Highlights from ASP-DAC 2014

Abde Ali Kagalwalla
Outline

• **Data Compression via Logic Synthesis**
  – L. Amaru, P.-E. Gaillardon, A. Burg and G. D. Micheli (EPFL, Switzerland)

• **Applying VLSI EDA to Energy Distribution System Design**
  – Sani Nassif, Gi-Joon Nam, Jerry Hayes (IBM Research) and Sani Fakhouri (UC Irvine)

• **Energy Efficient In-Memory Machine Learning for Data Intensive Image-Processing by Non-volatile Domain-Wall Memory**
  – H. Yu, Y. Wang, S. Chen, W. Fei (NTU Singapore), C. Weng, J. Zhao and Z. Wei (Huawei Shannon Lab, China)
Lossless Compression

• Two main steps in most data compression approaches
  – Data decorrelation: Reduce autocorrelation of data. E.g. DCT (Discrete Cosine Transform), dictionary
  – Entropy encoding: Compress the decorrelated data. E.g. Huffman coding, arithmetic coding

• Logic synthesis also compresses a binary sequence
  – Can it be used efficiently in compression?
  – Prior work less efficient compared to conventional compression
Example of Compression using Logic Synthesis

\[\begin{align*}
S_0 &\quad S_1 &\quad S_2 &\quad S_3 &\quad S_4 &\quad S_5 &\quad S_6 &\quad S_7 \\
000 &\quad 001 &\quad 010 &\quad 011 &\quad 000 &\quad 001 &\quad 110 &\quad 111 \\
\end{align*}\]

Define 3 output Boolean function \( G \)

\[
G_0 = BR(6) + BR(7) = I_0 I_1 \overline{I_2} + I_0 I_1 I_2 = I_0 I_1
\]

Objective: Find minimum number of gates to represent the 3-bit function \( G \)

Additional heuristic: Stimulated by \( BR(i) \) iff stimulated by previous term \( BR(i-1) \)
Overall Method for Compression

• Input: Binary string $B$
• Output: Compressed string $C$
• Summary of method
  1. Split $B$ in $L$-bit substrings
  2. Find sum of products expression for $L$-bit function $G$ corresponding to substrings of $B$
  3. Boolean minimization of every bit of $G$ to circuit $K$
  4. If $G_i$ too big or minimization too slow, use entropy encoding to compress
  5. Resource sharing of circuit $K$
Results for Benchmarks generated from Causal Process

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Original Data Size</th>
<th>ZIP</th>
<th>DCT+ZIP</th>
<th>bzip2</th>
<th>7zip</th>
<th>Our Approach</th>
<th>ZIP Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Data</td>
<td>2.2 MB</td>
<td>208 KB</td>
<td>868 KB</td>
<td>316 KB</td>
<td>60 KB</td>
<td>8 KB</td>
<td>0.3 s</td>
</tr>
<tr>
<td></td>
<td>25 MB</td>
<td>2.1 MB</td>
<td>8.3 MB</td>
<td>3.1 MB</td>
<td>888 KB</td>
<td>8 KB</td>
<td>2.1 s</td>
</tr>
<tr>
<td></td>
<td>287 MB</td>
<td>21 MB</td>
<td>81 MB</td>
<td>31 MB</td>
<td>3.4 MB</td>
<td>302 KB</td>
<td>32 s</td>
</tr>
<tr>
<td>Linear Data + Noise</td>
<td>2.2 MB</td>
<td>264 KB</td>
<td>872 KB</td>
<td>258 KB</td>
<td>212 KB</td>
<td>80 KB</td>
<td>0.4 s</td>
</tr>
<tr>
<td></td>
<td>25 MB</td>
<td>2.7 MB</td>
<td>8.4 MB</td>
<td>2.6 MB</td>
<td>2.4 MB</td>
<td>700 KB</td>
<td>3.0 s</td>
</tr>
<tr>
<td></td>
<td>287 MB</td>
<td>27 MB</td>
<td>84 MB</td>
<td>30 MB</td>
<td>23 MB</td>
<td>7.1 MB</td>
<td>43 s</td>
</tr>
<tr>
<td>Quadratic Data</td>
<td>3.3 MB</td>
<td>484 KB</td>
<td>816 KB</td>
<td>532 KB</td>
<td>272 KB</td>
<td>8 KB</td>
<td>1.0 s</td>
</tr>
<tr>
<td></td>
<td>39 MB</td>
<td>5.3 MB</td>
<td>7.6 MB</td>
<td>6.1 MB</td>
<td>3.3 MB</td>
<td>16 KB</td>
<td>6.1 s</td>
</tr>
<tr>
<td></td>
<td>449 MB</td>
<td>59 MB</td>
<td>71 MB</td>
<td>67 MB</td>
<td>40 MB</td>
<td>566 KB</td>
<td>64 s</td>
</tr>
<tr>
<td>Random (XOR-intensive network)</td>
<td>1.6 MB</td>
<td>116 KB</td>
<td>304 KB</td>
<td>124 KB</td>
<td>44 KB</td>
<td>8 KB</td>
<td>0.1 s</td>
</tr>
<tr>
<td></td>
<td>20 MB</td>
<td>1.2 MB</td>
<td>3.2 MB</td>
<td>1.5 MB</td>
<td>796 KB</td>
<td>8 KB</td>
<td>1.2 s</td>
</tr>
<tr>
<td></td>
<td>230 MB</td>
<td>12 MB</td>
<td>31 MB</td>
<td>15 MB</td>
<td>3.8 MB</td>
<td>234 KB</td>
<td>10 s</td>
</tr>
<tr>
<td>Average runtime (normalized to ZIP)</td>
<td></td>
<td>1</td>
<td>1.5 x</td>
<td>8 x</td>
<td>12 x</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Outline

