Implementation of Binary Edwards Curves for very-constrained devices

Ünal Kocabaş, Junfeng Fan and Ingrid Verbauwhede
Katholieke Universiteit Leuven, ESAT/SCD-COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{*unal.kocabas, junfeng.fan, ingrid.verbauwhede*}@*esat.kuleuven.be*

***Abstract*—Elliptic Curve Cryptography (ECC) is considered as the best candidate for Public-Key Cryptosystems (PKC) for ubiquitous security. Recently, Elliptic Curve Cryptography (ECC) based on Binary Edwards Curves (BEC) has been proposed and it shows several interesting properties, e.g., completeness and security against certain exceptional-points attacks. In this paper, we propose a hardware implementation of the BEC for extremely constrained devices. The w-coordinates and Montgomery powering ladder are used. Next, we also give techniques to reduce the register file size, which is the largest component of the embedded core. Thirdly, we apply gated clocking to reduce the overall power consumption. The implementation has a size of 13,427 Gate Equivalent (GE), and 149.5 ms are required for one point multiplication. To the best of our knowledge, this is the first hardware implementation of binary Edwards curves.**

***Keywords*-Public-Key Cryptography, Binary Edwards Curve**

I. Introduction

Public-Key Cryptography (PKC), introduced in 1976 by Diffie and Hellman [1], is now widely used for key establishment, digital signature and data encryption. The best-known and most commonly used public-key cryptosystems are RSA [2] and Elliptic Curve Cryptography (ECC) [3], [4]. The main benefit of ECC is that it offers equivalent security as RSA for much smaller parameter sizes. These advantages result in smaller data-paths, less memory usage and lower power consumption. ECC is widely considered as the best candidate for constrained devices such as RFID-tags.

Integrating a Public Key Cryptosystem into a constrained device such as a RFID-tag is a challenge due to the limitations in costs, area and power. A passive RFID-tag has no battery, thus the processing power of these devices is limited. On the other hand, security is required, in particular to prevent cloning or tracing [5], [6]. It was widely believed that devices with such constrained resources cannot carry out strong cryptographic operations such as Elliptic Curve Scalar Multiplication (ECSM). However, the feasibility of integrating PKCs into such devices have been recently proven by several implementations [7], [8], [9], [10].

Standard formulas for adding two points, say P and Q, on a Weierstrass-form elliptic curves fail if P is at infinity, or if Q is at infinity, or if P+Q is at infinity [11]. Attacks to explore this feature has been proposed [12]. Binary Edwards curves provides a different equation to define an elliptic curve which no longer has points at infinity [11]. This feature is known as $completeness$. In addition to completeness, new explicit addition formulas of Edwards curves make point addition slightly faster than known methods[13], [14].

In this paper, we present a hardware implementation of binary Edwards curves. We focus on different techniques to optimize the implementation to make it suitable for constrained devices. The optimization is mainly to reduce the area consumption and process time. By fixing the curve parameters (d_1 and d_2) and using a common-Z coordinate system [15] we reduce the number of registers from 9 to 6. Moreover, by using a squarer module we reduce the number of required cycles for a squaring operation to 1. As a result, the overall BEC core requires from 11.72 k to 14.53 k GE and the total process time is limited between 548 to 107 ms at 400 kHz for six different configurations, i.e, different digit sizes for the modular multiplier.

The rest of the paper is organized as follows. In section 2, we briefly recap previous implementations on ECC and introduce elliptic and binary Edwards curves. section 3 summaries several techniques to optimize the algorithm. The architecture of BEC and the main parts of the design are investigated in section 4. section 5 gives synthesis results and comparisons. section 6 provides a conclusion.

II. Background

A. Previous Implementations

In 2006, Kumar and Paar [8] proposed an affine coordinate ASIC implementation of the ECC processor, using a modified Montgomery point multiplication method for binary elliptic curves. An area consumption of 10k to 18k GE is obtained in a 0.35 μm CMOS process. Moreover, a bit-serial multiplier, a squarer unit, an adder and a memory, which consists of 6 random access registers, are used to compose an ECC-processor. This ECC-processor supports binary curves from GF(2^{113}) to GF(2^{193}). Furthermore, one scalar multiplication takes 31.8 ms running at 13.56 MHz over GF(2^{163}). In the same year, Sakiyama et al. [16] investigated the impact of the digit size on power consumption and area usage. Generally, bit-serial multipliers are more power efficient, whereas the digit-serial type is more energy efficient.

In 2007, Lee et al. [15] presented an optimized architecture based on the so-called Modular Arithmetic Logic

Unit. It uses a common Z-coordinate system for representing EC points to minimize memory requirements. Storage of intermediate values is usually the main contributor to area (roughly 66%). Area has a direct impact on the production cost of an integrated circuit. This work was further improved in [7].

