# A Systolic Architecture for Elliptic Curve Cryptosystems

*Wei-Chang Tsai and Sheng-Jyh Wang*
Department of Electronics Engineering
National Chiao Tung University
Hsinchu, Taiwan, China
Email: eea80616@smoltz.ee.nctu.edu.tw Fax no.: (+886)-3-5710580

**Abstract**

This article presents a new systolic architecture for the main operation in an elliptic curve cryptosystem over the finite field $GF(2^m)$. This proposed architecture is actually a finite field multiplier and is used to implement the addition operation in the elliptic curve cryptosystem. We apply partitioning scheme and parallelize the main operation in a straightforward systolic architecture to speed up the operation and then apply merging and re-timing schemes in the partitioned architecture to future improve the performance of this architecture. We compare this architecture with some previously proposed systolic architectures for the finite field arithmetics. The comparison shows that our architecture offers the lowest hardware complexity. This architectures can be easily adopted to build a low-complexity elliptic curve cryptosystem.

## I. Introduction

Due to the fast development in data communication, secure electronic transaction is becoming a major concern in many applications, such as electronic commerce. Among these security-related applications, public-key cryptography has been proven to be an attractive technique [1], [2]. This type of cryptosystem may deal with digital signatures, authentication, and key management. Among various public-key cryptosystems, elliptic curve cryptosystem (ECC) can offer a larger attack-resistance/key-length ratio and lead to a faster and smaller implementation. An ECC system could be constructed either in a finite field of odd characteristic or a finite field of characteristic 2. Within an ECC system, the main core is actually a finite field multiplier. In this paper, we propose a standard basis multiplier over finite filed $GF(2^m)$ for a high performance ECC system.

In this paper, we focus on the implementation of a non-supersingular elliptic curve cryptosystem. A non-supersingular elliptic curve ($E$) over a finite field $GF(2^m)$ can be defined as:

$$E : y^2 + xy = x^3 + ax^2 + b, \tag{1}$$

where $a, b \in GF(2^m)$ and $b \neq 0$. All the solutions of the above equation, together with a special point called the point at infinity $\mathcal{O}$, form the basic elements of the elliptic curve over $GF(2^m)$. If $P = (x_1, y_1) \in E$, then $-P = (x_1, y_1 + x_1)$. For all $P \in E$, $\mathcal{O} + P = P + \mathcal{O} = P$. Moreover, if $P = (x_1, y_1), Q = (x_2, y_2) \in E$, and $Q \neq -P$, then the sum of these two points is defined as $P + Q = (x_3, y_3)$, where

$$x_3 = \begin{cases} (\frac{y_1+y_2}{x_1+x_2})^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a & P \neq Q \\ x_1^2 + \frac{b}{x_1^2} & P = Q, \end{cases} \tag{2}$$

and

$$y_3 = \begin{cases} (\frac{y_1+y_2}{x_1+x_2})(x_1 + x_3) + x_3 + y_1 & P \neq Q \\ x_1^2 + (x_1 + \frac{y_1}{x_1})x_3 + x_3 & P = Q. \end{cases} \tag{3}$$

Note that all the additions, multiplications, and inverses operate in the finite field $GF(2^m)$. That is, the sum of two points in $E$ consists of a numbers of

0-7803-5747-7/00/$10.00©2000 IEEE.

multiplications, inverses, and additions over $GF(2^m)$. The inverse operation can be computed by applying Fermat's theorem: $x^{2^m} = x$ for all $x \in GF(2^m)$. This approach takes $m$ multiplications to finish an inverse operation ($x^{-1}$). To improve the operation speed, the homogeneous coordinates (or projective coordinates) are usually used to reduce the number of reversion operation. To compute $kP$, where $k$ is an positive integer and $P \in E$ in an elliptic curve cryptosystem, one necessary doubling and one possible addition are required in each iteration. If using homogeneous coordinates, 17 finite field multiplications are needed to computing $P + Q$ with $P \neq Q$, while 12 finite field multiplications are needed for computing $2P$.

