ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 7, Issue 5, November 2017

# Optimal speed real time implementations of modular multiplication

Zaid abdulsatar

*Abstract— This paper discuss design and implementation of four modular multipliers (Montgomery, faster Montgomery, standard interleaved, modified interleaved). The results are obtained using Very High Speed Integrated Circuit Description Language (vhdl ) through ISE 14.5 simulation tool family ZYNQ XC7Z045 device FFG900 -3 speed. The testing of the results according to time factor determining the speed of each multipliers and comparing the number of clocks of each multiplier with the number of the bits of each input of the multiplier and the maximum clock frequency can be used ( which is inverse of the period delay ) . however the main task of the paper is to find the optimum multiplier rakes minimum execution time . The faster Montgomery multiplier is the optimum one with number of clocks = number of bits of inputs × 3 + 2 , with clock frequency 36.547 MHZ.*

*Index Terms*—**Faster Montgomery, Modified Interleaved, Modular Multiplication, Montgomery, RSA, VHDL.**

## I. INTRODUCTION

The calculation of the modular exponentiation is an essential and important operation in many scientific fields, especially in the field of cryptography, The RSA (Rivest Shamir Adleman) is from the most common used public-key cryptosystems [5]. The encryption function is based on modular exponentiation with key size of 1024 or 2048 bits, and the modular multiplication is one of the major computation methods to implement modular exponentiation in cryptography systems [6]. The performance of the modular multiplication is the core arithmetic of the RSA cryptosystem comparing with other cryptography systems (e.g. international data encryption algorithm Diffie-Hellman key exchange [7, 8, 9], So this paper will discuss four types of modular multiplications, finding the best algorithm according to the time factor.

This paper is organized as follows: section II describes Montgomery multiplication; section III faster Montgomery multiplication; section IV standard interleaved multiplication algorithm; Section V describes modified interleaved multiplication algorithm; section VI presents results and discussion; section VII presents conclusion and future work.

## II .MONTGOMERY MULTIPLICATION

In 1985 Peter L. Montgomery invented a method to implement modular multiplication using, it computes P = $(X\times Y)\times(2^n)^{-1}$ , X, Y are two numbers, n is the number of bits in X, Montgomery multiplication performs first conversion of numbers to Montgomery domain and then the result is re-converted into Montgomery domain. This transformation operation exchanges division by several shift operations, Let X and Y be two n-bit numbers then Montgomery multiplication transformations in the following equations [2]:

$$Xm = X \times 2^n mod\ M \quad (1)[2]$$

$$Ym = Y \times 2^n mod\ M \quad (2)[2]$$

$$Z = X \times Y\ mod\ M \quad (3)[2]$$

$$Zm = Montgomery\ production\ (Xm, Ym, M) \quad (4)[2]$$

$$Zm = X \times Y \times 2^n mod\ M \quad (5)[2]$$

$$Zm = Z \times 2^n mod\ M \quad (6)[2]$$

The key concepts of the Montgomery algorithm are the following points [2,1] :

A. Adding a multiple of M to the intermediate results doesn't effect to the value of the final result, because the result is computed modulo M. and M is an odd number.

B. After each addition in the internal loop the least significant bit (LSB) of the intermediate result is requested. If it is 1 logic, the intermediate result is odd we add M to make it even. This even number can be divided by 2 with zero remainder; the division by 2 reduces the intermediate result to n+1 bits again.

C. After n steps these divisions a perform one division by $2^n$ . This algorithm is very easy to implement since it operates least significant bit first and does not require any comparisons, the hardware implementation is presented in algorithm 1 and described in fig(1). For detailed information of the Montgomery algorithm, can be seen to [2] and [1].

