FPGA Accelerated Tate Pairing Cryptosystems over Binary Fields

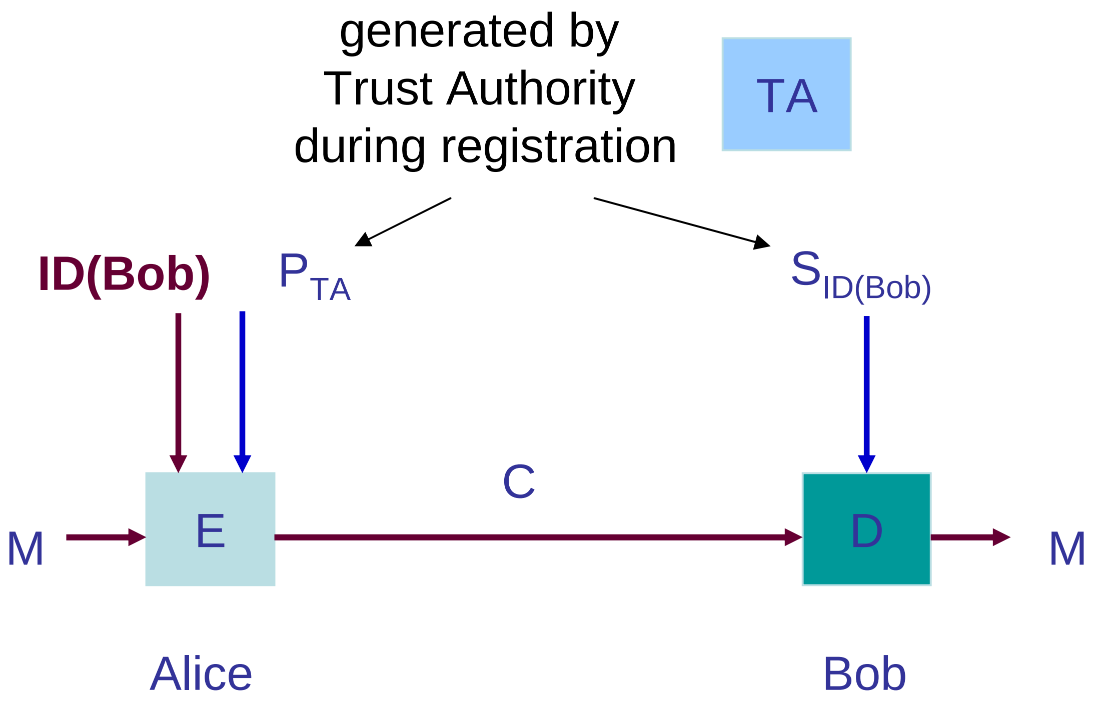
Chang Shu [1], Soonhak Kwon [2], and Kris Gaj [1]

[1] Dept. of ECE, George Mason University
Fairfax VA, USA
[2] Dept. of Mathematics, Sungkyukwan University
Suwon, Korea

Overview

- Introduction
  - Identity based encryption (IBE) scheme
  - Pairing based cryptography
- FPGA implementations
  - Algorithms
  - Architectures
  - Timing diagram
  - Results
- Comparison with software
- Comparison with previous work for comparable schemes
- Conclusions

Identity Based Encryption Scheme



Notations	
TA	trust authority
P_{TA}	TA’s public key
S	TA’s private key
ID(Bob)	Bob’s identity
$S_{ID(Bob)}$	Bob’s private key
M	Message
C	Cipher text
E	Encryption
D	Decryption

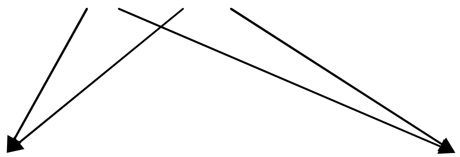
Features of IBE:

1. Use the receiver’s ID as a public key for encryption
2. A third trust authority party is involved in generating the receiver’s private key associated with one’s ID.

Pairing Based Cryptography (1)

- Pairing is a map between groups, where $e: G_1 \, x \, G_1 \rightarrow G_2$, $G_1 = E(F_q)$ and $G_2 = F_{qk}$

- The most important property of this map is bilinearity $e(aP, bQ) = e(P, Q)^{ab}$

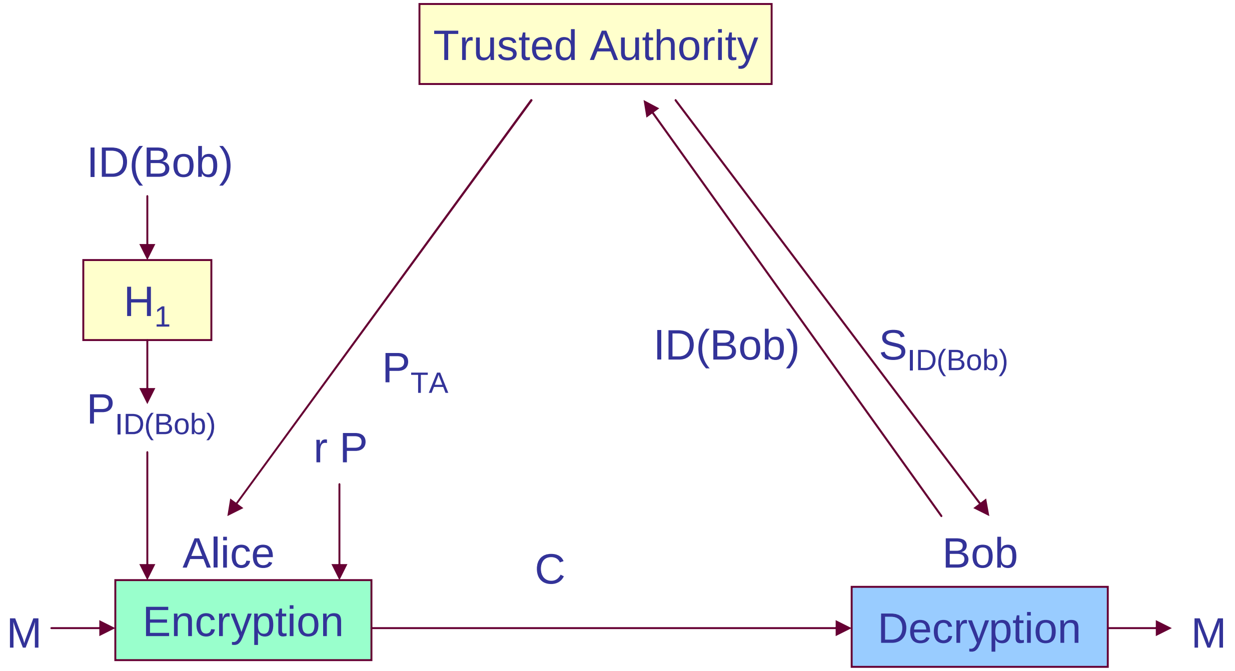


a, b: integers

P,Q: points on elliptic curves

- In practice, Tate or Weil pairing are used.

