High Speed ECC Implementation on FPGA over GF($2^m$)

Zia U. A. Khan and M. Benaissa
Department of Electronic and Electrical Engineering
University of Sheffield
Sheffield, UK

Int. Conf. on Field-programmable Logic and Applications (FPL)
2-4th September, 2015
Overview

Introduction

High Speed ECC

Comparison With The State Of Art

Conclusions
Elliptic Curve Cryptography (ECC)

- Public Key Cryptography (PKC) based on Elliptic Curve \( Q = kP \).
  Where, \( Q \) is public key, \( k \) is private key and \( p \) is a point of ECC.

- NIST Recommended Several Elliptic curves: Some area of applications => data transfer over internet, E-commerce, E-passport, sensor networks, RFID tags.

- Prime Field \((\text{GF}p)\) where \( p = 160 \) vs. Binary Field \((\text{GF}2^m)\) where \( m = 163 \).

Why ECC?
- **Smaller Key Sizes:** provide high security per bit.
- **Low bandwidth:** low transmission requirement.
- **Low storage:** small memory requirement.

We consider Binary Field due to as follows:
- Faster arithmetic circuit due to “Carry less” field operations (Multiplication, addition and squaring).
- Lower area complexity than prime field
- Suitable for hardware Implementation.

We consider Binary curve i.e. \( \text{GF}(2^{163}) \) for High Speed Implementation.
Elliptic Curve Cryptography (ECC)

- ECC based digital signature, ECDSA; Key agreement, ECDH etc.
- Main Operation of ECC is Point Multiplication: 
  \[ Q = kP = P + P + \ldots \ldots + P + P + P \], 
  where, 
  \( P \), a base point is a parameter of ECC protocol; 
  \( Q \), a point of Elliptic curve is user public key and 
  \( k \), an integer is its private key over the field.
- Point Addition \( Q = P + P \)
- Point Doubling \( Q = 2P \)
- Field Multiplication, Field Squaring, Field Addition, Field Inversion
Point multiplication \((Q = kP)\)

Scalar Point Multiplication is the main operation of Elliptic Curve Cryptography:

Point Multiplication Algorithm

Montgomery Point Multiplication Algorithm

Performance of the Elliptic curve cryptography depends on the point multiplication.

Point multiplications algorithm can affect the performance.

Advantages:
- Faster computation of \(Q = kP\)
- Inherent parallelism
- Partial Resistance of side-channel attack (Power attack)
- require less storage (only \(x\) and \(z\) coordinates are used)
High Speed ECC Design

Applications: Server end

Main requirement: Speed

How to achieve high speed in ECC?

To decrease Latency:

To increase Frequency

Point multiplication time

1. Reduce Latency (Clock cycles) for Point multiplication

2. Increase Frequency (Max. frequency in FPGA)

A) Use of larger digit serial/bit-parallel multiplier

B) Parallel operations: parallel multiplications

Reduce critical path delay using pipelining
High Speed ECC Design (cntd.)

- Drawbacks of Large digit size / bit-parallel multiplier
  - Long critical path delay (low operating frequency)
  - Large area requirement (optional for high speed design)

- To improve performance of the multiplier (to shorten critical path delay)
  - Pipelining stages improve frequency; hence, performance of the multiplier

- Maximum limit of pipelining stages. (each stage delays 1 clock cycle)
  - Pipelining stages may create bubble or idle clock cycles:
    - Due to data dependency in the point multiplications

- Idle clock cycle kills performance
  - Each clock cycle is important in the high speed design.

Introduction

Elliptic Curve Cryptography (ECC)
High Speed ECC Design (cntd.)

- How to remove data dependency to keep pipelining stages?

  - Smart pipelining
  - Careful scheduling of the point multiplication to avoid data dependency
Our proposed high speed ECC

Novel Full-precision Multiplier over GF($2^m$)

There are two stages pipelining:
- 1st stage pipelining is named “Segmented pipelining”

What is the Segmented pipelining?
- Divide $m$ into $w$ size segment
- Number segments, $n = m/w$
- Now, $n$ numbers of $MULGF2 (m \times w)$
- Result of each $MULGF2 = m+w$ bits
- Save each result in the $m+w$ bit register
- There $n$ number of $m+w$ bits registers
Our proposed high speed ECC (cntd.)

