This article was downloaded by:[Sakiyama, K.]
On: 28 August 2007
Access Details: [subscription number 779821987]
Publisher: Taylor & Francis
Informa Ltd Registered in England and Wales Registered Number: 1072954
Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK

International Journal of Electronics

Publication details, including instructions for authors and subscription information:

<http://www.informaworld.com/smpp/title~content=t713599654>

Reconfigurable modular arithmetic logic unit supporting high-performance RSA and ECC over GF(p)

Online Publication Date: 01 May 2007

To cite this Article: Sakiyama, K., Mentens, N., Batina, L., Preneel, B. and Verbauwhede, I. (2007) 'Reconfigurable modular arithmetic logic unit supporting high-performance RSA and ECC over GF(p)', International Journal of Electronics, 94:5, 501 - 514

To link to this article: DOI: 10.1080/00207210701293264

URL: <http://dx.doi.org/10.1080/00207210701293264>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.informaworld.com/terms-and-conditions-of-access.pdf>

This article maybe used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

© Taylor and Francis 2007

Reconfigurable modular arithmetic logic unit supporting high-performance RSA and ECC over $GF(p)$

K. SAKIYAMA*, N. MENTENS, L. BATINA,
B. PRENEEL and I. VERBAUWHEDE

Department Electrical Engineering – ESAT/SCD-COSIC,
Katholieke Universiteit Leuven/IBBT, Kasteelpark Arenberg 10,
B-3001, Leuven-Heverlee, Belgium

(Received 15 May 2006; in final form 14 November 2006)

This paper presents a reconfigurable hardware architecture for public-key cryptosystems. By changing the connections of coarse grain carry-save adders (CSAs), the datapath provides high performance modular operations that can be used for both RSA and elliptic curve cryptography (ECC). In addition, we introduce reconfigurable flip-flops in order to make an optimal choice of hardware resources. The proposed datapath is implemented with a 0.25- μm complementary metal oxide semiconductor (CMOS) technology and on a field programmable gate array (FPGA). We compare the performance of modular exponentiation for RSA and scalar multiplication for ECC based on the prototype implementation. The results show that higher performance is obtained for ECC on the same hardware platform.

Keywords: Public-key cryptosystem; RSA; Elliptic curve cryptography (ECC); FPGA implementation; Reconfigurable architecture

1. Introduction

The idea of public-key cryptography was introduced in the mid 1970s (Diffie and Hellman 1976). They showed that one can eliminate the need for prior agreement of a key in order to exchange some confidential data. Public-key cryptosystems also enable digital signatures. The best-known and most commonly used public-key cryptosystems are RSA and elliptic curve cryptography (ECC). The RSA public-key cryptosystem is named after its inventors Rivest *et al.* (1978). ECC is based on a different algebraic structure (Miller 1985, Koblitz 1987). In the case of ECC, the group used is the group of points on an elliptic curve. It is important to point out that ECC offers the same level of security as RSA for much smaller key sizes.

The contribution in this paper deals with an architectural solution for a reconfigurable datapath that is used for RSA and ECC over a field of a prime characteristic. The design is further evaluated with ASIC and FPGA implementations. The proposed reconfigurable datapath can achieve arbitrary

---

*Corresponding author. Email: Kazuo.Sakiyama@esat.kuleuven.be

precision up to 2048 bits, hence easily bridging the gap between the bit-lengths for ECC from 160 bits to 2048 bits long moduli for RSA. Modular multiplications are used based on Montgomery's method without any modular reduction which is also beneficial for side-channel attacks.

The results show that the proposed reconfigurable datapath is indeed a suitable solution for high-performance public-key cryptosystems, such as RSA and ECC. Comparing the two with the same hardware resources and with corresponding bit-lengths that provide similar security, we found that ECC-256p allows higher performance than RSA-2048. This research is of interest because, due to a constant progress in cryptography and security applications, an alternative solution for public-key services, such as signatures, key-distribution etc. is needed.

This paper is organized as follows. Section 2 lists some relevant previous work. Some mathematical background is explained briefly in §3. In §4, the details of our architecture are given. The main contribution of our work i.e. the reconfigurable datapath is explained in §5. The implementation results are given in §6. Section 7 concludes the paper.

