An Efficient Hardware Implementation of Radix-16 Montgomery Multiplication

Bien-Cuong Nguyen*, Cong-Kha Pham†
The University of Electro-Communications, Tokyo, Japan
Email: biencuong@vlsilab.ee.uec.ac.jp

***Abstract*—This paper proposes a radix-16 modular Montgomery multiplication based on modified Booth-16 encoding. The Booth-16 encoding is modified to reduce the complexity and critical path-delay of the generation of partial products. The proposed algorithm is evaluated on Xilinx FPGA Virtex7 XC7VLX330T-3 with ISE 14.7. The hardware synthesis result shows that the proposed algorithm reduces both the cost and latency by 9.9% and 21.3% respectively, in comparison to the previous works.**

***Index Terms*—Montgomery multiplication, modular multiplication, FPGA, high throughput, efficiency**

I. Introduction

Modular multiplication is the primary operation in the public-key cryptosystems (PKCs). Typically, the computation of PKCs requests the repeating modular multiplication. Therefore, the efficient implementation of the modular multiplication is a prerequisite condition to enhance the efficiency of the whole cryptosystem.

Montgomery multiplication (MM), which replaces the trial division by the efficient bit-shift and addition, has been considered as a fast modular multiplication for the hardware implementation [1]. However, the conventional MM processes 1 bit of the multiplier per round which leads to low throughput of the classical MM. Generally, high-radix MM is applied to improve the throughput of conventional MM. Radix-k MM reduces the number of iterations by k-time, as shown in algorithm 1. However, the critical path delay and computation complexity of x_iY and q_iM^* at step 4 of the algorithm 1 are increased rapidly as the increment of radix. It is a challenge for application of higher-radix MM.

Various methods have been proposed to reduce the complexity of high-radix MM [2]- [5]. In [2], a precomputation of M^* is proposed to efficiently compute q_i. Nevertheless, the height of the partial product of x_iY and q_iM^* is not reduced. In [3], the computation complexity of radix-4 MM is reduced by applying Booth-4 encoding. However, the higher-radix Booth encoding is hard to apply because of difficulty in the generation of odd partial products.

II. Radix-16 Montgomery Multiplication

In this paper, we propose an efficient method to improve the performance of radix-16 Montgomery multiplication based on modified Booth-16 encoding as shown in algorithm 2.

Firstly, to reduce the complexity of computation of x_iY, a modified Booth-16 encoding is proposed. Typically, the digit set of classical Booth-16 is the set of $(\pm 8Y, \pm 7Y, ..., \pm 1Y, 0)$. Total $3n$ full adders (FAs) are required to generate the odd partial products $(\pm 7Y, \pm 5Y, \pm 3Y)$. To overcome this drawback, we propose a modified Booth-16 encoding. The partial products of the proposed Booth-16 are efficiently computed by summation of two partial products of Booth-4 encoding as the formula, $B16_i = 2^2 B4_{2i+1} + B4_{2i}$, where $B16, B4$ are Booth-16 and Booth-4 encoding respectively. The computation of the proposed Booth-16 encoding is illustrated in Fig.1b. The proposed Booth-16 reduces resource utilization and critical path-delay of the computation of x_iY from $3n$ FAs to n FAs and two $(n+4)-bit$ 2's-complement conversions. The proposed encoding is applied to compute x_iY at step 3 and 7

Algorithm 1 Radix-k Montgomery multiplication [4]

```
    Input: $X, Y, M,\ R = 2^{k(n+2)}, 0 \le X, Y \le 2M^*$;
    $X = \sum_{i=0}^{n+2} (2^k)^i x_i, x_i \in \{0, 1, ..., 2^{k-1}\}, x_{n+2} = 0$;
    $M^* = (-M^{-1} mod\, 2^k)M, 4M^* < 2^{kn}$;
    Output: $S_{n+3} = X.Y.R^{-1} mod\, M$
 1: $S_0 \leftarrow 0$
 2: for $i \ge 0$ to n+2 do
 3:     $q_i \leftarrow S_i\, mod\, 2^k$;
 4:     $S_{i+1} \leftarrow (Si + q_iM^*)/2^k + x_iY$;
 5: end for
```