Novel Full-precision Multiplier over GF($2^m$)

There are two stages pipelining:
- 1st stage pipelining is named “Segmented pipelining”
- 2nd Stage pipelining after reduction;
  - The $n$ numbers $MULGF2$ results shifted and added ($\text{XOR}$)
  - We get $m \times m$ $MULGF2$ results = $2m-1$ bit
  - Full-precision reduction operation
  - Reduction result is $m$ bit output
  - Used $m$ bit register to save reduction result.

For $GF2GF(2^{163})$; we consider $w = 14$ bit: 12 number of 14 bit $MULGF2$ multipliers followed by reduction.
Our proposed high speed ECC
(cntd.)

Novel Full-precision Multiplier over GF($2^m$)

Comparison with bit parallel multiplier:
- Low critical path delay
  - Critical path delay can be modulated with the change of segment size, $d = w$
- Same area complexity (due to $m >> n$)
- Initial delay: 2 clock cycles; then, 1 clock cycle for each operation.

<table>
<thead>
<tr>
<th>Ref [8]</th>
<th>Latency, cc</th>
<th>Critical path delay</th>
<th>#XOR</th>
<th>#AND</th>
<th>#FFs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>$T_A + \left(\frac{m}{p} + \log_2(m)\right) T_X$</td>
<td>$m^2 + 3m$</td>
<td>$m^2$</td>
<td>$40m$</td>
</tr>
<tr>
<td>OursMul</td>
<td>1</td>
<td>$T_A + \left(\log_2\left(\frac{m}{d}\right)\right) T_X$ or $\left(\log_2\left(\frac{n+k}{d}\right)\right) T_X$</td>
<td>$m^2 + nm + 3m$</td>
<td>$m^2$</td>
<td>$n(m+d)+m$</td>
</tr>
</tbody>
</table>

$n$, #Segments $= \frac{m}{d}$, $w$ = segment size = $d$ = digit size, $k$ is the second higher order of irreducible polynomial. $T_A$ and $T_X$ are AND, and XOR gates delays respectively. $p = \#$ pipelining stages.
High Speed ECC

Point Multiplication

The modifications need

Our proposed high speed ECC (cntd.)

Parallel operation of Montgomery point multiplication

Algorithm 1: LD Montgomery Point Multiplication over GF(2^m)[6]

| INPUT: k = (k_{t-1}, ..., k_1, k_0) \_2 with k_{t-1} = 1, P = (x, y) ∈ E(F_{2^m}) |
| OUTPUT: kp |
| Initial Step: P(X_1, Z_1) ↔ (x, 1), 2P = Q(X_2, Z_2) ↔ (x^4 + b, x^2) |
| For i from t - 2 downto 0 do |
| If k_i = 1 then |

| Point addition: P(X_1, Z_1) = P(X_1, Z_1) + Q(X_2, Z_2) |
| Point Doubling: Q(X_2, Z_2) = 2Q(X_2, Z_2) |

1. Z_1 ← X_2, Z_1
2. X_1 ← X_1, Z_2
3. T ← X_1 + Z_1
4. X_1 ← X_1, Z_1
5. Z_1 ← T^2
6. T ← x · Z_1
7. X_1 ← X_1 + T
8. Return P(X_1, Z_1)

Conversion Step: x_3 ← X_1/Z_1 ; y_3 ← ((x^4 + x_1) / z_1) [((X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)) / (xZ_1Z_2)^{-1} + y.

- Main arithmetic operation: Multiplication
- 6 Muls, 5 Sqr's and 3 Adds operations

To achieve parallel operations:

- Need concurrent operations such as:
  - Mul || Sqr or Mul || add or Mul || Sqr || Add
- Need cascaded operation:
  - Mul → Sqr or Mul → Add or Mul → Add → Sqr
Our proposed high speed ECC (cntd.)