Pairing Based Cryptography (2)



Trusted Authority

ID(Bob)

H_1

$P_{ID(Bob)}$

P_{TA}

r P

ID(Bob)

$S_{ID(Bob)}$

Alice

C

Bob

M

Encryption

Decryption

M

s: secret value

P: public value

$P_{TA} = s\,P$
public key of TA

$P_{ID(Bob)} = H_1(ID(Bob))$
Bob’s public key

$S_{ID(Bob)} = s\,P_{ID(Bob)}$
Bob’s private key

r: random number

$C = (U, V) = (rP, M + H_2(e(P_{ID(Bob)}\,, P_{TA}\,)^r)$

$M = (V + H_2(e(S_{ID(Bob)}, U\,))$

By bilinearity, $e(S_{ID(Bob}, U) = e(sP_{ID(Bob)}, rP) = e(P_{ID(Bob)}, sP)^r = e(P_{ID(Bob)}\,, P_{TA}\,)^r$

Two Selected Algorithms (1)

Algorithm 1:

```
Input: $P=(x,y), Q=(\alpha,\beta)$
        $x,y,\alpha,\beta \in F_{2^m}$
Output: $C=\tau(P,Q)$ , where $C \in F_{2^{4m}}$
    $C \leftarrow 1,$
    $\alpha \leftarrow \alpha^4,\ \beta \leftarrow \beta^4,\ v \leftarrow x^2+1,\ \theta \leftarrow \alpha \cdot v$
    $u \leftarrow x^2+y^2+b+\frac{m-1}{2}$
for i=0 to m-1 do
    $A \leftarrow \beta+\theta+u+(\alpha+v)s+t$
    $C \leftarrow C^2$
    $C \leftarrow C \cdot A$
    $\alpha \leftarrow \alpha^4,\ \beta \leftarrow \beta^4,\ u \leftarrow u+v,$
    $v \leftarrow v+1,\ \theta \leftarrow \alpha \cdot v$
end for
$C \leftarrow C^{2^{2m}-1}$
```

Accumulative multiplication

Final powering

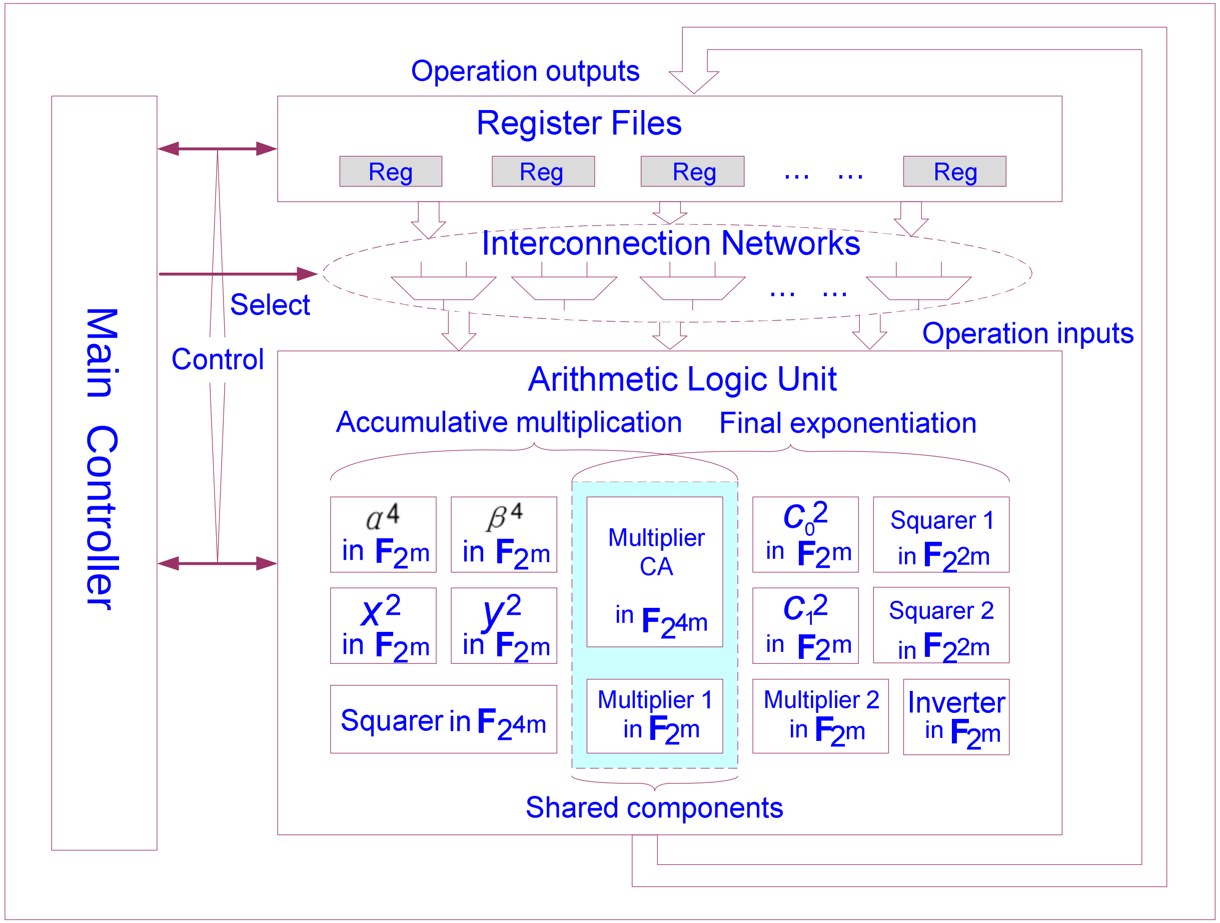
Algorithm 2:

```
Input: $P=(x,y), Q=(\alpha,\beta)$
        $x,y,\alpha,\beta \in F_{2^m}$
Output: $C=\tau(P,Q)$ , where $C \in F_{2^{4m}}$
    $C \leftarrow 1,\ \alpha \leftarrow \alpha^2+1, \beta \leftarrow \beta^2+1,$
    $u \leftarrow y+b+1, \theta \leftarrow \alpha \cdot v$
for i=0 to (m-1)/2 do
    $C \leftarrow C^2, C \leftarrow C \cdot A$
    if i<(m-1)2 then
        $\alpha \leftarrow \alpha^4,\ \beta \leftarrow \beta^4,\ u \leftarrow u+v+1,$
        $v \leftarrow v+1,\ \theta \leftarrow \alpha \cdot v$
    end if
end for
$A \leftarrow A+(\alpha^2+v+1)+s$
$C \leftarrow C \cdot A$
$C \leftarrow C^{MT},\ MT=(2^{2m}-1)(2^m \mp 2^{\frac{m+1}{2}}+1)(2^{\frac{m+1}{2}} \pm 1)$
```

