Available online at www.uofk.edu
Proceedings Vol. 1 pp. 171-176
Annual Conference of Postgraduate Studies and Scientific Research
17-20 February 2012-10-17 Friendship
Hall, Khartoum

A New MSB Modular multiplication Algorithm and its Implementation using 4-2 Compressor

Kauther M. Amer Ahmad S.Ashor

College of Communication Engineering, Academy of Graduate Studies,
Tripoli, Libya

Abstract: Currently, enhancing the performance of modular multiplication is very important for high performance microprocessors. Multiplication is inherently a slow operation as a large number of partial products. In this paper, Implementation of MSB modular multiplication using 4-2 compressor using the last two most significant bits is introduced. Using the two most significant bits (MSB's) for reduction in this technique, we will avoid the use of the carrier so, no carry store. However, we don't need to use the feedback which used in Montgomery Multiplication [13]. The total power-efficiency (power-delay-product) is reduced using low-power low-voltage 4-2 compressors. The circuits implemented using Matlab simulation.

Keywords: *MSB modular multiplier; modular multiplication algorithm; 4-2 compressor*.

1. INTRODUCTION

Recently computer security is one of the most important issues which have a wide attention from scholarships. Therefore computer cryptography becomes more and more important.
Modular multiplication is a widely used operation in cryptography. Several well-known applications, such as RSA algorithm, Digital Signature and elliptic curve cryptography, all use modular multiplication and modular exponentiation.
Multipliers implemented in hardware are generally faster than the software solutions. A small speed up for a single multiplication operation can lead to a substantial speed up for the cryptographic applications.
Some of the applications required the multiplication to be achieved at a faster rate while others tend to exploit less hardware and therefore more modest speed. In order to meet the demand of these applications, various design strategies have been exploited.
Many algorithms for modular multiplication have been proposed in the past [3]–[11].
In this work, we first present Modular Multiplication Algorithm using the two MSB's for reduction; we introduce an example to illustrate the modular multiplication algorithm.
The next section introduces the Modular multiplier using 4-2 compressor and then the Simulation Using MatLab is illustrated.
The conventional CMOS 4-2 compressor is replaced by the low-power 4-2 compressors presented in [2], In the next section; this compressor circuit contains 2 delays XOR which was 4 delays XOR in the previous compressor. The paper is ended by Conclusion and then References.

2. Modular Multiplication Algorithms

In RSA, the public encryption key is a pair of positive integers (E ,N), and the private decryption key is another pair of positive integers (D,N). To encrypt a message using the key (E,N), we first partition the message (a string of bits) into a sequence of blocks, and consider each block as an integer between 0 and N - 1. Then, we encrypt the message by raising each block M to the E th power modulo N , i.e., $C=M^E \bmod N$ for each message block M . Similarly, to decrypt the ciphertext using the key (D,N), we raise each ciphertext block to the power D modulo N, i.e., $M=C^D \bmod N$ for each ciphertext block C . Exponentiation is performed by repeated (iterated) squaring and multiplication operations. Let the binary representation of the exponent E be en-1 en-2 …..e1 e0 , then

$$M^E = M^{2^{n-1}\cdot e_{n-1}} \cdots M^{2^2 \cdot e_2} \cdot M^{2 \cdot e_1} \cdot M^{e_0}.$$

A simple way to perform modular exponentiation is to repeat the modular squaring and modular multiplication operations.

2.1 The New MSB Modular Multiplication Algorithm

The following is the notation used throughout this algorithm:

- M – modulus for modular multiplication.
- A – multiplier operand for modular multiplication.
- ai – a single bit of A at position i .
- B – multiplicand operand for modular multiplication.
- N – number of bits in the operands, operand's precision.
- Si – partial product in the multiplication process, final result of modular multiplication.

The algorithm require that M is prime, and

$$2^{(n-1)} < M > 2^{(n)}$$

Table (1). Radix-2 MSB Modular Multiplication Algorithm

Step	The algorithm
1:	S=0
2:	FOR i=n-1 to 0
3:	S=2S
4:	S=S+ a*B +S_n* ((complement M) +1) +$s_{(n-1)}$* 2*((complement M) +1)
	END FOR
5:	IF S>= M THEN S=S-M
	END IF

The main advantage of this algorithm is that we use the carrier in the reduction process so we don't need the feed back in our implementation.

Radix–2 denotes that the multiplier A is scanned one bit in each iteration of the loop, called computational loop or computational cycle.

