Pipelined Modular Multiplier Supporting Multiple Standard Prime Fields

Hamad Alrimeih
Cyber Security Center, KACST
Riyadh, Saudi Arabia
halrimeih@kacst.edu.ca

Daler Rakhmatov
ECE, University of Victoria
Victoria, Canada
daler@ece.uvic.ca
Outline

• Introduction
• Our implementation
• Fast reduction (NIST prime fields)
• Pipeline structure and timing
• Using proposed multiplier
Introduction

• Modular multiplications: **performance + flexibility**

• **Performance:** Use wide datapath, pipelining, and NIST-recommended prime fields (fast reduction)
  - P192 = $2^{192} - 2^{64} - 1$
  - P224 = $2^{224} - 2^{96} + 1$
  - P256 = $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
  - P384 = $2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
  - P521 = $2^{521} - 1$

• **Flexibility:** Support all five NIST primes
Our Implementation

• Xilinx Virtex-6 FPGA
  – 272-bit datapath
  – 100-MHz clock frequency
  – 8.4K slices + 289 DSP48 blocks

• Throughput:
  – $10^8$ mod. multiplications/s for P192, P224, P256
  – $125 \times 10^5$ mod. multiplications/s for P384, P521

• Latency:
  – 80 ns for P192, P224, P256
  – 200 ns for P384, P521
Fast Reduction Modulo P192

**INPUT:** 384-bit product $Z = (z_{11}...z_1z_0)_{2^{32}}$

**OUTPUT:** 192-bit result $R = Z \mod p_{192}$

1. **DEFINE** 192-bit data words
   
   $W_0 \equiv (z_5z_4z_3z_2z_1z_0)_{2^{32}}$,
   
   $W_1 \equiv (00z_7z_6z_7z_6)_{2^{32}}$,
   
   $W_2 \equiv (z_9z_8z_9z_800)_{2^{32}}$,
   
   $W_3 \equiv (z_{11}z_{10}z_{11}z_{10}z_{11}z_{10})_{2^{32}}$

2. $U = W_0 + W_1 + W_2 + W_3$

3. $R \equiv U \mod p_{192}$

4. **RETURN** $R$
Fast Reduction Modulo $p_{224}$

**INPUT:** 448-bit product $Z = (z_{13}...z_1z_0)_{2^{32}}$

**OUTPUT:** 224-bit result $R = Z \mod p_{224}$

1. **DEFINE** 224-bit data words

   $W_0 \equiv (z_6z_5z_4z_3z_2z_1z_0)_{2^{32}}$,  
   $W_1 \equiv (z_{10}z_9z_8z_7000)_{2^{32}}$,  
   $W_2 \equiv (0z_{13}z_{12}z_{11}000)_{2^{32}}$,  
   $W_3 \equiv (z_{13}z_{12}z_{11}z_{10}z_9z_8z_7)_{2^{32}}$,  
   $W_4 \equiv (0000z_{13}z_{12}z_{11})_{2^{32}}$

2. $U = W_0 + W_1 + W_2$, $V = W_3 + W_4$

3. $R \equiv (U - V) \mod p_{224}$

4. **RETURN** $R$
Fast Reduction Modulo P256

**INPUT:** 512-bit product $Z = (z_{15} \ldots z_1 z_0)_{2^{32}}$

**OUTPUT:** 256-bit result $R = Z \mod p_{256}$

1. **DEFINE** 256-bit data words
   
   $W_0 \equiv (z_7 z_6 z_5 z_4 z_3 z_2 z_1 z_0)_{2^{32}},$
   $W_1 \equiv (z_{15} z_{14} z_{13} z_{12} z_{11} 000)_{2^{32}},$
   $W_2 \equiv (0 z_{15} z_{14} z_{13} z_{12} 000)_{2^{32}},$
   $W_3 \equiv (z_{15} z_{14} 000 z_{10} z_9 z_8)_{2^{32}},$
   $W_4 \equiv (z_8 z_{13} z_{15} z_{14} z_{13} z_{11} z_{10} z_9)_{2^{32}},$
   $W_5 \equiv (z_{10} z_8 000 z_{13} z_{12} z_{11})_{2^{32}},$
   $W_6 \equiv (z_{11} z_9 000 z_{15} z_{14} z_{13} z_{12})_{2^{32}},$
   $W_7 \equiv (z_{12} 0 z_{10} z_9 z_8 z_{15} z_{14} z_{13})_{2^{32}},$
   $W_8 \equiv (z_{13} 0 z_{11} z_{10} z_9 0 z_{15} z_{14})_{2^{32}}$