- Parallel operation of Montgomery point multiplication

\[ K_i = 1 \]

\[ M_1 = X_1.Z_2 \]
\[ M_2 = X_1.Z_2 \]
\[ M_3 = X_2.Z_1 \]

\[ M_4 = X_2.Z_1 \]
\[ M_5 = X_1.Z_2 \]
\[ M_6 = X_1.Z_2 \]

Using one two-stage pipelined multiplier (M):

- Latency for a loop operation (combined point addition and point doubling):
  - \( 6 + 1 = 7 \) clock cycles
Our proposed high speed ECC (cntd.)

- Parallel operation of Montgomery point multiplication

- Using *two two-stage pipelined multiplier* (*M*):
  - Latency for a loop operation (combined point addition and point doubling):
    - \(4 + 1 = 5\) clock cycles
Our proposed high speed ECC (cntd.)

- Parallel operation of Montgomery point multiplication

- Using three two-stage pipelined multiplier (M):
  - Latency for a loop operation (combined point addition and point doubling):
  - \(2 + 2 = 4\) clock cycles
Our proposed high speed ECC (cntd.)

- Parallel operation of Montgomery point multiplication

\[ K_i = 1 \]

\[ M_1 = X_1 \cdot Z_2 \]

\[ M_2 = X_1 \cdot Z_2 \]

\[ M_3 = X_2 \cdot Z_1 \]

\[ M_4 = X_2 \cdot Z_1 \]

\[ M_5 = X_1 \cdot Z_2 \]

\[ M_6 = X_1 \cdot Z_2 \]

Using one two-stage pipelined multiplier (M):

- Latency for a loop operation (combined point addition and point doubling):
  \[ 6 + 1 = 7 \text{ clock cycles} \]

- To remove Idle clock cycles:
  “Careful Scheduling”

6 clock cycles

Save: 14.29% latency of loop operation!!
Our proposed high speed ECC (cntd.)

- Parallel operation of Montgomery point multiplication

\[ K_i = 1 \]
\[ M_1 = X_1, Z_2 \]
\[ M_2 = X_1, Z_2 \]
\[ M_1 = X_1, Z_2 \]

- Using two two-stage pipelined multiplier (M):
  - Latency for a loop operation (combined point addition and point doubling):
    - \( 4 + 1 = 5 \) clock cycles
  - To remove Idle clock cycles: "Careful Scheduling"
  - 4 clock cycles
  - Save: 20% latency of loop operation!!
Our proposed high speed ECC (cntd.)

- Parallel operation of Montgomery point multiplication

Using three two-stage pipelined multiplier(M):

- Latency for a loop operation (combined point addition and point doubling):
  - \(2 + 2 = 4\) clock cycles

To remove Idle clock cycles:

Not Possible due two stage pipelining

4 clock cycles
Our proposed high speed ECC (cntd.)

- Parallel operation of Montgomery point multiplication

\[ K_i = 1 \]

\[ K_i = 0 \]

\[ K_i = 1 \]

We Consider

- two two-stage pipelined multiplier(M):

“4 clock cycles for each loop operation”
Our proposed high speed ECC (cntd.)

- Cascaded Arithmetic Operations

- We exploit cascade arithmetic Circuit instead of standalone multiplier
  - To reduce latency
  - To simplify control operation
  - To reduce memory operation
High Speed ECC Implementation on FPGA over GF($2^m$)

Zia U. A. Khan and M. Benaissa

Overview
Introduction
High Speed ECC

High Speed ECC

Comparison With The State Of Art
Conclusions

Smart pipelining
The Key Strategies

Our proposed high speed ECC (cntd.)

