International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 02 | Feb 2019 www.irjet.net p-ISSN: 2395-0072

Radix 8 Booth Encoded Interleaved Modular Multiplication

Jhashank Gandhi[1], R. Sakthivel[2]

[1]M. Tech, VLSI Design, School of Electronics Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India
[2]Associate Professor, School of Electronics Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India

---

Abstract - *In most public key cryptosystems, such as Rivest-Shamir-Adleman (RSA), Elliptic curve cryptography (ECC), Digital Signature Algorithm (DSA), Diffie-Hellman (DH) key exchange, modular multiplication is the critical operation. To provide sufficient security strength, large numbers (order of 1024 or 2048 bits) are used. Usage of such large numbers reduces the speed of the whole cryptographic system. This paper introduces an efficient architecture for calculating modular multiplication of two large numbers A and B modulo a given modulus P. Radix–8 booth encoding techniques are used to modify the existing bit serial interleaved modular multiplication algorithm. The modified radix-8 version reduces the delay (number of clock cycles) required for performing one modular multiplication but at the cost of increased resources (hardware). The proposed architecture is implemented in Verilog HDL and simulated using Model Sim and Xilinx ISE 8.1i simulator. It is synthesized using Spartan3E FPGA and a comparison is made between 8 bits and 16 bits numbers for delay and hardware utilization.*

Key Words*:* Modular multiplication, radix-8 interleaved multiplication, booth encoding, cryptosystems, etc.

1. INTRODUCTION

Cryptography is the study and design of ways to protect valuable information over an insecure channel against attackers. Efficient cryptographic protocols are required to fulfill the growing demands for security in many different systems, such as large servers or small hand-held devices. Many restrictions such as computation time, silicon area, and power consumption must be taken care of by the designer of the hardware accelerators of cryptographic protocols.

Modular multiplication is a very time-consuming process that is extensively used in various cryptographic systems such as ECC, RSA etc. Elliptic curve cryptographic systems use much smaller key sizes as compared to RSA, which results in better bandwidth utilization, fewer storage requirements, and lower power consumption. Due to the computational complexity of ECC, there is a requirement of a dedicated hardware implementation to meet the time limitations of many real-time applications.

A simple approach to do modular multiplication ($x \times y \bmod p$) is to first compute ($x \times y$) then perform reduction under prime number p. Various algorithms have been proposed to avoid this reduction step as this step involves division by a prime number p which is a very costly operation to be performed in software and hardware platforms, as it takes a lot of time and more area.

To speed up modular multiplication technique many designs have been proposed. Two most used designs are Interleaved Modular Multiplication [1] and Montgomery Multiplication [2]. Various designs based on modified interleaved multiplication algorithm have been reported [3] [4] [5].

In this paper, the proposed modular multiplier is based on interleaved multiplication technique combined with booth encoding. Interleaved multiplication algorithm is proposed by Blakely [2]. The interleaved multiplication technique is based on iterative addition and reduction of partial products.

Partial products accumulation and intermediate results reduction are integrated in such a way so that the final division step is eliminated. Basic idea is to reduce the intermediate results below the modulus value in each iteration so that there is no need to perform the division operation. The Algorithm starts with traversing the multiplier from its most-significant-bit (MSB) to least-significant-bit (LSB).

Section II introduces the Radix-8 interleaved modular multiplication algorithm with booth encoding. Section III presents an architecture of the algorithm. Section IV provides the implementation results. Finally, the document ends with the conclusion in section V.

Partial products accumulation and intermediate results reduction are integrated in such a way so that the final division step is eliminated. Basic idea is to reduce the intermediate results below the modulus value in each iteration so that there is no need to perform the division operation. The Algorithm starts with traversing the multiplier from its most-significant-bit (MSB) to least-significant-bit (LSB).

2. RADIX-8 BIT INTERLEAVED MODULAR MULTIPLICATION (R8BIM)

In classical modular multiplication approach first, the product ($x \times y$) is computed and then the result is reduced by dividing the product with the given modulus p. In interleaved modular multiplication, multiplication and reduction overlap. At each step, the intermediate results are

reduced to the range {0 p-1} before resuming to multiplication process.

