Fewer cost and Superior Functioning Architecture of VSLI Premeditated with Multiplication of Montgomery

Meghana T.M, Pradeep Kumar S K
Meghana T.M PG student/ VLSI and ES, ECE, Kalpataru Institute of Technology Tiptur, Tumkur, India
Asst. Prof, Pradeep Kumar S K ECE Department, Kalpataru Institute of Technology Tiptur, Tumkur, India

Abstract

This paper is suggested forth with simple and efficacious Montgomery kind multiplication algorithm such that it is budgetary. Superior performance Montgomery modular multiplier can be enacted in congruence. The suggested multiplier acknowledges and turnouts of the data with the representations of binary utilized one level of Carry skip adder (CSA) to prohibit at the propagation at each summation type operation. This CSA can be premeditated to perform the respective operand at pre –guesstimation and the reformation of format basically from the format of carry save to the binary kind representation, which is following to a cost of low hardware and short cantankerous path lagging at extra clock cycles expense for making a culmination at one modular multiplication. To overcome this particularized weakness, configurable kind CSA could be one full adder or two of the serial half adders, which is suggested forth to lessen especially at the extra clock cycles for the pre-guesstimation operand and the format in reformation by a half. A mechanism is worked to investigate and consider skipping the unnecessary carry save addition operations in the one level CSA (CCSA) type architecture, while managing the short cantankerous path lagging, which has progressed.

Keywords*: Carry-save addition, Low-cost architecture, Montgomery modular multiplier, Public-key cryptosystem, Full carry save.*

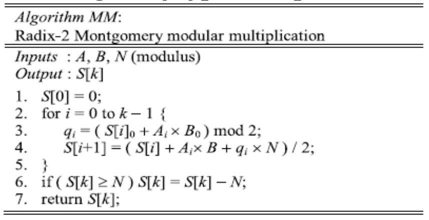
I. INTRODUCTION

Modular Multiplication is a cardinal operation in many application areas including public key cryptography [1] .This paper is written with motive at embellishing the performance of the carry skip adder(CSA) based Montgomery multiplier while managing lesser complexity in hardware.[4] Instead of the full carry-save(FCS) based multiplier with two level type CSA architecture semi-carry-save(SCS)based Montgomery modular multiplication(MM) type algorithm and its corresponding hardware in architecture with only one level CSA are suggested forth in this respective paper. Hardware architecture has many benefits and novel contributions over the previous respective designs can be known. First, one level CSA is premeditated forth to perform not only the summation operations in the iteration kind loop of the respective algorithm but also B plus N and the reformation of the format, which is following to a very short cantankerous path and cost being of the low relating to hardware.

First, one level CSA in premeditated to perform summation in the loop of iteration of Montgomery algorithm but also B in addition to N and the reformation of format, following to very short path in cantankerous and lower cost especially at hardware. Anyway, lot of extra clock cycles is essential to carry out B in addition to N and the respective reformation of format via the one level CSA type architecture. Advantage with the short cantankerous path will be lessened. To overcome those weaknesses, we then amend the one-level CSA architecture[3] which is to be able to perform the one three input carry save addition or two serial two input carry save additions,. Hence the extra clock cycles for the respective B in addition to N and the reformation format can be lessened to half. Ultimately, condition and circuit detection, which are basically different with that of the FCS-based multiplier (denoted as FCS MMM42 type multiplier) are progressed to pre-guesstimate quotients and skip those, which are of unnecessary carry save summation operations in the one level configurable CSA type architecture primarily while keeping a short cantankerous path in lagging. Hence, required cycles of clock for culminating one MM operation can be reduced to a greater extent. As the consequence, suggested Montgomery multiplier can be procured higher throughput and the smaller area time product than those of previously discussed Montgomery Multipliers.

II. MODULAR MULTIPLICATION ALGORITHMS [2]

A. *Montgomery type Multiplication*



```
Algorithm MM:
Radix-2 Montgomery modular multiplication
Inputs  : A, B, N (modulus)
Output : S[k]
1.  S[0] = 0;
2.  for i = 0 to k − 1 {
3.        q_i = ( S[i]_0 + A_i × B_0 ) mod 2;
4.        S[i+1] = ( S[i] + A_i × B + q_i × N ) / 2;
5.  }
6.  if ( S[k] ≥ N ) S[k] = S[k] − N;
7.  return S[k];
```