In each computational loop the multiplicand B is added to the partial product S depending on two bits on the multiplier A. The most significant bits (MSB's) ***S(n) & S(n-1)*** of the previous summation of S are shifted away in Step 3. These bits are used for reduction, the reduction is based on the table below:

Table (2). reduction cases

S(n)	**S(n-1)**	**Operation**
0	0	(no add)
0	1	Add (complement M) +1 to S
1	0	Add Right Shift of ((complement M) +1) to S
1	1	Add (complement M) +1 to S and Add Right Shift of ((complement M) +1) to S

In modular arithmetic adding\subtracting the modulus to\from a number does not affect the the number's value.

After the last iteration of the loop S holds the multiplication result, the final result must be a number less then the modulus. Therefore, in Step 5., called *final reduction step*, of this algorithm S is compared to M and is adjusted if needed.

2.2 Example to illustrate the algorithm

Assume that we have two binary numbers A(a3,a2,a1,a0) and B (b3,b2,b1,b0) ; A=11=[1011] and B=12=[1100] and the modular M=13=[1101]

We first calculate the complement of M which is named M', and then add M' to 1

M'+1=[0011]

To calculate 11*12 mod 13

```
B       1 1 0 0
A       1 0 1 1

S     0 0 0 0 0

a3*B    1 1 0 0
M'+1    0 0 0 0          (no add)

S     0 1 1 0 0
a2*B      0 0 0 0
M'+1      0 0 1 1        Add (complement M) +1

S     0 1 0 1 1
a1*B        1 1 0 0
M'+1        0 0 1 1      Add (complement M) +1

S     1 0 1 0 1
a0*B          1 1 0 0
M'+1            0 0 1 1 0      Add Right Shift of
((complement M) +1)

S           1 1 1 0 0    S > M
S – M         1 1 0 1

S           1 1 1 1      S > M
S – M         1 1 0 1

S             0 0 1 0    S < M    The final result
```

As discussed above we started with the MSB of A whish a3 and multiply it by B, in the first iteration the two most significant bits (MSB's) of S (s4 &s3) in the previous iteration are (0 0) then in this case (no add) which illustrated in Table 2.
In the second iteration S was (0 1) that means we will add S (after shift it in 2S) to the product a2*B and Add (complement M) +1. The same procedure is repeated in the other iterations.
The last result of S > M so we need to subtract S – M and repeat this step until S<M to have the final result.

3. The MSB Modular multiplier:

The proposed implementation of the modulo multiplier consists of four stages, for (n=4), each stage contains 4 compressors, and 5 FA.

For reduction we use one half adder and one XOR to calculate the n and n+1 MSB bits and use these bits for reduction.
We use a 4-2 compressor which consists of five inputs and three outputs and which implemented with two stages of full-adders (FA) connected in series as shown in Fig.1.

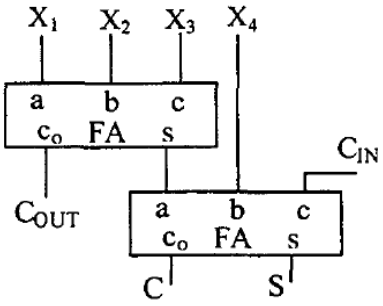


Fig. (1). 4-2 Compressor composed of two FAs.

The circuit of the MSB Modular multiplier is in the fig.2.