**Algorithm 1: Montgomery multiplication [1]**

```
— Inputs :X,Y,M with X>0,Y<M
— Output :P=(X*Y(2^n)^-1) mod M
— x_i : i bit of X;
— n:number of bits in x_i
— p0:LSB of p
— 1) p:=0;
— 2)for( i=0 ; i<n ; i++ ){
— 3)P:= P + X_i * Y;
— 4)P:= P + P_0 * M;
— 5)P:= P div 2;}
  6) if (P ≥ M) then P:= P-M
```

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 7, Issue 5, November 2017

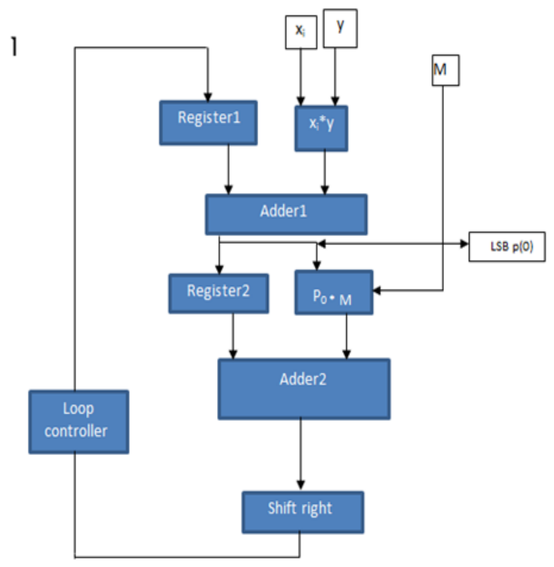


**Fig.(1) The inner loop of Montgomery algorithm [1].**

S0 : assign ( start = '0' ) , initialize the system with inputs (x , y , m) , flag counter =1 , p =0 , then make the ( start= '1' ), go to S1 .
S1 : if ( x(i) = '1' ) then load adder1 with values of p , y then go to S2.
S2 : if ( p(0) = '1' ) : load adder2 with values of values of p , m then go to S3 .
S3 : shift value of p .
S4 : check the counter if finished make flag counter =0 , and go to S5 else go to S1
S5 : if ( p >= m ) decrement p by m, then go to S8 .
S6 : go to S0[1].

## III. FASTER MONTGOMERY ALGORITHM

New optimized method for Montgomery multiplication , this would require considerable hardware resources to implement the architecture of the hardware behind this optimized algorithm is that to reduce the chip area for practical hardware implementation of Montgomery Algorithm . This is possible if we can recomputed the four Intermediate results to be added to the intermediate result in the loop in algorithm , by reducing the number of additions from 2 to 1 inside the loop in Montgomery . There are four possible scenarios[1]:

A. if the old value of the result is an even number, and if the bit xi of X is 0, then add none before we perform the reduction of result by division by 2(right shift).

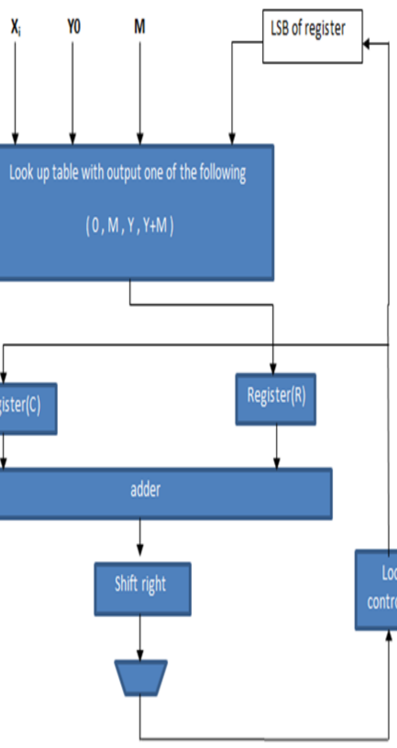
B. if the old value an odd number, and if the bit xi of X is 0, then add M to make the intermediate result even and then making the result divided by 2 (shifting right by 1) .

C. if the old value of the intermediate result in the loop is an even number, and if the bit xi of X is 1, with the incrimination of xi *Y is even, too, so there is no need to add M to make the intermediate result even. Then, in the loop we add Y before perform the division by 2. The same action is necessary if the result is odd, and if the bit xi of X is 1 and Y is odd as well as. In this case, (p + Y) is an even number, too.