Fig . 1 MM Algorithm

B. SCS-Based Montgomery type Multiplication

This does not present an efficacious approach basically to remove the 32-bit CPA with multiplexers and registers(denoted as CPA_FC)for the reformation of format and hence this type suffers from the cantankerous path. On an other hand, Zhang et al is re premeditated the two levels CSA type architecture to perform with the reformation of format hence CPA_FC can be premeditated out .

```
Algorithm SCS-based MM:
SCS-based Montgomery multiplication
Inputs  : $A$, $B$, $N$ (modulus)
Outputs : $S[k+2]$
1. $SS[0] = 0$;  $SC[0] = 0$;
2. for $i = 0$ to $k + 1$ {
3.     $q_i = (SS[i]_0 + SC[i]_0 + A_i \times B_0) \bmod 2$;
4.   $(SS[i+1], SC[i+1]) = (SS[i]+SC[i] + A_i \times B + q_i \times N) / 2$;
5. }
6. $S[k+2] = SS[k+2] + SC[k+2]$;
7. return $S[k+2]$;
```

Fig . 2 SCS-based Montgomery Multiplication Algorithm.

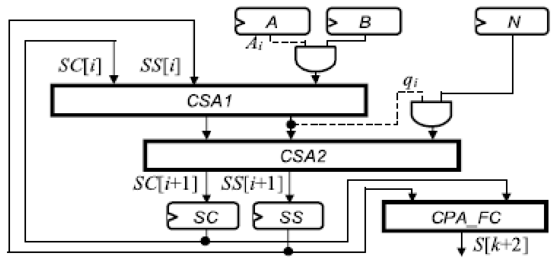


Fig . 3 SCS-MM-1 Multiplier.

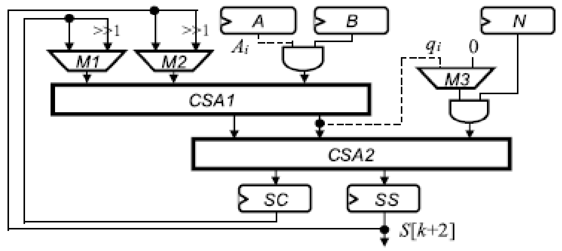


Fig . 4 SCS-MM-2 Multiplier.

C. FCS-Based Montgomery Multiplication

Cantankerous path of the FCS based Montgomery multiplier denoted as FCS-MM-2 multiplier may be slightly lessened with a significant increase in the respective hardware area, which when at comparison with the FCS based Montgomery multiplier denoted as FCS-MM-1 multiplier. Extra clock cycles for the reformation of format which is possibly lower than the performance of SCS-based type multipliers can be ascertained.

To further, we enhance the performance of the SCS reckoned multiplier, both the cantankerous path delay and the respective clock cycles for completing of one multiplication must be lessened while managing with low complexity of hardware.

```
Algorithm FCS-MM-1:
FCS-based Montgomery multiplication
Inputs  : $AS$, $AC$, $BS$, $BC$, $N$ (modulus)
Outputs : $SS[k+2]$, $SC[k+2]$
1. $SS[0] = 0$;  $SC[0] = 0$;
2. for $i = 0$ to $k + 1$ {
3.     $q_i = (SS[i]_0 + SC[i]_0 + A_i \times (BS_0 + BC_0)) \bmod 2$;
4.   $(SS[i+1], SC[i+1]) = (SS[i] + SC[i] + A_i \times (BS + BC)$
                                        $+ q_i \times N) / 2$;
5. }
6. return $SS[k+2]$, $SC[k+2]$;
```

Fig. 5 FCS-MM-1 Montgomery Multiplication Algorithm.

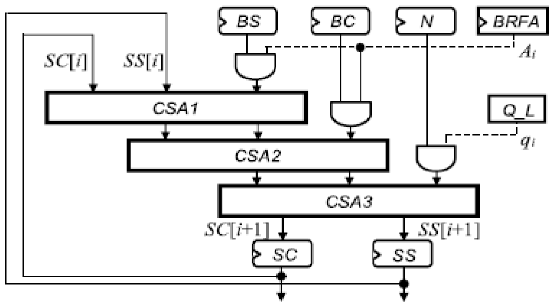
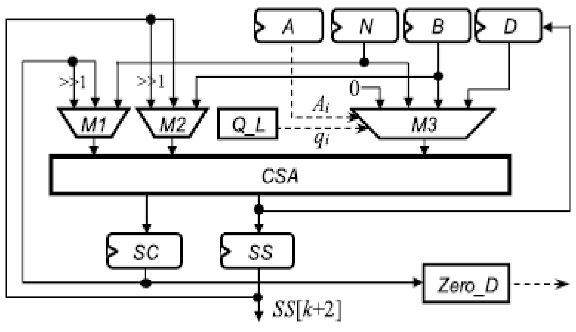