2. Related work

This section reviews some of the most relevant previous work in hardware implementations for RSA and ECC. To consider both RSA and ECC on the same platform has only recently become more popular, since ECC has proven to be a mature technology. Some of the work is done on FPGAs and only very few implementations are presenting an ASIC implementation of ECC in the field of prime characteristic.

More recent work on hardware implementation of RSA includes the work proposed in McIvor *et al.* (2003). They use carry save adders (CSAs) to perform the large word length additions required for Montgomery modular multiplication (MMM). The obtained performance for one 1024-bit RSA decryption on the Xilinx Virtex-2 board was 2.63 msec. Satoh and Takano (2003) present a dual field multiplier with the best performance so far in both binary and prime fields. The throughput of an elliptic curve scalar multiplication is maximized by use of the MMM and an on-the-fly redundant binary converter. The biggest advantage of their design is in scalability in operand size and also flexibility between speed and hardware area. The work by Tenca and Koç (2003) also introduces a scalable architecture for the computation of modular multiplication, based on the MMM. Their proposed multiplier works with any precision of the input operands, limited only by memory or control constraints. Andres *et al.* (2005) improve the version of the Tenca-Koç Montgomery multiplier and achieve half the latency and half the queue memory requirement.

3. Mathematical background

In this section, we give some mathematical background for the RSA cryptosystem and ECC over a prime field.

3.1 RSA

In a public-key cryptosystem, as introduced by Diffie and Hellman (1976), each user owns a pair of keys, the private and the public key. The private key of a user in this case consists of two large primes p and q and an exponent d . The public key consists of a pair (N, e) , where $N = p \cdot q$ is the modulus (at least 1024 bits) and the exponent e is such that

$$e = d^{-1} \bmod \lambda(N) \quad (1)$$

The corresponding p , q and d are kept secret. To encrypt a message M , the user computes $C = M^e \bmod N$ and decryption is described by $M = C^d \bmod N \equiv M^{1+k\varphi(N)} \equiv M \bmod N$. The previous equality follows from Fermat's theorem and the fact that $\lambda(N) = \text{lcm}(p-1, q-1)$ is a divisor of $\varphi(N) = (p-1)(q-1)$. The RSA encryption is the modular exponentiation with the public exponent e and the private exponent d is used for decryption.

Here, we explain the CRT implementation of RSA, which is an efficient way to reduce the work factor of the decryption process. The use of CRT for RSA was proposed in 1982 by Quisquater and Couvreur (1982). Namely, by means of the Chinese remainder theorem (CRT), the speed for the RSA decryption scheme can be increased up to 4 times (Koblitz 1994). This possibility is very attractive for practical applications. However, it includes some pitfalls on security, so it has to be carefully implemented.

The two most straightforward algorithms to implement modular exponentiation are given in Algorithm 1, where G is a finite abelian group and e is a positive integer.

Algorithm 1: Algorithms for left-to-right and right-to-left binary exponentiation.

Require: $g \in G$, $e = (e_{n-1}e_{n-2} \cdots e_1e_0)_2$

Ensure: g^e

```

1:  $A \leftarrow 1$ 
2: for  $i$  from  $n-1$  downto  $0$  do
3:    $A \leftarrow A \cdot A$ 
4:   if  $e_i = 1$  then
5:      $A \leftarrow A \cdot g$ 
6:   end if
7: end for
8: Return  $A$ 

```

Require: $g \in G$, $e = (1e_{n-2} \cdots e_1e_0)_2$

Ensure: g^e

```

 $A \leftarrow 1$ ,  $S \leftarrow g$ 
for  $i$  from  $0$  to  $n-1$  do
  if  $e_i = 1$  then
     $A \leftarrow A \cdot S$ 
  end if
   $S \leftarrow S \cdot S$ 
end for
Return  $A$ 

```

The basic operations in both algorithms are multiplications and squarings. To be able to use the same datapath for both operations and also for side-channel issues (Kocher *et al.* 1998) the squarings are not performed on a dedicated squarer, but on the multiplier. Taking into account an expected value of $n/2$ ones in e , the total number of multiplications in both algorithms is $3n/2$. In the left-to-right algorithm, the multiplications have to be performed consecutively requiring one memory location for intermediate values. In the right-to-left algorithm, the multiplications can be parallelized, which doubles the speed. However, the right-to-left algorithm uses two memory locations for intermediate values.

