiplication result of the inputs. If the decimal multiplication result of the inputs is 255, the modulo result is also same, i.e., 255.

CONCLUSION

This radix-8 booth modulo-1 multiplication algorithm performed mulitiplication operation to reduce the area and power dissipation without compromising system performance. Booth multiplication algorithm is more speed, when compared with the normal multiplication operation. And it is advanced method of the

Figure 7: Output Waveforms of Radix-8 Booth Encoded Modulo 2^n–1 Multiplier



radix-4 method. The percentage saving power product ranges from 2% to 35%. Further it can be implemented radix-16 booth algorithm, which can increase speed of the system to reduce the area and power dissipation.

REFERENCES

1. Bajard J C and Imbert L (2004), "A Full RNS Implementation of RSA", *IEEE Trans. Comput. – Brief Contributions*, Vol. 53, No. 6, pp. 769-774.
2. Bewick G W (1994), *Fast Multiplication: Algorithms and Implementation*.
3. Burch R, Najm F N, Yang P and Trick T N (1993), "A Monte Carlo Approach for Power Estimation", *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, Vol. 1, No. 1, pp. 63-71.
4. Cao B, Chang C H and Srikanthan T (2003), "An Efficient Reverse Converter for the 4-Moduli Set $\{2^{n}-1, 2^{n}, 2^{n}+1, 2^{2n}+1\}$ Based on the New Chinese Remainder Theorem", *IEEE Trans. Circuits Syst. I*, *Fundam. Theory Appl.*, Vol. 50, No. 10, pp. 1296-1303.
5. Cao B, Chang C H and Srikanthan T (2007), "A Residue-to-Binary Converter for a New Five-Moduli Set", *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, Vol. 54, No. 5, pp. 1041-1049.
6. Cao B, Low Y S, Chang C H and Srikanthan T (2010), "Performance Analysis of Different Special Moduli Sets for RNS-Based Inner Product Step Processor", Proc. 2010 Int. Conf. Green Circuits and Systems, pp. 236-241.
7. Cao B, Srikanthan T and Chang C H (2005), "Efficient Reverse Converters for Four-Moduli Sets $\{2^{n}-1, 2^{n}, 2^{n}+1, 2^{n+1}-1\}$ and $\{2^{n}-1, 2^{n}, 2^{n}+1, 2^{n-1}-1\}$", *IEE Proc. Comput., Dig. Techniq.*, Vol. 152, No. 5, pp. 687-696.
8. Cardarilli G C, Re A D, Nannarelli A and Re M (2005), "Low Power and Low Leakage Implementation of RNS FIR Filters", Proc. 39th Asilomar Conf. Signals, Systems and Computers, pp. 1620-1624.
9. Cherkauer B S and Friedman E G (1997), "A Hybrid Radix-4/Radix-8 Low Power Signed Multiplier Architecture", *IEEE Trans. Circuits Syst. II*, *Analog Digit. Signal Process*, Vol. 44, No. 8, pp. 656-659.
10. Dimitrakopoulos G and Paliouras V (2004), "A Novel Architecture and a Systematic Graph-Based Optimization Methodology for Modulo Multiplication", *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, Vol. 51, No. 2, pp. 354-370.
11. Dimitrakopoulos G, Nikolos D G, Vergos H T, Nikolos D and Efstathiou C (2005), "New Architectures for Modulo $2^{n}-1$ Adders", Proc. 12th IEEE Int. Conf. Electronics, Circuits and Systems, pp. 1-4.
12. Efstathiou C, Nikolos D and Kalamatianos J (1994), "Area-Time Efficient Modulo $2^{n}-1$ Adder Design", *IEEE Trans. Circuits Syst. II*, *Analog Digit. Signal Process*, Vol. 41, No. 7, pp. 463-467.
13. Efstathiou C, Vergos H T and Nikolos D (2004), "Modified Booth Modulo $2^{n}-

1$ Multiplier", *IEEE Trans. Comput.*, Vol. 53, No. 3, pp. 370-374.