Fig. 6 Fcs-Mm-1 Multiplier

III. PROPOSED MONTGOMERY MULTIPLICATION

A. Cantankerous Path Delay Reduction

The cantankerous path lagging of SCS based multiplier can be lessened by amalgamating the advantages of FCS-MM2 and SCS-MM2. [5]



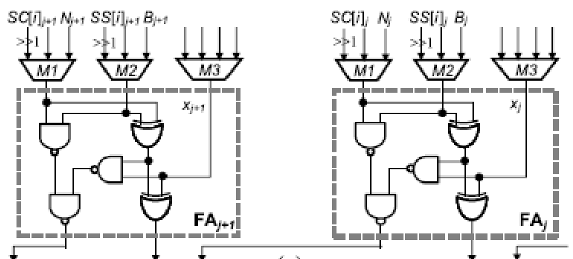
(b)

Fig .7(a) MSCS-MM multiplier

```
Algorithm Modified SCS-MM:
Modified SCS-based Montgomery multiplication
Inputs  : $A, B, N$ (modulus)
Output : $SS[k+2]$
1.  $(SS, SC) = (B + N + 0)$;
2.  while ($SC$ != 0)
3.      $(SS, SC) = (SS + SC + 0)$;
4.  $D = SS$;
5.  $SS[0] = 0$;  $SC[0] = 0$;
6.  for $i = 0$ to $k + 1$ {
7.      $q_i = (SS[i]_0 + SC[i]_0 + A_i \times B_0) \bmod 2$;
8.      if ($A_i = 0$ and $q_i = 0$)   $x = 0$;
9.      if ($A_i = 0$ and $q_i = 1$)   $x = N$;
10.     if ($A_i = 1$ and $q_i = 0$)   $x = B$;
11.     if ($A_i = 1$ and $q_i = 1$)   $x = D$;
12.     $(SS[i+1], SC[i+1]) = (SS[i] + SC[i] + x) / 2$;
13. }
14. while ($SC[k+2]$ != 0)
15.     $(SS[k+2], SC[k+2]) = (SS[k+2] + SC[k+2] + 0)$;
16. return $SS[k+2]$;
```
(a)

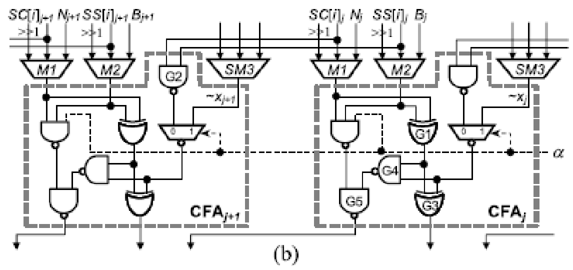
Fig .7(b) Modified SCS-based Montgomery Multiplication

Extra clock cycles basically for performing B plus N and the reformation of format through repeatedly executing the carry save summation, where to avoid a long carry propagation the intermediate result S of shifting modular addition carry save representation (SS, SC) being equal to SS+SC+0 are dependent on the propagation of longest carry in chain at *SS* plus *SC*. If *SS* = 111…1112 and *SC* = 000…0012, one level CSA architecture where k clock cycles are to complete SS plus SC.



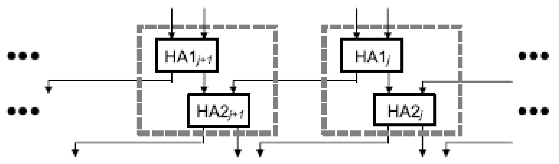
(a)

Fig .8(a) Conventional FA circuit.



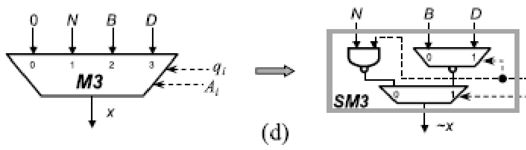
(b)

Fig. 8 (b) Suggested CFA circuit.



(c)

Fig . 8 (c) Two serial HAs.



(d)

Fig. 8 (d) Simplified multiplexer *SM3*.

We can ascertain that it is cantankerous to reduce the necessitated clock cycles of the SCS-based Montgomery multiplication(MSCS-MM type multiplier).

B .Clock Cycle Number Reduction

The crucial guesstimations in the respective for loop is performed in the following three to two carry save in addition.