In 2009, Hein et al. [9] presented an implementation of modular multiplications using 16-bit multipliers. Bit-serial multipliers clock on average $2*163$ = 326 registers per clock cycle. Their digit-level multiplier clocks roughly $4*16$ = 64 registers. So, the proposed approach gives a better power, but not energy, characteristic. Additionally, the core component of the datapath is a $16 * 16$ multiplier. The data width of the RAM circuit is adjusted to fit the 16-bit datapath. The RAM is organized as a 16-bit wide memory with 64 entries. The circuit features small silicon size (15k GE) and has low power consumption (8.57 μW). It computes 163-bit ECC point-multiplications in 296k cycles and has an ISO 18000-3 RFID interface.

B. Elliptic Curve's Algorithm

Let $\mathbb{K}$ be a finite field. An elliptic curve over $\mathbb{K}$ is defined by the *Weierstrass* equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (1)$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ and the discriminant $\Delta \neq 0$ [17]. The $\mathbb{K}$-rational points P(x,y) on curve E together with the point at infinity, $\mathcal{O}(\infty, \infty)$, form an abelian group $E(\mathbb{K})$.

When $Char(\mathbb{K}) = 2$, E can be transformed into the following form without loss of generality.

$$y^2 + xy = x^3 + a'_2x^2 + a'_6. \quad (2)$$

Given two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, one can compute $P_3(x_3, y_3) = P_1 + P_2$ as follow:

$$x_3 = \lambda^2+\lambda+x_1+x_2+a'_2, \quad y_3 = \lambda(x_1+x_3)+x_3+y_1, \quad (3)$$

where $\lambda = \frac{y_1+y_2}{x_1+x_2}$ if $P_1 \neq P_2$ or otherwise $\lambda = x_1 + \frac{y_1}{x_1}$.

For each point addition or doubling, one field inversion is required. Inversion is a quite costly operation (one inversion is equivalent to 8-10 multiplications [18] in $GF(2^{163})$), therefore projective coordinates were proposed. Using projective coordinates, a point P is represented as $(X, Y, Z) \in \mathbb{K}^3$, and the affine coordinates (x, y) can be retrieved as $(X/Z, Y/Z)$.

A scalar multiplication on an elliptic curve is the computation of kP, where k is a large integer and P is a point on the curve. The following algorithm gives a classical way to compute kP.

Due to the conditional branch at step 4-5, Algorithm 1 is considered insecure against Simple Power Analysis (SPA) and Timing Analysis (TA). The Montgomery powering ladder [19] is believed to be secure against SPA and TA. It performs point addition and point doubling in each step regardless of the value of k_i.

Algorithm 1 Add-and-double scalar multiplication.

```
Input: $P = (x, y)$, $k = (k_{l-1}k_{l-2} \cdots k_0)_2$
Output: $Q = (x', y') = kP$
1: $Q = \mathcal{O}$
2: for $i$ from $l - 1$ downto 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 $Q$
```

Algorithm 2 Montgomery ladder for scalar multiplication.

```
Input: $P = (x, y)$, $k = (k_{l-1}k_{l-2} \cdots k_0)_2$
Output: $Q = (x', y') = kP$
1: $Q[0] \leftarrow P$,$Q[1] \leftarrow 2P$,
2: for $i$ from $l - 2$ downto 0 do
3:     $Q[1 - k_i] \leftarrow Q[0] + Q[1]$, $Q[k_i] \leftarrow 2Q[k_i]$
4: end for
5: Return $Q[0]$
```

López and Dahab [13] observed that the Y coordinate is unnecessary when using the Montgomery ladder, which makes the Montgomery ladder a preferred choice in terms of both efficiency and security.

C. Binary Edwards Curves

Let d_1, d_2 be elements of $\mathbb{K}$ with $d_1 \neq 0$ and no element $t \in \mathbb{K}$ satisfying $t^2 + t + d_2 = 0$, then the binary Edwards curve with coefficients d_1 and d_2 is the affine curve [11]:

$$E_{B,d_1,d_2} : d_1(x+y)+d_2(x^2+y^2) = xy+xy(x+y)+x^2y^2 \quad (4)$$

This curve is symmetric in x and y and thus it has the property that if (x_1, y_1) is a point on the curve then so is (y_1, x_1). The point $(0, 0)$ is the neutral element of the addition law, while $(1, 1)$ has order 2.

Bernstein *et al.* [11] proposed to use the so called *w-coordinates* together with the Montgomery ladder. Assume that $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ and $P_3(x_3, y_3)$ are points on curve E_{B,d_1,d_2} (we assume $d_1 = d_2$ throughout this paper), and $P_3 = P_1 + P_2$, then $P_2 + P_3$ and $2P_2$ can be performed with only w-coordinates. Let $P_5(x_5, y_5) = P_2 + P_3$, $P_4(x_4, y_4) = 2P_2$, and $w_i = x_i + y_i$ for $i \in \{1, 2, 3, 4, 5\}$, the explicit formulas for Point addition (PA) and doubling (DA) are given below.

