Reconfigurable Hardware Implementation of Mesh Routing in the Number Field Sieve Factorization

Sashisu Bajracharya, Deapesh Misra, Kris Gaj
George Mason University

Tarek El-Ghazawi
The George Washington University
Objective & Outline

N = P · Q

P, Q – large integers

1. Number Field Sieve (NFS) Factorization
2. Mesh Routing Architecture for the Matrix Step of NFS
3. Implementation
4. Results
5. Conclusions
6. Reconfigurable computers & future work

FPGA Array
Number Field Sieve (NFS) Steps

- Polynomial Selection
- Sieving (Relation Collection)
- Matrix (Linear Algebra)
- Square Root

Two most computationally intensive steps
Previous work

2001
D. J. Bernstein, “Circuits for integer factorization: a proposal”
Mesh approach to the sieving and matrix steps
improves asymptotic complexity for NFS performance

2002
A. K. Lenstra, A. Shamir, J. Tomlinson, E. Tromer,
“Analysis of Bernstein's Factorization Circuit,”
Asiacrypt 2002
Detailed design for mesh routing

2003
W. Geiselmann, R. Steinwandt,
“Hardware to solve sparse systems of linear
equations over GF(2)”, CHES 2003
Distributing computations among multiple nodes
Our Objective

Design, describe in RTL VHDL, synthesize & simulate existing theoretical designs for the matrix step of NFS using current generation of FPGA devices.
Input to the Matrix Step

Matrix A:

D = number of the matrix columns and rows

D ∈ [10^7, 10^{11}]

d – column density (weight) = maximum number of ones per column

d << D,
e.g., d=100 for D=10^{10}
Function of the Matrix Step

Find linear dependency in the large sparse matrix obtained after sieving step

\[ c_{i1} \oplus c_{i2} \oplus \ldots \oplus c_{iL} = 0 \]

D = number of the matrix columns and rows
Block Wiedemann Algorithm for the Matrix Step of NFS

1) Uses multiple matrix-by-vector multiplications of the sparse matrix $A$ with $k$ random vectors $v_i$

$$A \cdot v_i, A^2 \cdot v_i, \ldots, A^k \cdot v_i$$

where $k = 2D/K$

2) Post computations leading to the determination of the linear dependence of the matrix columns

Most time consuming operation:

$$A_{[DxD]} \cdot v_{[Dx1]}$$
Mesh Routing Architecture for the Matrix Step
Matrix-by-Vector Multiplication

$\mathbf{v}^T = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$

$\mathbf{A} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}$

$\mathbf{A} \cdot \mathbf{v} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

Sparse Matrix
Matrix-by-Vector Multiplication for Sparse Matrices

\( \mathbf{A} \cdot \mathbf{v} \)

Sparse Matrix

\( \mathbf{v}^T \)
Mesh Routing

$m \times m$ mesh where $m = \sqrt{D}$

d = maximum number of non-zero entries per column
Mesh corresponding to the matrix A and vector v

<table>
<thead>
<tr>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Cell 1 representing Column 1

Row indices of non-zero matrix entries

Vector bit
Routing in the Mesh

Each time a packet arrives at the target cell, the packet’s vector’s bit is xored with the partial result bit on the target cell.
Packets generated by each cell in the mesh

<table>
<thead>
<tr>
<th>Cell 1 representing Column 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
</tbody>
</table>

Destination address

Vector bit
Only packets with non-zero vector bits need to be routed

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Mesh Routing with K parallel matrix by vector multiplications

Example for K=2

\[ A \cdot v_1 \text{ and } A \cdot v_2 \text{ computed in parallel} \]
Mesh corresponding to the matrix A and vectors $v_1, v_2$

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>0 1</th>
<th>4</th>
<th>1 0</th>
<th>3</th>
<th>0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td></td>
<td>9</td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1 1</td>
<td>2</td>
<td>0 0</td>
<td>5</td>
<td>0 1</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td></td>
<td>4</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1 0</td>
<td>3</td>
<td>0 1</td>
<td>2</td>
<td>1 0</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td></td>
<td>6</td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Packets with multiple vector bits generated by each cell in the mesh