(SS[*i* + 1], *SC*[*i* + 1]) = *(SS*[*i*] + *SC*[*i*] + *x)/*2 ……………… (1)

Where the variable *x* may be 0, *N*, *B*, or *D* depending on the values of *Ai* and *qi*.

skip*i*+1 =*∼(Ai*+1 ∨ *qi*+1 ∨ *SS*[*i* + 1]0*)* …………… (2)

, (ˆ *q*, ˆ*A*) = *(qi*+2, *Ai*+2*)* if skip*i*+1 = 1. Otherwise, (ˆ *q*, ˆ*A)* = *(qi*+1, *Ai*+1*)*.

C. Quotient Precomputation

It is easier to procure Ai+1 and Ai+2 in the ith iteration. Quotation is worked as referring Fig. 7(a) and 7(b) as follows:

qi+1 = (SS[i + 1]0 + SC[i + 1]0 + Ai+1 × B0) mod 2. ………(3)

By employing ˆ*N*

ˆN ={ N + 1, if N1:0 = 11
3N + 1, if N1:0 = 01. ………………………………………. (4)

The logic expression in (3) for generating *qi*+1 in the *i* th iteration can be rewritten as

qi+1 = *(SS*[*i*]1 ⊕ *SC*[*i*]1*)* ⊕ *(SS*[*i*]0 ∧ *SC*[*i*]0*)*……………. (5)

Similar to (3), the quotient *qi*+2 can be regulated in the *i* th iteration by the following equation:

qi+2 = *(SS*[*i* + 2]0 + *SC*[*i* + 2]0*)* mod 2. ……………….(6)

qi+2 = *(SS*[*i*]2 ⊕ *SC*[*i*]2*)* ⊕ *(qi* ∧ ˆ*N*2*)* ⊕ *(SS*[*i*]1 ∧ *SC*[*i*]1*)*. (7)

skip*i*+1 = *∼(Ai*+1 ∨ *qi*+1 ∨ *SS*[*i* + 1]0*)*
= *∼(Ai*+1 ∨ *(δ*1 ⊕ *δ*0*)* ∨ *δ*1*)*
= *∼(Ai*+1 ∨ *δ*1 ∨ *δ*0*)*

$= \sim(A_{i+1} \vee (SS[i]_1 \oplus SC[i]_1) \vee (SS[i]_0 \wedge SC[i]_0))$……….(8)

According to (8), we can quickly obtain skip$i$+1 in the i th iteration by $SS[i]_1$, $SC[i]_1$, $SS[i]_0$, and $SC[i]_0$.

D. Proposed Algorithm and Hardware Architecture

At considering cantankerous path delay reduction, clock cycle number reduction, and quotient precomputation mentioned above, a new SCS-based Montgomery type MM algorithm (i.e., SCS-MM-New algorithm shown in Fig. 10) utilizing one-level CCSA architecture is suggested to significantly minify the required clock cycles for completing one MM.

The hardware architecture of SCS-MM-New algorithm, expressed as SCS-MM-New multiplier. At the commencement of Montgomery multiplication, the FFs stored skip$i$+1, ˆ q, ˆA are first reset to 0 as shown in step 1 of SCS-MM-New algorithm, so that ˆD being equal to ˆB plus ˆN can be guesstimated via the one-level CCSA architecture. When considering the while loop, the skip detector Skip_D shown in Fig. 12 is utilized to produce skipi+1, ˆ q, and ˆA.

```
Algorithm SCS-MM-New:
Proposed SCS-based Montgomery multiplication

Inputs  : $A$, $B$, $\hat{N}$ (new modulus)
Output : $SS[k+5]$
1.  $\hat{B} = B << 3$;  $\hat{q} = 0$;  $\hat{A} = 0$;  $skip_{i+1} = 0$;
2.  $(SS, SC) = $ 1F_CSA$(\hat{B}, \hat{N}, 0)$;
3.  while ($SC$ != 0)
4.      $(SS, SC) = $ 2H_CSA$(SS, SC)$;
5.  $\hat{D} = SS$;
6.  $i = -1$;  $SS[-1] = 0$;  $SC[-1] = 0$;
7.  while ( $i \le k + 4$ ) {
8.      if ( $\hat{A} = 0$ and $\hat{q} = 0$)   $x = 0$;
9.      if ( $\hat{A} = 0$ and $\hat{q} = 1$)   $x = \hat{N}$ ;
10.     if ( $\hat{A} = 1$ and $\hat{q} = 0$)   $x = \hat{B}$ ;
11.     if ( $\hat{A} = 1$ and $\hat{q} = 1$)   $x = \hat{D}$ ;
12.     $(SS[i+1], SC[i+1]) = $ 1F_CSA$(SS[i], SC[i], x) >> 1$;
13.       compute $q_{i+1}$, $q_{i+2}$, and $skip_{i+1}$ by (5), (7) and (8);
14.     if ($skip_{i+1} = 1$){
15.         $SS[i+2] = SS[i+1] >> 1$;  $SC[i+2] = SC[i+1] >> 1$;
16.         $\hat{q} = q_{i+2}$;  $\hat{A} = A_{i+2}$;  $i = i + 2$;
17.     }
18.     else{
19.         $\hat{q} = q_{i+1}$;  $\hat{A} = A_{i+1}$;  $i = i + 1$;
20.     }
21. }
22. $\hat{q} = 0$;  $\hat{A} = 0$;
23. while ($SC[k+5]$ != 0)
24.     $(SS[k+5], SC[k+5]) = $ 2H_CSA$(SS[k+5], SC[k+5])$;
25. return $SS[k+5]$;
```