D. The same scenario is necessary if the old value of result is even, and the bit xi of X is 1 , and Y is odd. In this case, p +Y+M will be an even number, too. The computation of Y+M can be done prior to the loop. This saves one of the two additions which are replaced by the choice of the right operand can be added to the result, the hardware implementation is presented in algorithm 2 and described in fig(2) [1].

**Algorithm 2 : faster Montgomery multiplication[2]**

```
—   Inputs :X,Y,M with X>0,Y<M
—   Output :P=(X*Y(2^n)^-1) mod M
—   x_i : i bit of X;
—   n:number of bits in x_i
—   S0:LSB of s
—   1) p:=0, r := y + m ;
—   2)for( i=0 ; i<n ; i++ ){
—   3)if( x(i) = 0 ) then
—   p:=p + p_0 * m ;
—   Else
—    if ( p_0 xor y_0 ) then
—   p:= p + r ;
—   else if ( not (p_0 xor y_0 )) then
—   p:= p + y ;
—   end if;
—   end if;
—   4)p:= p div 2; }
—   5)if (P ≥ M) then P:= P-M
```



—

— **Fig (2). Inner loop of Faster Montgomery method[1]**

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 7, Issue 5, November 2017

S0 : assign ( start = '0' ) , initialize the system with inputs (x , y , m) , make flag counter =1 , p =0 , then make the ( start= '1' ), go to S1 .
S1 : initialize .look up table with values to get value of R register
S2 : load adder with values of values of p , m then go to S3 .
S3 : shift right value of p .
S4 : check the counter if finished make flag counter =0 , and go to S5 else go to S1
S5 : if ( p >= m ) decrement p by m, then go to S8 .
S6 : go to S0.

## IV. STANDARD INTERLEAVED MULTIPLICATION ALGORITHM

The purpose of the interleaved multiplication and reduction is to keep the intermediate results as short as possible, for n steps To find the result (P) this algorithm performs the following operations:

1. Shift left : 2*P
2. Partial product calculation: Xi * Y
3. Addition the first result with the second: 2*P+ Xi*Y
4. Two subtractions modulus from the result in third operation

If ( $P \geq M$) then $P := P - M$ ;
If ( $P \geq M$) then $P := P - M$ ;
the hardware implementation is presented in algorithm 3 and described in fig(3) [3, 4].

**Algorithm 3: standard interleaved modulo multiplication [3]**

Inputs: X, Y, with $0 \leq X , Y \leq M$
output: P= X * Y mod M
n : number of bits in X;
$x_i$: i th bit in X ;

```
1)P := 0 ;
2)For ( i = n-1 ; i≥0 ; i--) {
3)P := 2 *P ;
4)I := $x_i$ *Y;
5)P := P + I;
6)If ( P ≥M) then P := P – M ;
7)If ( P ≥M) then P := P – M ; }
```

The main advantages of this algorithm compared normal multiplication and division are the following:

- The whole algorithm required one loop only.
- The intermediate registers are not longer than ( n+2) bits so ( reducing the area).

  But there are some disadvantages:
- The algorithm requires one adder and two sub tractors in steps (5) , (6) and (7) .
- The latency to perform steps (4) , (5) because can't make the addition in step (5) unless comparison in step (4) is complete.
- The comparisons in steps (6) , (7) are of full bit length of P and can't be pipelined without delay because the result in step (7) depends on the result of step (6), and later this latency problem was solved by using modified interleaved algorithm[4] .