Cell 1 representing Column 1

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Destination address

Vector bits $v_1, v_2$
Mesh Routing with \( p \) multiple columns and \( K \) vectors

Example for \( p=2, K=2 \)

\[ A \cdot v_1 \] and \( A \cdot v_2 \) computed in parallel

\[
\begin{array}{c}
V_2 \\
V_1 \\
A
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\( p \) columns
**Mesh with multiple columns and multiple vectors per cell**

Cell 1 representing Columns 1,2

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>0 1</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>1 0</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1 0</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0 1</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>0 0</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>1 1</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>0 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Packets generated by the mesh with multiple columns and multiple vectors per cell

Cell 1 representing Columns 1, 2

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Clockwise Transposition Routing

Four iterations repeated
Compare-Exchange Cases
Between Columns

• Direction of travel = direction of exchange
• Result of comparison = Exchange the packets.
Compare-Exchange Cases
Between Columns

• Direction of travel ≠ direction of exchange

• Result of comparison = Exchange the packets.

• Rule for Exchange: Exchange iff the distance to target of the packet which is farthest from its destination gets reduced.
Implementation
Structure of the Xilinx Virtex FPGA

- Configurable Logic Block (CLB) slices
- I/O Blocks
- Block RAMs
- Block RAMs
- I/O Blocks
Two modes of operation of CLB slices

Logic mode

Combinational logic

1-bit register

1-bit register

Memory mode

RAM 16x1

1-bit register

RAM 16x1

1-bit register

CLB slice
Target FPGA Device

Xilinx Virtex II

XC2V8000
- 46,592 CLB slices
- 93,184 LUTs
- 93,184 FF (Flip-Flop)

I/O Block
CLB slice
Mesh parameters for single FPGA

\[ m^{(1)} = 12 \]

\[ p = 16 \]

\[ D^{(1)} = (m^{(1)})^2 \times p = (12)^2 \times 16 = 2304 \]

\[ K = 50 \]
Synthesis Results on one Virtex II XC2V8000 for Improved Mesh Routing Design

<table>
<thead>
<tr>
<th>Matrix Size</th>
<th>K</th>
<th>CLB slices</th>
<th>LUTs</th>
<th>FFs</th>
<th>Clock Period (ns)</th>
<th>Time for K mult (ns)</th>
<th>Time per 1 mult (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2304x2304 (Mesh 12x12, p=16)</td>
<td>1</td>
<td>6738 (14%)</td>
<td>10,438 (11%)</td>
<td>6,279 (7%)</td>
<td>14.5</td>
<td>11136</td>
<td>11136</td>
</tr>
<tr>
<td>2304x2304 (Mesh 12x12, p=16)</td>
<td>32</td>
<td>29,938 (64%)</td>
<td>50,983 (54%)</td>
<td>19,651 (21%)</td>
<td>16.7</td>
<td>12826</td>
<td>401</td>
</tr>
<tr>
<td>2304x2304 (Mesh 12x12, p=16)</td>
<td>50</td>
<td>43,402 (93%)</td>
<td>74,030 (89%)</td>
<td>27,406 (29%)</td>
<td>17.7</td>
<td>13593</td>
<td>271</td>
</tr>
</tbody>
</table>

\[ f_{\text{CLK-ROUTE}} = 55.5 \text{ MHz} \quad T_{\text{CLK-ROUTE}} = 18 \text{ ns} \]
Distributed Computation
(Geiselmann, Steinwandt, CHES 2003)

\[
A \cdot v = \left( \sum_{j=1}^{s} A_{1,j} \cdot v_j \right) :
\left( \sum_{j=1}^{s} A_{s,j} \cdot v_j \right)
\]
Using smaller FPGA arrays to perform the entire computation

1) FPGA array performs single sub-matrix by sub-vector multiplication
2) Reuse FPGA array for next sub-computation
Parallel Loading & Unloading of Data

