Design of a Scalable RSA and ECC Crypto-Processor

Ming-Cheng Sun, Chih-Pin Su, Chih-Tsun Huang and Cheng-Wen Wu
Laboratory for Reliable Computing (LaRC)
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 30013, ROC

Abstract—In this paper, we propose a scalable word-based crypto-processor that performs modular multiplication based on modified Montgomery algorithm for finite fields $GF(P)$ and $GF(2^m)$. The unified crypto-processor supports scalable keys of length up to 2048 bits for RSA and 512 bits for elliptic curve cryptography (ECC). Further extension of the key length can be done easily by enlarging the memory module or using the external memory resource. With the proposed parity prediction technique, our pipelined crypto-processor achieves a 512-bit RSA encryption rate of 276 Kbps and a 160-bit ECC encryption rate of 73.3 Kbps for a 220MHz clock rate.

I. Introduction

With the rapid advance in communication technology, more and more applications such as e-commerce and wireless networking are appearing. Protecting the sensitive information when transmitted along the insecure communication channel has become essential. Various cryptography systems have been investigated to prevent the information from snooped. Public-key cryptography, such as RSA algorithm [1], Elliptic Curve Cryptography (ECC) [2,3], DSA and Diffie-Hellman (DH) key exchange algorithm [4], plays a vital role in modern security system.

Most of the public key cryptography relies heavily on the finite field or modular multiplication which is the crucial part for high performance hardware for system applications, such as VPN (Virtual Private network), SSL (Secure Socket Layer), etc. In 1985, Montgomery proposed a modular multiplication algorithm to avoid iterative divisions, which is suitable for VLSI implementation [5]. Further improvement and modification of Montgomery algorithm can be found in [6,7]. Most of the conventional works mainly focused on the ASIC design. The efficient systolic array architectures for specific operand size have been investigated. As the key size of the cryptography growing with the demand of security system, the ASIC implementations suffer from the hardware complexity. Recently, scalable architectures for modular multiplication have been considered [8, 9] to trade off between performance and area overhead. A scalable multiplier architecture for finite field $GF(P)$ and $GF(2^m)$ was proposed in [10], which supports the basic multiplication for both ECC and RSA within the same hardware module.

In this paper, we propose a scalable crypto-processor for both RSA and ECC. Our pipelined dual-field crypto-module supports finite field multiplication over both $GF(P)$ and $GF(2^m)$. A parity prediction technique is presented to compensate pipeline stalls by the data dependency from the original algorithm, which simplifies the controller's design and further speeds up the computation of the finite field and modular multiplication. In addition, efficient dual-field adder and subtractor are used to accomplish modified Montgomery multiplication for the support of both RSA and ECC operations.

II. Word-based Multiplication Architecture with Parity Prediction

Let $P = (P^{w-1}, P^{w-2}, \dots, P^0)_r$ be a large prime number with w digit in radix r, where $r = 2^k$, and k is the word width. Let $B = (B^{w-1}, B^{w-2}, \dots, B^0)_r$ and $A = (A_{m-1}, A_{m-2}, \dots, A_0)_2$ are two large integers, which satisfy $m = wk$ and $0 \le A, B < P$. Let B_i^j be the i^{th} bit in the j^{th} word of B. For simplicity, $B_{i\dots k}^j$ denotes a series of bits from i^{th} to k^{th}in the j^{th} word of B. The word-based radix-2 Montgomery algorithm over $GF(P)$ is shown in Fig. 1(a), using the carry-save form to represent the intermediate results. Similarly the word-based Montgomery algorithm over $GF(2^m)$ is shown in Fig. 1(b), in which $A(x), B(x) \in GF(2^m)$ and $P(x)$ is the irreducible polynomial.