Two Selected Algorithms (2)

- Two main stages: accumulative multiplication and final powering
- Only 7 underlying field multiplications are needed in each iteration for both algorithms
- All these multiplications can be completed simultaneously.
- Half iterations can be saved when using Algorithm 2, however the computation for final powering is much easier for Algorithm 1.
- In our work, we use polynomial basis for both algorithms, and we choose $F_{2^{239}}$ and $F_{2^{283}}$ for our single FPGA implementations

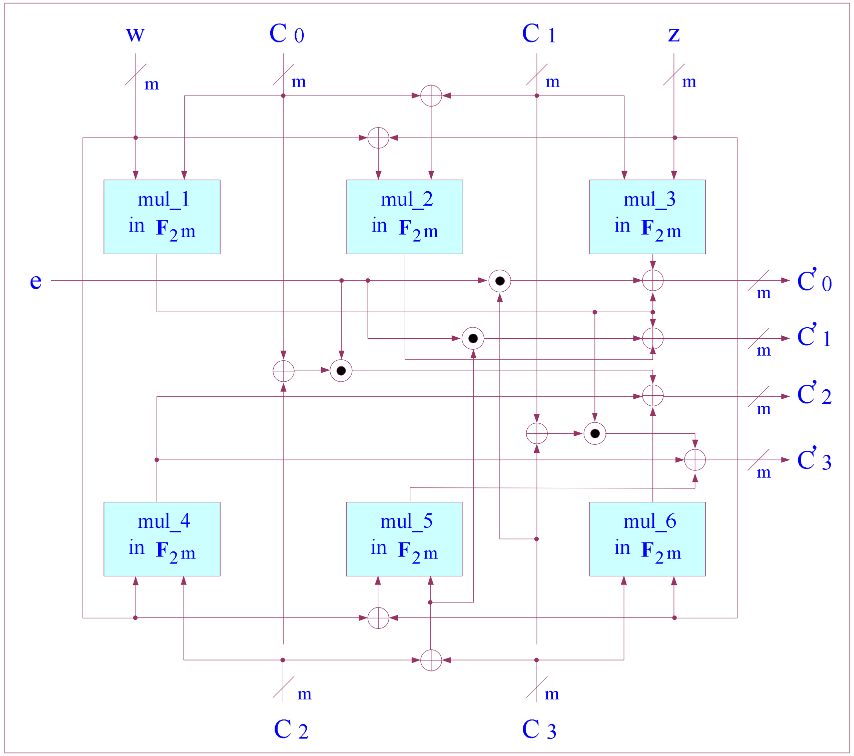
Top Architecture of Pairing Processor

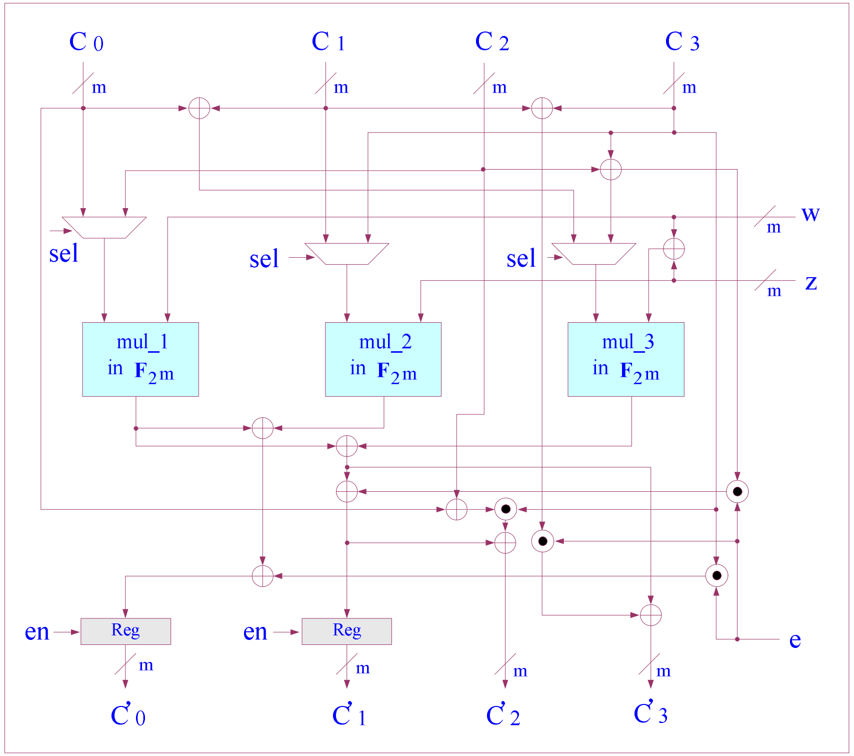

Operation outputs
Register Files
Reg
Reg
Reg
… …
Reg
Interconnection Networks
… …
Select
Control
Main Controller
Operation inputs
Arithmetic Logic Unit
Accumulative multiplication
Final exponentiation
α^4 in $\mathbf{F}_{2^m}$
β^4 in $\mathbf{F}_{2^m}$
Multiplier CA in $\mathbf{F}_{2^{4m}}$
c_0^2 in $\mathbf{F}_{2^m}$
Squarer 1 in $\mathbf{F}_{2^{2m}}$
x^2 in $\mathbf{F}_{2^m}$
y^2 in $\mathbf{F}_{2^m}$
c_1^2 in $\mathbf{F}_{2^m}$
Squarer 2 in $\mathbf{F}_{2^{2m}}$
Squarer in $\mathbf{F}_{2^{4m}}$
Multiplier 1 in $\mathbf{F}_{2^m}$
Multiplier 2 in $\mathbf{F}_{2^m}$
Inverter in $\mathbf{F}_{2^m}$
Shared components

Features:

- Hardwired logic instead of stored-programmed machine
- Iterative structure
- Register files for intermediate results
- Main controller designed as a finite state machine
- The extension field multiplier CA and Multiplier 1 are working for both stages