- Vector
- Non Zero Matrix Entries
- Result Vector
Sub matrix load-compute sequence

\[
\begin{bmatrix}
A_{1,1} & A_{1,2} & A_{1,3} \\
A_{2,1} & A_{2,2} & A_{2,3} \\
A_{3,1} & A_{3,2} & A_{3,3}
\end{bmatrix}
\times
\begin{bmatrix}
v_1 \\
v_2 \\
v_3
\end{bmatrix}
= 
\begin{bmatrix}
A'_{1} \\
A'_{2} \\
A'_{3}
\end{bmatrix}
\]

\[
A_{1,1} \times v_1 + A_{1,2} \times v_2 + A_{1,3} \times v_3 = A'_{1}
\]

\[
\text{Time}
\]

\begin{align*}
\text{load} & \quad \text{compute} & \quad \text{load} & \quad \text{compute} & \quad \text{load} & \quad \text{compute} & \quad \text{unload} \\
\text{load} & \quad \text{compute} & \quad \text{load} & \quad \text{compute} & \quad \text{load} & \quad \text{compute} & \quad \text{unload} \\
\text{load} & \quad \text{compute} & \quad \text{load} & \quad \text{compute} & \quad \text{load} & \quad \text{compute} & \quad \text{unload}
\end{align*}
Size of the mesh implemented using $f$ FPGAs

Matrix of $f$ FPGAs

Single FPGA

$\text{#cells}^{(f)} = \text{#cells}^{(1)} \cdot f$

$(m^{(f)})^2 = (m^{(1)})^2 \cdot f$

$m^{(f)} = m^{(1)} \sqrt{f}$
Size of the matrix handled using $f$ FPGAs

Sub-matrix handled by one FPGA

Sub-matrix handled by $f$ FPGAs

Entire matrix $A$

Sub-matrix $A^{(1)}$

Sub-matrix $A^{(f)}$

$D^{(1)}$

$D^{(f)}$

$D$

$D^{(1)} = (m^{(1)})^2 \times p$

$D^{(f)} = D^{(1)} \times f$

$d^{(f)} = \left\lfloor d \times D^{(f)}/D \right\rfloor$

$d$ – column density of $A$ = maximum number of ones per column of $A$

$d^{(f)}$ – column density of $A^{(f)}$
Sizes and the column densities of submatrices handled using f FPGAs

\[ D = 10^{10}, d = 100 \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( D^{(f)} )</th>
<th>( d^{(f)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,304</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>230,400</td>
<td>1</td>
</tr>
<tr>
<td>256</td>
<td>589,824</td>
<td>1</td>
</tr>
<tr>
<td>1024</td>
<td>2,359,296</td>
<td>1</td>
</tr>
<tr>
<td>10,000</td>
<td>23,040,000</td>
<td>1</td>
</tr>
</tbody>
</table>
Matrix Data

<table>
<thead>
<tr>
<th>Status (st)</th>
<th>Loading Address</th>
<th>Routing Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$r_L$</td>
<td>$c_L$</td>
</tr>
<tr>
<td></td>
<td>$r_R$</td>
<td>$c_R$</td>
</tr>
<tr>
<td>1</td>
<td>$k(f)$</td>
<td>$k(f)$</td>
</tr>
<tr>
<td></td>
<td>$k(f) + k_p$</td>
<td>$k(f) + k_p$</td>
</tr>
</tbody>
</table>

Matrix Data Size = $1 + 4 \cdot k(f) + 2 \cdot k_p = 1 + 4 \cdot \log_2 m^{(f)} + 2 \cdot \log_2 p$

<table>
<thead>
<tr>
<th>$f$ (#FPGAs)</th>
<th>Matrix Data Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>100</td>
<td>37</td>
</tr>
<tr>
<td>256</td>
<td>41</td>
</tr>
<tr>
<td>1024</td>
<td>45</td>
</tr>
</tbody>
</table>
### Format of the Packet

The format of the packet is shown in the table below:

