The public key cryptosystems health and social care essay

Abstract — The majority of the currently established Public-Key Cryptosystems (RSA, Die-Hellman, Digital Signature Algorithm (DSA), Elliptic Curves (ECC), etc., require modular multiplication in finite fields as their core operation which accounts for up to 99% of the time spent for encryption and decryption. In order to improve the performance of the overall cryptosystem, it is therefore crucial to optimize modular multiplication. Moreover Modular arithmetic operations such as inversion, multiplication, addition and exponentiation are used in several cryptography applications. A special moduli set of forms {2n-1, 2n, 2n +1} are preferred over the generic modulo 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. The modulo 2n-1 multiplier is usually the noncritical data path among all modulo multipliers in such high-DR RNS multiplier. With this precept, a family of radix-8 Booth encoded modulo 2n-1 multipliers, with delay adaptable to the RNS multiplier delay, is proposed in this paper. A CSA tree with end-around carry addition for accumulation of redundant partial-products. Modulo multiplier design has been presented using Sklansky & Kogge-Stone parallel-prefix structures. Index Terms — Public Key Cryptographic (PKC), Booth algorithm, modulo arithmetic, residue number system (RNS), Prefix structuresI. INTRODUCTIONPublic-key cryptography plays an important role in digital communication and storage systems. Processing public-key cryptosystems requires huge amount of computation, and, there is therefore, a great demand for developing dedicated hardware to speed up the computations. Speeding up the computation using specialized hardware enables the use of larger keys in public-key cryptosystems. 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. To create the private and public keys we have to deal with the modulo arithmetic operations like addition, subtraction and multiplication. When a number gets bigger, the arithmetic operations get more complex. Key sizes in the range of 512~1024 bits and 160~512 bits are typical in RSA and ECC, respectively [4]–[6]. Hence, the long carry propagation of large integer multiplication is the bottle-neck in hardware implementation of PKC. Hence, the arithmetic operations will be simple and can be performed on each residue independently giving the provision of parallel operations i. e. operations on all the residues can be done at the same time. 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 [7]–[9]. Hence, the modulo arithmetic operations will be simple and can be performed on each residue independently giving the provision of parallel operations i. e. arithmetic operations on all the residues can be done at the same time . RNS has also been successfully employed to design fault tolerant digital circuits . The Residue Number System (RNS) is a non-weighted number system that can map large numbers to smaller residues, without any need for carry propagations . Its most important property is that additions, subtractions, and multiplications are inherently carry-free. These arithmetic operations can be performed on residue digits concurrently and independently. Thus, using residue arithmetic, would in principle, increase the speed of computations RNS has shown high efficiency in realizing special purpose applications such as digital filters , image processing, RSA cryptography and specific applications for which only additions, subtractions and multiplications are used and the number dynamic range is specific. Special moduli sets have been used extensively to reduce the hardware complexity in the implementation of converters and arithmetic operations. Among which the triple moduli set {2n+1, 2n, 2n-1} have some benefits. Since the operation of multiplication is of major importance for almost all kinds of processors, efficient implementation of multiplication using modulo 2n-1 is important for the application of RNS. A RNS is defined by a set of pair-wise co-prime moduli,{L1, L2,.... LN} such that any integer X within the dynamic range (DR) i. e., is represented as a N-tuple{x1, , x2.... xN}, where xi is the residue of X modulo [4]-[6]. RNS multipliers based on generic moduli have been reported in [1]-[6]. However, special moduli of forms 2n or 2n ±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, {2n-1, 2n, 2n+1} which however has a DR of only 3n 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. It has been shown that the speed of RNS processor is increasingly dominated by the residue arithmetic operation rather than the one-time forward or reverse conversion. The paper is organized as follows. Section II describes the radix8 Booth encoding algorithm for modulo multiplication. Section III consists of proposed design for radix-8 booth enco- ded modulo multiplier has been presented. In Section IV, a brief introduction of modulo 2n-1 additon is given and proposed prefix structures are highlighted. The paper is concluded in Section V. II. RADIX-8 BOOTH ENCODED MULTIPLICATIONBooth 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. 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 2n-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| 2n-1 is to be generated by an n -bit end-around-carry addition of X and 2X. The modified Booth’s algorithm (radix-8 encoding) starts by appending a zero to the right of x0 (multiplier LSB). Quartets are taken beginning at position x–1 and continuing to the MSB with one bit overlapping between adjacent quartets. If the number of bits in X (excluding x-1) is odd, the sign (MSB) is extended one position to ensure that the last quartet contains 4 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. Letandreprese-nt the multiplicand and the multiplier of the modulo 2n-1 multiplier, respectively. The radix-8 Booth encodingFig. 1. Modulo (2n-1) multiplier architecturecan be viewed as a digit set conversion of four consecutive overlapping multiplier bits y3i+2y3i= 1y3i(y3i-1) to a signed digit, di, di. The digit set conversion is formally expressed as………………