Two Possible Architectures for CA





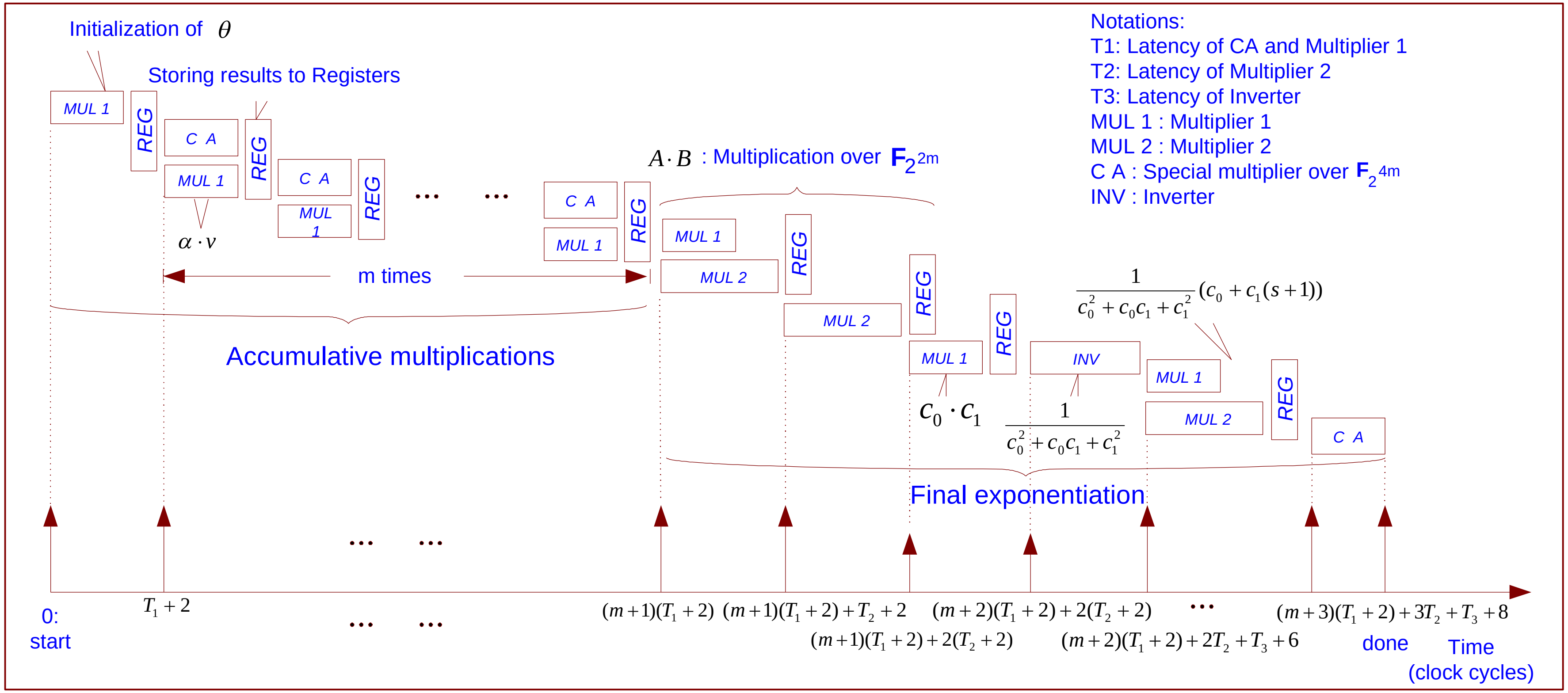
6 multipliers:

1. lower latency
2. larger area
3. lower product of latency by area

3 multipliers:

1. higher latency
2. smaller area
3. higher product of latency by area

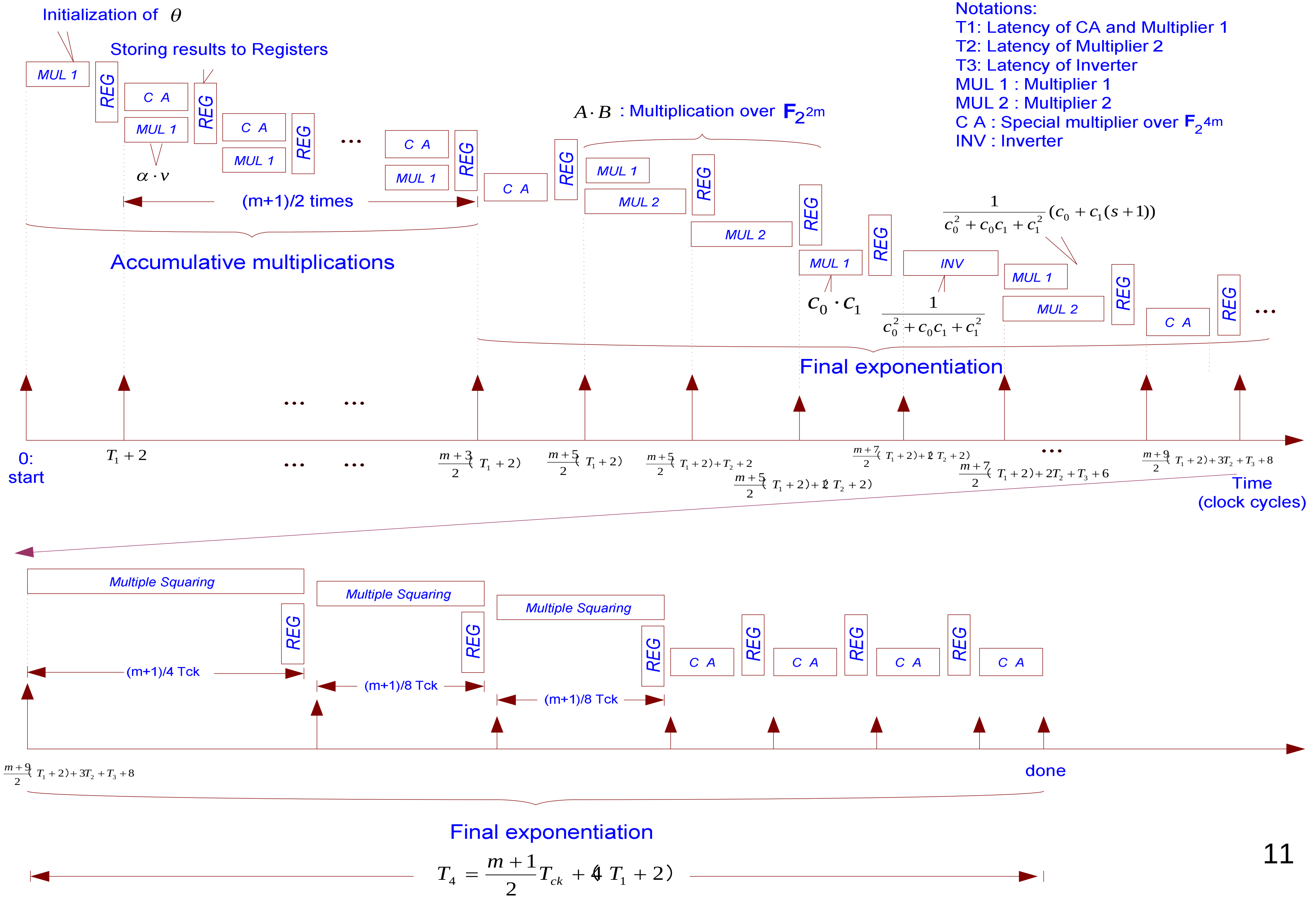
Timing Diagram for Algorithm 1



Notations:
T1: Latency of CA and Multiplier 1
T2: Latency of Multiplier 2
T3: Latency of Inverter
MUL 1 : Multiplier 1
MUL 2 : Multiplier 2
C A : Special multiplier over $\mathbf{F}_{2^{4m}}$
INV : Inverter