Regarding the finite field multiplications over $GF(2^m)$, assume there are two elements $A(\alpha)$ and $B(\alpha) \in GF(2^m)$. In standard basis representation, they can be expressed as

$$A(\alpha) = \sum_{i=0}^{m-1} a_i \alpha^i = a_{m-1}\alpha^{m-1} + a_{m-2}\alpha^{m-2} + ... + a_1\alpha + a_0, \quad (4)$$

and

$$B(\alpha) = \sum_{i=0}^{m-1} b_i \alpha^i = b_{m-1}\alpha^{m-1} + b_{m-2}\alpha^{m-2} + ... + b_1\alpha + b_0, \quad (5)$$

where $f_i$, $a_i$, and $b_i$ are binary number.

The addition operation over $GF(2^m)$ is computed by $S(\alpha) = A(\alpha) + B(\alpha) = \sum_{i=0}^{m-1} s_i \alpha^i$, where $s_i = a_i \oplus b_i$ and $\oplus$ denotes the exclusive OR operation. On the other hand, the multiplication operation $GF(2^m)$ can be computed as

**Multiplication Algorithm over $GF(2^m)$ by using Modulus Operation**

$R^0(\alpha) = 0$;
for $i = 1$ to $m$
  $R^i(\alpha) = (R^{i-1}(\alpha)\alpha + a_{m-i}B(\alpha)) \pmod{F(\alpha)}$;
end.
$P(\alpha) = R^m(\alpha)$,

where $a_j$ is the $j$th coefficient of $A(\alpha)$, $R^i(\alpha) = \sum_{j=0}^{m-1} r_j^i \alpha^j$ is the partial sum after the $i$th iteration, and $a_{m-i}B(\alpha) = \sum_{j=0}^{m-1}(a_{m-i}b_j)\alpha^j$.

For a high-speed multiplier over $GF(2^m)$, several designs [4], [5], [6] adopting the architecture of semi-systolic array have been proposed. However, all these semi-systolic architectures have to broadcast some global signals and it becomes more difficult to handle the broadcasting problem as the bit length $m$ becomes larger. Therefore, a "pure" systolic array, instead of a semi-systolic array, is usually a more appropriate choice for VLSI implementation [7]-[14]. Up to now, among these architectures, Yeh's design [7] is the fastest due to its shortest clock period, while one of Mekhallalati's designs (Systolic-II) [11] has a superior performance in the area-time product.

In this paper, we propose a new architectures to further improve the operation speed and to reduce the area complexity. We first partition the general cells in Kung's design [17] to shorten the clock period. Then we merge the cells to reduce the latency and area. For an elliptic curve cryptosystem with the Hamming weight of $k$ is 20 and $m = 155$, the throughput rate of our architecture is about $\frac{310}{155(12*154+17*20)*2.3*10^{-9}} =$ 397 Kbits per second. This speed is much faster than that of Agnew's design [20], which can be run at 60 Kbits per second. The organization of this paper is arranged as following. In Section 2, our systolic architecture is presented. In Section 3, the comparison with five systolic arrays and one semi-systolic array is made. We also present the performance of this architecture when used in an ECC system.

## II. Systolic Architecture

The main operation, $R^i(\alpha) = (R^{i-1}(\alpha)\alpha + a_{m-i}B(\alpha)) \pmod{F(\alpha)}$, of the multiplication algorithm can be rewritten as $R^i(\alpha) = (R^{i-1}(\alpha)\alpha \pmod{F(\alpha)}) + a_{m-i}B(\alpha)$. This is because $a_{m-i}B(\alpha)$ is already in SBR form. Hence, the computation of $R^i(\alpha)$ can be treated as the combination of a modular operation and an addition. The modular operation, $(R^{i-1}(\alpha)\alpha) \pmod{F(\alpha)}$, can be computed by converting the highest order term of $(R^{i-1}(\alpha)\alpha)$ into the SBR form first, and then adding the converted result with the remaining part of $(R^{i-1}(\alpha)\alpha)$. At the bit level, the Multiplication Algorithm becomes

**Bit-level Multiplication Algorithm over $GF(2^m)$**

$R^0(\alpha) = 0$; $r^k_{-1} = 0$, $k = 1$ to $m$;
for $i = 1$ to $m$
  $R^i(\alpha) = \sum_{j=0}^{m-1}(r^{i-1}_{m-1}f_j \oplus r^{i-1}_{j-1} \oplus a_{m-i}b_j)\alpha^j$;
end.
$P(\alpha) = R^m(\alpha)$.