3.2 ECC over $GF(p)$

The main operation in any ECC-based primitive is scalar multiplication. This operation is often performed by some sort of double-and-add algorithm that is using the point group operation (Blake *et al.* 1999). On the other hand, finite field operations in $GF(p)$, such as addition, subtraction, multiplication and inversion are required to perform the group operations.

When E is a curve defined over $GF(p)$ the typical equation is

$$E: y^2 = x^3 + ax + b \quad (2)$$

For an arbitrary point P on a curve E , an inverse of the point $P = (x_1, y_1)$ is $-P = (x_1, -y_1)$. The sum $P + Q$ of points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ (assume that $P, Q \neq \mathcal{O}$, and $P \neq \pm Q$) is point $R = (x_3, y_3)$ where

$$\left. \begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1} \\ x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= (x_1 - x_3)\lambda - y_1 \end{aligned} \right\} \quad (3)$$

For $P = Q$, we get the following, “doubling” formulae:

$$\left. \begin{aligned} \lambda &= \frac{3x_1^2 + a}{2y_1} \\ x_3 &= \lambda^2 - 2x_1 \\ y_3 &= (x_1 - x_3)\lambda - y_1 \end{aligned} \right\} \quad (4)$$

The point at infinity $\mathcal{O}$ plays a role analogous to that of the number 0 in ordinary addition. Thus, $P + \mathcal{O} = P$ and $P + (-P) = \mathcal{O}$ for all points P .

There are many types of coordinates in which an elliptic curve may be represented. In the equations above affine coordinates are used, but so-called projective coordinates have some implementation advantages. More precisely, in this case, the point addition can be done using field multiplications only and almost no inversion required, except only one at the end of a point multiplication. There are many types of projective coordinates that were proposed in the literature. In particular, a weighted projective representation (also referred to as Jacobian representation) is preferred in the sense of faster arithmetic on elliptic curves. In this representation, a triplet (X, Y, Z) corresponds to the affine coordinates

$(X/Z^2, Y/Z^3)$ for $Z \neq 0$. In this case, we have a weighted projective curve equation of the form

$$E: Y^2 = X^3 + aXZ^4 + bZ^6 \quad (5)$$

Weighted projective coordinates provide faster arithmetic than the “normal” projective coordinates. Conversion from projective to affine coordinates costs 1 inversion and 4 multiplications, while *vice versa* is trivial. If one implements addition and doubling as specified in the IEEE standard (IEEE P1363 2000), the total cost for general addition is $1I + 3M$ in affine coordinates and $16M$ in projective coordinates ($11M$ if $Z_1 = 1$, i.e., one point is given in affine coordinates, and the other one in projective coordinates). Here, I and M are denoting the modular inversion and multiplication operations, respectively. In the case of doubling (with $a = p - 3$), this relation is $1I + 4M$ in affine coordinates against $8M$ in projective coordinates. Thus, the choice of coordinates is determined by the ratio $I : M$. Therefore, multiplication in a finite field is the most important operation to focus on when working with projective coordinates. On the other hand, the extra inverter is required for affine coordinates' representation because one inversion has to be performed for every point operation.

Similar to the left-to-right and right-to-left binary algorithms for modular exponentiation, a point multiplication can be performed using Algorithm 2 (Menezes 1993), where P is a point on the elliptic curve and k is a positive integer. The point at infinity $\mathcal{O}$ is the identity element for elliptic curve operations.

Algorithm 2: Algorithm for left-to-right and right-to-left binary point multiplication

Require: $P = (x, y)$,

$$k = (k_{l-1}k_{l-2} \cdots k_0)_2$$

Ensure: $Q = (x', y') = kP$

1: $Q \leftarrow \mathcal{O}$

2: **for** i from $l-1$ down to 0 **do**

3: $Q \leftarrow 2Q$

4: **if** $k_i = 1$ **then**

5: $Q \leftarrow Q + P$

6: **end if**

7: **end for**

8: **Return** Q

Require: $P = (x, y)$,

$$k = (1k_{l-2} \cdots k_0)_2$$

Ensure: $Q = (x', y') = kP$

$$Q \leftarrow \mathcal{O}, S \leftarrow P$$

for i from 0 to $l-1$ **do**

if $k_i = 1$ **then**

$$Q \leftarrow Q + S$$

end if

$$S \leftarrow 2S$$

end for

Return Q

Similar to the modular exponentiation algorithms, the left-to-right algorithm will be used when the storage of intermediate values is the bottleneck, while the right-to-left algorithm will be used for higher speed when the datapath allows parallelism.

