An Optimized Montgomery Modular Multiplication Algorithm for Cryptography

P.Shenbagapriya(ME-II VLSI-design) Dr. G. Mahendran ME.,Ph.D,

ECE department, Associate professor and head
Syed ammal engineering college Ramanathapuram

Abstract:
*Montgomery modular multiplication is one of the fundamental operations used in cryptographicalgorithms, such as RSA andElliptic Curve Cryptosystem. The previous Montgomery multipliers perform a single Montgomery multiplication in approximately 2n clock cycles and it requires more numberof addition stages for large word length addition, where nis the size of of operandsoperands inbits. Inthis paper, new Montgomery modularmultiplier isproposed whichperformsthesame operationin approximatelyn clock cycles with almostsameclockperiod. The proposed multiplieruses carry selectadders (CSLAs) to perform largewordlengthadditions. Carry selectaddersis based on the concept of Binary to Excess-1convertor (BEC). The proposed algorithmusing the concept ofprecomputing partialresultsusingtwopossible assumptionseither zero or oneregarding the most significant bit of the previous word.The optimized algorithm is simulated using Xilinx ISE 12.1i and itis implemented using Virtex5 FPGA **device.***

Keyword *- Rivest, Shamir, Adleman(RSA), Carry Select Adders (CSLAs) and Binary to Excess-1convertor (BEC).*

1 Introduction

The Montgomery multiplication (MM) is the basic operation used in modular exponentiation,which is required in the Diffie-Hellman and RSA public-key cryptosystem.Recent implementations of the Montgomery Multiplication are focused on elliptic curve cryptography over the finite fields GF(p) and GF(2^m). A major design concern for multiplication units used in cryptography is the large number of operand bits, which causes large fan-out of signals, large wire delays, complex routing,.andthe cost of extra hardware resources.

Montgomeryalgorithm requires the built-in multipliersin the FPGA device to speed up thecomputation. This feature makes the implementation expensive. The systolic high-radix design by McIvor et al. described in [6] is capable of very high-speed operation, but suffers from the disadvantage of large area requirements for fast multiplier units. A different approach based on processing multiprecision operands in carry-save (CS) form has been presented in [7]. This architecture is optimized for the minimum latency and is particularly suitable for repeatedsequence of Montgomery multiplications, such as the sequence used in modular exponentiations

In this paper, the focus is given to the optimization of montgomery algorithm in order to minimize the number of clock cycles required tocompute an n-bit precision Montgomery multiplication and reducing number of gates used in multiplier by using carry select adders.Section 2 contains the introduction of Montgomery multiplication. Then, the new modified carry select adders are discussed in section 3. The new optimized algorithm, which is able toperform the n-bit precision MWR2MMalgorithm in approximatelyn clock cycles, is presented in Section 4. In Section 5, the experimental results of various architectures were discussed and the performance analysis has been done in section 6. Finally, the paper was concluded in section 7.

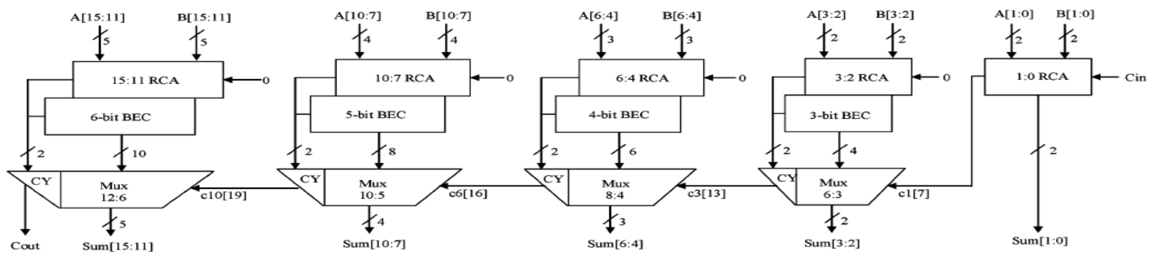


Fig.1 16 Bit Carry Select Adder

2 Montgomery Multiplication Algorithm

Let M >0 be an odd integer. In many cryptosystems, such as RSA, computing X .Y (mod M) is a crucial operation.The reduction of X .Y (mod M) is a more time-consuming step than the multiplication X.Y without reduction. Montgomery introduced a method for calculating products (mod M) without the costly reduction (mod M), since then known as Montgomery multiplicationobserved that the divisions canbe converted into simple shifts if multiplication isinstead performed on so-called *Montgomery residues*(*M-residues*). Montgomery multiplication(MM) of residues is defined as