In Radix-2 bit interleaved modular multiplication (R2BIM), the total number of loop iterations (clock cycles) is same as the number of bits in the multiplier. In radix-4 bit interleaved modular multiplication (R4BIM), the total number of loop iterations (clock cycles) is reduced from n to [n/2], where n is the number of bits of the multiplier. In R8BIM, the number of loop iterations (clock cycles) is further reduced to [n/3]. Therefore, with increasing radix clock cycles are getting reduced but at the cost of increased hardware utilization.

The radix-8 BE technique is given in Table 1 where every four bits are encoded as {0, ±1, ±2, ±3, ±4}. The effective number of bits processed in R8BIM is three, but we group four bits where one-bit acts as an overlapping bit which tells about the previous group. As in ECC, only positive numbers are used, so adding zeroes to the left of MSB is sufficient to complete the four bits group. The rule for adding zeroes to the left of MSB is as follows:

- If n mod 3 = 0 add three zeroes to the left of MSB
- If n mod 3 = 2 add two zeroes to the left of MSB
- If n mod 3 = 1 add single zero to left of MSB

One zero is added to the right of LSB bit to complete the first group, where that zero acts as an overlapping bit for the first group.

Table-1: Radix-8 Booth Encoding

Y_i	Y_{i-1}	Y_{i-2}	Y_{i-3}	Encoded Value
0	0	0	0	0
0	0	0	1	+1
0	0	1	0	+1
0	0	1	1	+2
0	1	0	0	+2
0	1	0	1	+3
0	1	1	0	+3
0	1	1	1	+4
1	0	0	0	-4
1	0	0	1	-3
1	0	1	0	-3
1	0	1	1	-2
1	1	0	0	-2
1	1	0	1	-1
1	1	1	0	-1
1	1	1	1	0

The modified R8BIM algorithm based on radix-8 and BE techniques is given below:

Algorithm-1 Radix-8 bit interleaved multiplication (R8BIM)

Input: x, y, p n bit numbers, $0 \le x, y \le p$

Output: $z = (x \times y) \bmod p$

1. *$z \leftarrow 0$,*
2. *$R1 \leftarrow 2x \bmod p$,*
3. $R2 \leftarrow 3x \bmod p$,
4. $R3 \leftarrow 4x \bmod p$,

N =
- n+3 , if n mod 3 = 0 append three zeroes to the left of MSB
- n+2 , if n mod 3 = 1 append two zeroes to the left of MSB
- n+1 , if n mod 3 = 2 append single zero to the left of MSB

5. $N \leftarrow N + 1$ append single 0 to the right of LSB of y
6. for($i = N, i \ge 3, i = i-3$) do
7. $z = 8z \bmod p$
8. switch ($y_{(i:i-3)}$) do
9. when {0000|1111} ➔ $z \leftarrow z$
10. when {0001|0010|1101|1110} ➔ $z \leftarrow z \pm x \bmod p$
11. when {0011|0100|1011|1100}➔ $z \leftarrow z \pm R_1 \bmod p$
12. when {0101|0110|1001|1010} ➔ $z = z \pm R_2 \bmod p$
13. else ➔ $z \pm R_3 \bmod p$
14. endsw
15. end
16. return z

The R8BIM algorithm takes [n/3] + 4 iterations to perform an n-bit modular multiplication. It consists of several steps that are as follows:

1. *In Step 1 of algorithm 1, 2x mod p, 3x mod p and 4x mod p values are calculated. These are pre-computed values done by a pre-computation process.*

2. *Step 4 of algorithm 1 is performed iteratively throughout the loop iterates. In this step value of accumulator, z is updated and is equal to three-bit left shift modulo p i.e, 8z mod p.*

3. *In steps 7-10, respective partial products are calculated according to current four bits of the multiplier y and are modularly added or subtracted from the accumulator z i.e, z = z ±{x, R_1, R_2, R_3, R_4}.*

3. HARDWARE ARCHITECTURE

The Hardware architecture of R8BIM is shown in Figure 1. The R8BIM architecture is composed of four major blocks: three doubling blocks and one add/sub block cascaded in series. As they are cascaded in series, it implies that each block output is input to the next block. Apart from these major blocks, there are also some minor blocks like BE, a temporary register, a multiplexer-based look-up table, and four n-bit data registers R1, R2, R3, and z.