(1)whereTable I summarizes the modulo-reduced multiples of for all possible values of the radix-8 Booth encoded multiplier digit, where CLS(X, j) denotes a circular-left-shift of Multiplicand by ‘ j’ bit positions. From Table 3, the necessary modulo-reduced partial products except ±3X can be generated by circular-left-shift operation and/or bit-wise complementation of the multiplicand, X. The generation of hard multiple ±3X i. e. (X+2X) requires a large word-length adder which increases the critical path delay of the multiplier significantly. III. PROPOSED RADIX-8 BOOTH ENCODEDMODULO MULTIPLIER DESIGNFig. 2 illustrates the computation of |+3X | 2n-1 by an n - bit end-around-carry addition of X2n-1 and 2X2n-1using RCAs for n= 8. To ensure that the radix-8 Booth encoded modulo 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 Fig. 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 end-Fig. 2. Generation of |+3X | 2n-1using two n-bit RCAs. 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 2n-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 [10]. GENERATION OF PARTIALLY-REDUNDANT HARD MULTIPLELet | X2n-1 and 2X2n-1be added by a group of M=(n/k) k-bit RCAs such that there is no carry propagation between the adders. Fig. 3 shows this addition for n= 8 and k= 4. Fig. 3. Generation of partially-redundant |+3X 2n-1 using k-bit RCAs

-

Fig. 4. Generation of partially-redundant | B+3X 2n-1|. where the sum and carry-out bits from the RCA block are represented as respectively. In Fig. 3, the carry-out of RCA 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 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. Fig. 4 illustrates how these bits in the sum and the carry outputs of RCA 0 and RCA 1 are modified. Table I : Modulo-Reduced Multiples for theRadix-8 Booth EncodingQuartet valueSigned-digitvalue, di| di . X| 2n-1000000…00001+1X0010+1X0011+2CLS(X, 1)0100+2CLS(X, 1)0101+3| +3X| 2n-10110+3| +3X| 2n-10111+4CLS(X, 2)1000-4CLS(, 2)1001-3| -3X| 2n-11010-3| -3X| 2n-11011-2CLS(, 1)1100-2CLS(, 1)1101-11110-11111-01…1The 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. 5 shows the biased simple multiples, | B+0| 2n-1, |

B+X| 2n-1, | B+2X| 2n-1 , | B+4X| 2n-1 represented in partially redundant form for n= 8. From Fig. 5, 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. Fig. 5. Generation of partially-redundant simplemultiples. Fig. 6. Modulo-reduced partial products and CCfor | X . Y| 28-1Fig. 6 illustrates the partial product matrix of | X . Y| 28-1 with partial products in partially-redundant representation. Each PPi consists of an n-bit vector, ppi7 ,…….. ppi1, ppi0 and a vector of n/k= 2, redundant carry bits qi1, qi0. Since qi0 and qi1 are the carry-out bits of the RCAs, they are displaced by k-bit positions for a given PPi. The bits, qij is displaced circularly to the left of q(i-1)j by 3 bits, i. e., q20 and q21 are displaced circularly to the left of q10 and q11 by 3 bits, respectively q10 and q11 are in turn displaced to the left of q00 and q01 by 3 bits, respectively. The last partial product in Fig. 6 is the Compensation Constant (CC) for the bias introduced in the partially- redundant representation. The generation of qij the modulo-reduced partial products, PP0, PP1, and PP2, in a partially-redundant representation using Booth Encoder (BE) and Booth Selector (BS) blocks are illustrated in Fig. 7. The BE block produces a signed one-hot encoded digit from adjacent overlapping multiplier bits as illustrated in Fig. 8(a). The signed one-hot encoded digit is then used to select the correct multiple to generate PPi. A bit-slice of the radix-8 BS for the partial product bit, PPij is shown in Fig. 8 (b). As the bit positions of do not overlap, as shown in Fig. 6, they can be merged into a single partial product for accumulation. The merged partial products, PPi and theFig. 7. Modulo-reduced partial product generation

•

Fig. 8.(a) Bit-slice of Booth Encoder (BE). 8. (b). Bit-slice of Booth Selector (BS). constant CC are accumulated using a CSA tree with end-around-carry addition at each CSA level and a final two-operand modulo 2n-1 adder as shown in Fig 9