In Fig. 1(a), $\{C_{out}, C^0\} = \{C^0, 0\}$ represents a bitwise left-shift of C^0, and C_{out} is the one-bit carry digit. The algorithm performs shifting instead of conventional division, resulting in a fast implementation for modular multiplication. However, due to the data dependency of the algorithm shown in Fig. 2(a), the processing unit (PU) which computes $C^j + S^j + a_i \cdot B^j$ has to wait until the result of $C^{j+1} + S^{j+1} + a_{i-1} \cdot B^{j+1} + parity \cdot P^{j+1}$ is generated. As a result, an extra pipeline stall exists between each column of PUs, which degrades the performance [10]. An extra constant 2^{-m} will be introduced after the Montgomery multiplication stage, i.e., $M = A \cdot B \cdot 2^{-m} \mod P$. In addition, M is in the range of $[0, 2P)$. A final reduction is thus required to ensure that $0 \le M \le P$. Therefore, improved Montgomery algorithm with two's complement numbers [7] is used in our design to prevent pipeline stalls by the sign digits, which is discussed in the following section.

A *parity prediction module* (PPM) is implemented to predict the *parity*, (i.e., $S_{T,0}^{(t)}$ in Fig. 1) and to compensate the pipeline stall, where the time instance is denoted as the superscript with

```
$WMM_p(A,B,P)$ // $A,B \in GF(P)$
{   $C^0 = S^0 = 0, C_{out} = 0$;
    for( $i = 0$ to $(m-1)$) { // modular multiplication
        $(C_T^0, S_T^0) = a_i \cdot B^0 + C^0 + S^0$;
        $\{C_{out}, C_T^0\} = \{C_T^0, 0\}$;
        $parity = S_{T,0}^{(0)}$;
        $(C^0, S^0) = parity \cdot P^0 + C_T^0 + S_T^0$;
        $S_{w-2...0}^{(0)} = S_{w-1...1}^0$;
        for( $j = 1$ to $(w-1)$) {
            $(C_T^j, S_T^j) = a_i \cdot B^j + C^j + S^j$;
            $\{C_{out}, C_T^j\} = \{C_T^j, C_{out}\}$;
            $(C^{(j)}, S^{(j)}) = parity \cdot P^{(j)} + C_T^j + S_T^j$;
            $\{S_{w-2...0}^j, S_{w-1}^{j-1}\} = \{S^j\}$;
        }
    }
    if ($A \cdot B > 0$) { // final adjustment
        for( $i = 0$ to $(w-1)$) {
            $(C^i, S^i) = C^i + S^{(i)} - P^i$;
        }
    return $M = (C + S)$;
}
                                (a)
$WMM_{2^m}(A(x), B(x), P(x))$
{   $M(x) = 0$;
    for( $i = 0$ to $(m-1)$) // finite-field multiplication
        $M^0(x) = M^0(x) + a_i \cdot B^0(x)$;
        $parity = M_0^0(x)$;
        $M^0 = M^0 + parity \cdot P^0$;
        $M_{w-2...0}^0 = M_{w-1...1}^0$;
        for( $j = 1$ to $(w-1)$) {
            $M^j(x) = M^j(x) + a_i \cdot B^j(x)$;
            $M^j(x) = M^j(x) + parity \cdot P^j(x)$;
            $\{M_{w-2...0}^j, M_{w-1}^{j-1}\} = \{M^j(x)\}$;
        }
    return $M(x)$;
}
                                (b)
```

Fig. 1. The word-based radix-2 Montgomery multiplication algorithms over (a) $GF(P)$ and (b) $GF(2^m)$.

parentheses and the bit position is denoted as the subscript. For example, $S_{T,0}^{(t)}$ is the least significant bit (LSB) of S_T generated at time instance t. From the data dependency shown in Fig. 3, $S_{T,0}^{(t)} = S_1^{(t-1)} \oplus C_0^{(t-1)} \oplus (b_0 a_i)$. Since $C_0^{(t-1)} = P_0 \cdot S_{T,0}^{(t-1)}$ and $P_0 = 1$ (because P is a prime), $C_0^{(t-1)} = S_{T,0}^{(t-1)}$. In addition, $S_1^{(t-1)} = C_{T,0}^{(t-1)} \oplus S_{T,1}^{(t-1)} \oplus P_1$. Therefore, $S_{T,0}^{(t)} = S_1^{(t-1)} \oplus C_0^{(t-1)} \oplus b_0 a_i = (C_{T,0}^{(t-1)} \oplus S_{T,1}^{(t-1)} \oplus P_1) \oplus S_{T,0}^{(t-1)} \oplus b_0 a_i$, where $C_{T,0}^{(t-1)}$ and $S_{T,1}^{(t-1)}$ can be computed using the outputs at time instant $(t-2)$ and $S_{T,0}^{(t-1)}$. Combining the parity $S_{T,0}^{(t-1)}$ at time $(t-2)$, we can generate the parity $S_{T,0}^{(t)}$ at time $(t-1)$, which can then be applied immediately at time (t). The resultant data flow graph is then shown in Fig. 2(b), where the add-on functional block Z represents the parity prediction module. Thus the pipeline stall can be eliminated.