Affine point addition with w-coordinates:

$$\begin{aligned} &A = w_2^2,\ B = A + w_2,\ C = w_3^2,\ D = C + w_3, \\ &w_5 = 1 + d_1\frac{1}{d_1 + B \cdot D} + w_1. \end{aligned} \quad (5)$$

Affine point doubling with w-coordinates:

$$A = w_2^2,\ B = A + w_2, w_4 = 1 + d_1 \frac{1}{d_1 + B^2}. \quad (6)$$

When using Montgomery ladder, $Q[1]$ is always $Q[0]+P$, thus w-coordinate is applicable. At the end of the loop, the x-coordinate of $2Q[0]$ can be retrieved from $w(Q[0])$ (=the w coordinate of $Q[0]$), $w(Q[1])$ and $P(x, y)$.

In order to recover x and y values from *w-coordinates*, we can use the following formula to compute $2(x_2, y_2)$ given x_1,y_1,w_2,w_3 [11].

$$\begin{aligned} x_2^2 + x_2 = \ & (w_3(d_1 + w_1 w_2(1 + w_1 + w_2) + \tfrac{d_2}{d_1} w_1^2 w_2^2) \\ & + d_1(w_1 + w_2) + (y_1^2 + y_1)(w_2^2 + w_2)) \\ & /(w_1^2 + w_1) \end{aligned}$$

This formula produces $x_2^2 + x_2$, then we use a half-trace computation to reveal either x_2 or $x_2 + 1$. The Algorithm 3 shows the half-trace computation.

Algorithm 3 Half-trace Computation in $GF(2^m)$

```
Input: $T = x^2 + x$, $H = 0$;
1: for $i$ from 0 to $\lfloor \frac{m}{2} \rfloor - 1$ do
2:     $H \leftarrow H + T^{(2^{(2i+1)})}$
3: end for
return $H$
```

Table I
Mixed w-coordinate Algorithms

Addition Algorithm	Doubling Algorithm
1. $T_1 \leftarrow W_3 + Z_3$	12. $W_2 \leftarrow T_1^2$
2. $W_3 \leftarrow W_3 T_1$	13. $Z_2 \leftarrow Z_2^2$
3. $T_1 \leftarrow W_2 + Z_2$	14. $Z_2 \leftarrow Z_2^2$
4. $T_1 \leftarrow W_2 T_1$	15. $Z_2 \leftarrow Z_2 d_1$
5. $W_2 \leftarrow T_1 W_3$	16. $Z_2 \leftarrow Z_2 + W_2$
6. $Z_3 \leftarrow Z_2 Z_3$	
7. $Z_3 \leftarrow Z_3^2$	
8. $Z_3 \leftarrow d_1 Z_3$	
9. $Z_3 \leftarrow Z_3 + W_2$	
10. $T_2 \leftarrow Z_3 w_1$	
11. $W_3 \leftarrow T_2 + W_2$	

Table II
Proposed PA and PD Algorithms

Addition Algorithm	Doubling Algorithm	Extra Steps
1. $T_1 \leftarrow W_3 + Z$	11. $W_2 \leftarrow T_1^2$	13. $W_2 \leftarrow W_2 T_2$
2. $W_3 \leftarrow W_3 T_1$	12. $T_1 \leftarrow Z + W_2$	14. $W_3 \leftarrow W_3 T_1$
3. $T_1 \leftarrow W_2 + Z$		15. $Z \leftarrow T_1 T_2$
4. $T_1 \leftarrow W_2 T_1$		
5. $W_2 \leftarrow T_1 W_3$		
6. $Z \leftarrow Z^4$		
7. $Z \leftarrow Z d_1$		
8. $T_2 \leftarrow W_2 + Z$		
9. $W_3 \leftarrow T_2 w_1$		
10. $W_3 \leftarrow W_2 + W_3$		

III. Optimization of the Algorithm

For an ECC processor integrated in a very-constrained device such as passive RFID-tags, the area and power consumption budget is very limited. In this case, reducing the area for temporary storage is crucial since it normally takes more than 50% of the overall area (see [7]). Therefore, reducing the number of intermediate results is important.

Although projective coordinates save us one inversion in each iteration, one drawback is that extra registers are required to store Z coordinates. The *w-coordinates* with Montgomery ladder consist of three points (P_1, P_2, P_3). Hence, at least seven registers (d_1, W_1, Z_1, W_2, Z_2, W_3, Z_3) are required, not even including two registers for intermediate values. Using mixed *w-coordinates*, which represent P_1 with only w_1, can save one register. Thus, we choose the mixed *w-coordinates* for the implementation.

A. Original Mixed w-coordinate [20]

Assume that w_1 is given as a field element, w_2, w_3 are given as fractions W_2/Z_2, W_3/Z_3 and w_4, w_5 are outputs of the formulas. The explicit formulas of addition and doubling are presented in register form in Table I. The register allocation of values are arranged using the linear scan method in [20].