Execution of the algorithm is done in two phases. The phases are as follows:

Phase 1:

1) 2x mod p, 3x mod p, and 4x mod p values are computed and assigned to R1, R2, and R3 respectively in the first clock cycle.

2) 2x mod p and 4x mod p is calculated by instantiating doubler block whereas 3x mod p is calculated using add/sub block. For 3x mod p, the inputs to add/sub block are x, 2x mod p, and cin. cin is set to zero to perform the addition operation.

Phase 2:

1) Iterations of the loop is controlled by a counter which is decremented by three after each iteration.

2) 8z mod p is performed by three doubler blocks cascaded in series. The register z is initially loaded with zero and after each iteration, it is three bits left shifted and reduced modulo p. Then a respective partial product is added or subtracted.

3) Steps 5-10 are executed by M block and add/sub block. Block M is a look-up table that selects the appropriate partial product to be modularly added or subtracted from z in the add/sub block.

4) BE block is the block which decides whether to add or subtract the partial product. It takes the four bits of a temporary register in each iteration and calculates cin accordingly. If cin is 1 modular subtraction is performed and if cin is 0 modular addition is performed.

A doubler block contains one adder and one multiplexer whereas an add/sub block contains two adders and one multiplexer. Therefore the critical path delay is:-

$$T_{\text{critical path}} = 5t_{adder} + 6t_{mux} \quad (1)$$

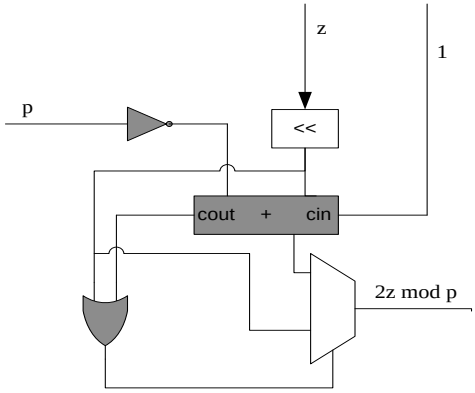

Fig – 1: Doubler block (2z mod p)