In this algorithm, the main operation can be computed bit-by-bit by adding three operands: $r^{i-1}_{m-1}f_j$, $r^{i-1}_{j-1}$, and $a_{m-i}b_j$. Because $r^{i-1}_{m-1}$, the MSB of $R^{i-1}(\alpha)$, is involved in the computation of $r^i_j$ for all $j$, it is more efficient to compute $R^{i-1}(\alpha)$ with the most significant bit calculated first. Hence, in the above algorithm, we compute $R^i(\alpha)$ starting from the MSB toward the LSB. A 2-D systolic architecture[8] for the implementation of this multiplication algorithm is illustrated in Figure 1. In this figure, the data dependency between bits and between iterations is shown. The cell on the $j$th column and $i$th row computes the $j$th bit of $R^i(\alpha)$ by computing

$$r^i_j = r^{i-1}_{m-1}f_j \oplus r^{i-1}_{j-1} \oplus a_{m-i}b_j, \tag{6}$$

where $r^{i-1}_{m-1}$ is the most significant coefficient of $R^{i-1}(\alpha)$.

The above architecture can be further improved by partitioning the main operation of the bit-level algorithm. In Equation 6, note that $f_j$, $a_{m-i}$, and $b_j$ could be available in advance. As $r^{i-1}_{m-1}$ is ready, $r^{i-1}_{m-1}f_j \oplus a_{m-i}b_j$ can be calculated immediately. Hence, the computation of $r^i_j$ can be partitioned into two simpler operations to shorten the clock period. These two operations are

$$p^i_j = r^{i-1}_{m-1}f_j \oplus a_{m-i}b_j, \tag{7}$$

and

$$r^i_j = r^{i-1}_{j-1} \oplus p^i_j. \tag{8}$$

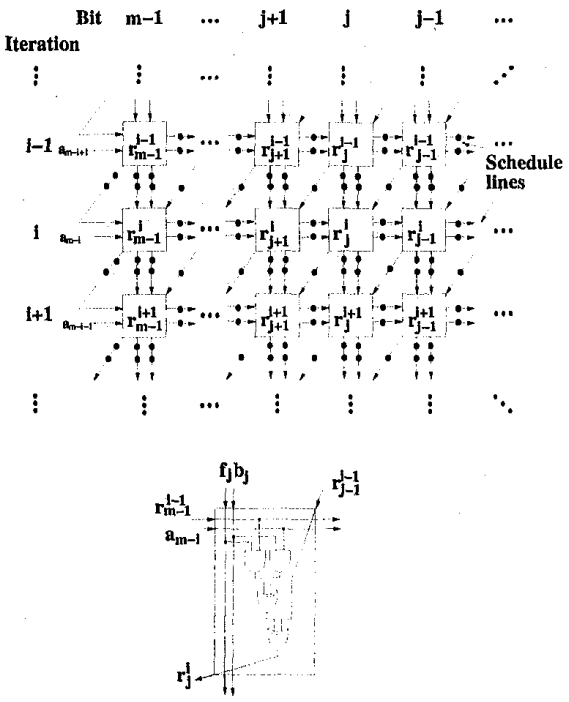


Fig. 1. A 2-D systolic architecture for the multiplication in $GF(2^m)$.

This partitioned architecture is shown in Figure 2. Two kinds of simple cells are used respectively to calculate $p^i_j$ and $r^i_j$; the upper layer cells compute $p^i_j$ while the lower layer cells compute $r^i_j$.