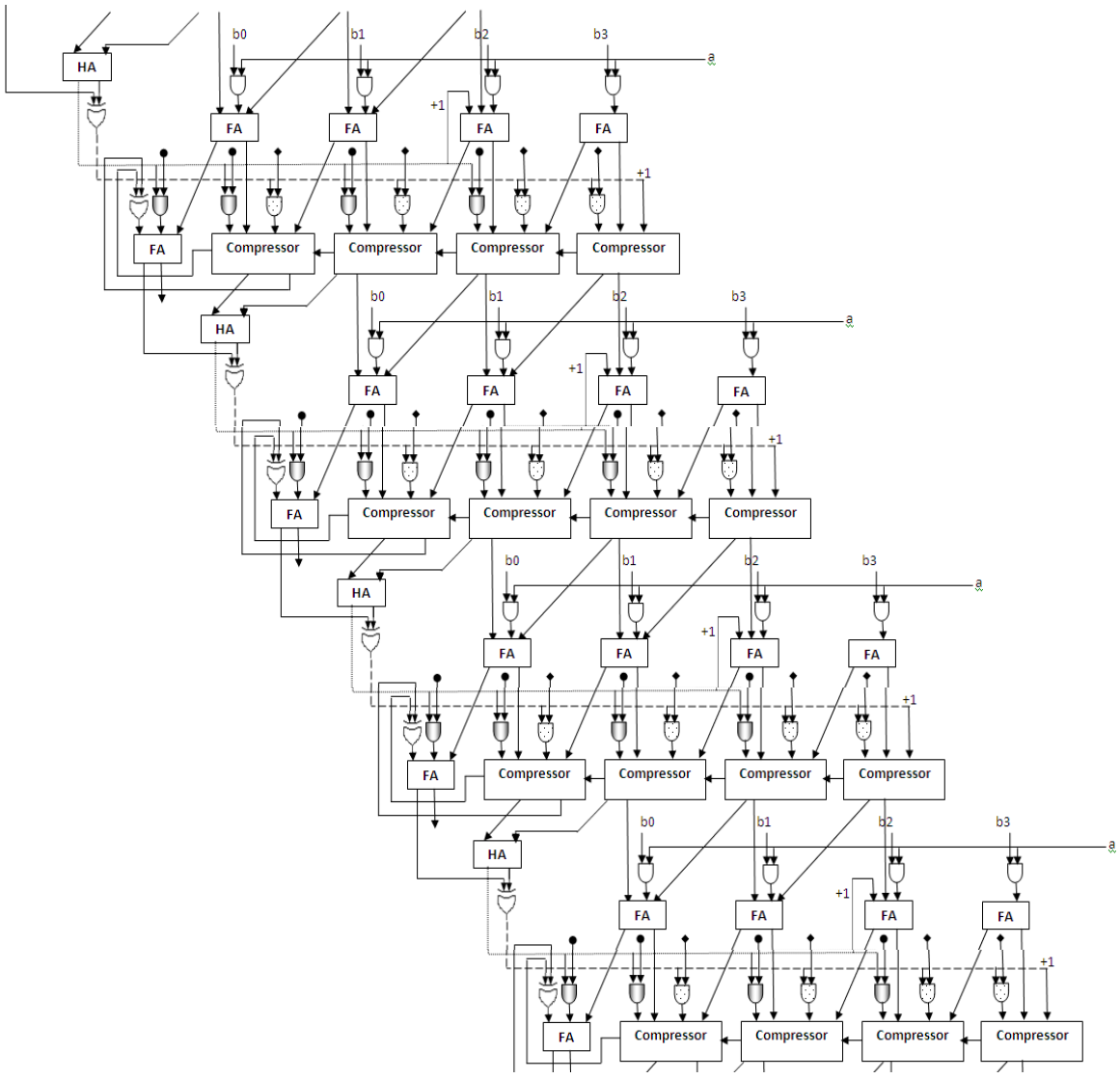


Fig. (2). The MSB Modular multiplier using 4-2 Compressor composed of two FAs

4. Simulation of the circuit:

The Modular multiplier is simulated using Matlab as shown below:

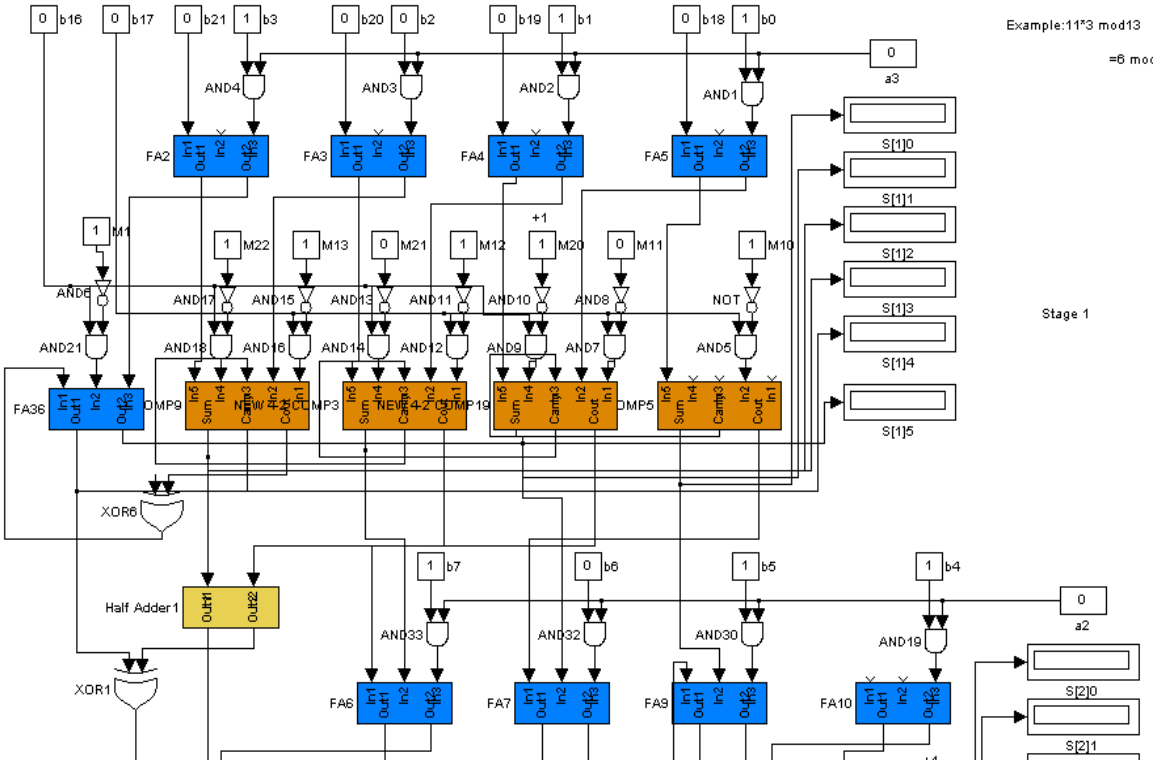
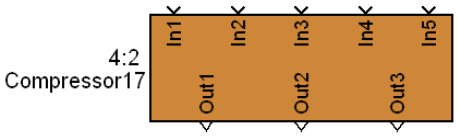