Modular addition formula uses 5M + 1S + 1D, where M, S and D denote field multiplication, field squaring and multiplying by the curve parameter d_1, respectively. Doubling formula uses 1M + 3S + 1D. These formulas can share the computation of T_1 (step 3-4) to reduce the total cost of differential addition and doubling to 5M + 4S + 2D. In order to perform these formulas, seven registers are required to store ($w_1, W_2, Z_2, W_3, Z_3, T_1, T_2$). These formulas can be transformed to a common Z-coordinate system at the cost of three extra field multiplications.

B. Mixed w-coordinate Using Common Z

The common Z-coordinate was introduced by Lee *et al.* [15] to save registers. After every iteration of Montgomery ladder, we ensure that $Z_2 = Z_3$. This can be done with three extra field multiplications as given below.

$$W_2 \leftarrow W_2 Z_3, W_3 \leftarrow W_3 Z_2, Z \leftarrow Z_2 Z_3. \quad (7)$$

In the first step of the Montgomery ladder ($Q[1] = 2P$), this condition is satisfied with $Z = Z_3$ and $W_2 = W_2 \cdot Z_3$, since the algorithm starts from $Z_2 = Z_3 = 1$. We apply this method to the mixed *w-coordinates* of BEC. The formulas of differential addition and doubling shown in Table I are transformed to the sequences in Table II.

In this situation, the formula uses 7M + 3S + 1D, which is much less than (5M + 4S + 2D) + (3M). Since after each iteration $Z_2 = Z_3$, step 6 of Table 1 becomes a squaring, and step 13-15 are omitted since they are the same to step 6-8. As a result, we trade (1M + 2S + 1D) with 3M + 1S, and the complexity of one iteration is 7M + 3S + 1D. Due to the use of common Z-coordinate, the proposed sequence requires only 6 registers (w_1,W_2,W_3,Z,T_1,T_2), or 7 registers if d_1 is not fixed. Furthermore, the mapping of variables in Table II is given in the following section in Table IV by using 6 registers.

C. Complexity

We compare the complexity of point addition and doubling using different coordinates and different elliptic curves. Table III shows the number of field operations for different combinations. Here we assume the Montgomery ladder is in use (as it is in [7], [9]). Since d_1 can be chosen $small$ [11], D can be very efficient. In this case, the cost of one Montgomery step on BEC is lower than normal curves or almost the same when using the common-Z trick.

Besides the coordinate system, the selection of curve parameters also has an impact on the complexity. For example, by selecting $a'_6 = 1$ in (2), one multiplication is omitted [7]. For extremely constrained devices, fixing one or two curve parameters helps to reduce the area and performance. For example, the implementation in [9] chose to implement the B-163 curve [21]. In this paper we choose $d_1 = d_2$ and d_1 is also fixed.

Table III
COMPLEXITY OF ONE STEP OF MONTGOMERY LADDER

Curve	Coordinates	PA+PD	Common Z
Weierstrass	Affine	2I + 4M+2S	-
	Mixed (Jacobian+Affine)	15M + 8S + M_2	-
	Lopéz-Dahab	6M + 5S	7M + 4S
Edwards ($d_1 = d_2$)	Affine	2I + 1M + 3S + 2D	-
	Projective w-Coord.	7M + 4S + 2D	-
	Mixed w-Coord.	5M + 4S + 2D	7M + 3S + 1D

IV. Implementation of a Binary Edwards Curve

As RFID systems become ubiquitous and used in several applications, the specification must be standardized. For example, ISO 18000-3 requires a 13.56 MHz clock frequency and a power consumption of less than 15 μW at 1.5 V to guarantee 1 m operating range [22]. Moreover, [7] states that the clock frequency is chosen to finish protocols within 250 ms and is a factor of 13.56 MHz. Taking into account these limitations, the goals to achieve in an RFID system are the low area and power consumption, and a short processing time. In this case, the number of registers is the most important issue, since the registers occupy more than 50% of the total area. Besides the common Z trick, we also optimize the register file for limited access. Clock gating is applied to reduce the power consumption.

A. Modular Arithmetic Logic Unit

In order to perform addition, multiplication and squaring operations in Table III, we design a compact Modular Arithmetic Logic Unit (MALU) [16] as illustrated in Figure 1. In the MALU architecture, operations are performed over finite fields as shown through Equation (8).

$$\begin{aligned} A(x) &= B(x) \cdot\ C(x) \bmod\ P(x) \\ A(x) &= A(x) + C(x) \\ C(x) &= C(x)^2 \ \bmod\ P(x) \end{aligned} \tag{8}$$

where $A(x) = \sum a_i x^i, B(x) = \sum b_i x^i, C(x) = \sum c_i x^i$ and $P(x) = x^{163} + x^7 + x^6 + x^3 + 1$.