Since there is no carry propagation problem in the same iteration, we can merge every $n$ cells together in a specific way, as shown in Figure 3. In this figure, we group $r^i_j$, $r^i_{j-1}$, ..., $r^i_{j-n-1}$, $p^{i+1}_j$, $p^{i+1}_{j-1}$, ..., and $p^{i+1}_{j-n-1}$ together, $r^i_{j-n-2}$, $r^i_{j-n-3}$, ..., $r^i_{j-2n-1}$,

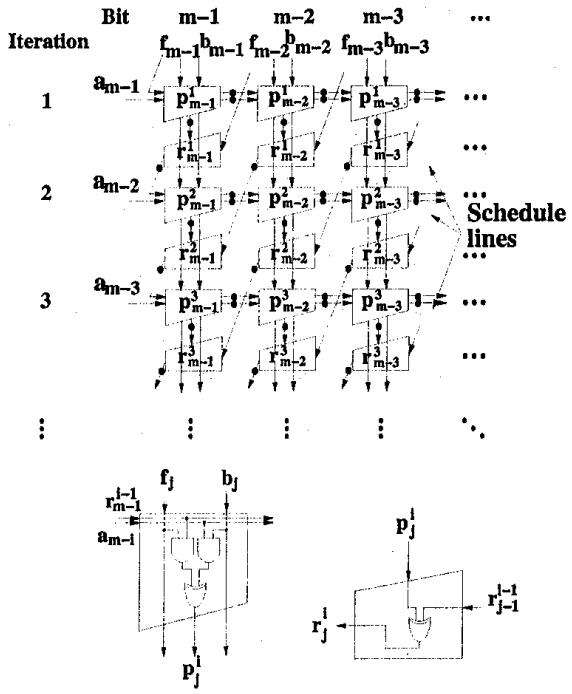


Fig. 2. Dependence graph for partitioned architecture.

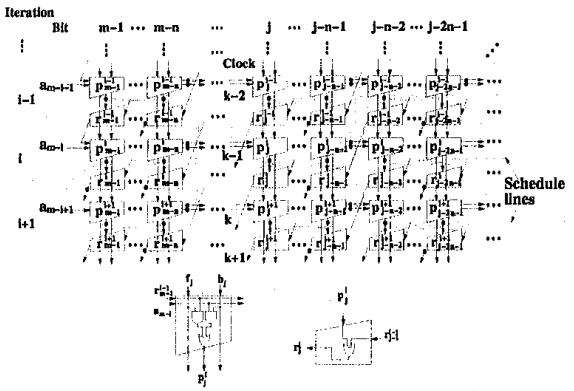


Fig. 3. 2-D Dependence graph of the merged architecture.

$p^{i+1}_{j-n-2}$, $p^{i+1}_{j-n-3}$, ..., and $p^{i+1}_{j-2n-1}$ together, ... and so on. With this arrangement, if $r^i_j$, ..., $r^i_{j-n-1}$, $p^{i+1}_j$, ..., and $p^{i+1}_{j-n-1}$ are computed in the $k$th clock cycle and $r^i_{j-n-2}$, ..., $r^i_{j-2n-1}$, $p^{i+1}_{j-n-2}$, ..., and $p^{i+1}_{j-n-2}$ are computed in the $(k+1)$th clock cycle, then $r^{i+1}_j$, $r^{i+1}_{j-1}$, ..., $r^{i+1}_{j-n-2}$, $p^{i+2}_j$, $p^{i+2}_{j-1}$, ..., and $p^{i+2}_{j-n-1}$ can also be calculated in the $(k+1)$th clock cycle. It looks infeasible, at the first glance, to have $r^{i+1}_{j-n-1}$ calculated in the $(k+1)$th clock cycle since the computation of

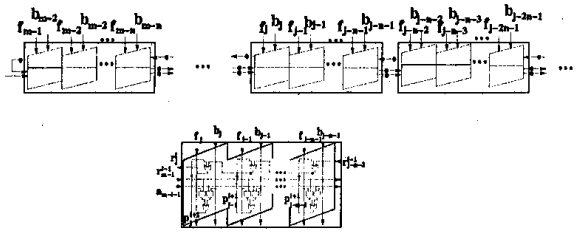

Fig. 4. 1-D Dependence graph of the merged architecture.