Fig. (3). One stage of the simulated circuit**.**



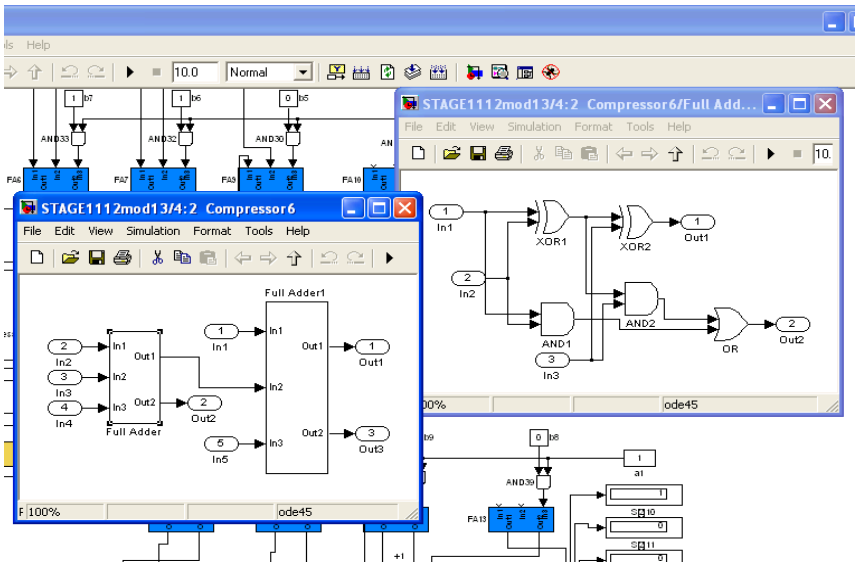


Fig. (4). The 4-2 Compressor contain by two full adders connecting as below

5. The MSB Modular multiplier:

Various approaches have been proposed in the literature to improve the 4-2 compressor efficiency. In [2], the low-power low-voltage 4-2 compressors is presented, we use this compressor in our MSB modular multiplier to have more efficiency by decreasing the XOR delay. The circuit of the second 4-2 compressor in fig 5

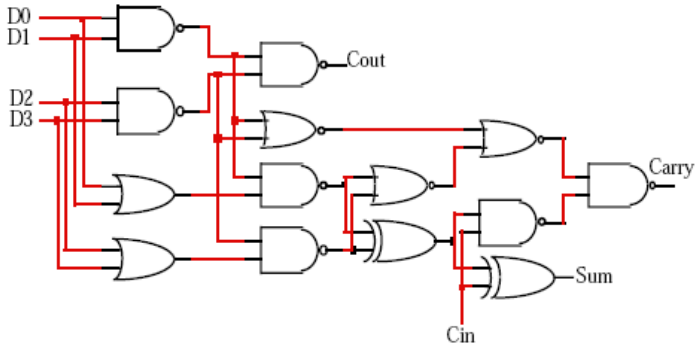


Fig. (5). Compressor composed of two FAs.