The cost of the field multiplication is $\frac{163}{d}$, with d the digit size, and for addition it is one clock cycle. In every cell, multiplication and addition can share the same XOR array. Therefore, the MALU can be easily scaled to different digit sizes by using cells in series. Furthermore, we design a squarer to perform squaring operations in one clock cycle independent of the digit size. The module of squaring is also added into the MALU with one extra control bit.

The MALU block does not contain any internal registers. To implement Equation (8), the register file keeps the intermediate value (shown in Figure 1 as RetA) and MALU performs the calculations. When the MALU performs a multiplication, each digit of multiplicand must be provided. That means in every cycle, RegB must be shifted to the left by d bits and the most significant digits turn back to the least significant bits as a circular shift. Moreover, the intermediate result of RegA must be reused as an input for $\frac{163}{d}$ times. The addition operation can be performed with the same hardware with some additional tricks using RegA and RegC values as operands. Then RegA keeps the result. The output of the MALU (RetA) is also used to return the result of the squarer.

B. Register File Architecture

The register file architecture is shown in Figure 2. This architecture was originally proposed in [7], and has been shown an effective way to reduce the area of register files. We modify it such that it fits binary Edwards curves. Six registers form a big circular shift register file, which has a lower complexity than random access register file. Each register is independently controlled for efficient management. RegA, RegB and RegC are used by MALU, and RegD is used as the input register. RegD has an 8-bit I/O through which data can be loaded and stored. Loading RegD with a new value can occur in parallel with multiplications. Furthermore, RegB is a circular shift register which can be shiftted by d bits (digit size of the data path) to the left. Two extra connections are added to avoid extra shifting in the circular register file

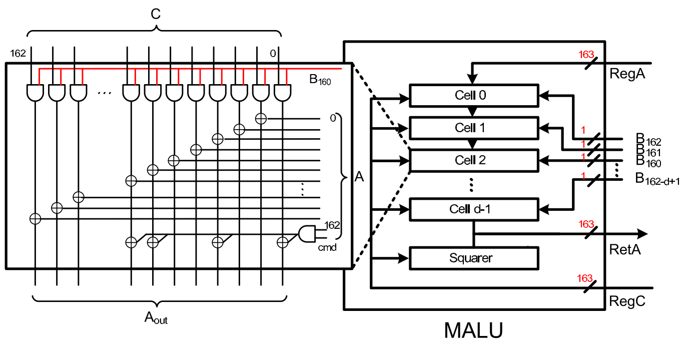

Figure 1. Modular Arithmetic Logic Unit

which are shown by dashed lines in Figure 2. Except for RegB and RegD, all the registers can be only updated by the preceding.

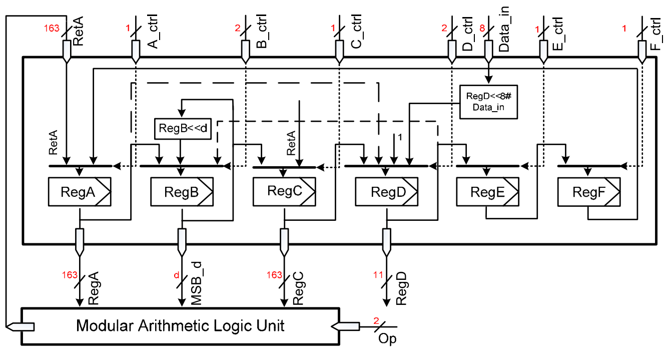

Figure 2. Register File Architecture

Clock-gating is applied to the circular shift register to reduce area and power consumption. An enable signal is created for all registers to control whether to update with new values or to keep previous values. As a result, the amount of multiplexers needed for the register file is reduced. RegE and RegF no longer require multiplexers.

C. Architecture of BEC Processor

The main architecture of the binary Edwards curves processor is shown in Figure 3. It consists of a processor, a small interface, a RAM, a ROM and front-end module. The ROM stores first point coordinates (x_1, y_1) and a key value (3×21-byte). Internal values and result points (W_2, W_3, Z) of the differential addition and doubling are kept in the register file. At the end of EC scalar multiplication, the final point is recovered by Algorithm 3 and sent to the RAM. The interface provides the connection from/to memory according to the address bits. The processor part of the design consists of a control block, a register file and a modular arithmetic logic unit (MALU). The control block has the finite state machine data path that manages the modules according to the addition, squaring and multiplication.

D. Low Power ASIC Design with Clock Gating

Clock gating is one of the power-saving techniques used on many synchronous circuits. It creates additional logic to generate clock trees for each register individually. Thus, flip-flops do not change state in disabled parts of the design. Their switching power consumption goes to zero, and only leakage currents are incurred. After clock gating is applied to the register file, the area and power consumptions are reduced by 8% and 20%, respectively, compared to the normal clocked version due to the removal of the multiplexers. The functionality of the clock gated version is verified with Modelsim SE. Moreover, the clock gated version is implemented on a Spartan-3E FPGA board with some transformations to verify the functionality.