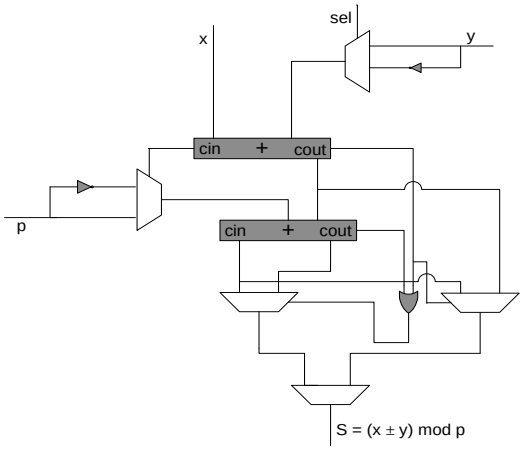

Fig-2: Add/Sub Block

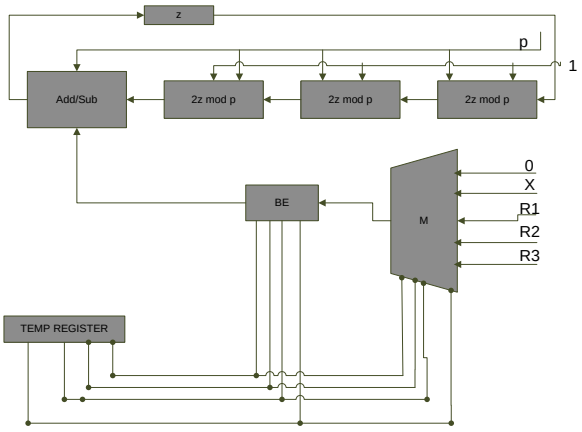

Fig-3: Hardware Architecture of R8BIM

4. IMPLEMENTATION AND RESULTS

The radix-8 multiplier is coded in Verilog HDL and synthesized targeting Spartan 3E FPGA. The Xilinx ISE 8.1i is used for synthesis. For behavioral simulation ModelSim is used. 8 bits implementation of R8BIM occupies 208 slices and 363 LUTs whereas, 16 bits implementation occupies 375 slices and 680 LUTs as shown in Table 2. For 8 bits implementation, R8BIM architecture computes one modular multiplication in 16.651ns whereas for 16 bits implementation it computes one modular multiplication in 19.579ns. Furthermore, the AT/b (area × time per bit) value is 0.432 for 8 bits whereas it is 0.458 for 16 bits. Different values of x, y, and p were taken, and output z was verified. Output came after four clock pulses for n = 8 bits and after nine clock cycles for n = 16 bits. Figure 4 shows the Finite State Machine of R8BIM Algorithm.

a) Inputs: x =20 (00010100), y = 28 (00011100),

   p = 29 (00011101)

   Output: z = 9 (00001001) **(For 8 bits)**

b) Inputs: x = 12548 (0011000100000100), y=12540(0011000011111100),

   p = 12553 (0011000100001001)

   Output: z = 65 (0000000001000001) **(For 16 bits)**

Table-2: R8BIM implementation comparison of different bits

No. of bits	Slices	Flip Flops	LUTs	AT/b	Delay (ns)
8 bits	208	43	363	0.432	16.651
16bits	375	49	680	0.458	19.579

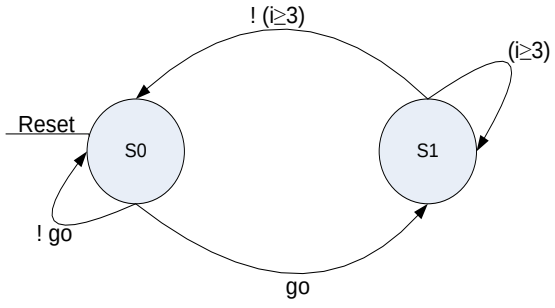

Figure-4: FSM of R8BIM

5. CONCLUSION AND FUTURE WORK

This paper introduces a comparison for radix-8 booth encoded modular multiplication with different bits. R8BIM takes [n/3] + 4 clock cycles to compute n-bit modular multiplication. For 8 bits multiplication, it takes 6 clock cycles whereas for 16 bits multiplication it takes 9 clock cycles. Modular interleaved multiplication will be implemented using a radix-16 booth encoding technique which will further reduce the number of the partial product (clock cycles) to [n/4] and hence will be faster but will utilize the additional hardware.

REFERENCES

[1] P.L. Montgomery, "Modular Multiplication without trial division, "Mathematics of Computation, vol. 44, no. 170, pp. 519-521, 1985.G.

[2] G. Blakely, "A computer algorithm for calculating the product ab modulo m," IEEE Transactions on Computers, vol. 32, no. 5, pp. 497–500, 1983

[3] K. Javeed and X. Wang, "Radix-4 and radix-8 booth encoded interleaved modular multipliers over general Fp," in Field Programmable Logic and Applications (FPL), 2014 24th International Conference on, Sept 2014, pp. 1–6.

[4] J. Wolkerstorfer, "Dual-field arithmetic unit for GF(p) and GF(2m)," in Cryptographic Hardware and Embedded Systems-CHES 2002. Springer, 2003, pp. 500–514.

[5] D. Narh Amanor, C. Paar, J. Pelzl, V. Bunimov, and M. Schimmler, "Efficient hardware architectures for modular multiplication on FPGAs," in Field Programmable Logic and Applications, 2005. International Conference on, Aug 2005, pp. 539–542.