- Cascaded Arithmetic Operations

Cascaded: $Sqr - Sqr = 4 - Sqr = ((x)^2)^2$

- We exploit cascaded $Sqr$
  - 4-sqr in single clock cycles
    - Use in the loop operation
    - Use to accelerate multipliciative inversion
Our proposed high speed ECC

Proposed ECC Architecture

We utilise:
- Two Full-precision multiplier
- Montgomery Point multiplication
- Careful scheduling
- Cascaded Arithmetic operations

Critical path delay of ECC processor:

<table>
<thead>
<tr>
<th>TABLE II</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRITICAL PATH DELAY ($T_{ECC}$) OF THE PROPOSED ECC OVER GF($2^m$)</td>
</tr>
</tbody>
</table>

Path1: $T_A + \left( \log_2 \left( \frac{m}{d} \right) \right) T_X + 4 T_{ux}$ or Path2: $\left( \log_2 ((n + k)) \right) T_X + T_{adder} + T_{sqr} = \left( \log_2 ((k)) T_X \right) + 3 T_{mux}$

$n = \# \text{ segments}, k$ is the second higher order of irreducible polynomial
Comparison with state of art

Maximum frequency (V5 and V4)

We use 2x163 Mul (complexity high)

Achieved the highest frequency, 153 MHz

<table>
<thead>
<tr>
<th>Ref.</th>
<th>FPGA</th>
<th>Slices (Sls)</th>
<th>LUTs</th>
<th>FFs</th>
<th>Freq. (MHz)</th>
<th>kP Time (μs)</th>
<th>Area x time, Sls xTime x10³</th>
<th>Clock Cycles</th>
<th>Multiplier</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10]</td>
<td>Virtex4</td>
<td>4080</td>
<td>7719</td>
<td>1502</td>
<td>197</td>
<td>20.56</td>
<td>84</td>
<td>4050</td>
<td>163 bit Karatsuba Multiplier</td>
<td></td>
</tr>
<tr>
<td>[14]</td>
<td>Virtex4</td>
<td>20807</td>
<td>-</td>
<td>-</td>
<td>185</td>
<td>7.72</td>
<td>161</td>
<td>1428</td>
<td>3 Core 82 bit Multipliers</td>
<td></td>
</tr>
<tr>
<td>[15]</td>
<td>Virtex4</td>
<td>24363</td>
<td>-</td>
<td>-</td>
<td>143</td>
<td>10.00</td>
<td>244</td>
<td>1446</td>
<td>3 GNB 55 bit Multipliers</td>
<td></td>
</tr>
<tr>
<td>[16]</td>
<td>Virtex4</td>
<td>17929</td>
<td>33414</td>
<td>-</td>
<td>250</td>
<td>9.60</td>
<td>172</td>
<td>2751</td>
<td>3 Digit Serial 55 bit Multipliers</td>
<td></td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex4</td>
<td>12834</td>
<td>22815</td>
<td>6683</td>
<td>196</td>
<td>17.20</td>
<td>221</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
<td></td>
</tr>
<tr>
<td>[18]</td>
<td>Virtex4</td>
<td>10417</td>
<td>-</td>
<td>-</td>
<td>121</td>
<td>9.00</td>
<td>94</td>
<td>1091</td>
<td>163 bit Karatsuba</td>
<td></td>
</tr>
<tr>
<td>[13]</td>
<td>Virtex5</td>
<td>6150</td>
<td>22936</td>
<td>-</td>
<td>259</td>
<td>5.48</td>
<td>34</td>
<td>1371</td>
<td>3 Digit Serial 81 bit Multipliers</td>
<td></td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex5</td>
<td>6536</td>
<td>17305</td>
<td>4075</td>
<td>262</td>
<td>12.90</td>
<td>84</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex5</td>
<td>10363</td>
<td>29095</td>
<td>6529</td>
<td>153</td>
<td>5.10</td>
<td>53</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex7</td>
<td>8736</td>
<td>27105</td>
<td>6529</td>
<td>223</td>
<td>3.50</td>
<td>31</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
<td></td>
</tr>
</tbody>
</table>
Comparison with state of art

Total Latency for $kP$ (V5 or V4)

