Runtime Reconfiguration

Abdullah AlWosaibi
Jing Gong
YiKai Wang
Runtime Reconfiguration

- Overview / Functional Density
- Improving Functional Density
- Parametrizable RegEx Operators
Reconfigurable Computing // Overview

- High performance of hardware
- (Some) flexibility of software
- Adaptable at runtime
Reconfigurable Computing is Good

- Optimise Limited Resources
- Expands Functionality
- Cost Effective
Reconfigurable Computing is Good

Replacing idle components

- Neural-network achieved 500% in neuron density. *

Partition large algorithms into sequential steps

- Image processing
- Huffman coding

Reconfigurable Computing is Good (?)

⚠️ Optimise Limited Resources

⚙️ Expands Functionality

💰 Cost Effective

⏰ Configuration Time
Functional Density Metric

$\text{Functional Density: } D = \frac{1}{\text{cost}} = \frac{1}{AT}$

- Compiles cost as a function of time and area.
- Factors in configuration times
- Directly compares static and configurable architectures
Functional Density Metric

Static Configuration: $D_S = \frac{1}{AT_E}$

Run-Time Reconfiguration: $D_R = \frac{1}{A(T_E + T_C)}$

Configuration ratio: $f = \frac{T_C}{T_E}$

$D_R = \frac{1}{AT_E(1+f)}$
Functional Density Metric

Improvement in percentage: 
\[ I = \frac{\Delta D \times 100}{D_S} = \frac{D_R - D_S}{D_S} \times 100 = \left( \frac{D_R}{D_S} - 1 \right) \times 100 \]

Recap: 
\[ D_R = \frac{1}{ATE(1+f)} \]

Max Improvement: 
\[ I_{max} = \lim_{f \to 0} \frac{D_R}{D_S} - 1 = \frac{D_{\text{MAX}}}{D_S} - 1 \]
Functional Density Metric

- Condition where configurable system density exceeds static:

\[ D_r = \frac{1}{A_r(T_r + T_c)} \geq D_s \]

\[ \frac{1}{D_s} \left( \frac{1}{A_r T_r} \right) \geq \frac{T_c}{T_r} + 1 \]

\[ \frac{D_{MAX}}{D_s} - 1 \geq \frac{T_c}{T_r} \]

\[ I_{max} \geq f \]
Improving Functional Density

• Reducing the configuration ratio $f$:
  
  • More time computing stuff ($+T_E$)
  
  • Reducing config time ($-T_C$)
    • Partial reconfiguration?

Run-Time Reconfiguration: $D_R = \frac{1}{A(T_E + T_C)}$

Configuration ratio: $f = \frac{T_C}{T_E}$

$D_R = \frac{1}{A T_E (1 + f)}$
Application Analysis: Run-Time Reconfigured Neural Network (RRANN)
## Circuit Parameters for a Neuron Network with 10,980 Connection

<table>
<thead>
<tr>
<th>Connections $n$</th>
<th>Static</th>
<th>RTR Maximum</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$ (CLBs)</td>
<td>14250</td>
<td>2702</td>
<td>2702</td>
</tr>
<tr>
<td>$T_e$ (ms)</td>
<td>1.93</td>
<td>1.93</td>
<td>1.93</td>
</tr>
<tr>
<td>$T_c$ (ms)</td>
<td>N/A</td>
<td>0</td>
<td>19.8</td>
</tr>
<tr>
<td>$f$</td>
<td>N/A</td>
<td>0</td>
<td>9.9</td>
</tr>
<tr>
<td>$D$</td>
<td>324</td>
<td>2079</td>
<td>189</td>
</tr>
<tr>
<td>$I$</td>
<td>0</td>
<td>4.28</td>
<td>-0.52</td>
</tr>
</tbody>
</table>
Partial Reconfiguration: reconfigure only a small subset of circuit, greatly reducing the configuration time.
## Circuit Parameters for a 60 Neuron Network on RRANN II