To realize the function of Y in Fig 2(c), our PU consists of a dual-field adder (DFA) array for both $GF(P)$ and $GF(2^m)$, a sign-bit generator (SG) and a PPM. Figure 4 shows the circuit

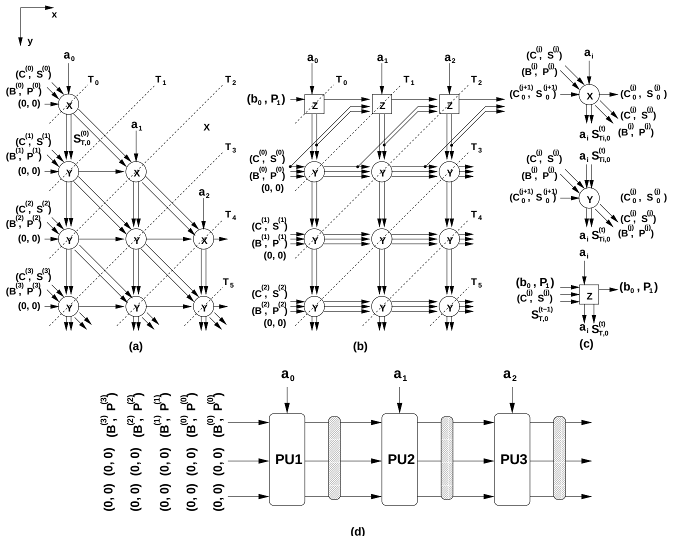

Fig. 2. Data flow graph of word-based modular multiplication (a) without parity prediction; (b) with parity prediction. (c) The signal description of each PU and (d) the block diagram of the projected architecture.

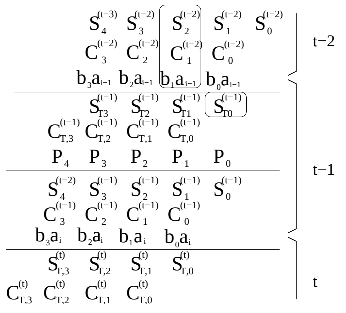

Fig. 3. The data dependency of parity $S_{T,0}^{(t-1)}$ and $S_{T,0}^{(t)}$.

of a PU without the PPM circuitry. The DFA array is based on the work in [10]. The hardware architecture of our crypto-module is shown in Fig. 5, which is obtained from Fig. 2(d) using the projection vector $\vec{d} = (0,1)$.

The PPM generates a new parity whenever the least significant word (LSW) of B is processed, and latches the parity until the most significant word (MSW) is done. The SG is enabled ($enable = 1$) when the MSW of B is processed, performing the proper sign extension. The function is summarized as follows:

Input		Output		
$C_{k-1,in}^{(w-1)}$	$S_{k-1,in}^{(w-1)}$	s_{out}	$C_{k-1,out}^{(w-1)}$	$S_{k-1,out}^{(w-1)}$
0	0	s_{in}	0	0
0	1	s_{in}	0	1
1	0	s_{in}	1	0
1	1	0	0	0

Whenever a new modular multiplication is started, operand B will be applied sequentially from the LSW to the MSW. $B^{(0)}$ will be applied two times so that the PPM in PU$_1$ will generate the parity. From the 2nd to w^{th} clock cycles, the PUs process the data in sequence. This procedure will proceed until the LSW of B reaches PU$_n$. If the number of PUs, n, is large than