14. Flynn M J and Oberman S F (2001), *Advanced Computer Arithmetic Design*, Wiley.
15. Hamada M, Ootaguro Y and Kuroda T (2001), "Utilizing Surplus Timing for Power Reduction", Proc. IEEE Conf. Custom Integrated Circuits, pp. 89-92.
16. Hiasat A A and Abdel-Aty-Zohdy H S (1998), "Residue-to-Binary Arithmetic Converter for the Moduli Set $\{2^{k}, 2^{k}-1, 2^{k-1}-1,\}$", *IEEE Trans. Circuits Syst. II*, *Analog Digit. Signal Process*, Vol. 45, No. 2, pp. 204-209.
17. Hiasat A A (2000), "New Efficient Structure for a Modular Multiplier for RNS", *IEEE Trans. Comput.*, Vol. 49, No. 2, pp. 170-174.
18. Juang T-B, Chiu C-C and Tsai M-Y (2010), "Improved Area-Efficient Weighted Modulo $2^{n}+1$ Adders Design with Simple Correction Schemes", *IEEE Trans. Circuits and Syst. II*, *Exp. Briefs*, Vol. 57, No. 3, pp. 198-202.
19. Kalampoukas L, Nikolos D, Efstathiou C, Vergos H T and Kalamatianos J (2000), "High-Speed Parallel-Prefix Modulo $2^{n}-1$ Adders", *IEEE Trans. Comput.*, Vol. 49, No. 7, pp. 673-680.
20. Koblitz N (1987), "Elliptic Curve Cryptosystems", *Mathemat. of Comput.*, Vol. 48, No. 177, pp. 203-209.
21. Kouretas I and Paliouras V (2009), "A Low-Complexity High-Radix RNS Multiplier", *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, Vol. 56, No. 11, pp. 2449-2462.
22. Lenstra A K and Verheul E R (2001), "Selecting Cryptographic Key Sizes", *J. Cryptol.*, Vol. 14, No. 4, pp. 255-293.
23. McIvor C, McLoone M and McCanny J V (2004), "Modified Montgomery Modular Multiplication and RSA Exponentiation Techniques", *IEE Proc. Comput. and Dig. Techniq.*, Vol. 151, No. 6, pp. 402-408.
24. McIvor C, McLoone M and McCanny J V (2006), "Hardware Elliptic Curve Cryptographic Processors Over $GF(p)$", *IEEE Trans. Circuits Syst. I, Reg. Papers*, Vol. 53, No. 9, pp. 1946-1957.
25. Miller V (1986), "Use of Elliptic Curves in Cryptography", Proc. Advances in Cryptology-CRYPTO'85, Lecture Notes in Computer Science, Vol. 218, pp. 417-426.
26. Mohan P V A and Premkumar A B (2007), "RNS-to-Binary Converters for Two Four-Moduli Sets $\{2^{n}-1, 2^{n}, 2^{n}+1, 2^{n+1}-1\}$ and $\{2^{n}-1, 2^{n}, 2^{n}+1, 2^{n+1}+1\}$", *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, Vol. 54, No. 6, pp. 1245-1254.
27. Mohan P V A (2002), "Residue Number Systems: Algorithms and Architectures", Kluwer
28. Mohan P V A (2007), "Reverse Converters for a New Moduli Set $\{2^{2n}-1, 2^{n}, 2^{2n}+1 \}$", *Circuits, Syst. Signal Process*, Vol. 26, No. 2, pp. 215-227.
29. Mohan P V A (2007), "RNS-to-Binary Converter for a New Three Moduli Set $\{2^{n+1}-1, 2^{n}, 2^{n}-1\}$", *IEEE Trans. Circuits Syst. II*, *Exp. Briefs*, Vol. 54, No. 9, pp. 775-779.

30. Molahosseini A S, Navi K, Dadkhah C, Kavehei O and Timarchi S (2010), "Efficient Reverse Converter Designs for the New 4-Moduli Sets $\{2^{n}-1, 2^{n}, 2^{n}+1, 2^{2n+1}-1\}$ and $\{2^{n}-1, 2^{n}+1, 2^{2n}, 2^{2n}+1\}$ Based on New CRTs", *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, Vol. 57, No. 4, pp. 823-835.

31. Muralidharan R and Chang C H (2010), "Fast Hard Multiple Generators for Radix-8 Booth Encoded Modulo $2^{n}-1$ and Modulo $2^{n}+1$ Multipliers", Proc. 2010 IEEE Int. Symp. Circuits and Systems, pp. 717-720.

32. Nagendra C, Irwin M J and Owens R M (1996), "Area-Time-Power Tradeoffs in Parallel Adders", *IEEE Trans. Circuits Syst. II*, *Analog Digit. Signal Process*, Vol. 43, No. 10, pp. 689-702.