$r^{i+1}_{j-n-1}$ depends on $r^i_{j-n-2}$. However, assume we remove the latch between $r^i_{j-n-2}$ and $r^{i+1}_{j-n-1}$; that is, in the $(k+1)$th clock cycle, $r^{i+1}_{j-n-1}$ is to be computed right after $r^i_{j-n-2}$ is computed. Then, all $r^{i+1}_j$, $r^{i+1}_{j-1}$, ..., $r^{i+1}_{j-n-1}$, $p^{i+2}_j$, $p^{i+2}_{j-1}$, ..., and $p^{i+2}_{j-n-1}$ can be calculated in the $(k+1)$th clock cycle without time conflict. Similarly, we remove the latch between $r^i_{j-2n-1}$ and $r^{i+1}_{j-2n-2}$, the latch between $r^i_{j-3n-1}$ and $r^{i+1}_{j-3n-2}$, ..., and so on. The removal of these latches only forms a local data propagation and won't create any global propagation path. Moreover, after merging the cells, we can remove lots of the latches that have been used to keep $a_i$'s, $r^i_{m-1}$'s, and the control signals. This removal can save a huge amount of latches. The area and power consumption of the architecture can thus be greatly reduced after eliminating these latches.

After merging the cells, in iteration $i$, the $k$th general cell of the merged architecture computes the following operations:

$$
\begin{aligned}
&r^i_j = r^{i-1}_{j-1} \oplus p^i_j, \\
&r^i_{j-1} = r^{i-1}_{j-2} \oplus p^i_{j-1}, \\
&\cdots \\
&r^i_{j-n-1} = r^{i-1}_{j-n-2} \oplus p^i_{j-n-1}, \\
&p^{i+1}_j = r^i_{m-1} f_j \oplus a_{m-i-1} b_j, \\
&p^{i+1}_{j-1} = r^i_{m-1} f_{j-1} \oplus a_{m-i-1} b_{j-1}, \\
&\cdots \\
&p^{i+1}_{j-n-1} = r^i_{m-1} f_{j-n-1} \oplus a_{m-i-1} b_{j-n-1},
\end{aligned}
$$

where $j = nk$.

Figure 3 shows the implementation of this algorithm. The upper layer cells compute $r^{i-1}_j$, $r^{i-1}_{j-1}$, ..., and $r^{i-1}_{j-n-1}$ while the lower layer cells compute $p^i_j$,

$p^i_{j-1}$, ..., and $p^i_{j-n-1}$. The computation sequence is shown in Figure 5. In the MSB cell, $r^i_{m-1}$ is calculated and then immediately used for the computation of $p^{i+1}_{m-1}$ and $p^{i+1}_{m-2}$. This temporal dependency, which includes one XOR operation for $r^i_{m-1}$ and one AND plus one XOR operation for $p^{i+1}_{m-1}$ or $p^{i+1}_{m-2}$, forms the critical path of this architecture. The detail comparisons of this architecture with some other proposed architectures will be presented later in the next section.

Time

$$
\begin{array}{llll}
r^{i-1}_{m-1} & r^{i-1}_{m-n-1} & r^{i-1}_{m-2n-1} & \cdots \\
r^{i-1}_{m-2} & r^{i-1}_{m-n-2} & r^{i-1}_{m-2n-2} & \cdots \\
\cdot & \cdot & \cdot & \cdots \\
r^{i-1}_{m-n} & r^{i-1}_{m-2n} & r^{i-1}_{m-3n} & \cdots \\
p^{i}_{m-1} & p^{i}_{m-n-1} & p^{i}_{m-2n-1} & \cdots \\
p^{i}_{m-2} & p^{i}_{m-n-2} & p^{i}_{m-2n-2} & \cdots \\
\cdot & \cdot & \cdot & \cdots \\
p^{i}_{m-n} & p^{i}_{m-2n} & p^{i}_{m-3n} & \cdots \\
 & r^{i}_{m-1} & r^{i}_{m-n-1} & r^{i}_{m-2n-1} \quad \cdots \\
 & r^{i}_{m-2} & r^{i}_{m-n-2} & r^{i}_{m-2n-2} \quad \cdots \\
 & \cdot & \cdot & \cdot \quad \cdots \\
 & r^{i}_{m-n} & r^{i}_{m-2n} & r^{i}_{m-3n} \quad \cdots \\
 & p^{i+1}_{m-1} & p^{i+1}_{m-n-1} & p^{i+1}_{m-2n-1} \quad \cdots \\
 & p^{i+1}_{m-2} & p^{i+1}_{m-n-2} & p^{i+1}_{m-2n-2} \quad \cdots \\
 & \cdot & \cdot & \cdot \quad \cdots \\
 & p^{i+1}_{m-n} & p^{i+1}_{m-2n} & p^{i+1}_{m-3n} \quad \cdots
\end{array}
$$

Fig. 5. Timing sequence of $p^k_l$'s and $r^k_l$'s.