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Maximum</th>
<th>RTR</th>
<th>Global</th>
<th>Partial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connections</td>
<td>10980</td>
<td>10980</td>
<td>10980</td>
<td>10980</td>
<td>10980</td>
</tr>
<tr>
<td>( A ) (cells)</td>
<td>50764</td>
<td>18904</td>
<td>18904</td>
<td>18904</td>
<td>18904</td>
</tr>
<tr>
<td>( T_e ) (ms)</td>
<td>1.49</td>
<td>1.49</td>
<td>1.49</td>
<td>1.49</td>
<td>1.49</td>
</tr>
<tr>
<td>( T_c ) (ms)</td>
<td>N/A</td>
<td>0</td>
<td>2.42</td>
<td>1.16</td>
<td></td>
</tr>
<tr>
<td>( f )</td>
<td>N/A</td>
<td>0</td>
<td>1.62</td>
<td>.78</td>
<td></td>
</tr>
<tr>
<td>( D )</td>
<td>146</td>
<td>391</td>
<td>148</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>( I )</td>
<td>0</td>
<td>1.68</td>
<td>.014</td>
<td>.50</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>$I_{max}$</td>
<td>$f$</td>
<td>$I_{actual}$</td>
<td>Break-Even</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>------</td>
<td>--------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>RRANN$_{60}$</td>
<td>4.28</td>
<td>9.9</td>
<td>-0.52</td>
<td>138 neurons</td>
<td></td>
</tr>
<tr>
<td>RRANN-II$_{global}$</td>
<td>1.68</td>
<td>1.62</td>
<td>.014</td>
<td>58 neurons</td>
<td></td>
</tr>
<tr>
<td>RRANN-II$_{partial}$</td>
<td>1.68</td>
<td>.78</td>
<td>.50</td>
<td>28 neurons</td>
<td></td>
</tr>
<tr>
<td>Template$_{global}$</td>
<td>.588</td>
<td>.394</td>
<td>.139</td>
<td>$89 \times 89$ image</td>
<td></td>
</tr>
<tr>
<td>Template$_{partial}$</td>
<td>.588</td>
<td>.025</td>
<td>.564</td>
<td>$18 \times 18$ image</td>
<td></td>
</tr>
<tr>
<td>Edit$_{5000}$</td>
<td>.79</td>
<td>2.64</td>
<td>-0.51</td>
<td>17,700 chars</td>
<td></td>
</tr>
</tbody>
</table>
State of Art
Xilinx Dynamic
Partial
Reconfiguration
State of Art
Partial Reconfiguration
Processor Configuration
Access Port (PCAP) and
Internal Configuration
Access Port (ICAP)
ZyCap
Comparison of Resource Utilization for Different PR Methods on the Zynq.

<table>
<thead>
<tr>
<th>Method</th>
<th>Resource Utilisation</th>
<th>Throughput (MBytes/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FFs</td>
<td>LUTs</td>
</tr>
<tr>
<td>PCAP</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Xilinx ICAP (non-DMA)</td>
<td>443</td>
<td>296</td>
</tr>
<tr>
<td>Xilinx ICAP (with DMA)</td>
<td>443</td>
<td>296</td>
</tr>
<tr>
<td>ZyCAP</td>
<td>806</td>
<td>620</td>
</tr>
</tbody>
</table>
Parameterizable RegEx Operators

YiKai Wang
Regular Expression (String matching)

```
Select  
ItemName, StockingUOM  
from dbo.NES_TrackItem  
where    
    (StockingUOM LIKE '[a-z][0-9]%'  
    AND StockingUOM NOT LIKE '[a-z][0-9]%[^0-9]%')  
AND ItemName = 'V03AB15010005P'
```
Regular expression matching operator

- Databases do not operate on a fixed set of regular expressions
  - use the FPGA not only for a single instance of a query
    - but for a large number of queries.
NFA

- every component in regular expression translates to a state
- the cost of reloading the configuration is only 2 clock cycles
  - State transition only 1 clock cycles
<table>
<thead>
<tr>
<th>Pattern</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>'P.O. Box'</td>
<td>low</td>
</tr>
<tr>
<td>'Next.*Day.*Shipping'</td>
<td>medium</td>
</tr>
<tr>
<td>'a(REQIMG</td>
<td>RVWCFG)b'</td>
</tr>
<tr>
<td>'Max-dotdot[ \n]*[0-9]{3,}'</td>
<td>medium</td>
</tr>
<tr>
<td>'(P.O. Box</td>
<td>PB).*(87[0-9]{4})'</td>
</tr>
<tr>
<td>'SITE[\t\r\n\v\f]+NEWER'</td>
<td>high</td>
</tr>
</tbody>
</table>
Fig. 11: Impact of hit percentage with $P_3$
Fig. 9: Time to process a fixed number of strings using scanning and index techniques ($P_1$)
**Fig. 12**: Throughput as function of pattern complexity

- Hyper
- RE2
- DBx
- FPGA
Future trends

- The memory bandwidth of the FPGA will be increased
- Dynamic scheduling of operators on the FPGA and possibly sharing it between multiple operators
- TCAMs
- Many different database operators can be deployed dynamically
Thank you!

Questions?