Timing Diagram for Algorithm 2

Notations:
T1: Latency of CA and Multiplier 1
T2: Latency of Multiplier 2
T3: Latency of Inverter
MUL 1 : Multiplier 1
MUL 2 : Multiplier 2
C A : Special multiplier over $\mathbf{F}_{2^{4m}}$
INV : Inverter

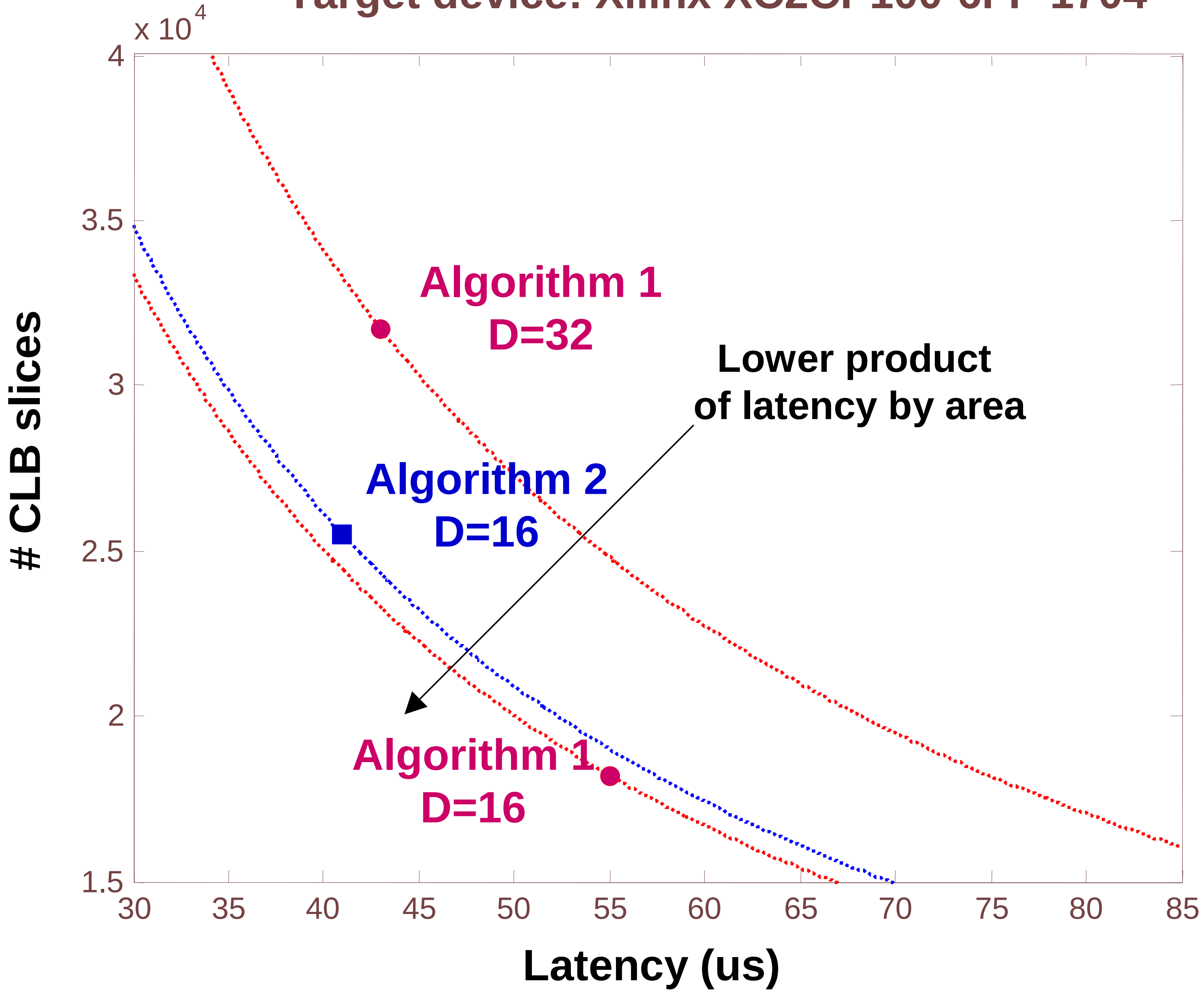

Initialization of θ
Storing results to Registers
MUL 1
REG
C A
MUL 1
REG
C A
MUL 1
REG
...
C A
MUL 1
REG
$\alpha \cdot v$
(m+1)/2 times
Accumulative multiplications
C A
REG
$A \cdot B$: Multiplication over $\mathbf{F}_{2^{2m}}$
MUL 1
MUL 2
REG
MUL 2
REG
MUL 1
$c_0 \cdot c_1$
REG
INV
$\frac{1}{c_0^2 + c_0 c_1 + c_1^2}$
$\frac{1}{c_0^2 + c_0 c_1 + c_1^2}(c_0 + c_1(s+1))$
MUL 1
MUL 2
REG
C A
REG
...
Final exponentiation
... ...
0:
start
$T_1 + 2$
... ...
$\frac{m+3}{2}(T_1+2)$
$\frac{m+5}{2}(T_1+2)$
$\frac{m+5}{2}(T_1+2)+T_2+2$
$\frac{m+5}{2}(T_1+2)+2(T_2+2)$
$\frac{m+7}{2}(T_1+2)+2(T_2+2)$
$\frac{m+7}{2}(T_1+2)+2T_2+T_3+6$
...
$\frac{m+9}{2}(T_1+2)+3T_2+T_3+8$
Time
(clock cycles)
Multiple Squaring
REG
(m+1)/4 Tck
Multiple Squaring
REG
(m+1)/8 Tck
Multiple Squaring
REG
(m+1)/8 Tck
C A
REG
C A
REG
C A
REG
C A
$\frac{m+9}{2}(T_1+2)+3T_2+T_3+8$
done
Final exponentiation
$T_4 = \frac{m+1}{2}T_{ck} + 4(T_1+2)$

Timing Diagrams

- The large extension field multiplier CA works for both stages, and two different sources of data
- Multiplier 1 also works for both stages
- Compared with CA and Multiplier 1, Multiplier 2 has smaller digit size.
- Only one subfield inversion is needed
- Compared with Algorithm1, one half of iterations can be saved for Algorithm 2, however more multiplications and squares are involved.
- For Algorithm 2, CA has 4 different sources of data, i.e., more levels of multiplexers are used.