33. National Institute of Standards and Technology [online] available at http://csrc.nist.gov/publications/PubsSPs.html

34. Nozaki H, Motoyama M, Shimbo A and Kawamura S (2001), "Implementation of RSA Algorithm Based on RNS Montgomery Multiplication", *Proc. Workshop on Cryptographic Hardware and Embedded Systems*, Vol. 2, pp. 364-376.

35. Patel R A, Benaissa M and Boussakta S (2007), "Fast Parallel-Prefix Architectures for Modulo $2^{n}-1$ Addition with a Single Representation of Zero", *IEEE Trans. Comput.*, Vol. 56, No. 11, pp. 1484-1492.

36. Piestrak S J (2001), "Design of Squarers Modulo a with Low-Level Pipelining", *IEEE Trans. Circuits Syst. II*, *Analog Digit. Signal Process*, Vol. 49, No. 1, pp. 31-41.

37. Pontarelli S, Cardarilli G C, Re M and Salsano A (2008), "Totally Fault Tolerant RNS Based FIR Filters", Proc. 14th IEEE Int. on-Line Testing Symp., pp.192-194.

38. Rabaey J M, Chandrakasan A and Nikolic B (2003), *Digital Integrated Circuits*, Prentice-Hall.

39. Ravi S, Raghunathan A, Kocher P and Hattangady S (2004), "Security in Embedded Systems: Design Challenges", *ACM Trans. Embedded Comput. Syst.*, Vol. 3, No. 3, pp. 461-491.

40. Rivest R, Shamir A and Adleman L (1978), "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Commun. ACM*, Vol. 21, No. 2, pp. 120-126.

41. Satzoda R K, Chang C H and Srikanthan T "Monte Carlo Statistical Analysis for Dynamic Power Simulation of RTL Designs Using Synopsys Power Compiler", Synopsys Users\' Group Conf.

42. Schinianakis D M, Fournaris A P, Michail H E, Kakarountas A P and Stouraitis T (2009), "An RNS Implementation of an Fp Elliptic Curve Point Multiplier", *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, Vol. 56, No. 6, pp. 1202-1213.

43. Soderstrand M A, Jenkins W K, Jullien G A and Taylor F J (1986), "Residue Number System Arithmetic: Modern Applications in Digital Signal Processing", IEEE Press.

44. Soudris D J, Paliouras V, Stouraitis T and Goutis C E (1997), "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.

45. Sousa L and Chaves R (2005), "A Universal Architecture for Designing Efficient Modulo $2^{n}+1$ Multipliers", *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, Vol. 52, No. 6, pp. 1166-1178.

46. Steiner I (2005), "A Fault-Tolerant Modulus Replication Complex FIR Filter", Proc. 16th IEEE Int. Conf. Application-Specific Systems, Architecture and Processors, pp. 387-392.

47. Stouraitis T and Paliouras V (2001), "Considering the Alternatives in Low-Power Design", *IEEE Circuits Devices Mag.*, Vol. 17, No. 4, pp. 22-29.

48. Stouraitis T, Kim S W and Skavantzos A (1993), "Full Adder-Based Arithmetic Units for Finite Integer Rings", *IEEE Trans. Circuits Syst. II*, *Analog Digit. Signal Process*, Vol. 40, No. 11, pp. 740-745.

49. Szabo N S and Tanaka R I (1967), *Residue Arithmetic and its Applications to Computer Technology*, McGraw-Hill.

50. Tomczak T (2008), "Fast Sign Detection for RNS $\{2^{n}-1, 2^{n}, 2^{n}+1\}$", *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, Vol. 55, No. 6, pp. 1502-1511.

51. TSMC 0.18 $\mu{\rm m}$ Process 1.8 Volt Sage-X™ Standard Cell Library Databook (2003), Artisan Components Inc.

52. Vergos H T and Efstathiou C (2008), "A Unifying Approach for Weighted and Diminished-1 Modulo $2^{n}+1$ Addition", *IEEE Trans. Circuits and Syst. II*, *Exp. Briefs*, Vol. 55, No. 10, pp. 1041-1045.

53. Wang Z, Jullien G A and Miller W C (1996), "An Algorithm for Multiplication Modulo $(2^{n}-1)$", Proc. 39th IEEE Midwest Symp. Circuits and Systems, pp. 1301-1304.

54. Zimmermann R (1999), "Efficient VLSI Implementation of Modulo $(2^{n}\pm1)$ Addition and Multiplication", Proc. 14th IEEE Symp. Computer Arithmetic, pp. 158-167.