Scalable High-Throughput SRAM-based Architecture for IP-lookup Using FPGA

Hoang Le, Weirong Jiang, Viktor K. Prasanna
Department of Electrical Engineering
University of Southern California, USA

FPL 2008 Sep. 23, 2008
Our solution

• **Dual Linear Pipeline Architecture for IP Lookup (DuPI)**
  - Linear pipeline
  - Memory sizes proportional to the sizes of supported routing tables

• **Complete binary tree structure (BST)**
  - Addresses of the child nodes eliminated
  - Amount of memory doubled in the next level
  - Number of pipeline stages determined by the size of the supported routing tables

• **Large routing tables with external SRAMs**
  - Last few stages moved onto external SRAMs
  - Number of external stages depends on number of I/O pins
Pre-processing

• Convert given set of prefixes to a set of non-overlapping ranges:
  • Convert prefixes to ranges
    • Pad ‘0’ for lower bound, ‘1’ for upper bound
  • Map all ranges onto \([0 - (2^{32} - 1)]\)
  • Algorithm to cut overlapped ranges
    • Scan from left to right, cut at any prefix’s bound
    • Assign routing index of the longest prefix among the overlapping prefixes to the cut range

• Set of \(N\) prefixes
  • At most \(2N\) distinct bounds
  • \(N\) non-overlapping ranges (best case)
  • \((2N - 1)\) non-overlapping ranges (worst case)
## Pre-processing (example)

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Lower bound</th>
<th>Upper bound</th>
<th>Range(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0* 00000000</td>
<td>0 01111111</td>
<td>R2</td>
</tr>
<tr>
<td>P2</td>
<td>000* 00000000</td>
<td>0 00011111</td>
<td>R1</td>
</tr>
<tr>
<td>P3</td>
<td>010* 01000000</td>
<td>64 01011111</td>
<td>R3, R5</td>
</tr>
<tr>
<td>P4</td>
<td>01001* 01001000</td>
<td>72 01001111</td>
<td>R4</td>
</tr>
<tr>
<td>P5</td>
<td>01011* 01011000</td>
<td>88 01011111</td>
<td>R6</td>
</tr>
<tr>
<td>P6</td>
<td>011* 01100000</td>
<td>96 01111111</td>
<td>R7</td>
</tr>
<tr>
<td>P7</td>
<td>110* 11000000</td>
<td>192 11011111</td>
<td>R8</td>
</tr>
<tr>
<td>P8</td>
<td>111* 11100000</td>
<td>224 11111111</td>
<td>R9</td>
</tr>
</tbody>
</table>

The diagram below illustrates the ranges and bounds for each prefix:

- **P1:** Ranges [0-31], [32-63], [64-127]
- **P2:** Ranges [0-31]
- **P3:** Ranges [32-63], [72-79], [80-87], [88-95], [96-127]
- **P4:** Ranges [72-79], [88-95], [96-127]
- **P5:** Ranges [72-79], [88-95]
- **P6:** Ranges [96-127]
- **P7:** Ranges [192-223], [224-255]
- **P8:** Ranges [224-255]
- Sort all non-overlapping ranges using lower bounds in increasing order (assume $N$ ranges)

**Case 1**

- # nodes = $2^n - 1$

**Case 2**

- # nodes < $2^n - 1 + 2^{n-1}$

**Case 3**

- # nodes > $2^n - 1 + 2^{n-1}$
**IP look up**

- **Start from root**

- **At each node**
  - Compare incoming IP address with node’s lower bound
  - If greater or equal
    - Compare IP address with node’s upper bound
    - If less than or equal, replace routing information with node’s routing information
      - Else Traverse to the right by appending 1 to the memory address
  - Else Traverse to the left by appending 0 to the memory address
  - Memory address and routing information are forwarded
Each node stores: 2 bounds, routing index
FPGA Implementation (Non-optimized)

- Each node needs approx. 71 bits
  - Lower bound: 32 bits
  - Upper bound: 32 bits
  - Routing index: 6 bits (arbitrarily chosen to represent ports, actions ...)
  - Status: 1 bit

- Dual-ported Memory
  - 71-bit data for each port
  - Address width equals stage index
  - Total of \((142 + 2 \times \text{stage\_index})\) pins

- Further memory optimizations are possible to achieve the compact design in FPL’08
**FPGA Implementation (non-optimized)**

- **Virtex-4 FX140:**
  - **9936 Kb BRAM:**
    - Up to 17 stages
    - Up to 128K nodes
    - 92% BRAM utilization
  - **1517 I/O pins**
    - Interface up to 6 external dual-ported SRAM banks
    - Requires 969 pins coming into FPGA
  - **Support up to 8M nodes**
    - 8M prefixes (best case)
    - 4M prefixes (worst case)
• Throughput: over 300 MLPS (150 MHz)

• Required external SRAM:

<table>
<thead>
<tr>
<th>External stage</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem (Mb)</td>
<td>9</td>
<td>27</td>
<td>63</td>
<td>135</td>
<td>279</td>
<td>567</td>
</tr>
<tr>
<td># nodes</td>
<td>256K</td>
<td>512K</td>
<td>1M</td>
<td>2M</td>
<td>4M</td>
<td>8M</td>
</tr>
</tbody>
</table>

• Architecture with duplicated pipelines

<table>
<thead>
<tr>
<th># duplications</th>
<th>External SRAMs (Mb)</th>
<th># nodes</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>2x</td>
<td>31.5 x 2</td>
<td>512M</td>
<td>600 MLPS (192 Gbps)</td>
</tr>
<tr>
<td>3x</td>
<td>13.5 x 3</td>
<td>256K</td>
<td>900 MLPS (288 Gbps)</td>
</tr>
</tbody>
</table>

• State-of-the-art TCAM: 533 MLPS
## Power Dissipation: SRAM vs. TCAM

<table>
<thead>
<tr>
<th></th>
<th>TCAM (18 Mb)</th>
<th>SRAM (18 Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock rate (MHz)</td>
<td>266</td>
<td>250</td>
</tr>
<tr>
<td>Cell size (# of transistors per bit)</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>Power consumption (W)</td>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
P_{\text{f=150 MHz, 18Mb-SRAM}} = 1 \times \frac{150}{250} = 0.6 \text{ W}
\]

Source: Samsung Specification
### Power Dissipation (Non-optimized)

<table>
<thead>
<tr>
<th></th>
<th>Amount of Memory</th>
<th># prefixes</th>
<th>Power consumption (W)</th>
<th></th>
<th></th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Logic</td>
<td>Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCAM</td>
<td>36 Mb</td>
<td>819.2K</td>
<td>N/A</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>USC-4</td>
<td>135 Mb</td>
<td>1M</td>
<td>&lt; 1</td>
<td>4.5</td>
<td>&lt; 5.5</td>
<td></td>
</tr>
<tr>
<td>USC-2x</td>
<td>63 Mb</td>
<td>256K</td>
<td>&lt; 1</td>
<td>2.1</td>
<td>&lt; 3.1</td>
<td></td>
</tr>
</tbody>
</table>

USC-4: no duplication
USC-2×: 2× duplication

+ 4 external SRAMs
+ 3 external SRAMs

**Worst case**

Source: Xilinx