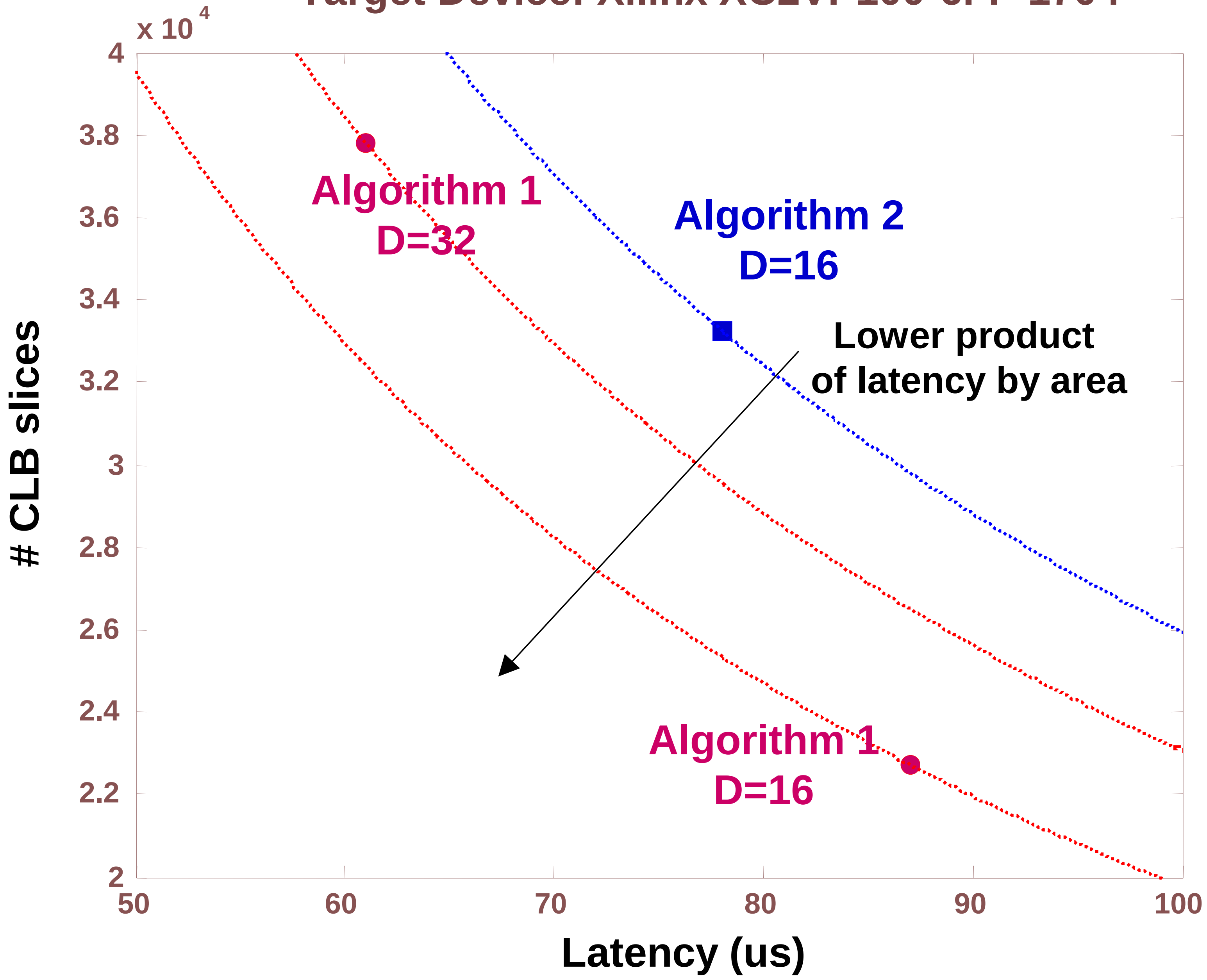
Implementation Results for GF(2^{239})

Target device: Xilinx XC2CP100-6FF-1704



Implementation Results for GF(2^{283})