The same previous circuit of the Modular multiplier is used but the compressor circuit is exchanged to the low-power low-voltage 4-2 compressors which in fig 5.
The number of the XORs in the compressor is reduced by half when we used this construction.

6. Comparison and evaluation:

A performance comparison of the new proposed architectures with similar structures available in the literature [,,] in terms of both the time, number of latches and the area is presented in Table 4.
It should be mentioned that the unit gate area (GA) and the unit gate delay (Δ) represent the area required and the propagation delay of a NAND gate or of a NOR gat. The area required (A) and the propagation delay (T) for the typical gate functions, a full adder, 2-bit Multiplexer (MUX),and a latch are listed in Table 3 according to references [,,]

Table (3). area required (A) and the propagation delay (T) for the typical gate functions

Structure or Gate Function	Area (A)	Propagation Delay (T)
AND	$A_{AND} = 2G_A$	$T_{TND}=2\Delta$
OR	$A_{OR}=2\ G_A$	$T_{OR}=2\Delta$
NAND	$A_{NAND}=1\ G_A$	$T_{NAND}=1\Delta$
NOR	$A_{NOR}=1\ G_A$	$T_{NOR}=1\Delta$
Inverter(INF)	$A_{INF}=1\ G_A$	$T_{INF}=1\Delta$
XOR	$A_{XOR}=3\ G_A$	$T_{XOR}=3\Delta$
Full Adder (FA)	$A_{FA}=10\ G_A$	$T_{FA}=6\Delta$
2-bit Multiplexer (MUX)	$A_{MUX}=4\ G_A$	$T_{MUX}=3\Delta$
Latch	$A_{Latch}=7\ G_A$	$T_{Latch}=5\Delta$

7. Conclusion

The algorithm and implementation of MSB modular multiplier using 4-2 compressor is presented. This algorithm avoids carry store and feedback which used in Montgomery multiplier [13]. The use of the low-power low-voltage 4-2 compressors, improve the total delay. A generalization of this algorithm for higher radix can be using booths algorithms.

REFERENCES

[1] Martin Margala and Nelson G. Durdle," low-power low-voltage 4-2 compressors for VLSI applications ", IEEE- 1999.

[2] D. Radhakrishnan, A.P. Preethy, "Low Power CMOS Pass Logic 4-2 Compressor for High-speed Multiplication", IEEE-Aug, 2000.

[3] G.R. Blakley, “A computer algorithm for the produlo m,” IEEEtRANS. Comput., vol.C-32, pp. 497-500, 1983

[4] E. F. Brickell. “A fast modular multiplication algorithm with application to two-key cryptography,” in Proc. CRYTO’ 82 Advances Cryptology, 1982, pp. 51-60.

[5] -,” A survey of hardware implementations of RSA,” IN P ROC. Crypto’89 Advances Cryptology, Berlin , Germany , 1989, pp.368-370.

[6] H. Orup and P. Kornerup, “ Ahigh-radix hardware algorithm for calculating the exponential M^E modulo

N," IN Proc. 10th IEEE Symp. Comput. Arithmetic, Grenoble, , France, june1991, pp.51-56.

[7] P.Kornerup, "High-radix modular multiplication for cryptosystems," in Proc.11th IEEE Symp. Comput. Arithmetic, Windsor, ON , Canada, June 1993, pp. 277-283.

[8] C. K. Koc, " RSA Hung, " Bit-level systolic arrays for modular multiplication "J. VLSI Signal Processing, vol. 3 no. 3, pp. 215-223,1991.

[9] C. K. Koc, "RSA Hardware Implementation." RSA Laboratories, Tech. Rep., RSA Data Security Inc., Redwood, CA, 1995.

[10] N.Takagi and S. Yajima, "Modular multiplication hardware algorithms with a redundant representation and their application to RSA cryptosystem," IEEE Trans. Comput., vol. 41, pp. 887-891, July 1992.

[11] N.Takagi, " Aradix-4 modular multiplication hardware algorithm for modular exponentiation, " IEEE Trans. Comput., vol. 41, pp. 949-956, Aug. 1992.

[12] ThesisGeorgi Todorov "ASIC Design Implementation and Analysis of A Scalable High- Radix Montgomery Multiplier" ,December 2000

[13] Jin-Hua Hong, Member, IEEE, and Cheng-Wen Wu, Senior Member, IEEE "Cellular-Array Modular Multiplier for Fast RSA Public-Key Cryptosystem Based on Modified Booth's Algorithm" 2003