The point operations in Algorithm 2 are point additions and point doublings. In our case, a point addition and a point doubling, respectively, consist of 14 and 21 multiply/add operations by the MALU in the underlying finite field. Therefore, the total number of multiplications for point multiplication is estimated as $(49l)/2$. However, as we allocate two MALUs for the ECC case, the number becomes $2l$ by processing point additions and doublings in parallel.

4. Modular arithmetic logic unit

4.1. Datapath of the MALU

The proposed architecture is an MMM with digit-serial multiplications (Algorithm 1). Four-to-two (4-2) CSAs (figure 1a) are used in the hardware implementation because they are considered as one of the most optimal solutions for a multi-operand addition.

The cell, a column of the datapath of the MALU uses d sets of 4-2 CSAs (figure 1b), i.e., the inputs and outputs of the cell are presented in 2-bit CS-form during the operation. Therefore, the cell needs $2d$ sets of FAs. The critical path of the datapath is estimated based on the critical path delay of the cell as follows.

$$T_{4-2CSAs} = 2dT_{FA} \quad (6)$$

Here, we assumed that the delay for the sum and carry calculations are the same. The propagation of $s_{i,j}$ goes through d sets of the cell and uses two FAs in every cell. The right-most cell, $\text{cell}(i,0)$ provides the m_i vector for the rest of the cells. As expressed in Equation (7), the path for generating a bit of m_i only consists of a 3-input XOR in the right-most cell.

$$\left. \begin{aligned} m_i[0] &= vs_{i,0} \oplus vc_{i,0} \oplus x_i[0]y_0 \\ m_i[1] &= s_{i,0}[1] \oplus c0_{next}[0] \oplus x_i[1]y_0 \\ &\vdots \\ m_i[d-1] &= s_{i,0}[d-1] \oplus c0_{next}[d-1] \oplus x_i[d-1]y_0 \end{aligned} \right\} \quad (7)$$

The worst combination of the paths through the logic generating m_i results in

$$dT_{m_i} + dT_{FA} \quad (8)$$

This path delay is assumed to be equivalent to or shorter than $T_{4-2CSAs}$. As can be seen from the hardware configuration and the delay calculations, the datapath of the MALU has an area and delay that can be adapted with the size of d . In this way, the propagation can be tuned to the speed of the system. The proposed array is flexible regarding the size of d ; a trade-off can be made between performance and cost.

Figure 1. Configuration of the MALU with 4-2 CSAs.

4.2 Functionality of the MALU

Before explaining the general case, the main functionality of the MALU is explained with the case that $d=1$. In this configuration, each cell is composed of one 4-2 CSA (figure 1a). The 4-2 CSA sums up the four-bit inputs xy, mn, s and c and outputs two bits in the redundant CS-form whose value is $2(c_{next}) + s_{next}$ where s and c are the virtual sum and carries. The bit multiplications xy and mn are the main inputs for computing the bit level of the Montgomery multiplication in Algorithm 1, i.e., $(T + xy + mn)/2$.

Algorithm 3: Algorithm for d -digit serial montgomery modular multiplication over $GF(p)$ without final subtraction.

Require: $N = (n_{k-1} \cdots n_1 n_0)_2$, $X = (x_k \cdots x_1 x_0)_2$, $Y = (y_k \cdots y_1 y_0)_2$
with $x, y \in [0, 2N - 1]$, $R = 2^{k+2}$, $\gcd(N, 2) = 1$

Ensure: $T = XYR^{-1} \bmod 2N$

1. $T \leftarrow 0$
2. **for** i **from** 0 **to** $k+1$ **do**


```

3.   $m_i \leftarrow t_0 \oplus x_i y_0$ 
4.   $T \leftarrow (T + x_i y + m_i N)/2$  // addition stage
5.   $m_{i+1} \leftarrow t_0 \oplus x_{i+1} y_0$ 
6.   $T \leftarrow (T + x_{i+1} y + m_{i+1} N)/2$  // addition stage
     $\vdots$ 
1+2d.  $m_{i+d-1} \leftarrow t_0 \oplus x_{i+d-1} y_0$ 
2+2d.  $T \leftarrow (T + x_{i+d-1} y + m_{i+d-1} N)/2$  // addition stage
3+2d. end for
4+2d. Return  $T$ 