Table IV
MAPPING OF THE POINT MULTIPLICATION (SEE TABLE 2 FOR THE OPERATION OF EACH STEP)

Operation	RegA	RegB	RegC	RegD	RegE	RegF
	Z	W_2	W_3	W_2	w_1	Z
	Z	W_2	W_2	W_3	w_1	Z
ADD (3)	W_2+Z	W_2	W_2	W_3	w_1	Z
	W_2+Z	W_2+Z	W_2	W_3	w_1	Z
MULT (4)	C	W_2+Z	W_2	W_3	w_1	Z
	Z	W_3	W_2	C	w_1	Z
	Z	W_3	W_3	C	w_1	Z
ADD (1)	W_3+Z	W_3	W_3	C	w_1	Z
	W_3+Z	W_3+Z	W_3	C	w_1	Z
MULT(2)	D	W_3+Z	W_3	C	w_1	Z
	Z	D	W_3	C	w_1	Z
	Z	Z	D	C	w_1	w_1
	Z	Z	Z	D	C	w_1
SQR (6)	Z	Z	Z^2	D	C	w_1
SQR (6)	Z	d_1	Z^4	D	C	w_1
MULT (7)	X	d_1	Z^4	D	C	w_1
	w_1	X	Z^4	D	C	C
	C	D	X	w_1	C	C
	C	C	D	X	w_1	C
MULT (5)	V	C	D	X	X	w_1
	w_1	V	C	X	X	X
SQR (11)	X	V	W_4	w_1	X	X
	X	V	V	W_4	w_1	X
ADD (8)	Z_4	V	V	W_4	w_1	X
	X	V	V	Z_4	W_4	w_1
ADD (12)	Z_5	V	V	Z_4	W_4	w_1
	w_1	Z_5	V	Z_4	W_4	w_1
	w_1	w_1	Z_5	V	Z_4	W_4
MULT (9)	$w_1 * Z_5$	w_1	Z_5	V	Z_4	W_4
	$w_1 * Z_5$	V	w_1	Z_5	Z_4	W_4
	$w_1 * Z_5$	Z_5	V	w_1	Z_4	W_4
ADD (10)	W_5	Z_5	Z_5	w_1	Z_4	W_4
	W_4	Z_5	Z_5	W_5	w_1	Z_4
	W_4	W_4	Z_5	W_5	w_1	Z_4
MULT (13)	W_2New	W_4	Z_5	W_5	w_1	Z_4
	Z_4	W_5	Z_5	W_2New	w_1	Z_4
	Z_4	Z_4	W_5	Z_5	W_2New	w_1
MULT (14)	W_3New	Z_4	W_5	Z_5	W_2New	w_1
	W_3New	Z_5	Z_4	W_3New	W_2New	w_1
MULT (15)	Z	Z_5	Z_4	W_3New	W_2New	w_1
	w_1	W_3	Z_4	Z	W_2	W_2
	W_2	W_3	W_3	w_1	Z	W_2
	W_2	W_3	W_3	W_2	w_1	Z
	Z	W_2	W_3	W_2	w_1	Z

E. Verifying Design on an FPGA

Global free-running clocks in an FPGA are distributed using dedicated interconnects specifically designed to connect and supply clock inputs to various resources in the FPGA. These clock networks are optimally designed to have low skew, low power, and improved jitter tolerance. Gating a global clock signal with a logic circuit forces the gated clock signal to traverse the much slower routing network, thus introducing significant skew [23]. This can lead to hold errors. Therefore, we resort to manual transformation of registers of clock gating while maintaining identical functionality. It is achieved by re-declaring every gated flip-flop with two nodes, a free-running clock node and an enable node. That is, flip-flops that use the gated clocks are changed to enabled flip-flops. Such flip-flops are then clocked by the free-running clock and enabled by the gate signal. Consequently, the functionality of a low power ASIC design with clock gating is successfully tested on an FPGA.

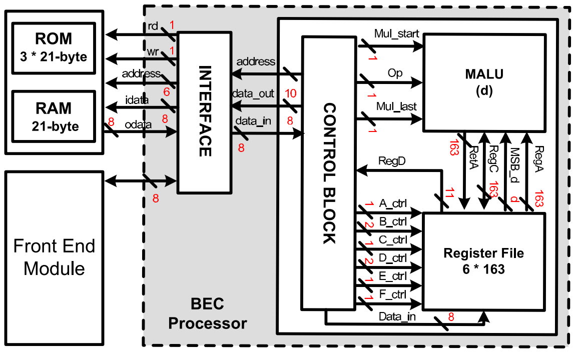

Figure 3. BEC Processor Architecture

V. Synthesis Result and Discussions

The implementation of binary Edwards curves is written in the GEZEL [24] hardware description language and it is optimized for fast computation and low-area consumption. The implementation is synthesized for an ASIC design using a low leakage library of UMC's 0.13 μ m using the Synopsys Design Compiler tools [25]. Table V shows the area, power and delay estimations of the proposed design using different digit sizes.