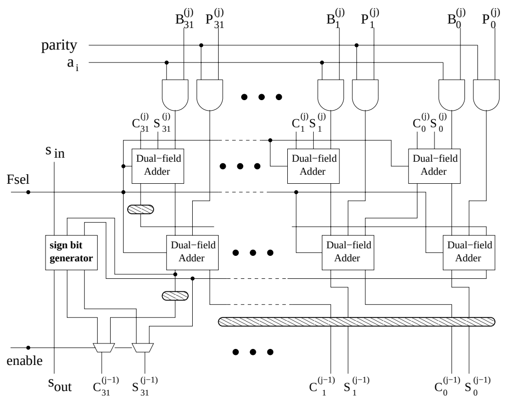
Fig. 4. The block diagram of the PU.

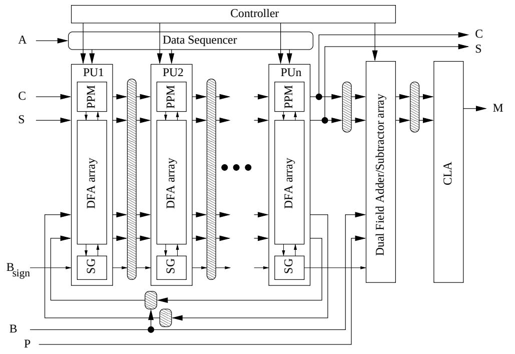
Fig. 5. The architecture of the scalable word-based dual-field crypto-module.

w, the output of PU$_n$ can be fed back immediately to PU$_1$, otherwise an extra buffer is required to store the temporary data until PU$_1$ is available. The total computation time, C_C (clock cycles), is summarized as follows,

$$C_C = \begin{cases} (w+1)+(n-1)+(\lceil \frac{m}{n} \rceil - 1)\cdot n, & \text{if w} \le \text{n}, \\ (w+1)+(n-1)+(\lceil \frac{m}{n} \rceil - 1)\cdot w, & \text{otherwise}, \end{cases}$$

Using the proposed parity prediction, the total computation time can be reduced by $(\lceil \frac{m}{n} \rceil - 1)\cdot n + (n-1)$ clocks when $w \le n$ and by $(n-1)$ clocks otherwise, as compared with that in [10, 11]. The area overhead of the PPM and SG is approximately 4% as compared with DFA array.

In addition to the PUs, an extra stage is needed to ensure that the result is within the range of $(-P,P)$. The relation between the output of PU$_n$, O, and the final result, R, is given as

$$R = \begin{cases} O-P, & \text{if } A>0, B>0, \\ O-B, & \text{if } A<0, B>0, \\ O, & \text{if } A>0, B<0, \\ O+B-P, & \text{if } A<0, B<0. \end{cases}$$

Such part of function can be easily implemented by a dual-field adder/subtractor.

The function of final adjustment can be extended to support the finite field addition and subtraction both for $GF(P)$ and $GF(2^m)$, which are also the basic operations to compute ECC. To ensure that the result of two's complement addition and subtraction is within the range of $(-P,P)$, the result R with respect to the given operand A and B has the following adjustment,

$$R = \begin{cases} A+B-P, & \text{if } A>0, B>0, \\ A+B, & \text{if } A>0, B<0, \\ A+B, & \text{if } A<0, B>0, \\ A+B+P, & \text{if } A<0, B<0, \end{cases}$$

for addition over $GF(P)$. Similarly for subtraction over $GF(P)$, the result R will be

$$R = \begin{cases} A-B, & \text{if } A>0, B>0, \\ A-B-P, & \text{if } A>0, B<0, \\ A-B+P, & \text{if } A<0, B>0, \\ A-B, & \text{if } A<0, B<0. \end{cases}$$

Fig. 6 shows the block diagram of the dual-field adder/subtractor array. To support different arithmetic operations, the multiplexers are implemented to select proper operands for the DFA array. Note that since P is a prime, $P_0 = 1$ and the LSB of its two's complement is also 1. Similarly $c_{out} = 1$ when subtracting by B. The addition and subtraction over $GF(2^m)$ is simply the bitwise exclusive-OR operation. In addition, a pipelined carry lookahead adder (CLA) is used to convert the result of modular addition, subtraction or multiplication from redundant carry-save form to the irredundant form.