```

Simply thinking, a multiplication can be computed with $(k + \alpha)^2$ times 4-2 CSA operations if the multiplicand and multiplier have $(k + \alpha)$ bit. However, considering that there are no carry propagations in the j -direction shown on figure 1, it is natural to allocate $(k + \alpha)$ sets of cells in the j -direction to take the speed merit. This CSA array is defined as the minimal configuration of our proposed MALU. The connection of the CSA arrays in the i -direction are determined by the bit weights of the CSA's outputs (numbers in parenthesis in figure 1a, b) and the division of the bit-level Montgomery algorithm (1-bit right-shift). The connection is latched with $(2k + 2\alpha - 1)$ sets of F/Fs for virtual carries.

The explanation of the MALU for a general d is given as follows. As illustrated in figure 1(c), the introduced MALU with 4-2 CSAs has four types of input vectors, $X = (x_g \cdots x_1 x_0)_{2^d}$, $Y = (y_{k+1} \cdots y_1 y_0)_2$, $N = (n_{k-1} \cdots n_1 n_0)_2$, and $S = (s_{h, k+\alpha-1} \cdots s_{1, k+\alpha-1} s_{0, k+\alpha-1})_{2^d}$ where $g = \lceil (k+1)/d \rceil$ and $h = \lceil k/d \rceil$. Here, X is the multiplier, Y is the multiplicand and N is the modulus. The augend vector S is provided to the MALU by d bits in every cycle and eventually added to the result of the modular multiplication of X and Y (modulo N). The intermediate results are stored in $VS = (vs_{i, k+\alpha-1} \cdots vs_{i, 1} vs_{i, 0})_2$ and $VC = (vc_{i, k+\alpha-1} \cdots vc_{i, 1} vc_{i, 0})_2$. They are reset to zero when a modular multiplication starts to execute ($i=0$). After finishing a Montgomery multiplication, the result is output from the right-most cell by d bits in every cycle as $Sout = (sout_g \cdots sout_1 sout_0)_{2^d}$.

The MALU has two independent stages for $GF(p)$ operation. One is the Carry-Save(CS)-stage that implements the Montgomery algorithm in a CS-form. Another converts the CS-form integer into a normal integer by propagating carries, namely the Carry-Propagate(CP)-stage. Moreover, the CP-stage is capable of adding/subtracting S to/from the result of the CS-stage. When subtracting S from XY , we use the 2's complement of S . More precisely, each bit of S is inverted in setting a register for S and $(2N + 1)$ is provided from the inputs of mn at the first cycle of the CP-stage. For reducing the hardware cost and the critical path delay, the CP calculations are executed in the same datapath of the MALU as the CS-calculations. The operation of the MALU is explained in equation (9)

$$MALU_N(XR, YR, SR) = (XY \pm S)R \bmod N. \quad (9)$$

Here R is selected as $R = 2^{k+\alpha}$ where k is the bit-length of the secret key (i.e., $N < 2^k$) and α is a value determined so that the final reductions can be avoided. In our case, we chose $\alpha = 4$. The details are explained by the following Lemma based on previous work by Batina *et al.* (2004).

Lemma 1: If the Montgomery parameter R satisfies the following inequality $R > 16N$, then for inputs $X, Y < 4N$ and $S < 2N$ the result T will satisfy $T < 4N$ (as required).

Proof: The Montgomery multiplication as implemented in the MALU calculates the following:

$$\begin{aligned} T = \text{MALU}_N(X, Y, S) &= \frac{XY + MN}{R} + S \\ &< \frac{4N \cdot 4N}{16N} + N + 2N \leq 4N \end{aligned} \quad (10)$$

While the reduction step was needed in the original notation of Montgomery's algorithm, we use a method which does not require reduction. For convenience of repeating usage of equation (9), the so-called Montgomery form is applied because the output is in the Montgomery form as well. The latency to calculate a MALU_N is $2 \cdot \lceil (k + \alpha)/d \rceil$ cycles in total.