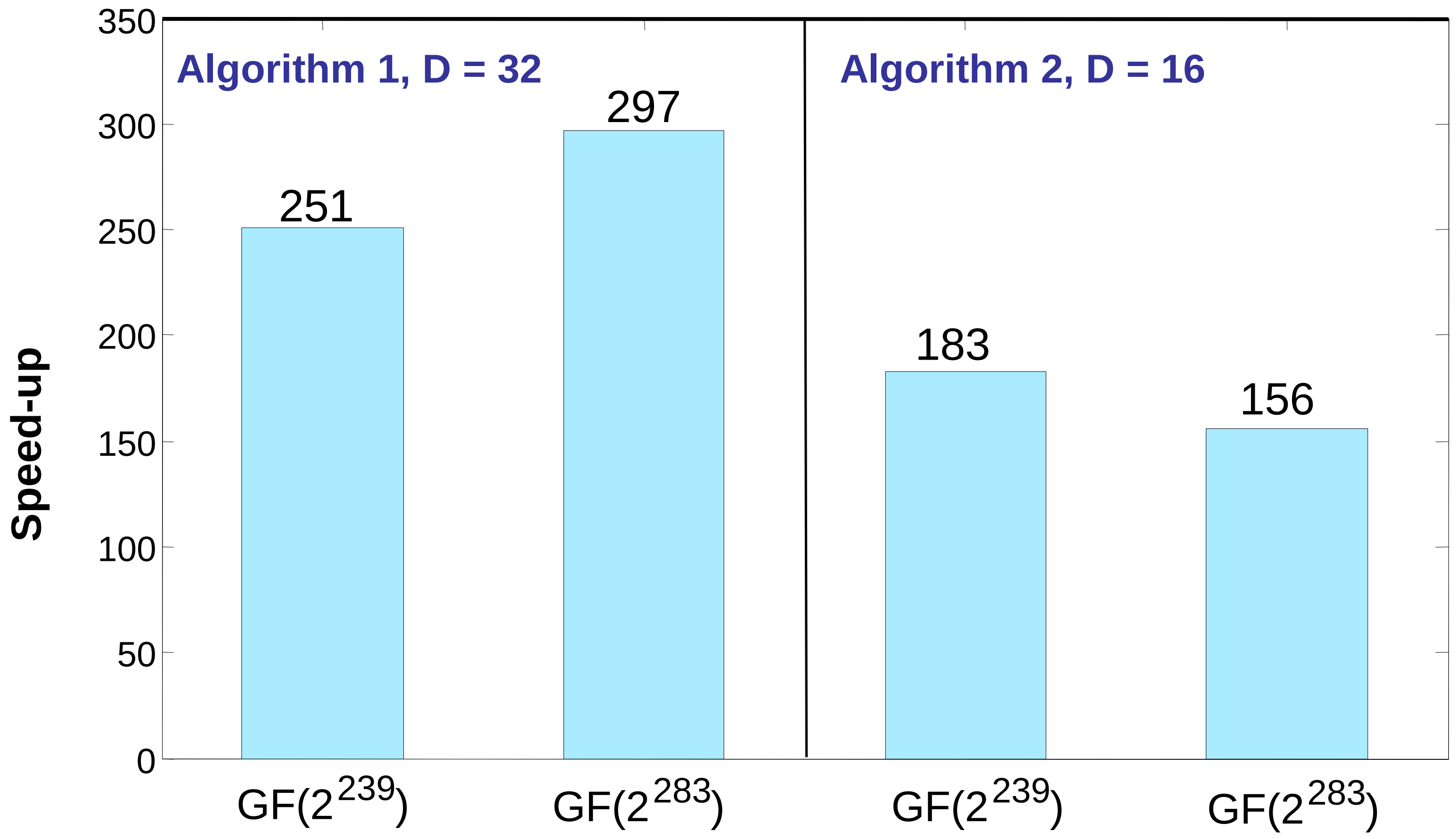
Target Device: Xilinx XC2VP100-6FF-1704



Speed-up over Software

Software Platform: Intel Xeon 2.8 GHz; C++ library,
LiDIA, for subfield arithmetic

Hardware Platform: Xilinx XC2VP100-6FF-1704

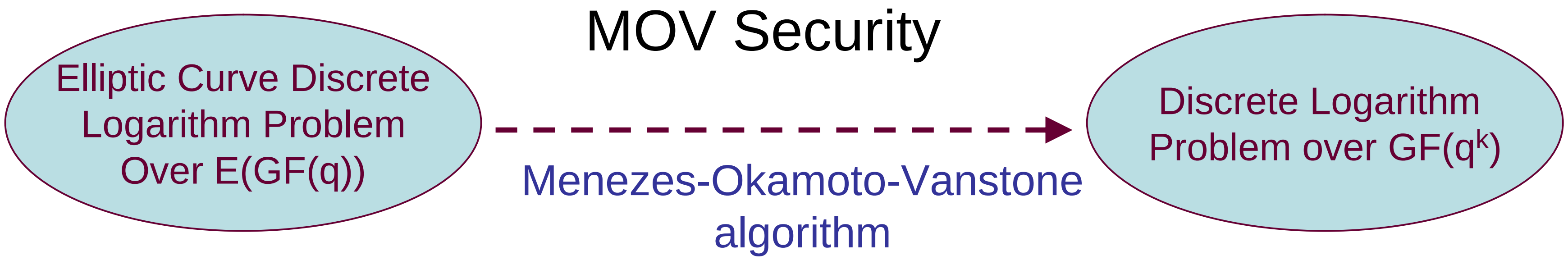


Comparison with Hardware Implementation of Comparable Schemes (1)

Comparable Schemes

	Binary elliptic (our scheme)	Cubic elliptic	Binary hyper-elliptic
Field F_q	$q = 2^m$	$q = 3^m$	$q = 2^m$
Curve	elliptic	elliptic	hyper-elliptic
Embedded Degree, k	4	6	12

Comparison with Hardware Implementation of Comparable Schemes (2)



MOV Security

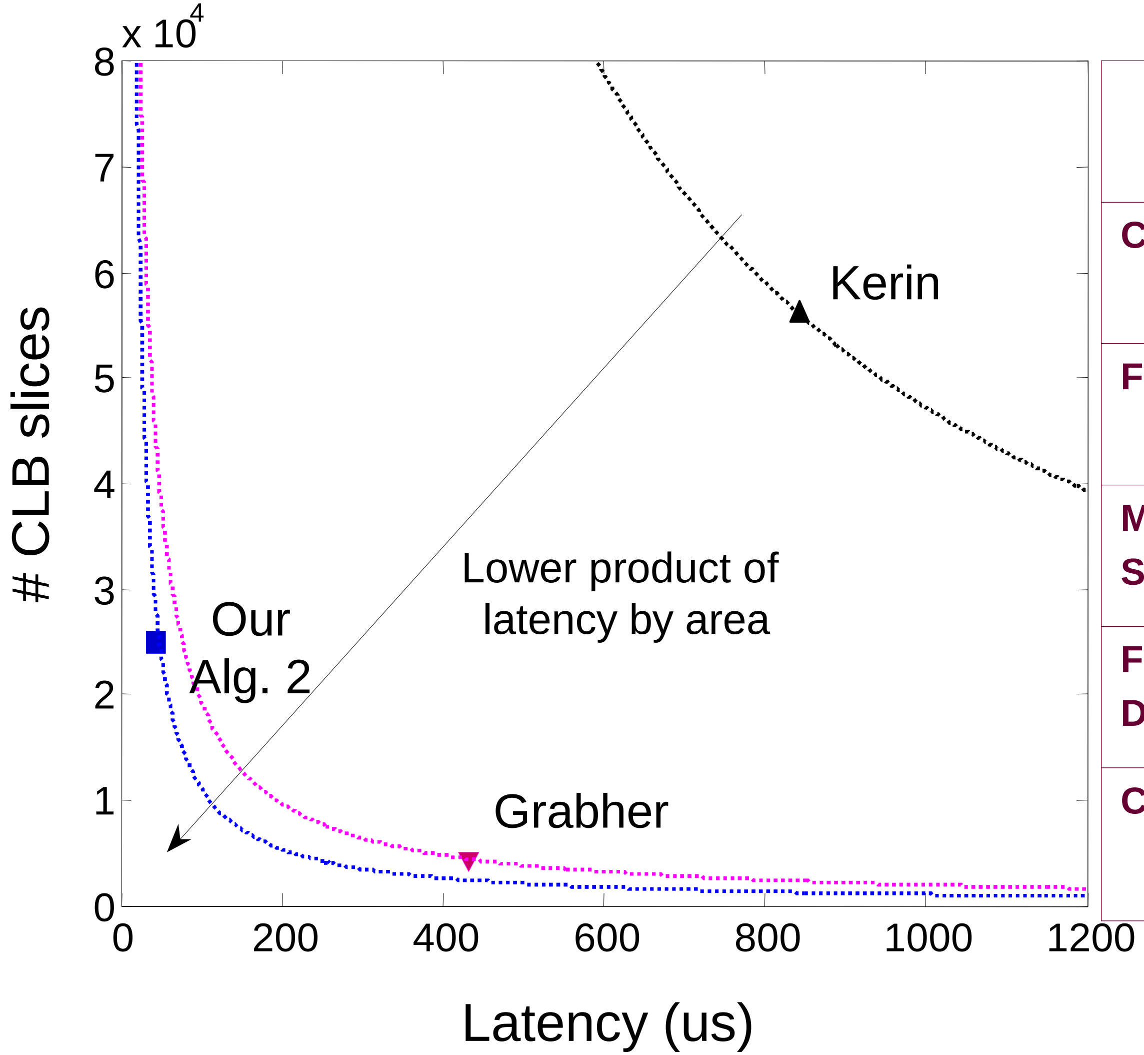
Elliptic Curve Discrete Logarithm Problem Over E(GF(q))