## III. Comparisons for Multiplication over $GF(2^m)$

In this section, we compare our design with several systolic arrays and a semi-systolic array for their performance in computing multiplications over $GF(2^m)$. Figure 6 shows the time-area complexity for different $n$ when we merge every $n$ cells together. We can know from this figure that $n = 4$ has the lowest complexity. For a fair comparison, we add the serial output circuits to our architecture as shown in Figure 7 and add a control signal to reset all flip flops. The comparisons of the average speed, area, and area-time complexity are shown in Table I. The timing and area estimation is based on the delay and gate count information of TSMC $0.35\mu$ cell library. Since there is no 5-input XOR gate in the library, we estimate its delay and gate count ourselves.

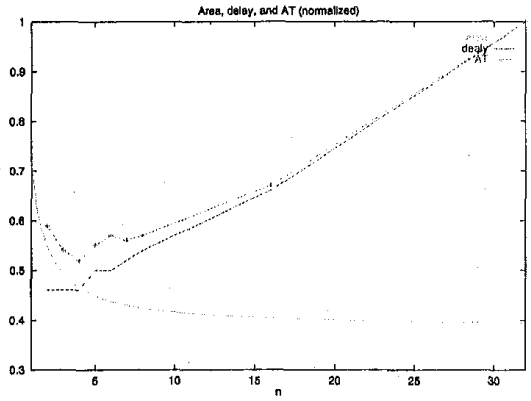


Fig. 6. AT complexity for different $n$ .

Since there is an inherent one-clock-cycle gap problem in a bit-by-bit systolic architecture, all the bit-by-bit structures (including [7], [13], and [8]) need $2m$ clock cycles per operation for dependent multiplications and $m$ clock cycles per operation for independent multiplications. On the other hand, for most architectures which are not bit-by-bit structure, like our architecture and Mekhallalati's systolic-II design [19], the average time becomes $m$ clock cycles per operation for both dependent and independent multiplications. Note that, however, Mekhallalati's systolic-Ib design [19] needs $2m$ clock cycles per operation for both dependent and independent multiplications. This is due to the fact that when this architecture is computing a multiplication, no other multiplication can be computed in parallel.

In Table I, we can see [7] and [15] have the shortest delay among all these architectures. This is because all these designs have applied partitioning on their architectures. However, due to their complicated ways to handle the one-clock-cycle gap problem, these two

| Author | Cycles | Delay per cell(ns) | Total(ns) | AT |
|---|---|---|---|---|
| | # of cells | Area per cell (gate count) | Total area | ($m^2$ ns) |
| Yeh et al. [7] | $m^*$ | $T_{and} + T_{xor} + T_{latch} \approx 1.66$ | $\approx 1.66m^*$ | 150* |
| | $m$ | $3A_{and} + 2A_{xor} + 10A_{latch} + A_{mux} = 90.5$ | $90.5m$ | |
| Wang et al. [8] | $m^*$ | $T_{and} + T_{3xor} + T_{latch} \approx 1.84$ | $\approx 1.84m^*$ | 163* |
| | $m$ | $3A_{and} + A_{3xor} + A_{mux} + 10A_{latch} = 88.5$ | $88.5m$ | |
| Hasan et al. [13] | $m^*$ | $T_{and} + T_{xor} + T_{latch} \approx 1.66$ | $\approx 1.66m^*$ | 163* |
| | $m$ | $3A_{and} + 2A_{xor} + A_{mux} + 11A_{latch} = 98$ | $98m$ | |
| Systolic-Ib [11] | $2m$ | $T_{and} + T_{5xor} + T_{mux} + T_{latch} \approx 2.65$ | $\approx 5.3m$ | 182 |
| | $0.5m$ | $4A_{and} + A_{5xor} + 7A_{latch} = 68.5$ | $34.3m$ | |
| Systolic-II [11] | $m$ | $T_{and} + T_{5xor} + T_{mux} + T_{dmux} + T_{latch} \approx 3.1$ | $\approx 3.1m$ | 141 |
| | $0.5m$ | $4A_{and} + 4A_{xor} + 2A_{mux} + 9A_{latch} = 95.5$ | $47.8m$ | |
| 1-D Jain et al.[6] | $m$ | $T_{and} + T_{xor} + T_{latch} + T_{broadcast_{155}} \approx 3.3$ | $\approx 3.3m$ | 160 |
| | $m$ | $2A_{and} + 2A_{xor} + 5A_{latch} = 48.5$ | $48.5m$ | |
| Our architecture(n=4) | $m$ | $T_{and} + 2T_{xor} + T_{latch} + T_{broadcast_4} \approx 2.3$ | $\approx 2.3m$ | 88 |
| | $0.25m$ | $8A_{and} + 8A_{xor} + 4A_{mux} + 14A_{latch} = 152$ | $38m$ | |