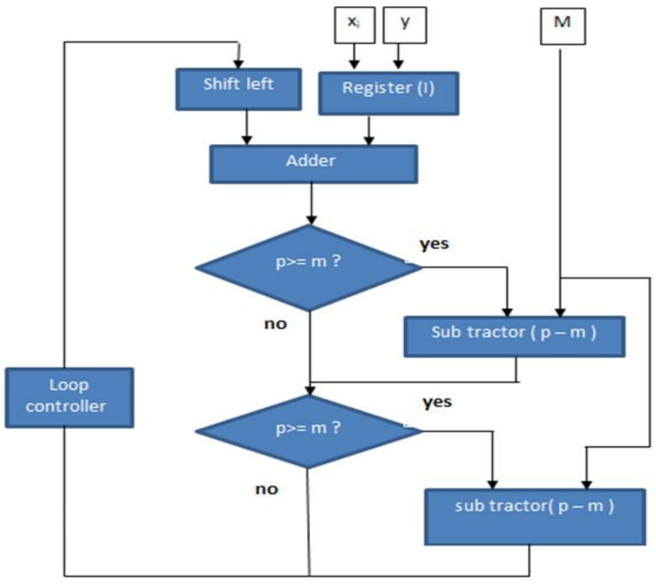


**Fig (3). Architecture of interleaved multiplication method[3].**

S0 : assign ( start = '0' ) , initialize the system with inputs (x , y , m) , make flag counter =1 , p =0 , then make the ( start= '1' ) declaring that data is ready, go to S1 .
S1 : shift the value of (P) , if ( x(i) = '1' ) then load I register with Y else clear I register ,
go to S2 .
S2 : load adder with values of p , I registers then go to S3.
S3 : if the value of P larger than M decrement M from this value, go to S4.
S4 : if the value of P again larger than M decrement M from this value, go to S5.
S5 : check the index of bits if finished make the flag counter = 0 and then finish else go to state S1 .
S6 : go to S0.

## V. MODIFIED INTERLEAVED MULTIPLICATION

Newer approach of interleaved multiplication used to decrease latency from ( 4 to 3) inside the loop , it involved finding recomputed value (R) which is ( $2 \times M$ ) and then abbreviation of algorithm 3 line (6) and (7) in one line (6) expressing two conditions :

1) If the previous value of P larger than R then decrease from P the value (R) notice that R equal to ( $2 \times M$ ) .

2) Else if the previous value of P larger than M then decrease from P the value (M).

This method decrease the latency inside the loop with one clock, the hardware implementation is presented in algorithm 4 and described in fig(4) [4].

***Algorithm4: modified interleaved modulo multiplication [4]***

Inputs: X, Y, with $0 \leq X , Y \leq M$
output: P= X * Y mod M
n : number of bits in X;

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 7, Issue 5, November 2017

```
 xi: ith bit in X ;
1)P := 0 , R := 2 * M;
2)For ( i = n-1 ; i≥0 ; i--) {
3)P := 2 *P ;
4)I := $x_i$ *Y;
5)P := P + I;
6)If ( P ≥R) then  P := P – R ;
      Else If ( P ≥M) then  P := P – M  ; }
```

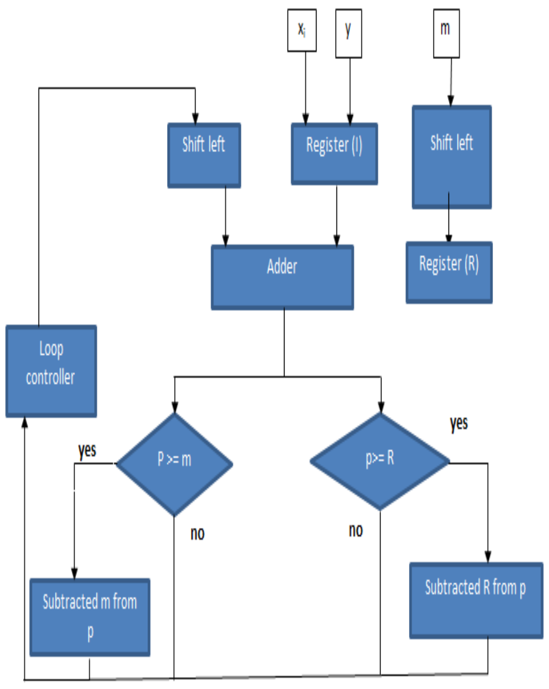


Fig (4). Architecture of modified interleaved method [4].

S0 : assign ( start = '0' ) , initialize the system with inputs (x , y , m) , make flag counter =1 , p =0 , then make the ( start= '1' ) declaring that data is ready, find R ( shift left m ), go to S1 .
S1 : shift the value of (P) , if ( x(i) = '1' ) then load I register with Y else clear I register ,
go to S2 .
S2 : load adder with values of p , I registers then go to S3.
S3 : if the value of P larger than M decrement M from this value, else if the value of P again larger than R decrement R from this value, go to S5.
go to S4.
S4 : check the index of bits if finished make the flag counter = 0 and then finish else go to state S1 .
S5 : go to S0.