$Z=MM(X,Y)=XY\, r^{-1}$ mod M

Observe that

$Z=XYr^{-1}$mod M=$xryrr^{-1}$modM

=(xy)r modM=zrmod M

so the Montgomery product of two Montgomeryresidues is the Montgomery residue of the product of the two corresponding integers.

```
Z = 0
fori= 0 to n-1
Z = Z + xi *Y
            ifZ is odd then Z = Z + M
 Z = Z/2
             ifZ = M then Z = Z – M
```

Simple radix-2 Montgomerymultiplication algorithm

This algorithm computes S = MM(X, Y) = X r^{-1}mod *M*. It uses *n* steps, as in a word-serialmultiplication algorithm. On step *i*, it adds the *Y* to the running sum if the *i*th bit of *X* (*xi*) is true. Also on each step, it divides by two. If the running sum was odd, it first adds *M* so the result can be divided by two without loss of information. This is permissible because adding *M* mod *M* does not change the result. After *n* steps of dividing by two, the algorithm hasdivided by *r* = 2^n. The algorithm might produce a result as large as 2*M*-1, so it concludes by subtracting *M* if necessary to restore the result to the legal range.The final subtraction can be

avoided for repetitive multiplications used in exponentiation.

3. Carry Select Adder

Carry Select Adder (CSLA) is one of the fastest adders used in many data-processing processors to perform fast arithmetic functions. However,the CSLA is not area efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by consideringcarry input c_{in} = 0 and c_{in}= 1 then the final sum and carry are selected by the multiplexers (mux).

The basic idea of this work is to use Binary to Excess-1 Converter(BEC) instead of RCA with c_{in}= 1 in the regular CSLA to achieve lower area and power consumption .[2] The main advantage of thisBEC logic comes from the lesser number of logic gates than the n-bit Full Adder (FA) structure.a structure of a 4-b BEC and 16 bit CSLA are shown in Fig. 2.and Fig 1.

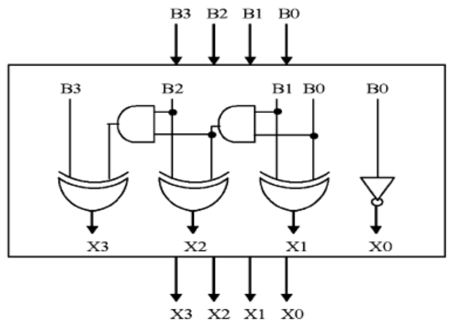


Fig.2 4 bit BEC

4. Optimized Montgomery Multiplier Using Carry Select Adders

In thiscontext, the optimization of hardware block for single wordand multiword Radix-2(MWR2MM) inorder to minimize the number of clock cycles required tocompute an n-bit precision Montgomery multiplication

Was focussed.Ituses carry select adders (CSLAs) to

```
    Input: odd $M, n = \lfloor \log_2 M \rfloor + 1, X = \sum_{i=0}^{n-1} x_i \cdot 2^i$, with
           $0 \leq X, Y < M$
    Output: $Z = MP(X, Y, M) \equiv X \cdot Y \cdot 2^{-n} \pmod{M}, 0 \leq Z < M$
1.1 $S[0] = 0$;
1.2 for $i = 0$ to $n - 1$ do
1.3     $q_i = (x_i \cdot Y_0) \oplus S[i]_0$;
1.4     $S[i+1] = (S[i] + x_i \cdot Y + q_i \cdot M)/2$;
1.5 if $S[n] > M$ then
1.6     $S[n] = S[n] - M$;
1.7 return $Z = S[n]$;
```

perform large word length additions[1].