<table>
<thead>
<tr>
<th>status</th>
<th>row destination</th>
<th>column destination</th>
<th>local result position</th>
<th>vector bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>st</td>
<td>r</td>
<td>c</td>
<td>l</td>
<td>v</td>
</tr>
</tbody>
</table>

- \( k^{(f)} = \log_2 m^{(f)} \)
- \( k_p = \log_2 p \)
- \( m^{(f)} = m^{(1)} \cdot \sqrt{f} \)
- \( K = 50 \)

<table>
<thead>
<tr>
<th># FPGAs</th>
<th>packet size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63</td>
</tr>
<tr>
<td>100</td>
<td>69</td>
</tr>
<tr>
<td>256</td>
<td>71</td>
</tr>
<tr>
<td>1024</td>
<td>73</td>
</tr>
</tbody>
</table>
Loading Unit

Memory of packet destination addresses

Memory of vector data

vector data

result

packet

vector/result data

matrix data

matrix data

R[i]

P[i]

en

d

K

p

decode

data

vector/data

data

matrix/data

comb

Current Packet Unit

Diagram:

- Current Packet Unit (CP)
- Packet
- Annihilate (annih)
- New Packet
- Exchange
- Current Packet
Result Calculation

Memory of the result vector bits

P'[i]

en

addr2

result

new packet excluding vector bits

new packet vector bits

cell coordinate

Check Dest
Resource Proportion for LUT Usage

Mesh of 12x12

P=16
K=50

10.3

89.7

logic

RAM

%LUT RAM
% LUT logic
$$T_{CLK-IO} = 6 \text{ ns}$$

$$T_{CLK-ROUTE} = 18 \text{ ns}$$

$$m^{(1)} \ast \text{packet size}$$

FPGA boundary
Slowdown caused by crossing the chips boundaries

\[ x = \text{multiplexing factor} = \frac{\text{#bits crossing the boundary}}{\text{# of pins per boundary}} = \]

\[ = \frac{m^{(1)} \times \text{packet size}(f) \times 2}{\left(\frac{\text{#FPGA_pins}}{4}\right)} = \frac{12 \times 73 \times 2}{277} = 7 \]

for \( f=1024 \)

**Slowdown factor**

\[ = \frac{4 \cdot T_{\text{CLK-IO}} + (x-1) T_{\text{CLK-IO}} + T_{\text{CLK-ROUTE}}}{T_{\text{CLK-ROUTE}}} \]

\[ = \frac{10 \cdot T_{\text{CLK-IO}} + T_{\text{CLK-ROUTE}}}{T_{\text{CLK-ROUTE}}} = 4.33 \]
Results and Analysis
## Results for a 512-bit number N

K = number of concurrent multiplications = 50  
D = number of columns in matrix A = $6.7 \times 10^6$  
m$^{(f)}$ = mesh dimension  
p = number of columns handled in one cell  
d$^{(f)}$ = density of sub-matrix handled by mesh  
n = number of times to repeat sub-multiplications

$T_{\text{route}}$ = time for K multiplications in the mesh  
$T_{\text{Load}}$ = time for loading and unloading for K multiplications  
$T_{\text{Total}}$ = total time for the Matrix Step = $3 \cdot (D/K) \cdot n \cdot (T_{\text{route}} + T_{\text{Load}})$  

$$R = \left( \frac{\#\text{FPGA} \cdot T_{\text{Total}}}{1 \text{ FPGA} \cdot T_{\text{Total}} \text{ for 1 FPGA}} \right)$$