2. $U = W_0 + W_1 + W_1 + W_2 + W_2 + W_3 + W_4,$
   $V = W_5 + W_6 + W_7 + W_8$

3. $R \equiv (U - V) \mod p_{256}$

4. **RETURN** $R$
Fast Reduction Modulo P384

**INPUT:** 768-bit product $Z = (z_{23}...z_1 z_0)_{2^{32}}$

**OUTPUT:** 384-bit result $R = Z \mod p_{384}$

1. **DEFINE** 384-bit data words
   - $W_0 \equiv (z_{11} z_{10} z_9 z_8 z_7 z_6 z_5 z_4 z_3 z_2 z_1 z_0)_{2^{32}}$
   - $W_1 \equiv (00000 z_{23} z_{22} z_{21} 0000)_{2^{32}}$
   - $W_2 \equiv (z_{23} z_{22} z_{21} z_{20} z_{19} z_{18} z_{17} z_{16} z_{15} z_{14} z_{13} z_{12})_{2^{32}}$
   - $W_3 \equiv (z_{20} z_{19} z_{18} z_{17} z_{16} z_{15} z_{14} z_{13} z_{12} z_{23} z_{22} z_{21})_{2^{32}}$
   - $W_4 \equiv (z_{19} z_{18} z_{17} z_{16} z_{15} z_{14} z_{13} z_{12} z_{20} 0 z_{23} 0)_{2^{32}}$
   - $W_5 \equiv (00000 z_{23} z_{22} z_{21} z_{20} 0000)_{2^{32}}$
   - $W_6 \equiv (000000 z_{23} z_{22} z_{21} 00 z_{20})_{2^{32}}$
   - $W_7 \equiv (z_{22} z_{21} z_{20} z_{19} z_{18} z_{17} z_{16} z_{15} z_{14} z_{13} z_{12} z_{23})_{2^{32}}$
   - $W_8 \equiv (0000000 z_{23} z_{22} z_{21} z_{20} 0)_{2^{32}}$
   - $W_9 \equiv (0000000 z_{23} z_{22} 000)_{2^{32}}$

2. $U = W_0 + W_1 + W_1 + W_2 + W_3 + W_4 + W_5 + W_6$
   $V = W_7 + W_8 + W_9$

3. $R \equiv (U - V) \mod p_{384}$

4. **RETURN** $R$
Fast Reduction Modulo P521

**INPUT:** 1042-bit product \( Z \)

**OUTPUT:** 521-bit result \( R = Z \mod p_{521} \)

1. **DEFINE** 521-bit data words \( W_0 \equiv Z \mod 2^{521} \) and \( W_1 \equiv (Z - W_0)2^{-521} \)
2. \( U = W_0 + W_1 \)
3. \( R \equiv U \mod p_{521} \)
4. **RETURN** \( R \)
Sum-Carry Pairs (P192, P224, P256)

- Non-modular product: $Z = S[Z] + C[Z]$
  - Incomplete sum of partial products ($N$ 32-bit words): $S[Z] = (s_j[Z]), j = 0, 1, 2, ..., N - 1$
  - Saved multibit carries: $C[Z] = (c_j[Z]), j = 1, 2, ..., N$
- Fast reduction: $Z \mod p = (U - V) \mod p$

\[
S[U] = (s[N-1[U] ... s_1[U] s_0[U]])_{2^{32}}, \\
S[V] = (s[N-1[V] ... s_1[V] s_0[V]])_{2^{32}}, \\
C[U] = (c[N[U] ... c_2[U] c_1[U]])_{2^{32}}, \\
C[V] = (c[N[V] ... c_2[V] c_1[V]])_{2^{32}}.
\]
Sum-Carry Pairs (P384)

• Non-modular product: $Z_r = S^{[Zr]} + C^{[Zr]}

  – Subscript $r$ represents LL (low-low), LH (low-high), HL (high-low), or HH (high-high), referring to half-operand multiplications

• Fast reduction: $Z_r \mod p = (U_r - V_r) \mod p$

\[
S^{[U_r]} = \left( s_0^{[U_r]} s_1^{[U_r]} \ldots s_{N-1}^{[U_r]} \right)_{2^{32}}, \\
S^{[V_r]} = \left( s_0^{[V_r]} s_1^{[V_r]} \ldots s_{N-1}^{[V_r]} \right)_{2^{32}}, \\
C^{[U_r]} = \left( c_0^{[U_r]} c_1^{[U_r]} c_2^{[U_r]} \ldots c_N^{[U_r]} \right)_{2^{32}}, \\
C^{[V_r]} = \left( c_0^{[V_r]} c_1^{[V_r]} c_2^{[V_r]} \ldots c_N^{[V_r]} \right)_{2^{32}}.
\]
### TABLE III.  MAPPING BETWEEN INDICES $k$ AND $j = 1, 2, \ldots, N - 1$ FOR EACH $\{s_j^{[U]}, c_j^{[U]}\}$ GIVEN PRIMES $p = p_{192}, p_{224}, p_{256}$ AND FOR EACH $\{s_j^{[U_r]}, c_j^{[U_r]}\}$ GIVEN PRIME $p = p_{384}$.