• Data Compression via Logic Synthesis

• Applying VLSI EDA to Energy Distribution System Design
  – Sani Nassif, Gi-Joon Nam, Jerry Hayes (IBM Research) and Sani Fakhouri (UC Irvine)

• Energy Efficient In-Memory Machine Learning for Data Intensive Image-Processing by Non-volatile Domain-Wall Memory
  – H. Yu, Y. Wang, S. Chen, W. Fei (NTU Singapore), C. Weng, J. Zhao and Z. Wei (Huawei Shannon Lab, China)
Motivation

• Using rigorous modeling, analysis and optimization (as done in EDA) for solving problems in power distribution
Load Balancing for Reliability

- Lifetime of costly components (like transformers) depends on usage level relative to rating
- Two techniques: Adding tieline or reconfigure switches
Load Balancing Optimization Method

- Key idea is to construct grid graph and define slack (rating - load) for each graph edge
- Simple greedy algorithms. E.g. for tieline addition, connect vertices with high load differentials and less physical distance
Other Problems Mentioned

• Static simulation of entire grid → Use techniques commonly deployed in power grid analysis of VLSI designs
• Contingency planning → Add redundancies to network to account for geography, weather, societal need (e.g. hospitals)
• Component upgrade
• Loss minimization
Outline

• Data Compression via Logic Synthesis

• Applying VLSI EDA to Energy Distribution System Design
  – Sani Nassif, Gi-Joon Nam, Jerry Hayes (IBM Research) and Sani Fakhouri (UC Irvine)

• Energy Efficient In-Memory Machine Learning for Data Intensive Image-Processing by Non-volatile Domain-Wall Memory
  – H. Yu, Y. Wang, S. Chen, W. Fei (NTU Singapore), C. Weng, J. Zhao and Z. Wei (Huawei Shannon Lab, China)
Domain Wall Nanowire

- Antiparallel spin direction has higher resistance (GMR effect)
- Spin based non-volatile memory → No standby power
- Can be used directly as part of in-memory logic
- Goal: Use this technology to implement neural network in hardware
Binary sum using DW-nanowire
LUT Based Sigmoid Function
## SPICE Simulation Results at 32nm

### Domain wall nanowire device

<table>
<thead>
<tr>
<th>Operation</th>
<th>Speed (cycles)</th>
<th>Energy (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>write</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>shift</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

### Domain wall nanowire logic

<table>
<thead>
<tr>
<th>Logic</th>
<th>Speed (cycles)</th>
<th>Energy (pJ)</th>
<th>Area (um²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit full adder</td>
<td>54</td>
<td>40</td>
<td>2.6</td>
</tr>
<tr>
<td>8-bit multiplier</td>
<td>163</td>
<td>308</td>
<td>18.9</td>
</tr>
<tr>
<td>8-bit sigmoid (LUT)</td>
<td>2</td>
<td>116</td>
<td>31.8</td>
</tr>
</tbody>
</table>
## CACTI Simulation Results for an Image Processing Application

<table>
<thead>
<tr>
<th>Platform</th>
<th>DW-NN</th>
<th>GPP (with on-chip memory)</th>
<th>GPP (with off-chip memory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation resources utilized</td>
<td>1×Processor 7714×DW-ADD ER 7714×DW-MUL 551×DW-LUT 1×controller</td>
<td>1×Processor</td>
<td>1×Processor</td>
</tr>
<tr>
<td>Area of computational units</td>
<td>18 mm² (processor) + 0.5 mm² (accelerators)</td>
<td>18 mm²</td>
<td>18 mm²</td>
</tr>
<tr>
<td>Power (Watt)</td>
<td>10.1</td>
<td>12.5</td>
<td>12.5</td>
</tr>
<tr>
<td>Throughput (GBytes/s)</td>
<td>108MBytes/s</td>
<td>9.3MBytes/s</td>
<td>9.3MBytes/s</td>
</tr>
<tr>
<td>Energy efficiency (nJ/bit)</td>
<td>7</td>
<td>389</td>
<td>642</td>
</tr>
</tbody>
</table>