Fig. 10 SCS-MM-New algorithm.

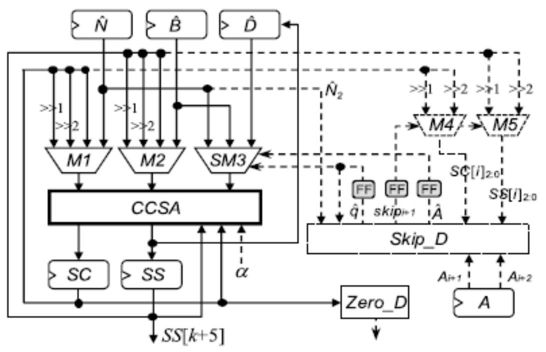


Fig. 11 SCS-MM-New multiplier.

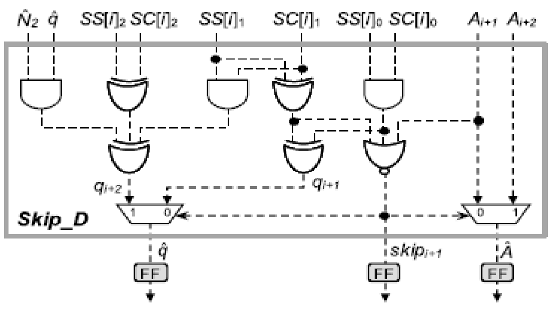


Fig. 12 Skip detector Skip_D.

III. EXPERIMENTAL RESULTS

Table. 1 Results

Library ami05	Area(µm2)	Power(nW)
SCS_MM_1	280.181	26160275.47
SCS_MM_2	536.20	3220778.09
FCS_MM_1	525.131	28560203.06
MSCS_MM	254.806	12717265.53
SCS_MM_NEW_MULTIPLIER	274.4519	14420499.72

IV. CONCLUSION

FCS based multipliers manages the input and turnouts at the operands of the Montgomery MM in the respective format of carry save format to escape from the reformation of format, following to fewer clock cycles but larger area than SCS reckoned multiplier. To embellish the performance of the Montgomery MM whiles managing the lower complexity in hardware, where this paper has expressed with the amendments the SCS based algorithm of Montgomery and suggested economical low cost and high performance Montgomery modular multiplier. Suggested multiplier utilized one level CCSA type architecture and skipped the unnecessary carry save addition type operations to lessen at the cantankerous path lagging and required clock cycles for the culmination of one MM operation. Exploratory results expressed that the suggested approaches are indeed capable of enhancing the respective performance of 2

CSA reckoned Montgomery multiplier while managing with hardware complicatedness at lesser level.

REFERENCES

[1] V. Bunimov, M. Schimmler, and B. Tolg, "A complexity-effective version of Montgomery's algorithm," in Proc. Workshop Complex. Effective Designs, May 2002.

[2] P. L. Montgomery, "Modular multiplication without trial division," Math. Comput., vol. 44, no. 170, pp. 519–521, Apr. 1985.

[3] Y.-Y. Zhang, Z. Li, L. Yang, and S.-W. Zhang, "An efficacious CSA architecture for Montgomery modular multiplication," Microprocessors Microsyst., vol. 31, no. 7, pp. 456–459, Nov. 2007.

[4] G. Perin, D. G. Mesquita, F. L. Herrmann, and J. B. Martins, "Montgomery modular multiplication on reconfigurable hardware: Fully systolic array vs parallel implementation," in Proc. 6th Southern Program. Logic Conf., Mar. 2010, pp. 61–66.

[5] J. Han, S. Wang, W. Huang, Z. Yu, and X. Zeng, "Parallelization of radix-2 Montgomery multiplication on multicore platform," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 12, pp. 2325–2330, Dec. 2013.