5. Reconfigurable datapath

In order to perform a high-performance modular operation (equation (9)), we need to allocate $(k + \alpha - 1)$ cells for the datapath of the MALU. For instance, the case of ECC-256p and RSA-2048 need 260 and 2,052 cells, respectively. For the general case ($K \times D$ sets of $\text{MALU}_{k \times d}$), the block diagram of the reconfigurable datapath is illustrated in figure 2.

Since we target a platform which supports both ECC-256p and RSA-2048 (with or without CRT), we introduce a coarse grain datapath of the $\text{MALU}_{260 \times 1}$ ($k = 256$

Figure 2. Reconfigurable Datapath using $K \times D$ sets of $\text{MALU}_{k \times d}$.

Figure 3. Reconfigurable datapath using 2×4 sets of $MALU_{260 \times 1}$. (a) The case of RSA-2048: $MALU_{2080 \times 1}$; (b) the case of RSA-2048(CRT): $2 \times MALU_{1040 \times 1}$; (c) the case of ECC-256: $2 \times MALU_{260 \times 4}$.

Figure 4. Flip-flops and multiplexors of the $MALU_{k \times d}$.

and $d=1$) and allocate its clones. Figure 3 shows the interconnection for ECC and RSA in this case.

The datapath can be configured by changing the interconnection of the $MALU_{260 \times 1}$ that is determined by three multiplexors (figure 4) and two- or three-bit registers for selecting them. The signals *sel1* and *sel2* are used for configuring the datapath, and *sel3* is used for configuring the flip-flops. Those multiplexors and flip-flops for the configuration are considered as the area overhead (denoted as *AO*), introduced by the reconfigurable feature. It is approximately estimated as follows:

$$\begin{aligned}
 AO_{base} &= KD(A_{MUX(sel1)} + A_{MUX(sel2)} + A_{MUX(sel3)} + A_{FF(config)}) \\
 &= \{(k + 6d)KD - 2Kd\} \cdot A_{2-1MUX} + \{3KD - K\} \cdot A_{FF}
 \end{aligned} \tag{11}$$

Here, we ignored the area increase caused by the complexity of the wiring. In addition to AO_{base} , some more flip-flops are not used depending on the configuration. As an example, we consider the case of using eight clones of $MALU_{260 \times 1}$ ($K=2$ and $D=4$). When supporting RSA-2048, the datapath is configured as $MALU_{2080 \times 1}$. In this configuration, the horizontal connections of the MALUs are not re-timed by flip-flops for S and R (REG_S and REG_R). Therefore, the total area overhead becomes as follows:

$$\begin{aligned} AO_{RSA-2048} &= AO_{base} + 14A_{FF} \\ &= 11,448A_{2-1MUX} + 36A_{FF} \end{aligned} \quad (12)$$

Likewise, for RSA-2048 with CRT (Quisquater and Couvreur 1982), the datapath is configured to have two sets of $MALU_{1040 \times 1}$. Therefore, the area overhead for RSA-2048 with CRT is estimated as follows:

$$\begin{aligned} AO_{RSA-2048(CRT)} &= AO_{base} + 12A_{FF} \\ &= 11,448A_{2-1MUX} + 34A_{FF} \end{aligned} \quad (13)$$

For ECC-256p, we configure the datapath, so that two sets of $MALU_{260 \times 4}$ can be used in parallel. This configuration uses vertical series of $MALU_{260 \times 1}$. The intermediate values, the virtual carry and sum, are stored in the flip-flops for VS and VC only at the bottom of figure 2. In this configuration, only one fourth of REG_X , REG_Y and REG_N are used in each $MALU_{260 \times 1}$. Therefore, the total area of the overhead becomes as follows:

$$\begin{aligned} AO_{ECC-256p} &= AO_{base} + (260 \times 12 + 260 \times 24 \times 3/4)A_{FF} \\ &= 11,448A_{2-1MUX} + 7,822A_{FF} \end{aligned} \quad (14)$$