We achieved the lowest latency: 780 Clock Cycles

<table>
<thead>
<tr>
<th>Ref.</th>
<th>FPGA</th>
<th>Slices (Sls)</th>
<th>LUTs</th>
<th>FFs</th>
<th>Freq. (MHz)</th>
<th>$kP$ Time (μs)</th>
<th>Area x time, Sls xTimex$10^{-3}$</th>
<th>Clock Cycles</th>
<th>Multiplier Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10]</td>
<td>Virtex4</td>
<td>4080</td>
<td>7719</td>
<td>1502</td>
<td>197</td>
<td>20.56</td>
<td>84</td>
<td>4050</td>
<td>41 bit Karatsuba Multiplier</td>
</tr>
<tr>
<td>[14]</td>
<td>Virtex4</td>
<td>20807</td>
<td>-</td>
<td>-</td>
<td>185</td>
<td>7.72</td>
<td>161</td>
<td>1428</td>
<td>3 Core 82 bit Multipliers</td>
</tr>
<tr>
<td>[15]</td>
<td>Virtex4</td>
<td>24363</td>
<td>-</td>
<td>-</td>
<td>143</td>
<td>10.00</td>
<td>244</td>
<td>1446</td>
<td>3 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>[16]</td>
<td>Virtex4</td>
<td>17929</td>
<td>33414</td>
<td>-</td>
<td>250</td>
<td>9.60</td>
<td>172</td>
<td>2751</td>
<td>3 Digit Serial 55 bit Multipliers</td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex4</td>
<td>12834</td>
<td>22815</td>
<td>6683</td>
<td>196</td>
<td>17.20</td>
<td>221</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>[18]</td>
<td>Virtex4</td>
<td>10417</td>
<td>-</td>
<td>-</td>
<td>121</td>
<td>9.00</td>
<td>94</td>
<td>1091</td>
<td>163 bit Karatsuba</td>
</tr>
<tr>
<td>[13]</td>
<td>Virtex5</td>
<td>6150</td>
<td>22936</td>
<td>-</td>
<td>250</td>
<td>5.48</td>
<td>34</td>
<td>1371</td>
<td>3 Digit Serial 81 bit Multipliers</td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex5</td>
<td>6536</td>
<td>17305</td>
<td>4075</td>
<td>262</td>
<td>12.90</td>
<td>84</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex5</td>
<td>10363</td>
<td>29095</td>
<td>6529</td>
<td>153</td>
<td>5.10</td>
<td>53</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex7</td>
<td>8736</td>
<td>27105</td>
<td>6529</td>
<td>223</td>
<td>3.50</td>
<td>31</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
</tr>
</tbody>
</table>
### Comparison with the State of Art

#### Total time for $kP$ (V5 or V4)

![We achieved the fastest speed: 5.10µs](image-url)

**TABLE III**

<table>
<thead>
<tr>
<th>Ref.</th>
<th>FPGA</th>
<th>Slices (Sls)</th>
<th>LUTs</th>
<th>FFs</th>
<th>Freq. (MHz)</th>
<th>$kP$ Time (µs)</th>
<th>Area x time, Sls xTime$10^{-3}$</th>
<th>Clock Cycles</th>
<th>Multiplier Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10]</td>
<td>Virtex4</td>
<td>4080</td>
<td>7719</td>
<td>1502</td>
<td>197</td>
<td>20.56</td>
<td>84</td>
<td>4050</td>
<td>41 bit Karatsuba Multiplier</td>
</tr>
<tr>
<td>[14]</td>
<td>Virtex4</td>
<td>20807</td>
<td>-</td>
<td>-</td>
<td>185</td>
<td>7.72</td>
<td>161</td>
<td>1428</td>
<td>3 Core 82 bit Multipliers</td>
</tr>
<tr>
<td>[15]</td>
<td>Virtex4</td>
<td>24363</td>
<td>-</td>
<td>-</td>
<td>143</td>
<td>10.00</td>
<td>244</td>
<td>1446</td>
<td>3 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>[16]</td>
<td>Virtex4</td>
<td>17929</td>
<td>33414</td>
<td>-</td>
<td>250</td>
<td>9.60</td>
<td>172</td>
<td>2751</td>
<td>3 Digit Serial 55 bit Multipliers</td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex4</td>
<td>12834</td>
<td>22815</td>
<td>6683</td>
<td>196</td>
<td>17.20</td>
<td>221</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>[18]</td>
<td>Virtex4</td>
<td>10417</td>
<td>-</td>
<td>-</td>
<td>121</td>
<td>9.00</td>
<td>94</td>
<td>1091</td>
<td>163 bit Karatsuba</td>
</tr>
<tr>
<td>[13]</td>
<td>Virtex5</td>
<td>6150</td>
<td>22936</td>
<td>-</td>
<td>250</td>
<td>5.48</td>
<td>34</td>
<td>1371</td>
<td>3 Digit Serial 81 bit Multipliers</td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex5</td>
<td>6536</td>
<td>17305</td>
<td>4075</td>
<td>262</td>
<td>12.90</td>
<td>84</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex5</td>
<td>10362</td>
<td>29095</td>
<td>6529</td>
<td>153</td>
<td>5.10</td>
<td>53</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex7</td>
<td>8736</td>
<td>27105</td>
<td>6529</td>
<td>223</td>
<td>3.50</td>
<td>31</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
</tr>
</tbody>
</table>