## VI. RESULTS AND DISCUSSION

Figs (5 , 6 , 7 ) shows execution of Montgomery multiplier for ( 4 , 8 , 32 ) bits respectively, Figs (8 , 9 , 10 ) shows execution of faster Montgomery multiplier for ( 4 , 8 , 32 ) bits respectively , Figs (11 , 12 , 13 ) shows execution of standard interleaved multiplier for ( 4 , 8 , 32 ) bits respectively, Figs (14 , 15 , 16 ) shows execution of modified interleaved multiplier for ( 4 , 8 , 32 ) bits respectively, when start signal is 1 that means the start execution of the algorithm until done signal is 1 that means the algorithm is finished and signal p is the result, So The number of clocks when start is 1 until done equal 1 That is the number of Table(1) shows the number of clocks for each algorithm and for the same number of bits, table(2) shows the then maximum clock frequency that can be used for each multiplier.

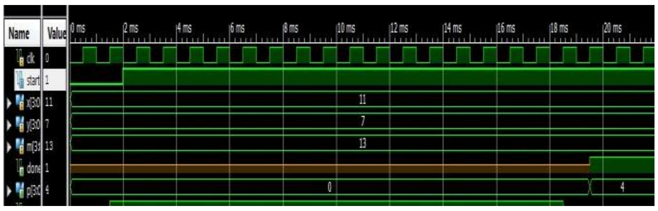


Fig (5). 4 bit Montgomery multiplier.

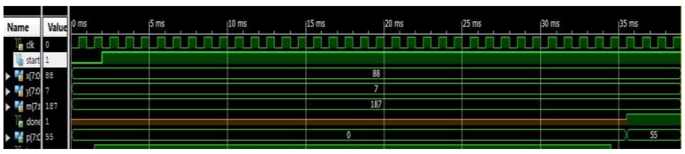


Fig (6). 8 bit Montgomery multiplier.

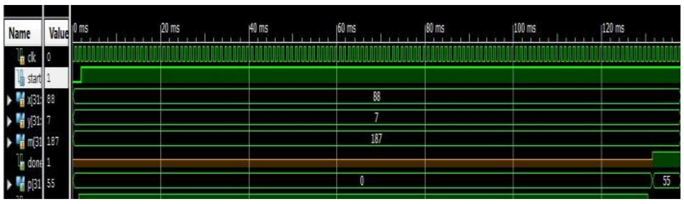


Fig (7). 32 bit Montgomery multiplier.

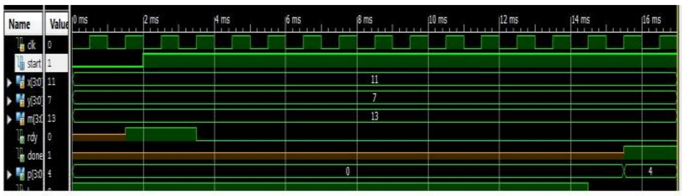


Fig (8). 4 bit faster Montgomery multiplier.

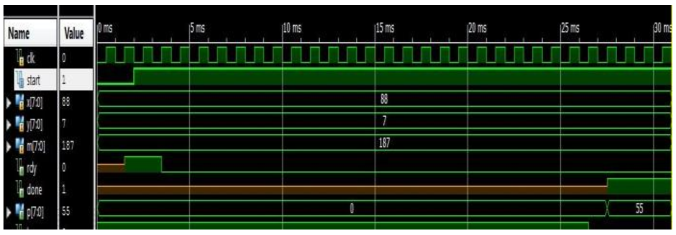


Fig (9). 8 bit faster Montgomery multiplier.