In the case of the RSA configuration, we can utilize the flip-flops with almost no waste, while the configuration of ECC cannot use them effectively. In order to exploit the unused flip-flops, we introduce another kind of reconfigurability. Different from RSA, ECC needs to store the intermediate variables during point operations. For this purpose, two sets of 14 words of 260-bit RAM (28×260 -bit RAM) can be configured with the unused flip-flops. In this case, the area overhead becomes as follows:

$$AO_{ECC-256p} = 11,448A_{2-1MUX} + 542A_{FF} \quad (15)$$

Thus, we can make the best use of the hardware resources also for the ECC configuration. For the critical path delay for each configuration, we have different delays as follows:

$$\left. \begin{aligned} T_{RSA-2048} &= T_{RSA-2048(CRT)} = 2T_{FA} + 2T_{2-1MUX} + T_{FF} + T_{wiring} \\ T_{ECC-256p} &= 8T_{FA} + 4T_{2-1MUX} + T_{FF} + 4T_{wiring} \end{aligned} \right\} \quad (16)$$

We assumed that the wiring delay in the critical path of ECC is four times longer than that of RSA. As can be seen from the equation (16), the critical path delay of ECC is about four times longer than RSA. Therefore, we need to assume a circumstance where we can use two different clock frequencies, e.g., providing a divided clock for the ECC configuration, in order to facilitate high performance for both configurations.

6. Performance comparison

6.1. Performance estimation for RSA and ECC

We implemented the proposed datapath on a Xilinx FPGA (Spartan 3). The place-and-route result of Xilinx ISE is shown in table 1. The same design is also synthesized with a 0.25- μm CMOS technology by using synopsys design vision and the pre-layout netlist is used for the cost and performance estimation. As for the constraints of the synthesis, the design is set to the ECC configuration and the critical path delay for the RSA configuration is estimated by the result of (Static Timing Analysis) STA.

Based on the required number of multiplications for RSA and ECC, we estimate the performance for both and compare them with each other. The result is summarized in table 2. For the ECC case, the latency of one point multiplication is used for the performance, and the latency of modular exponentiation is estimated for the RSA case. As can be seen from the result, the performance of the modular exponentiation for RSA-2048 is lower than ECC by a factor of about 8 and 12 for the results of the FPGA implementation and the synthesis with a 0.25- μm CMOS technology respectively. Even when applying CRT acceleration, ECC-256p outperforms RSA-2048. Moreover, considering that ECC-256p offers stronger security than RSA-2048 (Lenstra and Verheul 2000), ECC-256p shows significantly higher performance than RSA-2048 in our design case.

6.2 Comparison with previous work

In this part, we elaborate on the state-of-the-art in Montgomery multipliers. For comparison, we focus on multipliers for RSA, implemented on FPGA. Table 3 lists the previous work and compares it with our results. As for the flexibility of the design, in most of the previous work the operation size is fixed to facilitate high-speed modular multiplication. On the contrary, our design can support 256-bit

Table 1. Implementation result of the proposed datapath.

Target platform	Number of slices	Critical path (n sec)	
		RSA config.	ECC config.
XC3S5000-5	27,597 Slices	10.6	25.3
0.25 μm CMOS	243 Kgates	4.0	6.3

Table 2. Estimated performance of RSA-2048, RSA-2048 with CRT, and ECC-256p (FPGA / 0.25 μm CMOS).

Type of public-key cryptography	MALU configuration	Max. clock freq. [MHz]	Performance [msec]
RSA-2048	$\text{MALU}_{2048 \times 1}$	95/248	133.1/50.8
RSA-2048(CRT)	$2 \times \text{MALU}_{1024 \times 1}$	95/248	33.3/12.8
ECC-256p	$2 \times \text{MALU}_{260 \times 4}$	40/159	17.7/4.3

Table 3. Performance comparison of Montgomery's multipliers over $GF(p)$.

Reference	FPGA/ Technology	Field length bit length	Latency [μ sec]	Max. clock freq. [MHz]	Size of area
This work	XC3S5000-5 0.25 μ m CMOS	2048	43.4	95	27,597 Slices
		1024 (2-way)	21.7		
		256(2-way)	3.20	40	
		2048	16.4	250	243 Kgate
		1024 (2-way) 256(2-way)	8.20 0.80	160	
McIvor <i>et al.</i>	XC2V6000*	2048	29.0	71	23,060 Slices
	XC2V3000*	1024	13.5	76	11,617 Slices
Andres <i>et al.</i>	XC2V250-6	256	2.10	53	1514 LUTs (757 Slices [‡])
Satoh <i>et al.</i>	0.13 μ m CMOS	256	0.48	144	96 Kgates
Tenca <i>et al.</i>	0.50 μ m CMOS	1024	43	80	28 Kgates
		256	7.4		

*: No speed-grade was reported. ‡: Authors' estimate.