<table>
<thead>
<tr>
<th>Virtex II chips (f)</th>
<th>D</th>
<th>p</th>
<th>d$^{(f)}$</th>
<th>m$^{(f)}$</th>
<th>n</th>
<th>$T_{\text{route}}$ (ns)</th>
<th>$T_{\text{Load}}$ (ns)</th>
<th>$T_{\text{Total}}$ (days)</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$6.7 \times 10^6$</td>
<td>16</td>
<td>1</td>
<td>12</td>
<td>8.4 $\times 10^6$</td>
<td>13594</td>
<td>1568</td>
<td>596.6</td>
<td>1.00</td>
</tr>
<tr>
<td>10$^2$</td>
<td>$6.7 \times 10^6$</td>
<td>14</td>
<td>2</td>
<td>120</td>
<td>1105</td>
<td>8.9 $\times 10^5$</td>
<td>6.1 $\times 10^4$</td>
<td>4.9</td>
<td>0.82</td>
</tr>
<tr>
<td>16$^2$</td>
<td>$6.7 \times 10^6$</td>
<td>8</td>
<td>3</td>
<td>192</td>
<td>516</td>
<td>1.3 $\times 10^6$</td>
<td>1.3 $\times 10^5$</td>
<td>3.3</td>
<td>1.42</td>
</tr>
<tr>
<td>32$^2$</td>
<td>$6.7 \times 10^6$</td>
<td>4</td>
<td>6</td>
<td>384</td>
<td>129</td>
<td>2.5 $\times 10^6$</td>
<td>4.3 $\times 10^5$</td>
<td>1.8</td>
<td>3.07</td>
</tr>
</tbody>
</table>
Results for a 1024-bit number $N$

$K =$ number of concurrent multiplications $= 50$

$D =$ number of columns in matrix $A = 10^{10}$

$m^{(f)} =$ mesh dimension

$p =$ number of columns handled in one cell

$d^{(f)} =$ density of sub-matrix handled by mesh

$n =$ number of times to repeat sub-multiplications

$T_{\text{route}} =$ time for $K$ multiplications in the mesh

$T_{\text{Load}} =$ time for loading and unloading for $K$ multiplications

$T_{\text{Total}} =$ total time for the Matrix Step $= 3 \cdot (D/K) \cdot n \cdot (T_{\text{route}} + T_{\text{Load}})$

$R = (#\text{FPGAs} \cdot T_{\text{Total}})/(1 \text{ FPGA} \cdot T_{\text{Total}} \text{ for 1 FPGA})$

<table>
<thead>
<tr>
<th>Virtex II chips $(f)$</th>
<th>$D$</th>
<th>$p$</th>
<th>$d^{(f)}$</th>
<th>$m^{(f)}$</th>
<th>$n$</th>
<th>$T_{\text{route}}$ (ns)</th>
<th>$T_{\text{Load}}$ (ns)</th>
<th>$T_{\text{Total}}$ (days)</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$10^{10}$</td>
<td>16</td>
<td>1</td>
<td>12</td>
<td></td>
<td>$1.8 \times 10^{13}$</td>
<td>$13,593$</td>
<td>$1,568$</td>
<td>$1.9 \times 10^{12}$</td>
</tr>
<tr>
<td>$10^2$</td>
<td>$10^{10}$</td>
<td>16</td>
<td>1</td>
<td>120</td>
<td></td>
<td>$1.8 \times 10^9$</td>
<td>$4.8 \times 10^5$</td>
<td>$4.5 \times 10^4$</td>
<td>$6.9 \times 10^9$</td>
</tr>
<tr>
<td>$16^2$</td>
<td>$10^{10}$</td>
<td>16</td>
<td>1</td>
<td>192</td>
<td></td>
<td>$2.8 \times 10^8$</td>
<td>$8.2 \times 10^5$</td>
<td>$7.5 \times 10^4$</td>
<td>$1.8 \times 10^9$</td>
</tr>
<tr>
<td>$32^2$</td>
<td>$10^{10}$</td>
<td>16</td>
<td>1</td>
<td>384</td>
<td></td>
<td>$1.8 \times 10^7$</td>
<td>$1.7 \times 10^6$</td>
<td>$1.6 \times 10^5$</td>
<td>$2.3 \times 10^8$</td>
</tr>
</tbody>
</table>
Time vs. #FPGAs

\[ D = 10^{10} \]

\[ \sim \frac{1}{(#\text{FPGA})^{3/2}} \]
Cost*Time vs. #FPGAs

$D = 10^{10}$

$\#FPGAs \times \#days$

$210$ bln$/year$
Summary & Conclusions (1)

- First practical hardware implementation of Mesh Routing for the Number Field Sieve implemented and verified using timing simulation

- Practical numbers, based on post-placing & routing static timing analysis, obtained for an array of Xilinx Virtex II 8000 FPGAs

- A two-dimensional array of Virtex II chips can perform computations faster than a single FPGA by a factor approximately proportional to \((\text{number of FPGAs})^{3/2}\)
Matrix step for a 512-bit RSA key takes about 5 days on the rectangular array of 100 FPGAs.