Algorithm 1 shows the pseudocode for the radix-2Montgomery multiplication, where we choose n=($\log_2$M)+1 . n is the size of M in bits.

Algorithm 1

In [3], [4], Tenca and Koc¸ proposed a scalable architecturebased on the Multiple-Word Radix-2 Montgomery MultiplicationAlgorithm, shown as Algorithm 2.

Algorithm 2

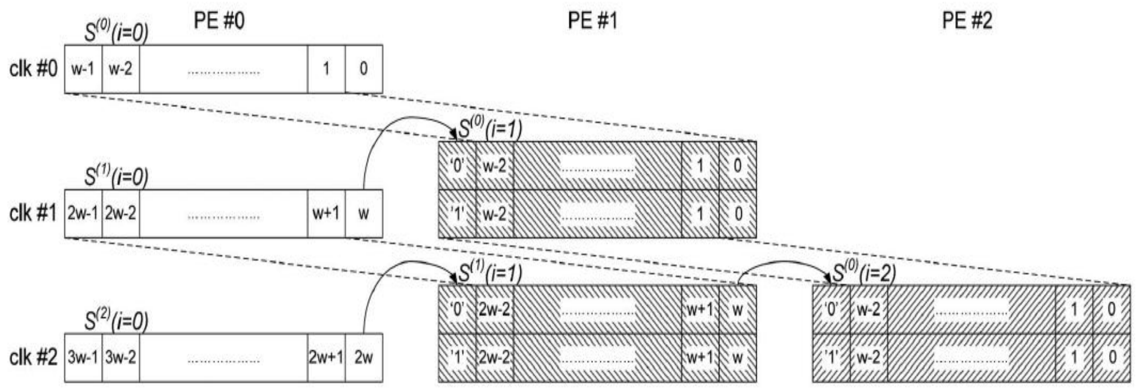
In Algorithm 2,[1] the operand Y (multiplicand) is scannedword-by-word, and the operand X is scanned bit-by-bit.The operand length is n bits, and the wordlength is w bits.e =(n+1)/w words are required to store S since its range is[0, 2M -1]. The original M and

```
Input: odd $M$, $n = \lfloor \log_2 M \rfloor + 1$, word size $w$, $e = \lceil \frac{n+1}{w} \rceil$,
       $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$, $Y = \sum_{j=0}^{e-1} Y^{(j)} \cdot 2^{w \cdot j}$,
       $M = \sum_{j=0}^{e-1} M^{(j)} \cdot 2^{w \cdot j}$, with $0 \le X, Y < M$
Output: $Z = \sum_{j=0}^{e-1} S^{(j)} \cdot 2^{w \cdot j} = MP(X, Y, M) \equiv X \cdot Y \cdot 2^{-n}$
        (mod $M$), $0 \le Z < 2M$
2.1 $S = 0$;                    /*initialize all words of S*/
2.2 for $i = 0$ to $n-1$ do
2.3     $q_i = (x_i \cdot Y_0^{(0)}) \oplus S_0^{(0)}$;
2.4     $(C^{(1)}, S^{(0)}) = x_i \cdot Y^{(0)} + q_i \cdot M^{(0)} + S^{(0)}$;
2.5     for $j = 1$ to $e$ step 1 do
2.6         $(C^{(j+1)}, S^{(j)}) = C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)} + S^{(j)}$;
2.7         $S^{(j-1)} = (S_0^{(j)}, S_{w-1..1}^{(j-1)})$;
2.8     $S^{(e)} = 0$;
2.9 return $Z = S$;
```

Y are extended by one extra bit of 0 as the most significant bit. Presented as vectors,

$(c^{(j+1)}, s(j)) = c(j) + x_i Y^{(j)} + q_i M_j + s(j)$
Assuming that $c^{(j)} \le 3$, we obtain
$(c^{(j+1)}, s(j)) = c(j) + x_i Y^{(j)} + q_i M_j + s(j)$
$\le 3 + 3(2^w - 1)$
$\le 3.2^w$
we have $c^{(j+1)} \le 3$. By induction, $c(j) \le 3$ is ensured for any $0 \le j \le e-1$. Additionally, based on thefact that $S \le 2M$, we have $c^e \le 1$.