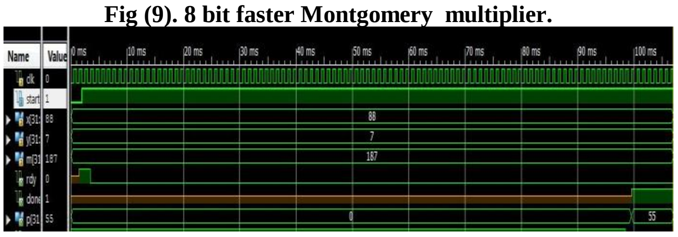


Fig (10). 32 bit faster Montgomery multiplier.

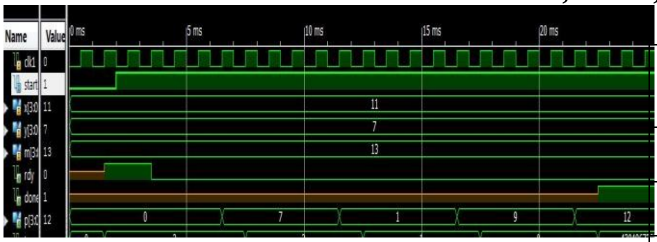


Fig (11). 4 bit standard interleaved multiplier.

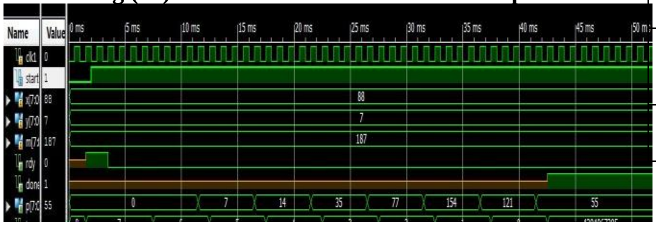


Fig (12). 8 bit standard interleaved multiplier.

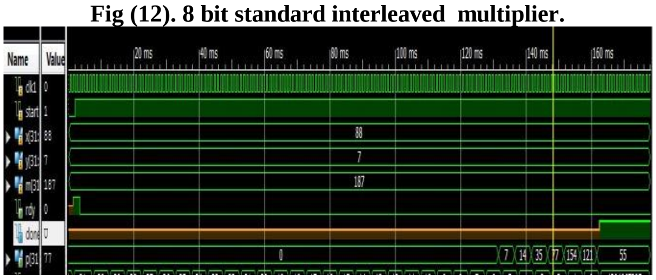


Fig (13). 32 bit standard interleaved multiplier.

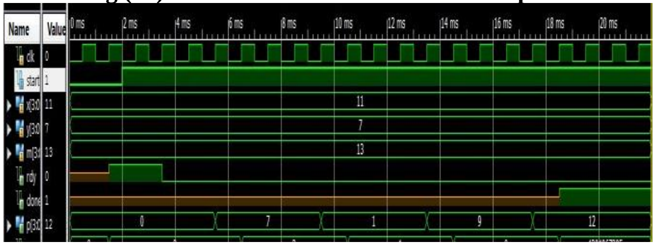


Fig (14). 4 bit modified interleaved multiplier.

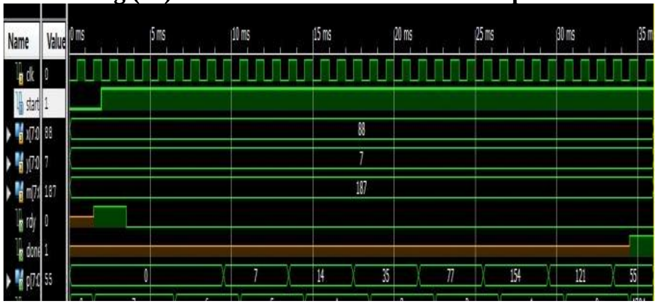


Fig (15). 8 bit modified interleaved multiplier.