Algorithm 2 Proposed radix-16 Montgomery multiplication

```
    Input: Let $X, Y$, are n+1-bits 2's-complement numbers,
    M, n+1-bits odd integer, $-M \le X, Y < M$
    $R = 16^{\lfloor (n+5)/4 \rfloor}$, $M^* = -M^{-1} mod\, 2^4$;
    Output: $S_{\lceil \frac{(n+5)}{4} \rceil} = X.Y.R^{-1} mod\, M$
 1: $S_0 \leftarrow 0, x_{-1} \leftarrow 0, x_{n+2} : x_{n+5} \leftarrow x_{n+1}$;
 2: for $i$ from 0 to $\lfloor \frac{(n+5)}{4} \rfloor$ do
 3:     $B_i \leftarrow 2^2 B_4(x_{4i+3}, x_{4i+2}, x_{4i+1}) + B_4(x_{4i+1}, x_{4i}, x_{4i-1})$;
 4:     $q_i \leftarrow S_iM^*\, mod\, 2^4$; where $B_4$ is Booth-4
 5:     $q_{enc} \leftarrow ENC(q_i)$;
 6:     $S_i^* \leftarrow signext(S_i + q_{enc}M)/2^4$;
 7:     $S_{i+1} \leftarrow S_i^* + B_iY$;
 8: end for
 9: if $S_{\lceil \frac{(n+5)}{4} \rceil} < 0$ then
10:     $S_{\lceil \frac{(n+5)}{4} \rceil} \leftarrow S_{\lceil \frac{(n+5)}{4} \rceil} + M$;
11: end if
```

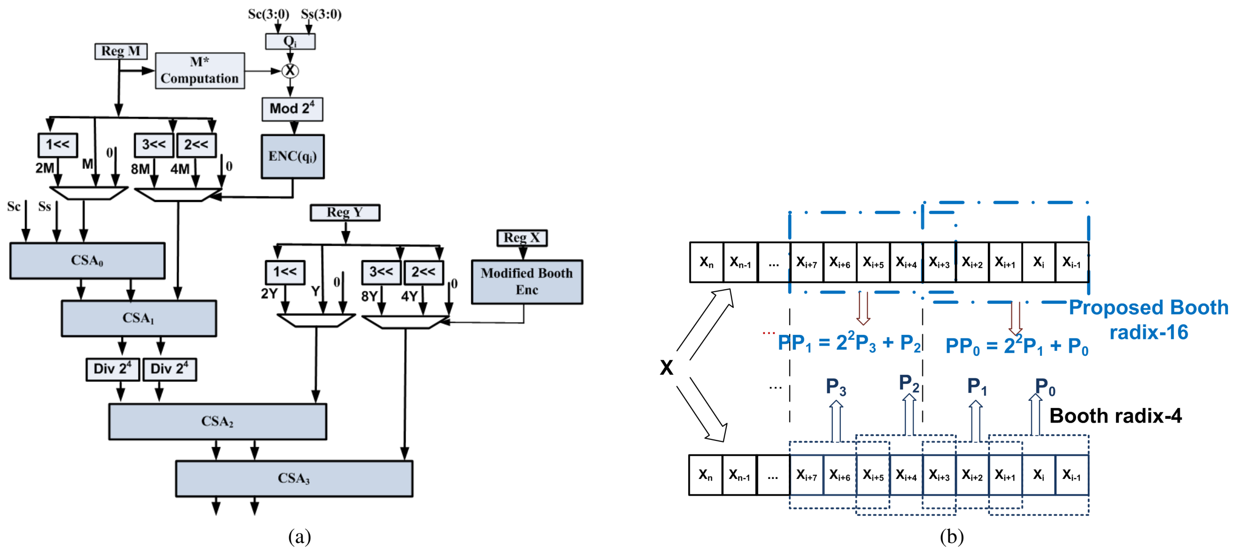

Fig. 1: The proposed architecture ((a) proposed MM, (b) proposed modified Booth encoding)

of the algorithm 2.