In order to reduce the two-clock-cycle delay to half, we propose an approach of precomputing the partial results using two possible assumptions regarding the most significantbit of the previous word in order to reduce the two-clock-cycle delay to half, we propose an approach of precomputing the partial resultsusing two possible assumptions regarding the most significant bit



$M = (0, M^{(e-1)}, \ldots\ldots M^{(1)}, M^{(0)})$
$Y = (0, Y^{(e-1)}, \ldots\ldots Y^{(1)}, Y^{(0)})$
$S = (0, S^{(e-1)}, \ldots\ldots S^{(1)}, S^{(0)})$
And $X = (x_{n-1}, \ldots\ldots\ldots\ldots x_1, x_0)$
The carry variable $C^{(j)}$ has two bits, as explained below.Assuming $C^{(0)} = 0$, each subsequent value of $C^{(j+1)}$ isgiven by

Algorithm 3. Computations in Task D

```
Input: $x_i$, $Y^{(0)}$, $M^{(0)}$, $S_0^{(1)}$, $S_{w-1..1}^{(0)}$
Output: $q_i$, $C^{(1)}$, $S_{w-1..1}^{(0)}$
3.1 $q_i = (x_i \cdot Y_0^{(0)}) \oplus S_1^{(0)}$;
3.2 $(CO^{(1)}, SO_{w-1}^{(0)}, S_{w-2..0}^{(0)}) = (1, S_{w-1..1}^{(0)}) + x_i \cdot Y^{(0)} + q_i \cdot M^{(0)}$;
3.3 $(CE^{(1)}, SE_{w-1}^{(0)}, S_{w-2..0}^{(0)}) = (0, S_{w-1..1}^{(0)}) + x_i \cdot Y^{(0)} + q_i \cdot M^{(0)}$;
3.4 if $S_0^{(1)} = 1$ then
3.5     $C^{(1)} = CO^{(1)}$;
3.6     $S_{w-1..1}^{(0)} = (SO_{w-1}^{(0)}, S_{w-2..1}^{(0)})$;
3.7 else
3.8     $C^{(1)} = CE^{(1)}$;
3.9     $S_{w-1..1}^{(0)} = (SE_{w-1}^{(0)}, S_{w-2..1}^{(0)})$;
```

of the previous word. As shown in Fig. 3, PE #1 can take the (w -1) most significant bits of $S^{(0)}(i=0)$ from PE #0 at the beginning of clock #1, do a right shift, andcompute two versions of $S^{(0)}(i=0)$, based on the two different assumptions about the most significant bit of this word at the start of computations. At the beginning of theclock cycle #2, the previously missing bit becomes available as the least significant bit of $S^{(1)}(i=0)$. This bit can be used to choose between the two precomputed versions of$S^{(1)}(i=1)$. Similarly, in the clock cycle #2, two different versions of $S^{(0)}(i=2)$ and $S^{(1)}(i=1)$ are computed by PE #2 and PE #1, respectively, based on two different assumptions about the most significant bits.

Fig. 3 Data Operation Of Optimized Algorithm

of these words at the start of computations. At the beginning of the clockcycle #3, the previously missing bits become available as the least significant bits of $S^{(1)}(i=1)$ and $S^{(2)}(i=0)$, respectively.These two bits can be used to choose between the two precomputed versions of these words. The same pattern of computations is repeated in subsequent clock cycles.

Furthermore, since e words are enough to represent the values in S, $S^{(e)}$ is discarded in our designs. Therefore, e clock cycles are required to compute one iteration oS.