•

Fig. 9. Modulo-reduced partial product accumulation. IV. MODULO 2N-1 ADDITIONA. PARALLEL PREFIX ADDITONThe two-operand modulo adder implemented in final stage is a parallel-prefix structure with an additional prefix level for the end-around-carry addition. Modulo 2n-1(Mersenne-numbers) addition is one of the most common operations that has been put to hardware implementations because of its circuit efficiency. The need for a Parallel Prefix adder is that it is primarily fast when compared with ripple carry adders. Parallel Prefix adders (PPA) are family of adders derived from the commonly known carry look ahead adders. These adders are best suited for adders with wider word. The common method of doing modulo addition is to utilize the carry-lookahead equations and use carry-generates/propagates signals recursively to get the final result in just one step. A general prefix addition process structure is shown in fig 10. In this paper, modified parallel-prefix structure is employed to achieve the fastest architecture. Many techniques can help modulo addition. The schemes we present here is Sklansky Parallel Prefix Structure with end-around carry addition and Kogge-Stone prefix structure. [11] and [12]A parallel prefix adder can be seen as a 3-stage process namely Pre-computation, Prefix and Post-computationFig. 10. Parallel Prefix Addition Process StepsPRE-COMPUTATION: In pre-computation stage, each bit computes its carry generate (g)/propagate (p) signals and a temporary sum as below. These two signals are said to describe how the Carry-out signal will be handled. gi = ai . bipi = ai ^ bici +1 = gi + pi . ciPREFIX: In the prefix stage, the group carry generate/propagate signals are computed to form the carry chain and provide the carry-in for the adder below. Various signal graphs/architectures can be used to calculate the carry-outs for the final sum. A few of them are as follows. POST-COMPUTATION: In the post-computation stage, the sum and carry-outare finally produced. The carry-out can be omitted if only a sum needs to be producedSi = Pi ^ CiB. PROPOSED PARALLEL PREFIX STRUCTURESThe square (□) and diamond (◊) nodes form the Pre and Postprocessing stages, respectively. The black nodes (●) evaluate the prefix operator andWhite nodes (○) pass the signals unchanged to the next prefix levelFig. Proposed End-around carry ParallelPrefix adder structurewhere ai and bi are the operand input signals, gi and pi the generate and propagate, Ci the carry, and Si the sum output signals at bit position i. C0 and Cn correspond to the carry-in and carry-out Cout , respectivelyFig . Sklansky prefix structureThe Sklansky prefix tree has Minimal depth and High fan-out nodes, where as Kogge-Stone adder results in low depth, high node count (implies more area) and minimal fan-out of 1 at each node (implies faster performance)Fig . Kogge-Stone prefix structure

V. RESULT

Table II . Devize utilization summary

Device Utilization(w. r. t3s500efg-320)Modulo multiplier with Skalansky adderModulo Multiplier with Kogge-Stone adderNumber of

Slices10697Number of 4 input LUTs186167Number of IOs2526Number of bonded IOBs2426Max delay27. 404ns25. 458ns

V1. CONCLUSION

A new approach for low-power modulo 2n-1 multiplication with high speed parallel prefix adders where the delay of the proposed multiplier is controlled by the word-length of the small parallel RCAs was proposed. These RCAs are used to compute the requisite hard multiple of the radix-8 Booth encoded multiplication in a partially-redundant form. Similar to the binary multiplier, the generation of the partial products is accomplished by Radix-8 modified booth algorithm. The CSA tree is applied to reduce the speed for compression of column size from N to two. The resultant propagated carry and sum from CSA adder is fed to parallel-prefix adder to achieve modulo multiplier output. The simulation results of proposed modulo multiplier have been verified using Skalanksy & Kogge-Stone prefix adders. The design has been implemented using verilog HDL and synthesized using Xilinx ISE w. r. t Spartan3E fpga device. The proposed multiplier modulo (2n- 1) shows an extremely regular structure and is very suitable for VLSI implementation. VII.

REFERENCES[1] R. Rivest, A. Shamir, and L. Adleman, " A method for obtaining digital signatures and public key cryptosystems," Commun. ACM, vol. 21, no. 2, pp. 20-126, Feb. 1978.[2] V. Miller, " Use of elliptic curves in cryptography," inProc. Advances in Cryptology-CRYPTO’85, Lecture Notes in Computer Science, 1986, vol. 218, pp. 417-426.[3] N. Koblitz, " Elliptic curve cryptosystems," Mathemat. of Comput., vol. 48, no. 177, pp. 203–209, Jan. 1987[4] National Institute of Standards and Technology [Online]. Available: http://csrc. nist. gov/publications/PubsSPs. html[5] A. K. Lenstra and E. R.

Verheul, " Selectingcryptographic 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 exponent- iation techniques," IEEE Proc Comput. and Dig. techn- ique vol. 151, no. 6, pp. 402–408, Nov. 2004.[7] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas, and T. Stouraitis, " An RNS implementati- on an FP Elliptic curve point multiplier," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 6, pp. 1202–1213, Jun 2009.[8] 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.[9] H. Nozaki, M. Motoyama, A. Shimbo, and S. Awamura, " Implementation of RSA algorithm based on RNS Mont- go mery multiplication," in Proc. Workshop on Crypto-graphic Hardware and EmbeddedSystems, Paris, France, May 2001, pp. 364–376.[10] G. W. Bewick, " Fast multiplication: Algorithms and implementation," Ph. D. dissertation, Stanford Univ., Stanford, CA, 1994[11]. R. Zimmermann, " Efficient VLSI implementation of Modulo addition and 2n± 1multiplication," inProc. 14th IEEE Symp. Computer Arithmetic, Adelaide, Australia Apr. 1999, pp. 158–167.[12]. Parallel-prefix structures for binary and modulo{2n-1, 2n, 2n +1} adders - Jun Chen, 2004