$T_{and}$: Delay of a two-input AND gate (0.33ns).
$T_{xor}$: Delay of a two-input XOR gate (0.47ns).
$T_{mux}$: Delay of a two-input Multiplex gate (0.45ns).
$T_{dmux}$: Delay of a two-output Demultiplex gate (0.45ns).
$T_{3xor}$: Delay of a three-input XOR gate (0.64ns).
$T_{5xor}$: Delay of a five-input XOR gate (1ns).
$T_{latch}$: Delay of a latch gate (0.87ns).
$T_{broadcast_4}$: Delay of 4-bit broadcasting signal (0.18ns).
$T_{broadcast_{155}}$: Delay of 155-bit broadcasting signal (1.63ns).

$A_{and}$: Are of a two-input AND gate (1.5).
$A_{xor}$: Area of a two-input XOR gate (4).
$A_{mux}$: Area of a two-input Multiplex gate (3).
$A_{dmux}$: Are of a two-output Demultiplex gate (3).
$A_{3xor}$: Are of a three-input XOR gate (6).
$A_{5xor}$: Are of a five-input XOR gate (10).
$A_{latch}$: Are of a latch gate (7.5).
$m$ is the order of the finite field $GF(2^m)$.
*: double if the *dependent* multiplications are computed.

TABLE I

COMPARISONS FOR COMPUTING MULTIPLICATIONS.

designs consume larger area. In fact, the computation speed of our architecture is the fastest when calculating independent multiplications. Moreover, as mentioned before, a huge amount of latches can be removed after merging. Therefore, the area size and power consumption are greatly reduced in Mekhallalati's designs and our design. Here, we also compare our architectures with a recent semi-systolic design [6]. To have a fair comparison, we project Jain's design into a 1-D architecture. In this 1-D semi-systolic architecture, the delay of the broadcast signal for a 155-bit design is estimated to be about 1.63ns without considering the wire loading. This delay could be further shortened, but with more area consumed. In this table, we can see that our architecture still offers a smaller area-time complexity than Jain's design.

## IV. Conclusion

In this paper, we propose a new low-complexity architecture to increase the performance of multiplication over $GF(2^m)$ by increasing the number of pipeline stage to shorten the clock cycle period and by merging the partitioned cells to avoid the one-clock-cycle gap problem. This architecture offers a low area-time complexity when used as the main core

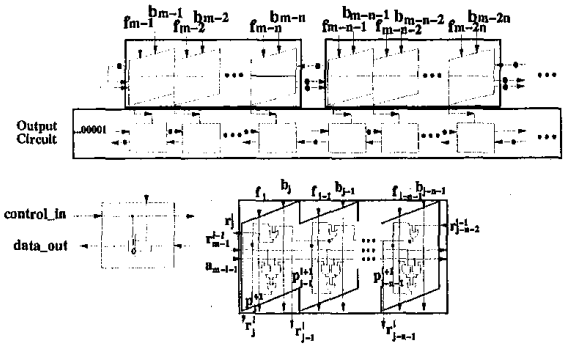


Fig. 7. Our architecture with serial output circuits.

in an elliptic curve cryptosystem.