Algorithm 4. Computations in Task E

```
    Input: $q_i, x_i, C^{(j)}, Y^{(j)}, M^{(j)}, S_0^{(j+1)}, S^{(j)}_{w-1..1}$
    Output: $C^{(j+1)}, S^{(j)}_{w-1..1}, S_0^{(j)}$
4.1 $(CO^{(j+1)}, SO^{(j)}_{w-1}, S^{(j)}_{w-2..0}) =$
    $(1, S^{(j)}_{w-1..1}) + C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)};$
4.2 $(CE^{(j+1)}, SE^{(j)}_{w-1}, S^{(j)}_{w-2..0}) =$
    $(0, S^{(j)}_{w-1..1}) + C^{(j)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)};$
4.3 if $S_0^{(j+1)} = 1$ then
4.4     $C^{(j+1)} = CO^{(j+1)};$
4.5     $S^{(j)}_{w-1..1} = (SO^{(j)}_{w-1}, S^{(j)}_{w-2..1});$
4.6 else
4.7     $C^{(j+1)} = CE^{(j+1)};$
4.8     $S^{(j)}_{w-1..1} = (SE^{(j)}_{w-1}, S^{(j)}_{w-2..1});$
```

In Task D consists of three steps, the computation of q_i, the calculation of two sets of possible results, and the selection between these two sets of results using an additional input S_0^1 , which becomes available at the end ofthe processing time for Task D. These three steps are shown in Algorithm 3. Task E corresponds to two steps, as shown in Algorithm 4. The data forwarding of $s_0^{(j)}$ and $S^j_{(w-1,\ldots 1)}$ from one circle E to the two circles in the right column takesplace at the same time. However, S_0^j is used for selecting the two partial results of S^{j-1}, and $S^j_{(w-1,\ldots 1)}$ is used for generating the two partial results of $S^{j.}$ Additions of words in above algorithm is calculated by CSLA shown in fig 4.

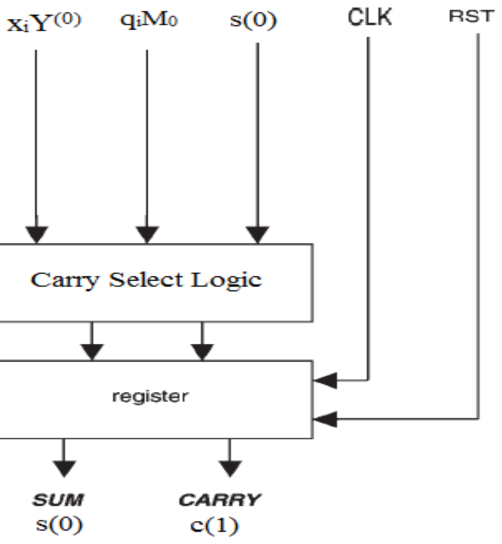


Fig .4 Word Additions By CSA

Fig .5 Mont Multiplier Unit With CSA

Instead of using normal adder we can use this carry select adder to reduce delay and the area consumption.This adder is used to add x_i , q_iM_0 and S(i) which have been mentioned in the algorithm.the Montgomery multiplier unit is shown in fig 5.

Parameter	**Montgomery algorithm**	**Montgomery algorithm with CSA**	**Montgomery algorithm with BEC**
Number of Slice LUTs	144	132	142
Number of bonded IOBs	22	22	22
Time (ns)	17.534	19.660	20.466

5. Experimental Results

The proposed Montgomery multiplication algorithm, Montgomery algorithm with carry select adder (CSA) and with Binary to excess-3 Converter (BEC) are simulated by using Xilinx ISE 12.1i and implemented in Virtex-5 FPGA processor. The experimental results are given in Table 1.

6. Performance Analysis

The performance of the Montgomery algorithm with various types of adders are analysed based on the area and time consumption which was given in table 1.

Table 1. Experimental Results

Based on the analysis, it is come to known that the Montgomery multiplication which uses the carry select adder gives the better reduction in area and time when compared to the other two methods and is shown in the fig.5.