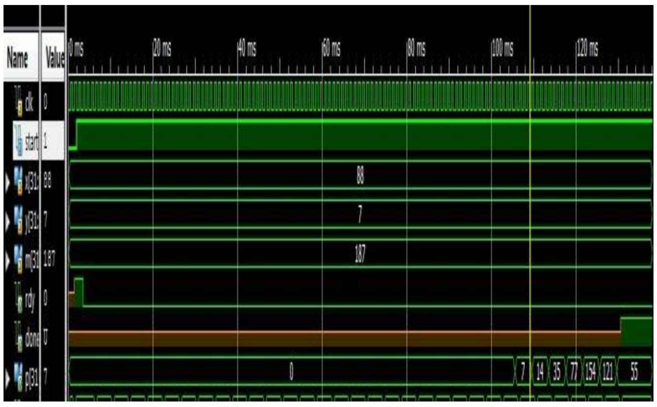


Fig (16). 32 bit modified interleaved multiplier.

Table(1) Number of clocks for each multiplier.

| Number of bits | Montgomery | Faster | Interleaved | modified |
|---|---|---|---|---|
| 4 | 18 | 14 | 20 | 16 |
| 6 | 26 | 20 | 30 | 24 |
| 8 | 34 | 26 | 40 | 32 |
| n | $n \times 4 + 2$ | $n \times 3 + 2$ | $n \times 5$ | $n \times 4$ |

Table(2) maximum frequency for each multiplier.

| Number of bits | Montgomery | Faster | Interleaved | Modified |
|---|---|---|---|---|
| 128 | 328.342 | 298.009 | 328.342 | 323.771 |
| 256 | 215.257 | 201.791 | 215.257 | 213.283 |
| 512 | 127.461 | 122.616 | 127.461 | 126.766 |
| 1024 | 70.198 | 68.703 | 70.198 | 69.987 |
| 2048 | 36.974 | 36.547 | 36.974 | 36.915 |

## VII . CONCLUSION AND FUTURE WORK

From Table(1*)* the number of clocks of faster multiplier fastest one of the others comes after it modified multiplier is faster than Montgomery and interleaved , then Montgomery multiplier is the last one, when the number of bits increased the number of clocks to get the result increased too, and from compartments 4, 8 , 32 bits multipliers it can be made equation between the number of bits and the number of clocks needed by each multiplier as shown in the same table , it can be noticed that when the number of bits increased the maximum frequency decreased ( minimum delay increased ) reached 36 MHZ in 2048 bit multiplier.

The speed of modular exponentiation which is used in the space of cryptography specifically for RSA algorithm [8] depends basically on modular multiplication, so using faster Montgomery multipliers is the optimal speed for real time implementation of such kind of cryptographic algorithms as a future work.

## REFERENCES

[1] V. Bunimov, M. Schimmler, "Area-Time Optimal Modular Multiplication", Embedded Cryptographic Hardware: Methodologies and Architectures, 2004.

[2] P. L. Montgomery, "Modular multiplication without trial division", Math. Computation, vol. 44, pp.519 - 521, 1985.

[3] G. R. Blakley, "A computer algorithm for the product AB modulo M", IEEE Transactions on Computers, 32(5): pp 497 - 500, May 1983.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 7, Issue 5, November 2017

[4] K. R. Sloan, Jr. Comments on "A computer algorithm for the product AB modulo M", IEEE Transactions on Computers, 34(3): pp 290 - 292, March 1985.

[5] M. E. Kaihara and N. Takagi, "A Hardware Algorithm for Modular Multiplication/ Division", IEEE Transactions on Computers, Vol. 54, No. 1, pp. 12-21, Jan. 2005.

[6] R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W. Fichtner, "A 177 Mb/sVLSI Implementation of International Data Encryption Algorithm," IEEE Journal Solid-StateCircuits,Vol.29,No.3,pp. 303-307, March 1994.

[7] F. J. Taylor, "Residue Arithmetic: A Tutorial with Examples," IEEE Computer Magazine, Vol. 17, No. 5, pp. 50-62, May 1984.

[8] S. Masui, K. Mukaida, M. Takenaka, and N. Torii, "Design Optimization of a High-Speed, Area-Efficient and Low-Power Montgomery Modular Multiplier for RSA Algorithm," IEICE Transactions on Electronics, Vol. 88, No. 4, pp. 576-581, April 2005.

[9] X. Lai and J.L.Massey, "A Proposal for a New Block Encryption Standard," in EUROCRYPT '90, Arahus, Denmark, May 1990.