High Speed ECC Implementation on FPGA over GF($2^m$)

Zia U. A. Khan and M. Benaissa

FPL 2015, London, UK
Comparison with state of art

ECC in the new technology (V7)

- We achieved the best area-time performance: 31.
- The Fastest speed for $kP$: 3.50 µs
- Max. Frequency: 223 MHz

<table>
<thead>
<tr>
<th>Ref.</th>
<th>FPGA</th>
<th>Slices (Sls)</th>
<th>LUTs</th>
<th>FFs</th>
<th>Freq. (MHz)</th>
<th>kP Time (µs)</th>
<th>Area x time, Sls x Timex10^3</th>
<th>Clock Cycles</th>
<th>Multiplier Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10]</td>
<td>Virtex4</td>
<td>4080</td>
<td>7719</td>
<td>1502</td>
<td>197</td>
<td>20.56</td>
<td>84</td>
<td>4050</td>
<td>41 bit Karatsuba Multiplier</td>
</tr>
<tr>
<td>[14]</td>
<td>Virtex4</td>
<td>20807</td>
<td>-</td>
<td>-</td>
<td>185</td>
<td>7.72</td>
<td>161</td>
<td>1428</td>
<td>3 Core 82 bit Multipliers</td>
</tr>
<tr>
<td>[15]</td>
<td>Virtex4</td>
<td>24363</td>
<td>-</td>
<td>-</td>
<td>143</td>
<td>10.00</td>
<td>244</td>
<td>1446</td>
<td>3 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>[16]</td>
<td>Virtex4</td>
<td>17929</td>
<td>33414</td>
<td>-</td>
<td>250</td>
<td>9.60</td>
<td>172</td>
<td>2751</td>
<td>3 Digit Serial 55 bit Multipliers</td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex4</td>
<td>12834</td>
<td>22815</td>
<td>6683</td>
<td>196</td>
<td>17.20</td>
<td>221</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>[18]</td>
<td>Virtex4</td>
<td>10417</td>
<td>-</td>
<td>121</td>
<td>9.00</td>
<td>9.00</td>
<td>94</td>
<td>1091</td>
<td>163 bit Karatsuba Multiplier</td>
</tr>
<tr>
<td>[13]</td>
<td>Virtex5</td>
<td>6150</td>
<td>22936</td>
<td>-</td>
<td>250</td>
<td>5.48</td>
<td>34</td>
<td>1371</td>
<td>3 Digit Serial 81 bit Multipliers</td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex5</td>
<td>6536</td>
<td>17305</td>
<td>4075</td>
<td>262</td>
<td>12.90</td>
<td>84</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex5</td>
<td>10363</td>
<td>29095</td>
<td>6529</td>
<td>153</td>
<td>5.10</td>
<td>53</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex7</td>
<td>8736</td>
<td>27105</td>
<td>6529</td>
<td>223</td>
<td>3.50</td>
<td>31</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
</tr>
</tbody>
</table>
Comparison with state of art

Can the previous state of art meet the speed if they are re-implemented?