<table>
<thead>
<tr>
<th>Index $j$</th>
<th>$p_{192}$</th>
<th>$p_{224}$</th>
<th>$p_{256}$</th>
<th>$p_{384}$ ($r = LL$)</th>
<th>$p_{384}$ ($r = LH$)</th>
<th>$p_{384}$ ($r = HL$)</th>
<th>$p_{384}$ ($r = HH$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0, 6, 10</td>
<td>0</td>
<td>0, 8, 9</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>12, 20, 21</td>
</tr>
<tr>
<td>1</td>
<td>1, 7, 11</td>
<td>1</td>
<td>1, 9, 10</td>
<td>1</td>
<td>13</td>
<td>13</td>
<td>13, 22, 23</td>
</tr>
<tr>
<td>2</td>
<td>2, 6, 8, 10</td>
<td>2</td>
<td>2, 10, 11</td>
<td>2</td>
<td>14</td>
<td>14</td>
<td>14, 23</td>
</tr>
<tr>
<td>3</td>
<td>3, 7, 9, 11</td>
<td>3, 7, 11</td>
<td>2, 11, 11, 12, 12, 13</td>
<td>3</td>
<td>12, 15</td>
<td>12, 15</td>
<td>12, 15, 20, 21</td>
</tr>
<tr>
<td>4</td>
<td>4, 8, 10</td>
<td>4, 8, 12</td>
<td>4, 12, 13, 13, 13, 14</td>
<td>4</td>
<td>12, 13, 16</td>
<td>12, 13, 16</td>
<td>12, 13, 16, 20, 21, 21, 22</td>
</tr>
<tr>
<td>5</td>
<td>5, 9, 11</td>
<td>5, 9, 13</td>
<td>5, 13, 14, 14, 14, 15</td>
<td>5</td>
<td>13, 14, 17</td>
<td>13, 14, 17</td>
<td>13, 14, 17, 21, 22, 22, 23</td>
</tr>
<tr>
<td>6</td>
<td>–</td>
<td>6, 10</td>
<td>6, 13, 14, 14, 14, 15, 15</td>
<td>6</td>
<td>6, 14, 15</td>
<td>6, 14, 15</td>
<td>14, 15, 18, 22, 23, 23</td>
</tr>
<tr>
<td>7</td>
<td>–</td>
<td>–</td>
<td>7, 8, 15, 15, 15</td>
<td>7</td>
<td>7, 15, 16</td>
<td>7, 15, 16</td>
<td>15, 16, 19, 23</td>
</tr>
<tr>
<td>8</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>8</td>
<td>8, 16, 17</td>
<td>8, 16, 17</td>
<td>16, 17, 20</td>
</tr>
<tr>
<td>9</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>9</td>
<td>9, 17</td>
<td>9, 17</td>
<td>17, 18, 21</td>
</tr>
<tr>
<td>10</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>18, 19, 22</td>
</tr>
<tr>
<td>11</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>19, 20, 23</td>
</tr>
</tbody>
</table>
## Index Mapping (2)

### TABLE IV. Mapping between indices $k$ and $j = 1, 2, ..., N - 1$ for each $\{s_j^{[V]}, c_j^{[V]}\}$ given primes $p = p_{192}, p_{224}, p_{256}$ and for each $\{s_j^{[V_r]}, c_j^{[V_r]}\}$ given prime $p = p_{384}$.

<table>
<thead>
<tr>
<th>Index $j$</th>
<th>Indices $k$ used to calculate $\sum (s_k^{[Z]} + c_k^{[Z]})$, given that $p = ...$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p_{192}$</td>
</tr>
<tr>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>–</td>
</tr>
<tr>
<td>5</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>7</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>–</td>
</tr>
<tr>
<td>9</td>
<td>–</td>
</tr>
<tr>
<td>10</td>
<td>–</td>
</tr>
<tr>
<td>11</td>
<td>–</td>
</tr>
</tbody>
</table>
Pipeline Timing: P192, P224, P256

Stage I
Cycle 1  Cycle 2
Stage A  Stage B
Cycle 3  Cycle 4
Stage C  Stage D
Cycle 5  Cycle 6
Stage E  Stage F
Cycle 7  Cycle 8
Stage H
Cycle 9
Summary

• Our modular multiplier supports all five NIST prime fields (192/224/256/384/521-bit ECC)

• Its 272-bit datapath has 9 pipeline stages

• It takes 8 clock cycles to produce a multiplication result when using P192, P224, or P256

• It takes 20 clock cycles to produce a multiplication result when using P384 or P521
ECC Application

- (a) Jacobian point doubling:
  \[ Q_j \leftarrow 2Q_j \]

- (b) Affine-Jacobian point addition or subtraction:
  \[ Q_j \leftarrow \pm P_A + Q_j \]
Overall Architecture (1)
Overall Architecture (2)

Our Multiplier

- MM
- MAS0
- MAS1

MM_In[271:0]
MM_Out[271:0]
MAS0_In[271:0]
MAS0_Out[271:0]
MAS1_In[271:0]
MAS1_Out[271:0]

272 Prime
272 Prime+1

RFA

272 272 DM_DOUT0
272 272 DM_DOUT1
RFA_R0
RFA_R1

R3
R2

RFB

272 RFB_OUT0[271:0]
272 RFB_OUT1[271:0]

PNG

272 PNG_OUT[271:0]
Thank You!

Questions?