A. Area, Power and Energy Estimations

In order to find the best trade-off on digit size, we evaluated six different digit sizes. For example, the implementation (d=4) uses 13,427 GE and finishes one EC scalar multiplication in 59,800 clock cycles. The power consumption is estimated by using both Design Vision and ModelSim SE, which provides average power consumptions by including switching activity of gates. The total dynamic power is around 12 μW at 400 kHz. Since one point multiplication takes 149.5 ms, the energy consumption is 1.79 μJ. According to the synthesis results, the power consumption is low enough to implement the proposed design in a passive RFID-tag.

B. Comparison with Previous Implementations

Table VI shows the comparison of this work with related work in GF(2^{163}). Our strategy is similar to that of [7]. It uses a digit-serial multiplier and a cyclic register structure, but no squarer modules. Although the area consumption of [7] is smaller than our design, the power consumption is still larger at 400 kHz. Moreover, the energy cost of one EC scalar multiplication is halved in our design.

The result of [8] shows 15,094 GE which is larger than our proposal and requires almost seven times as many cycles. [8] uses affine coordinates rather than projective or mixed coordinates, so each iteration takes considerably more time than our implementation due to inversions. The design in [9] has the lowest power consumption among the related works, but is much slower than our design.

The synthesis results do not include ROM or RAM in Table VI, therefore the consumption of area and power of memories should be added. Table VII compares the number of memory access and memory requirements in one EC scalar multiplication. Every design must have at least three ROM blocks of size 21-byte to store the main point (x_1, y_1) and a key value. Moreover, the number of RAM blocks is chosen depending on the representation of the final point. By storing the base point in shared memory, one 163-bit register can be saved (Type-1 in [7]). However, the ECC core needs to load 21 bytes in each iteration. This loading dramatically increases the number of memory accesses. Our design only reads the EC base point (x_1, y_1) once in the beginning to calculates w_1. For one EC scalar multiplication, the RAM and ROM block are accessed 63 and 84 times, respectively. The requirements of memory units and the number of memory access are not indicated in [8] and [9], thus we add minimum requirements of ordinary ECC curves on Table VII. Based on these comparisons, this work requires only a limited number of memory units and accesses.

Table V
Implementation Results at 400 kHz

	Area	Power & Energy Est.		Delay	
Digit Size	Total Area $[GE]$	T . Dynamic Power $[\mu W]$	Energy$\star$ $[\mu J]$	Time $[msec]$	Cycles
d=1	11,720	7.27	3.98	547.87	219,148
d=2	12,348	9.1	2.58	283.57	113,428
d=3	12,862	10.19	1.99	195.28	78,112
d=4	13,427	11.997	1.79	149.5	59,800
d=5	13,970	12.69	1.57	123.34	49,336
d=6	14.530	13.8	1.48	106.99	42,796

$\star$ Energy for one EC scalar multiplication

Table VI
Comparison to Related Work

Ref.	Digit Size	Area $[GE]$	Cycles	Freq. $[kHz]$	Perf. $(msec)$	Power $[\mu W]$	Energy$\star$ $[\mu J]$
BEC This work $[0.13\mu m]$	1	11,720	219,148	400	547.87	7.27	3.98
	2	12,348	113,428		283.57	9.1	2.58
	3	12,862	78,112		195.28	10.19	1.99
	4	13,427	59,800		149.5	11.99	1.79
	5	13,970	49,336		123.34	12.69	1.57
	6	14,530	42,796		106.99	13.8	1.48
ECC† [7] $[0.13\mu m]$	1	10,106	275,816	1,130	244.08	36.63	8.94
	2	11,383	144,842	590	245.49	21.55	5.29
	3	12,236	101,183	411	246.19	15.75	3.88
	4	12,863	78,544	323	243.17	12.08	2.94
ECC [8] $[0.35\mu m]$	N/A	15,094	430,654	13,560	31.8	N/A	N/A
		16,207	376,864		27.9		
ECC [9] $[0.18\mu m]$	N/A	11,904	296,000	106	2,792	N/A	N/A
		13,250				8.57	31.3

$\star$ Energy for one EC scalar multiplication
† Extra memory is required

VI. Conclusion

In this paper, the first hardware implementation of a binary Edwards curve is presented. We propose a compact architecture for a binary Edwards curve. We suggest the use of mixed w-coordinate with the common Z-coordinates

Table VII
REQUIREMENTS OF THE MEMORY ACCESS

Ref.	ECC Core		Shared Memory		Memory Access		
	Register File	RAM	ROM	RAM	Read		Write
					ROM	RAM	RAM
BEC	6*163	-	3*21-byte	21-byte	84	21	42
ECC	6*163 †	-	4*21-byte	21-byte	4,330	-	21
[7]	7*163	-	4*21-byte	21-byte	928	-	21
ECC	7*163	-	3*21-byte ⋆	21-byte ⋆	63	-	42
[8]	8*163	-	3*21-byte ⋆	21-byte ⋆	63	-	42
ECC	-	64*16	3*21-byte ⋆	21-byte ⋆	63	-	42
[9]	-	7*163	3*21-byte ⋆	21-byte ⋆	63	-	42

⋆ Minimum memory requirements for ordinary ECC curves
† Type-1 in the original paper

which reduces the size of the register file. According to the synthesis results, the implementation of a binary Edwards curve can provide the same level of efficiency as previous ECC designs.