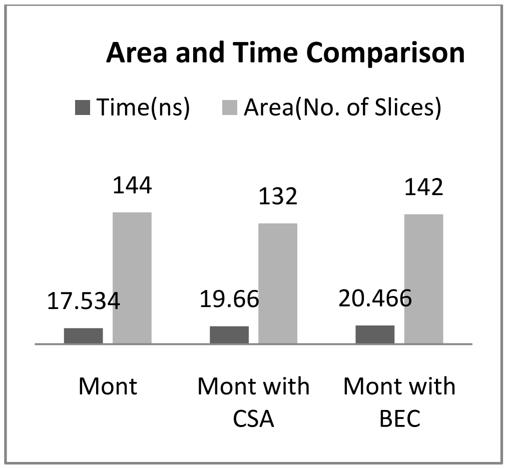


Fig.6 Area and Time Comparison

7. Conclusion

In general, Montgomery multiplication algorithmused for cryptography requires more area and time. In order to reduce both the parameters required, the various adders are used in the case of addition process. Among three types of addition processes which have discussed above, the Montgomery algorithm with carry select adder (CSLA) produces the optimized result. Because, it reduces both area and time considerably. Further the time consumption may also be reduced with the other new technique used for Montgomery multiplication algorithm.

References

[1] Miaoqing Huang, Kris Gaj, and Tarek El- Ghazawi,"New Hardware Architectures for Montgomery Modular Multiplication Algorithm",IEEE transactions on computers, vol. 60, no. 7, july 2011

[2] B. Ramkumar and Harish M Kittur," Low-Power and Area-Efficient Carry Select Adder",Ieee Transactions On Very Large Scale Integration (VLSI) Systems, vol. 20, no. 2, february 2012.

[3] A.F. Tenca and C¸ .K. Koc¸, "A Scalable Architecture for Montgomery Multiplication," Proc. First Int'l Workshop Cryptographic Hardware and Embedded Systems (CHES '99), pp. 94-108, 1999.

[4] A.F. Tenca and C¸ .K. Koc¸, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm," IEEE Trans.Computers, vol. 52, no. 9, pp. 1215-1221, Sept. 2003.

[5] Mathew, and S. Hsu, "An Improved Unified Scalable Radix-2 Montgomery Multiplier,"Proc. 17th IEEE Symp. Computer Arithmetic (ARITH), pp. 172-178, June. 2005.

[6]C.McIvor, M. McLoone, and J.V.McCanny, "High-Radix Systolic Modular Multiplication on ReconfigurableHardware," Proc. IEEE Int'l Conf. Field-Programmable Technology (ICFPT '05), pp. 13-18, Dec. 2005.

[7] C. McIvor, M. McLoone, and J.V. McCanny, "Modified Montgomery ModularMultiplication and RSA Exponentiation Techniques,"IEE Proc.- Computers and Digital Techniques, vol. 151,no. 6, pp. 402-408, Nov. 2004.

[8] Zhang Y.-Y. Li Z. Yang L. and Zhang S-W. 'An efficient CSA architecture for Montgomery modular multiplication',IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 31, no. 7, pp. 456–459, june 2009.

[9] Miyamoto A. Homma N. Aoki T. And Satoh A.,'Systematic design of RSA processors based on high-radix Montgomery multipliers', IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 7, pp. 1136–1146.Dec. 2011.

[10] Kuang S.-R. Wang J.-P. Chang K.-C. And Hsu H.-W. , 'Energy - efficient high - throughput Montgomery modular multipliers for RSA cryptosystems', IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 11, pp. 1999–2009. July 2013.

[11]J oão Carlos Néto1, Alexandre Ferreira Tenca2 and Wilson Vicente Ruggiero1.'A Parallel k-Partition Method to perform Montgomery Multiplication',IEEE Trans .Very Large Scale Integr. (VLSI) Syst., vol. 18, no.9, pp.4577 –1292. Feb 2011.

[12] Hong J.-H, and Wu C.-W. 'Cellular-array modular multiplier for fast RSA public-key cryptosystem based on modified Booth's algorithm', IEEE Trans.Very Large Scale Integr. (VLSI) Syst., vol. 11, no.3, pp. 1063-8210. Sept 2012.