May not possible: due to
Could not achieve 780 clock cycles due to pipelining.
showed poor frequency even using low complexity circuit

<table>
<thead>
<tr>
<th>Ref.</th>
<th>FPGA</th>
<th>Slices (Sls)</th>
<th>LUTs</th>
<th>FFs</th>
<th>Freq. (MHz)</th>
<th>kP Time (µs)</th>
<th>Area x time, Sls x Time x 10^-3</th>
<th>Clock Cycles</th>
<th>Multiplier Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10]</td>
<td>Virtex4</td>
<td>4080</td>
<td>7719</td>
<td>1502</td>
<td>197</td>
<td>20.56</td>
<td>84</td>
<td>4050</td>
<td>41 bit Karatsuba Multiplier</td>
</tr>
<tr>
<td>[14]</td>
<td>Virtex4</td>
<td>20807</td>
<td>-</td>
<td>-</td>
<td>185</td>
<td>7.72</td>
<td>161</td>
<td>1428</td>
<td>3 Core 82 bit Multipliers</td>
</tr>
<tr>
<td>[15]</td>
<td>Virtex4</td>
<td>24363</td>
<td>-</td>
<td>-</td>
<td>143</td>
<td>10.00</td>
<td>244</td>
<td>1446</td>
<td>3 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>[16]</td>
<td>Virtex4</td>
<td>17929</td>
<td>33414</td>
<td>-</td>
<td>250</td>
<td>9.60</td>
<td>172</td>
<td>2751</td>
<td>3 Digit Serial 55 bit Multipliers</td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex4</td>
<td>12834</td>
<td>22815</td>
<td>6683</td>
<td>196</td>
<td>17.20</td>
<td>221</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>[18]</td>
<td>Virtex4</td>
<td>10417</td>
<td>-</td>
<td>-</td>
<td>121</td>
<td>9.00</td>
<td>94</td>
<td>1091</td>
<td>163 bit Karatsuba</td>
</tr>
<tr>
<td>[13]</td>
<td>Virtex5</td>
<td>6150</td>
<td>22936</td>
<td>-</td>
<td>250</td>
<td>5.48</td>
<td>34</td>
<td>1371</td>
<td>3 Digit Serial 81 bit Multipliers</td>
</tr>
<tr>
<td>[17]</td>
<td>Virtex5</td>
<td>6536</td>
<td>17305</td>
<td>4075</td>
<td>262</td>
<td>12.90</td>
<td>84</td>
<td>3379</td>
<td>2 GNB 55 bit Multipliers</td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex5</td>
<td>10363</td>
<td>29095</td>
<td>6529</td>
<td>153</td>
<td>5.10</td>
<td>53</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
</tr>
<tr>
<td>Ours</td>
<td>Virtex7</td>
<td>8736</td>
<td>27105</td>
<td>6529</td>
<td>223</td>
<td>3.50</td>
<td>31</td>
<td>780</td>
<td>2x163 bit Multiplier</td>
</tr>
</tbody>
</table>

Table III
Comparision of proposed ECC with published state of arts over GF(2^m) after place and route on FPGA
The fastest ECC processor to date on FPGA!

\[ kP = 3.50 \, \mu s \, (V7) \]

- Main contributions to achieve the speed:
  - Novel full-precision multiplier using segmented pipelining
  - Parallel Montgomery point multiplication
  - Careful scheduling
  - Cascaded arithmetic operations
  - Maintaining pipelining to shorten critical path delay of the ECC architecture
  - Use of 4-sqr circuit to accelerate multiplicative inversion
  - Finally: utilising Tools- Timing closure techniques
Thank you.

For Further enquiry about the paper, please contact:
Zia U. A. Khan
elp10zuk@sheffield.ac.uk
M. Benaissa:
m.benaissa@sheffield.ac.uk