ACKNOWLEDGEMENTS

This work was supported in part by K.U. Leuven-BOF (OT/06/40), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by FWO project G.0300.07, and in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

REFERENCES

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] V. S. Miller, "Use of elliptic curves in cryptography," in *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*. New York, NY, USA: Springer-Verlag New York, Inc., 1986, pp. 417–426.

[4] N. Koblitz, "Elliptic Curve Cryptosystem," *J. Cryptology Math. Comp.*, vol. 48, pp. 203–209, 1987.

[5] S. Vaudenay, "On Privacy Models for RFID," in *ASIACRYPT*, 2007, pp. 68–87.

[6] P. Tuyls and L. Batina, "RFID-Tags for Anti-counterfeiting," in *CT-RSA*, 2006, pp. 115–131.

[7] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede, "Elliptic-Curve-Based Security Processor for RFID," *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1514–1527, 2008.

[8] S. Kumar and C. Paar, "Are standards compliant elliptic curve cryptosystems feasible on RFID?" in *Proceedings of Workshop on RFID Security*, 2006.

[9] D. Hein, J. Wolkerstorfer, and N. Felber, "ECC is Ready for RFID - A Proof in Silicon," in *Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science, R. Avanzi, L. Keliher, and F. Sica, Eds. Springer-Verlag, 2009, in press.

[10] A. C. Atici, L. Batina, J. Fan, I. Verbauwhede, and S. Berna Ors Yalcin, "Low-cost implementations of NTRU for pervasive security," in *ASAP '08: Proceedings of the 2008 International Conference on Application-Specific Systems, Architectures and Processors*, 2008, pp. 79–84.

[11] D. J. Bernstein, T. Lange, and R. R. Farashahi, "Binary Edwards Curves," in *CHES '08: Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 5154. Springer-Verlag, 2008, pp. 244–265.

[12] T. Izu and T. Takagi, "Exceptional procedure attack on elliptic curve cryptosystems," *PKC 2003, LNCS*, vol. 2567, pp. 224–239, 2003.

[13] J. López and R. Dahab, "Fast Multiplication on Elliptic Curves over GF(2m) without Precomputation," in *CHES '99: Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems*. London, UK: Springer-Verlag, 1999, pp. 316–327.

[14] M. Stam, "On Montgomery-Like Representations for Elliptic Curves over GF(2^k)," in *Public Key Cryptography*, 2003, pp. 240–253.

[15] Y. K. Lee and I. Verbauwhede, "A Compact Architecture for Montgomery Elliptic Curve Scalar Multiplication Processor," in *WISA*, 2007, pp. 115–127.

[16] K. Sakiyama, L. Batina, N. Mentens, B. Preneel, and I. Verbauwhede, "Small-footprint ALU for Public-Key Processors for Pervasive Security," in *Proc. Workshop RFID Security (RFIDSec '06)*, 2006.

[17] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. New York: Springer, 2004.

[18] T. Itoh and S. Tsujii, "Effective recursive algorithm for computing multiplicative inverses in GF(2^m)," vol. 24(6). Electronic Letters, 1988, pp. 334–335.

[19] M. Joye and S.-M. Yen, "The Montgomery Powering Ladder," in *CHES*, 2002, pp. 291–302.

[20] M. Poletto and V. Sarkar, "Linear scan register allocation," *ACM Trans. Program. Lang. Syst.*, vol. 21, no. 5, pp. 895–913, 1999.

[21] National Institute for Standards and Technology, "Digital Signature Standard (DSS)," Technical report, January 2000.

[22] ISO/IEC 18000-3:2004, "Information Technology - Radio Frequency Identification (RFID) for Item Management - Part 3: Parameters for air interface communications at 13.56 MHz."

[23] P. H. Wang, J. D. Collins, C. T. Weaver, B. Kuttanna, S. Salamian, G. N. Chinya, E. Schuchman, O. Schilling, T. Doil, S. Steibl, and H. Wang, "Intel ®atom ™processor core made FPGA-synthesizable." in *FPGA*, 2009, pp. 209–218.

[24] P. Schaumont and I. Verbauwhede, "Interactive Cosimulation with Partial Evaluation," in *DATE*, 2004, pp. 642–647.

[25] Synopsys Inc., "Design Compiler Tutorial Using Design Vision," 2006.