Assuming matrix size $D=10^{10}$ matrix step for a 1024-bit RSA key appears to be prohibitive from the point of view of full (throughput) cost.
Mapping designs to existing reconfigurable platforms

Generic Array of FPGAs

Existing General – Purpose Reconfigurable SuperComputers
What is a reconfigurable computer?

Microprocessor system

μP

... μP

μP memory

... μP memory

I/O

Interface

FPGA system

FPGA

... FPGA

FPGA memory

... FPGA memory

Interface

I/O
Example: SRC 6E System

http://www.srccomp.com/

Source: [SRC, MAPLD04]
SRC MAP™ Reconfigurable Processor

Source: [SRC, MAPLD04]
SRC Hi-Bar™ Based Systems

- Hi-Bar sustains 1.4 GB/s per port with 180 ns latency per tier
- Up to 256 input and 256 output ports with two tiers of switch
- Common Memory (CM) has controller with DMA capability
- Controller can perform other functions such as scatter/gather
- Up to 8 GB DDR SDRAM supported per CM node

Source: [SRC, MAPLD04]
SRC Programming

μP system

FPGA system

HDL (VHDL)

HLL (C)

Library Developer

Application Programmer
Other reconfigurable supercomputers

• Cray XD1 (formerly Octiga Bay 12 K) from Cray Inc.

• SGI Altix 3000 from Silicon Graphics

• Star Bridge Hypercomputer from Star Bridge Systems
Advantages of reconfigurable computers

• general-purpose: cost distributed among multiple users with different needs

• behaves like hardware:
  - parallel processing
  - distributed memory
  - specialized functional units, etc.

• can be programmed by mathematicians themselves using traditional programming languages or GUI environments

• encourage innovation and experimentation
Our future goal

- Polynomial Selection
- Sieving
- Matrix (Linear Algebra)
- Square Root

Next target
Questions?
Backup slides
Sources of optimism

- emergence of **new companies** supporting reconfigurable supercomputing, including major players in the area of traditional supercomputing, such as Cray Inc. and SGI

- constant progress in the **Capabilities, performance, and flexibility** of existing reconfigurable computing platforms

- constant progress in the compiler technology, and logic synthesis of **high level programming languages**.
Parameters

\( f \) = number of FPGAs in the FPGA array

\( m^{(1)} \) = mesh dimension for one FPGA = 12

\( m^{(f)} \) = mesh dimension for \( f \) FPGAs = \( m^{(1)} \cdot \sqrt{f} \)

\( p \) = number of columns of matrix handled in one cell of the mesh

\( D^{(f)} \) = matrix dimension handled by \( f \) FPGAs

\[
D^{(f)} = (m^{(f)})^2 \cdot p = (m^{(1)} \cdot \sqrt{f})^2 \cdot p = f \cdot (m^{(1)})^2 \cdot p = f \cdot D^{(1)}
\]
Parameters

\[ D \quad = \text{number of columns in matrix } A \]
\[ D^{(1)} \quad = \text{number of columns of sub-matrix of } A \text{ handled by one FPGA in the mesh} \]
\[ = (m^{(1)})^2 \cdot p \]
\[ D^{(f)} \quad = \text{number of columns of sub-matrix of } A \text{ handled by } f \text{ FPGAs in the mesh} \]
\[ = (m^{(f)})^2 \cdot p = (m^{(1)})^2 \cdot f \cdot p = D^{(1)} \cdot f \]

