MSCU: Accelerating CNN Inference with Multiple Sizes of Compute Unit on FPGAs

Author: Bao Zhenshan, Guo Junnan, Li Xiaqing, Zhang Wenbo

Lecturer: Guo Junnan

CONTENTS

1 Motivation
2 Idea
3 Solution
4 Experiment
5 Conclusion
Motivation

Deep learning Algorithm

AI Application

Motivation  Object detection on UAV

- High Accuracy
- Low power consumption
- Low latency

Accelerator for CNN
Motivation  Current FPGA-based CNN Accelerator


Motivation  DAC-SDC Skynet

Motivation Observations of Skynet

Comparison of actual computation and theoretical computation on Skynet

Motivation Invalid Computation

Actively used Idle hardware

CU size

CU size

CU size

CU size
Motivation Extension

Optimization problem

Idea Workflow

Stage one
- Dynamic selecting to each layers
- Voting from multiple layers

Stage two
- Implementing HLS design
- Deploying MSCU on FPGA

Step 1: generating a set of candidate CUs.

CU0  CU1 ... CUk

Step 2: Dynamic programming for best combination of CUs.

Step 3: generate single layer solution.
Solution Accelerator Design

Solution Compute Unit
Solution Task Scheduling for Single Layer

Definition of scheduling optimization

\[
\begin{align*}
\text{min} & \quad L(\text{num}_1, \text{num}_2, \ldots, \text{num}_n) \\
\text{s.t.} & \quad \text{Latency}_1 > \text{Latency}_2 > \ldots > \text{Latency}_n > 0 \\
& \quad \sum_{x=1}^{n} \text{DSP}_x \leq \text{target}_{\text{DSP}} \\
& \quad \sum_{x=1}^{n} \text{BRAM}_x \leq \text{target}_{\text{BRAM}} \\
& \quad \text{num}_1, \text{num}_2, \ldots, \text{num}_n \in N
\end{align*}
\]

\[
L(\text{num}_1, \text{num}_2, \ldots, \text{num}_n) = \text{Latency}_1 \times \text{num}_1 + \sum_{x=2}^{n} (\text{Latency}_x \times \max(0, \text{num}_x - \text{num}_{x-1}))
\]

Solution Generate Whole Accelerator by Voting

Algorithm 4: Voting from multiple layers

Input: \( n \) is the number of CNN. \( \text{Layer}_{\text{solution}}[n] \) are the best solution for each layers.

Output: \( \text{final}_{\text{solution}} \) is the whole CNN solution.

1. Create a map \( \text{VOTE\langle solution, int \rangle} \), which is used to store voting results;
2. for \( i = 0; i < n; i += 1 \) do
3.   \[ \text{VOTE}[\text{Layer}_{\text{solution}}[i]]++ \]
4. \( \text{final}_{\text{solution}} \) = the solution in VOTE with the most votes;
5. return \( \text{final}_{\text{solution}} \);
**Experiment**

**Software setup**
- Xilinx HLS 2018.3
- Xilinx Vivado 2018.3

**Hardware setup**
- Ultra96 V2
  - Xilinx XCZU3EG FPGA chip
  - ARM Cortex-A53
  - 2G DDR3 memory

---

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>Fixed point</td>
<td>32bits float</td>
<td>Weight:11bits Activation:9bits</td>
<td>Weight:8bits Activation:9bits</td>
</tr>
<tr>
<td><strong>Frequency</strong></td>
<td>150MHz</td>
<td>100MHz</td>
<td>215MHz</td>
<td>300MHz</td>
</tr>
<tr>
<td><strong>Platform</strong></td>
<td>Virtex6 VLX240T</td>
<td>Virtex7 VX485T</td>
<td>Zynq ZU3EG</td>
<td>Zynq ZU3EG</td>
</tr>
<tr>
<td><strong>FPGA Capacity</strong></td>
<td>37680 Slices &amp; 768 DSP</td>
<td>75900 Slices &amp; 2800 DSP</td>
<td>8800 Slices &amp; 360 DSP</td>
<td>8800 Slices &amp; 360 DSP</td>
</tr>
<tr>
<td><strong>CNN</strong></td>
<td>-</td>
<td>Alexnet</td>
<td>Skynet</td>
<td>Ultramen</td>
</tr>
<tr>
<td><strong>Model Size</strong></td>
<td>2.74GMAC</td>
<td>1.33GLOP</td>
<td>0.46GMAC</td>
<td>0.20GMAC</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>17.00GOPs</td>
<td>61.62GOPs</td>
<td>23.15GOPs</td>
<td>29.59GOPs</td>
</tr>
<tr>
<td><strong>Performance Density</strong></td>
<td>4.5x10^6 GOPs/Slices</td>
<td>8.12x10^6 GOPs/Slices</td>
<td>2.63x10^7 GOPs/Slices</td>
<td><strong>3.36x10^8 GOPs/Slices</strong></td>
</tr>
</tbody>
</table>

---

Available: https://github.com/heheda365/ultra-net
### Experiment

#### Configuration

<table>
<thead>
<tr>
<th>No.</th>
<th>Input Size</th>
<th>Layer Type</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32<em>20</em>10</td>
<td></td>
<td>64<em>10</em>5</td>
</tr>
<tr>
<td>2</td>
<td>32<em>104</em>104</td>
<td>CONV3x3(32, 64)</td>
<td>64<em>52</em>52</td>
</tr>
<tr>
<td>3</td>
<td>32<em>208</em>208</td>
<td>RELU(64, 64)</td>
<td>64<em>104</em>104</td>
</tr>
<tr>
<td>4</td>
<td>32<em>416</em>416</td>
<td>DOWN(64, 64)</td>
<td>64<em>208</em>208</td>
</tr>
<tr>
<td>5</td>
<td>32<em>160</em>80</td>
<td>MaxPooling (64, 64)</td>
<td>64<em>80</em>40</td>
</tr>
<tr>
<td>6</td>
<td>32<em>220</em>160</td>
<td></td>
<td>64<em>160</em>80</td>
</tr>
</tbody>
</table>

Comparison of the latency cycles of customize layers

### Experiment

#### Combination

<table>
<thead>
<tr>
<th>Combination</th>
<th>CU Size</th>
<th>10*10</th>
<th>20*20</th>
<th>40*40</th>
<th>80*80</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1-1</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1-2</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1-3</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>C1-4</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2-1</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2-2</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>C2-3</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>C3-1</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>C4-1</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Comparison of the latency cycles
## Experiment

<table>
<thead>
<tr>
<th>Combination</th>
<th>Amount</th>
<th>FM Size</th>
<th>Invalid Rate</th>
<th>Cycles (x10³)</th>
<th>LW/C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1-1</td>
<td>1764</td>
<td>420*420</td>
<td>2%</td>
<td>149.1</td>
<td>0.33</td>
</tr>
<tr>
<td>C1-2</td>
<td>441</td>
<td>420*420</td>
<td>2%</td>
<td>130.1</td>
<td>0.16</td>
</tr>
<tr>
<td>C1-3</td>
<td>121</td>
<td>440*440</td>
<td>11%</td>
<td>141.4</td>
<td>0.15</td>
</tr>
<tr>
<td>C1-4</td>
<td>36</td>
<td>480*480</td>
<td>25%</td>
<td>167.6</td>
<td>0.14</td>
</tr>
</tbody>
</table>

*FM Size: the size of feature map
LW/C: the proportion of load and write (LW) and computation (C) cycles*
Conclusion

More efficient practical deep learning

- Balance between accuracy, latency and resource
- From customized computing to automatic designing

Reference


Thank you

Faculty of Information Technology
Beijing University of Technology