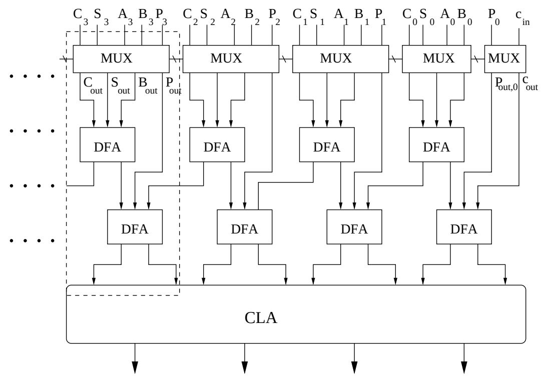
Fig. 6. The circuit of Dual-Field Adder/Subtractor array.

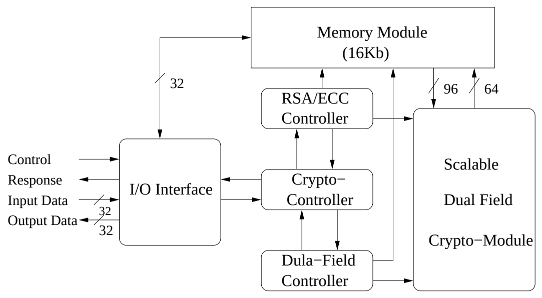

Fig. 7. The overall architecture of our crypto-processor.

III. The Crypto-processor Architecture

The overall architecture of the crypto-processor core is shown in Fig. 7. There is an I/O interface for transferring the data from and to the on-chip bus with standard protocol. Therefore, the crypto-processor can be easily plugged into a system chip. The crypto-controller manages the information exchange from the I/O interface and the different cryptographic

TABLE I
COMPARISONS ON AREA AND PERFORMANCE (512-BIT RSA).

	Year	Gate Count	# Clock Cycles	Tech.	Clock	Baud Rate	NB	NB /Gate Count	Scalable Key Length
Yang [6]	1998	74K	390K	0.6μm	125M	164K	286K	3.9	N
Su [7]	1999	76K	510K	0.6μm	100M	100K	174.8K	2.3	N
Hong [12]	2002	77K	530K	0.6μm	300M	289K	512K	6.7	N
Hsieh (8-bit) [13]	1999	4.5K	6.5M	0.6μm	125M	10.5K	18.6K	2.37	Y
Lin (16-bit) [8]	2001	13.1K	810K	0.35μm	125M	79K	79K	6	Y
Ours (32-bit)	2002	40K	405K	0.35μm	220M	276K	276K	6.9	Y

operations of both RSA and ECC. When a cryptographic process begins, the controller will access the necessary multiplicand, multiplier and the prime or irreducible polynomial into the memory module. Proper microinstructions are generated from the crypto-controller and fed into the RSA/ECC controller. Then the RSA/ECC controller will access the dual-field crypto-module for proper data flow. The RSA/ECC controller also assigns each memory block as a read or write buffer during the encryption and decryption. The dual-field controller selects either $GF(P)$ or $GF(2^m)$ arithmetic operations in the crypto-module. There are 16 PUs with 32-bit word in the crypto-module. The memory module consists of $2048Kb \times 6$ two-port memory blocks as the register files to store the intermediate codeword. Additional $2048Kb \times 2$ FIFOs are used as the buffer to store the temporary data when the key length is greater than 512 bits. The key length is scalable by 32-bit words. As a result, the overall crypto-processor is capable of processing 2048-bit RSA and 512-bit ECC cryptography. However, it is extensible simply with a larger memory module, or using external memory resource as the buffer.