Menezes-Okamoto-Vanstone algorithm

Discrete Logarithm Problem over $GF(q^k)$

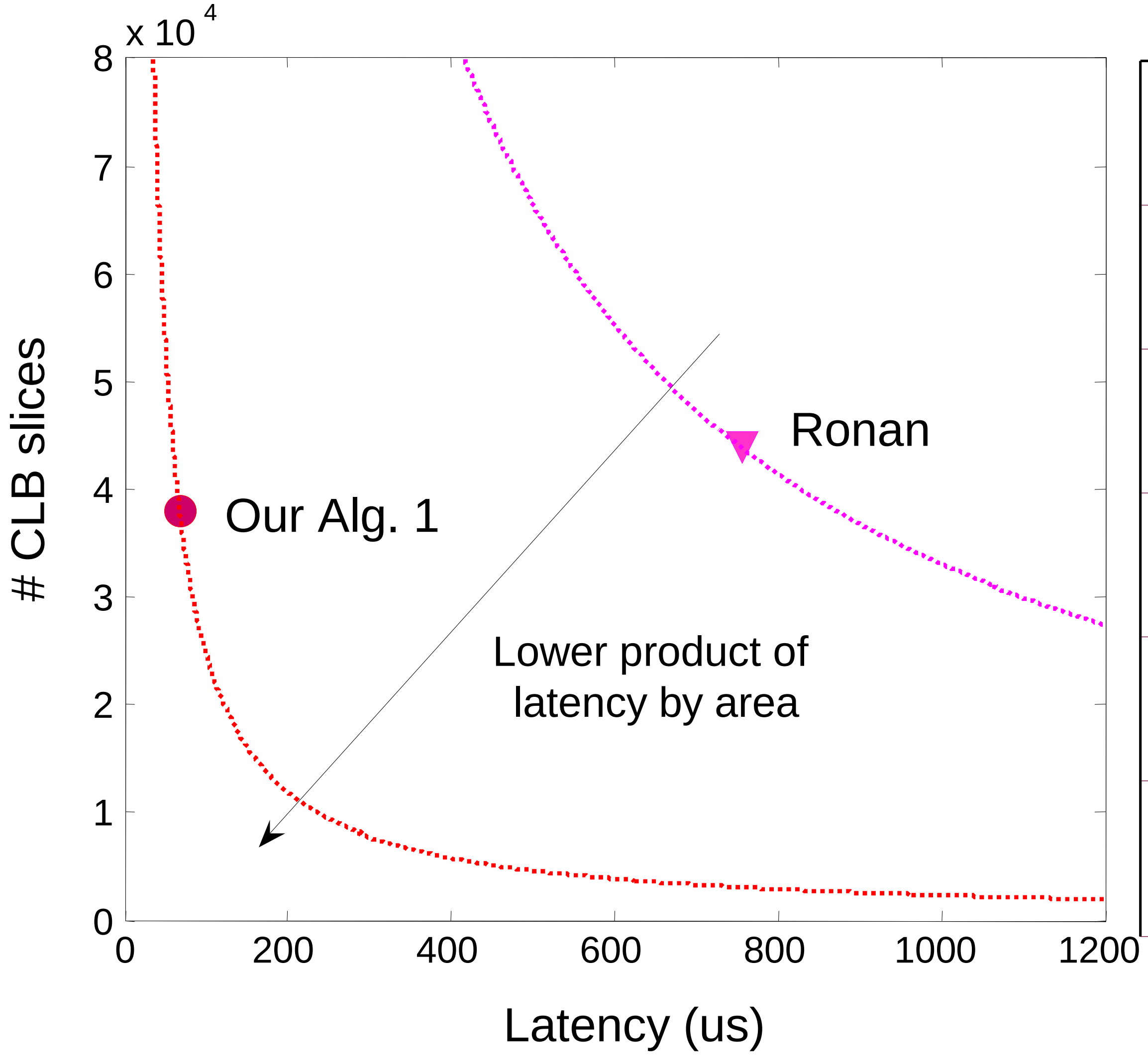
	Field F_q	MOV Security	
Binary elliptic	$q = 2^m$	$k \cdot m$	4 m
Binary hyper-elliptic			12 m
Cubic elliptic	$q = 3^m$	$k \cdot (\log_2 3) \cdot m$	9.5 m

Comparison with Hardware Implementation of Comparable Schemes (3)



	Our Alg. 2	Kerin	Grabher
Curves	Elliptic	Elliptic	Elliptic
Fields	GF(2^{239})	GF(3^{97})	GF(3^{97})
MOV Security	956	922	922
FPGA Device	XC2VP 100	XC2VP 125	XC2VP4 FF672
Controller	Hard wired logic	Hard wired logic	Micropr ocessor

Comparison with Hardware Implementation of Comparable Schemes (4)



	Alg. 1	Ronan
Curves	Elliptic	Hyper-elliptic
Fields	GF(2^{283})	GF(2^{103})
MOV Security	1132	1236
FPGA Device	XC2VP 100	XC2VP 125
Controller	Hardwired logic	Hardwired logic

Conclusions

- First FPGA implementation of the Tate pairing schemes for binary elliptic curves.
- Two algorithms improved, implemented and compared
- Algorithm 2 is faster, but its implementation takes more area
- Speed-ups in the range 150-300 demonstrated for Xilinx XC2VP100 vs. Xeon 2.8 GHz
- Our designs outperform existing implementation of comparable schemes in terms of the execution time by a factor 10-20, the product of latency by area by a factor 12-46.