\[ d \quad = \text{column density of matrix } A \]
\[ d^{(1)} \quad = \text{density of sub-matrix handled by one FPGA} \]
\[ = d \cdot D^{(1)} / D \]
\[ d^{(f)} \quad = \text{density of sub-matrix handled by } f \text{ FPGAs} \]
\[ = d \cdot D^{(f)} / D \]
Routing Parameters

\[ T_{\text{CLK\_mult}} = \text{multiplication clock period} \]
\[ T_{\text{CLK\_IO}} = \text{IO clock period} \]
\[ x = \text{bits to exchange between FPGAs / (bus size between FPGAs)} \]
\[ x = 2 \cdot (1 + 2 \cdot k^{(f)} + k_p + K) \cdot (m^{(1)})^2 / 277 \]
\[ T_{\text{step}} = \text{Total time needed for transfer of packet between cells across FPGAs} \]
\[ T_{\text{step}} = 4 \cdot T_{\text{CLK\_IO}} + (x-1) T_{\text{CLK\_IO}} + T_{\text{CLK\_mult}} \]

\[ h_c = \text{slowdown factor due to limited inter-FPGA IO connections} \]
\[ h_c = T_{\text{step}} / T_{\text{CLK\_mult}} \]

\[ T_{\text{route}} = \text{routing time for sub-multiplication in the mesh} \]
\[ T_{\text{route}} = \#\text{entries per cell} \cdot \#\text{steps} \cdot T_{\text{CLK\_mult}} \cdot h_c \]
\[ T_{\text{route}} = p \cdot d^{(f)} \cdot 4 \cdot m^{(f)} \cdot T_{\text{CLK\_mult}} \cdot h_c \]
Loading Unloading Parameters

\[ b^{(1)} = \text{#pins for data transfer for 1 FPGA} = \frac{\text{#maximum FPGA IO}}{2} \]

\[ b^{(f)} = \text{#pins for data transfer for } f \text{ FPGAs} \]
\[ = (\text{#maximum FPGA IO } /4) \cdot \sqrt{f} \]

\[ s_{\text{IO}} = \text{clock stages between two FPGA connections} \]

\[ T_{\text{CLK_load}} = \text{loading clock period} \]

\[ T_{\text{load}} = \text{time for loading and unloading for a sub-multiplication} \]
\[ = \left[ (\text{(#matrix entries bits}) + (\text{#vector bits to load}) + \right. \]
\[ (\text{#vector bits to unload}) \bigg) / \ b^{(f)} + \]
\[ s_{\text{IO}} \cdot (m^{(f)} - 1) \cdot m^{(f)} \ast \text{loading packet bits} / b^{(f)} \bigg] \cdot T_{\text{CLK_load}} = \]

\[ \frac{(1+4\cdot k^{(f)} + 2\cdot k_p) \cdot d \cdot D^{(f)}) + (K \cdot D^{(f)}) + K \cdot D^{(f)} \cdot D^{(f)} / D}{b^{(f)}} + s_{\text{IO}} \cdot (m^{(f)} - 1) \cdot m^{(f)} \cdot (1+4\cdot k^{(f)} + 2\cdot k_p + K) / b^{(f)} \bigg] \cdot T_{\text{CLK_load}} \]
Parameters

\[ n = \text{number of times to repeat sub-multiplications} \]
\[ = \frac{D^2}{(D^{(f)})^2} = \frac{D^2}{((m^{(f)})^2 p)^2} \]

\[ T_{\text{Total}} = \text{total time for a Matrix step} = 3 \cdot \frac{D}{K} \cdot n \cdot (T_{\text{route}} + T_{\text{load}}) \]
\[ d(f) \cdot p = \left\lfloor d \cdot p \cdot f \cdot (m^{(1)})^2 / D \right\rfloor \cdot p \leq \text{threshold area on FPGA} \]
$m^{(1)}$ cells

$p$ entries

$D^{(f)}$
Total time of routing

- Total routing takes maximum $d \cdot 4 \cdot m \cdot$ compare-exchange operations,
  where
  - $d$ – matrix density = maximum number of non-zero entries per column
  - $m$ – mesh size = $\sqrt{D}$, where $D$ is the matrix size