IV. COMPARISONS

Table I compares different designs of the 512-bit RSA cryptography with normalized clock rate and baud rate, where the NB represents the normalized baud rate with respect to $0.35\mu m$ technology. The first three ASIC designs are systolic array design, while the design in [12] requires no broadcasting signal and achieves the best NB per gate for RSA. The last three designs are processor-based implementations. Our crypto-processor achieves the highest performance with the measurement of the NB per gate, regardless the scalability and the maximum key length that are the outperformance of the proposed design. The NB is 276 Kbps by a standard cell-library design flow using $0.35\mu m$ technology with 40K gates and a 220MHz clock rate from synthesis result. In addition, the pipelined crypto-processor is unified for both RSA and ECC with scalable key length. For ECC computation, projective coordinates are used to reduce the requirement of the modular inversion in affine coordinates. The resultant baud rate of 160-bit ECC is 73.3 Kbps for $GF(P)$ and 65.9 Kbps for $GF(2^m)$.

V. CONCLUSIONS

We have presented a new scalable and unified crypto-processor based on a modified Montgomery algorithm for both RSA and ECC. Effective pipeline architecture is implemented to perform the modular multiplication, addition and subtraction over $GF(P)$ and $GF(2^m)$, which are the basic operations in RSA and ECC. The word-based crypto-processor supports scalable keys of the length up to 2048 bits for RSA and 512 bits for ECC. The key length can be increased easily with a larger memory module or using external memory resource, without affecting the overall architecture. Using a $0.35\mu m$ CMOS technology, our crypto-processor achieves a 512-bit RSA encryption rate of 276 Kbps and a 160-bit ECC encryption rate of 73.3Kbps for a 220MHz clock rate.

REFERENCES

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[2] N. Koblitz, "Elliptic curve cryptosystems", in *Mathmatics of Computation*, 1987, pp. 203–209.

[3] V. S. Miller, "Use of elliptic curve in cryptography", in *Advances in Cryptology—Crypto'85 Proceedings*, 1986, pp. 417–426.

[4] W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Trans. Information Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[5] P. L. Montgomery, "Modular multiplication without trial division", *Math. Computation*, vol. 44, no. 7, pp. 519–521, 1985.

[6] C.-C. Yang, T.-S. Chang, and C.-W. Jen, "A new RSA cryptosystem hardware design based on Montgomery's algorithm", *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 7, pp. 908–913, July 1998.

[7] C.-Y. Su, S.-A. Hwang, P.-S. Chen, and C.-W. Wu, "An improved Montgomery algorithm for high-speed RSA public-key cryptosystem", *IEEE Trans. VLSI Systems*, vol. 7, no. 2, pp. 280–284, June 1999.

[8] Y.-C. Lin, "A word-based RSA public-key crypto-processor core for IC smart card", Master thesis, Dept. Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan, June 2001.

[9] A. F. Tenca, G. Todorov, and Ç. K. Koç, "High-radix design of a scalable modular multiplier", in *Cryptographic Hardware and Embedded Systems (CHES) 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. 2001, number 2162 in LNCS, pp. 189–205, Springer-Verlag.

[10] E. Savaş, A. F. Tenca, and Ç. K. Koç, "A scalable and unified multiplier architecture for finite fields $GF(p)$ and $GF(2^m)$", in *Cryptographic Hardware and Embedded Systems (CHES) 2000*. 2000, LNCS, pp. 281–296, Springer-Verlag.

[11] A. F. Tenca and Ç. K. Koç, "A scalable architecture for Montgomery multiplication", in *Cryptographic Hardware and Embedded Systems (CHES) 1999*. 1999, LNCS, pp. 94–108, Springer-Verlag.

[12] J.-H. Hong and C.-W. Wu, "Cellular array modular multiplier for the RSA public-key cryptosystem based on modified Booth's algorithm", *IEEE Trans. VLSI Systems*, 2002 (accepted).

[13] Y.-H. Hsieh, "Design and implementation of an RSA encryption/decryption processor on IC smart card", Master Thesis, Dept. Electrical Engineering, National Taiwan University, Taipei, Taiwan, June 1999.