ECC and 2048-bit RSA effectively by changing the settings of the datapath configuration. The performance of our design is nevertheless competitive, compared to previously designed high-performance modular multipliers that are dedicated to ECC or RSA in the previous work.

7. Conclusions

We presented a new reconfigurable datapath that enables modular operations for different bit-widths. In addition, the flip-flops are also reconfigured depending on the configuration in order to use hardware resources effectively. The estimated performance based on an FPGA implementation is shown as a case study of RSA-2048 (with and without CRT) and ECC-256p. The results prove that our proposed datapath is suitable for a high-performance cryptosystem supporting both RSA and ECC over $GF(p)$. Especially in our case, ECC-256p shows higher performance than RSA-2048 using the same amount of hardware resources.

Acknowledgements

Kazuo Sakiyama, Nele Mentens and Lejla Batina are funded by FWO projects (G.0450.04, G.0475.05). This research has been also supported by IBBT-QoE and the EU IST FP6 projects SESOC, ECRYPT.

References

- M. Andres, S. Mathew, D. Harris, R. Krishnamurthy and S. Hsu, "An improved unified scalable radix-2 Montgomery multiplier", in *Proceedings of 14th IEEE Symposium on Computer Arithmetic (ARITH-17)*, Cape Cod, MA, USA, 27–29 June 2005, pp. 172–178.
- L. Batina, G. Muurling and S.B. Örs, "Flexible hardware design for RSA and elliptic curve cryptosystems", in *Topics in Cryptology – CT-RSA – The Cryptographers' Track at the*

- RSA Conference*, T. Okamoto Ed., number 2964n in Lecture Notes in Computer Science, Berlin, Germany: Springer-Verlag, San Francisco, CA, USA, 23–27 February 2004.
- I. Blake, G. Seroussi and N.P. Smart, “*Elliptic Curves in Cryptography*”, London Mathematical Society Lecture Note Series 265, Cambridge: Cambridge University Press, 1999.
- W. Diffie and M.E. Hellman, “New directions in cryptography”, *IEEE Trans. Infor. Theory*, 22, pp. 644–654, 1976.
- IEEE P1363. *Standard Specifications for Public Key Cryptography*, <http://grouper.ieee.org/groups1363/>, 2000.
- N. Koblitz, “Elliptic curve cryptosystem”, *Math. Comp.*, 48, pp. 203–209, 1987.
- N. Koblitz. *A Course in Number Theory and Cryptography*, volume 114 of *Graduate Text in Mathematics*, 2nd ed., Berlin, Germany: Springer-Verlag, 1994.
- P. Kocher, J. Jaffe and B. Jun, “Introduction to differential power analysis and related attacks”, url: <http://www.cryptography.com/dpa/technical>, 1998.
- A. Lenstra and E. Verheul, “Selecting cryptographic key sizes”, in *Proceedings of Third International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000)*, H. Imai and Y. Zhengnumber Eds. 1751 in Lecture Notes in Computer Science, Berlin: Springer-Verlag, 2000, pp. 446–465.
- C. McIvor, M. McLoone, J. McCanny, A. Daly and W. Marnane, “Fast montgomery modular multiplication and RSA cryptographic processor architectures”, in *Proceedings of 37th Annual Asilomar Conference on Signals, Systems and Computers*, Monterey, CA, USA, November 2003, pp. 379–384.
- A.J. Menezes, “*Elliptic Curve Public Key Cryptosystems*”, Norwel, MA, USA: Springer-Verlag, 1993.
- V. Miller, “Uses of elliptic curves in cryptography”, in *Advances in Cryptology: Proceedings of CRYPTO’85* H.C. Williams, Ed., number 218 in Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1985, pp. 417–426.
- J.-J. Quisquater and C. Couvreur, “Fast decipherment algorithm for RSA public-key cryptosystem”, *Elect. Lett.*, 18, pp. 905–907, 1982.
- R.L. Rivest, A. Shamir and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, *Communicat. of the ACM*, 21, pp. 120–126, 1978.
- A.F. Tenca and Ç.K. Koç, “A scalable architecture for modular multiplication based on Montgomery’s algorithm”, *IEEE Trans. Comput.*, 52, pp. 1215–1221, 2003.