## REFERENCES


[1] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, November 1976.

[2] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, February 1978.

[3] D. Denning, "Cryptography and data security", Addison-Wesley, 1982.

[4] S. Bandyopadhyay and A. Sengupta, "Algorithms for multiplication in Galois field for implementation using systolic arrays," *IEE Proceedings*, Vol. 135, Pt. E, No. 6, pp. 336–340. November, 1988.

[5] J.-H. Guo and C.-L. Wang, "Digit-serial systolic multiplier for Finite Fields $GF(2^m)$, *IEE. Proc.-Comput. Digit. Tech.* Vol. 145, No. 2, PP. 143–148, March, 1998.

[6] S. K. Jain, L. Song, and K. K. Parhi, "Efficient SemiSystolic Architectures for Finite-Field Arithmetic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 6, No. 1, pp. 101–113. March, 1998.

[7] C.-S. Yeh, I. S. Reed, and T. K. Troung, "Systolic Multipliers for Finite Fields $GF(2^m)$," *IEEE Transactions on Computers*, Vol. c-33, pp. 357-360, 1984.

[8] C.-L. Wang and J.-L. Lin, "Systolic Array Implementation of Multipliers for Finite Fields $GF(2^m)$," *IEEE Transactions on Circuits and Systems*, Vol. 38, No. 7, pp. 796-800, July, 1991.

[9] S.-W. Wei, "VLSI Architectures for Computing Exponentiations, Multiplicative Inverses, and Divisions in $GF(2^m)$," *IEEE Transactions on Circuits and Systems - II: Analogs and Digital Signal Processing*, Vol. 44, No. 10, pp. 847–855, October, 1997.

[10] J.-H. Guo and C.-L. Wang, "Systolic Array Implementation of Euclid's Algorithm for Inversion and Division in $GF(2^m)$," *IEEE International Symposium on Circuits and Systems*, pp. 481–484, 1996.

[11] M. C. Mekhallalati, M. K. Ibrahim, and A.S. Ashur, "New low complexity bidirectional systolic structures for serial multiplication over the finite field $GF(q^m)$," *IEE Proc.-Circuits Devices System.*, vol. 145, no. 1, pp. 55–60, February, 1998.

[12] A. Ghafoor and A. Singh, "Systolic architecture for finite filed exponentiation", in *IEE Proceedings*, Vol.136, Pt. E, No.. 6, November, 1989.

[13] M. A. Hasan and V. K. Bhargava, "Bit-Serial Systolic Divider and Multiplier for Finite Fields $GF(2^m)$," *IEEE Transactions on Computers*, Vol. 41, No. 8, pp. 972-980, July, 1992.

[14] C.-L. Wang, "A Systolic Exponentiator for Finite Field $GF(2^m)$" in *Proceedings of the 34th Midwest Symposium on Circuits and Systems*, Vol.1, pp. 279–282.

[15] M. A. Hasan and V. K. Bhargava, "Division and bit-serial multiplication over $GF(q^m)$," *IEE Pro. Comput. Digit*, 139, (3), pp. 230-236, May, 1992.

[16] P. Scott, S. Taverns, and L. Peppard, "A fast multiplier for $GF(2^m)$," *IEEE Trans. Comput.*, 37, (6), pp. 735–739, 1988.

[17] S.Y. Kung, "On supercomputing with systolic/wavefront array processors", *Proc. IEEE*, pp. 867–884, July, 1984.

[18] I. Hsu, T. Troung, L. Deustsch, and I. Reed, "A comparison of VLSI architecture of finite field multiplier using dual, normal or standard bases," *IEEE Trans. Comput.* 37, (6), pp. 735–739, 1998.

[19] M. C. Mekhallalati and A.S. Ashur, "Novel Structures for Serial Multiplication over the Finite Field $GF(2^m)$," in *VLSI Signal Processing IX*, pp. 65–74. October, 1996.

[20] G. B. Agnew, R. C. Mullin, and S. A. Vanstone, "An Implementation of Elliptic Curve Cryptosystems Over $F_{2^{155}}$," *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 5, pp. 804–813, June, 1993.