Secondly, an efficient computation of q_iM is proposed. The q_i is computed at step 4 of the algorithm 1, where the coefficient q_i is the result of multiplication modulo 2^4. Consequently, the value of coefficient q_i is the set of $(0, 1, ..., 14, 15)$. Therefore, $4n$ FAs are required to compute the q_iM. To reduce the complexity of computation q_iM, we represent digit set of q_i by $(-7, -6, ..., -1, 0, 1, ..., 8)$. The computation requirement of the modified q_iM is only $3n$ FAs. However, the computation difficulty of odd numbers $(\pm 3M, \pm 5M, \pm 7M)$ still exists. To overcome this problem, the coefficient q_i is divided into T_1 and T_0, with $q_{enc} = ENC(q_i) = 2^2T_1 + T_0$, where $T_1, T_0 \in (\pm 2, \pm 1, 0)$. The proposed encoding decreases the computation complexity of q_iM from $3n$ FAs to n FAs and two $(n+4) - bit$ 2's-complement conversions. The proposed q_i encoding is applied in step 5 and 6 in the algorithm 2.

III. Hardware architecture and experiment results

A straight forward hardware architecture of the algorithm 2 is illustrated in Fig.1a. The partial products of $q_{enc}M$ and proposed B_iY are simply implemented by 3, 2 or 1 left bit shift operation. The negative values which are represented in two's-complement are obtained from positive ones by inverting and adding 1 to inverted sequence bits.

To avoid the large carry-propagation, the carry-save addition is adopted like many previous designs [5], [2]. The sign extension in CSA form may issue some errors. Therefore, the 4-bit sign extension is adopted [6] to ensure the correctness of sign extension. Total of four carry-save adders and four 2's-complement conversions are used for the computation of the inner loop of the proposed algorithm.

The proposed algorithm is evaluated on Virtex7 XC7VLX330T-3. The synthesis result shows that the proposed architecture obtains both high throughput and low cost in comparison to design [5] and [2].

TABLE I: Comparison the 2048-bit Montgomery Multiplication on Virtex7 XC7VLX330T-3

	LUT	FF	Period	Cycles	Time	Area*Time
			(ns)		(μs)	(FF+LUT)*T
[5]	33,734	10,315	4.81	518	2.49	109.68
[2]	36,238	15,066	4.57	522	2.39	122.61
Our work	32,744	6,938	3.63	518	1.88	75.66

IV. Conclusion

In this paper, we propose an efficient hardware implementation of radix-16 Montgomery multiplication. The proposed Booth-16 and q_i encoding decrease the computation complexity of MM significantly. The resource of the proposed MM is reduced by 9.9% resource utilization (LUT+FF) while throughput is speeded up 1.32 times as high as that of the design in [2]. Future work will extend for the general case of higher-radix Montgomery multiplication and improve the coefficient q_i conversion.

References

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

[2] S. S. Erdem, T. Yank, and A. elebi, "A general digit-serial architecture for montgomery modular multiplication," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 1658–1668, May 2017.

[3] Jin-Hua Hong and Cheng-Wen Wu, "Cellular-array modular multiplier for fast rsa public-key cryptosystem based on modified booth's algorithm," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 11, pp. 474–484, June 2003.

[4] T. Blum and C. Paar, "High-radix montgomery modular exponentiation on reconfigurable hardware," *IEEE Transactions on Computers*, vol. 50, pp. 759–764, July 2001.

[5] J.-S. Pan, "Efficient digit-serial modular multiplication algorithm on fpga," *IET Circuits, Devices and Systems*, vol. 12, pp. 662–668(6), September 2018.

[6] A. F. Tenca, S. Park, and L. A. Tawalbeh, "Carry-save representation is shift-unsafe: the problem and its solution," *IEEE Transactions on Computers*, vol. 55, pp. 630–635, May 2006.