Transparent Binary Acceleration Using Reconfigurable Array

Jongkyung Paek, Yangsoo Kim, and Kiyoung Choi
School of EECS
Seoul National University

Jongeun Lee
School of ECE
UNIST

Outline

• Introduction
• FloRA
• Configurable Range Memory
• Experiments
• Conclusion
Introduction

• Binary acceleration

- FloRA: coarse-grained reconfigurable array (CGRA)
- Wide range of application kernels
- Low run-time configuration overhead

• Transparency
  - Automatic insertion of memory management code
    • For memory-centric communication
    • Configurable range memory (CRM) allows efficient and transparent data communication between the processor and the CGRA
  - Need static analysis of binary code
    • Currently not 100% transparent
FloRA

• Architecture

Processor ➔ Cache ➔ Network

Memory Controller

Shared Memory

Exclusive Memory

scratchpad memory

RCM Control Unit

PE Array

Configuration Cache

FloRA

• Binary acceleration on FloRA

Processor ➔ Cache ➔ Network

Memory Controller

Shared Memory

scratchpad memory

RCM Control Unit

Processor

Cache

Network

Memory Controller

Shared Memory

Exclusive Memory

scratchpad memory

RCM Control Unit

PE Array

Configuration Cache

Binary ➔ Static Analysis ➔ HLS ➔ CGRA Code

Frequent Loops

Data Preparation + Modify Binary ➔ Partitioned CGRA Code

Modified Binary ➔ HLS

Partition

Binary

Static Analysis

HLS

CGRA Code
Configurable Range Memory

• Memory-centric communication

Main Proc.  L1$  L2$  Main Memory

SPM  ←→  RCM

Configurable Range Memory

• Memory-centric communication

Main Proc.  L1$  L2$  Main Memory

CRM  ←→  RCM
Configurable Range Memory

- CRM
  - Range recognizer

Address

<table>
<thead>
<tr>
<th>Tag</th>
<th>Index</th>
<th>Blk Offset</th>
</tr>
</thead>
</table>

Tag Reg

- Valid Bit
- Dirty Bit

Address Space

A block (Index=0)
All addresses here have the same tag (=0x4000)
Valid blocks

Example:
Addr=0x4000_30_00
Tag=0x4000
Index=0x30
Hit if Tag Reg=0x4000 and BValid[0x30]=1

Configurable Range Memory

- CRM
  - Multiple range recognizers per bank
  - Multiple banks

Tag Reg0
Tag Reg1
Tag Reg2
Tag Reg3

Bank0
Bank1

Array_A
Array_B
Experiments

- Experimental setup
  - Main processor: ARM926EJ-S
  - D cache has 32KB
  - RCM: 4x4 array of PEs
  - 4 banks of CRM
  - 4KB (8KB) for each bank
  - 2 range recognizers per bank
  - AHB is used for the bus
  - Access to CRM takes 1 cycle
Experiments

• **DSPstone benchmark result**

<table>
<thead>
<tr>
<th></th>
<th>SW</th>
<th>RCM</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Comp.</td>
<td>Comm.</td>
<td>Conv.</td>
</tr>
<tr>
<td>Complex multiply (N=16)</td>
<td>2928</td>
<td>24</td>
<td>1199</td>
</tr>
<tr>
<td>Complex updates (N=16)</td>
<td>3807</td>
<td>24</td>
<td>1744</td>
</tr>
<tr>
<td>Convolution (16-point)</td>
<td>939</td>
<td>16</td>
<td>373</td>
</tr>
<tr>
<td>Dot product (N=16)</td>
<td>1429</td>
<td>16</td>
<td>601</td>
</tr>
<tr>
<td>FIR (16 taps)</td>
<td>1492</td>
<td>17</td>
<td>319</td>
</tr>
<tr>
<td>FIR 2D (3x3 coeff)</td>
<td>5782</td>
<td>40</td>
<td>651</td>
</tr>
<tr>
<td>IIR biquad (4 sections)</td>
<td>949</td>
<td>16</td>
<td>67</td>
</tr>
<tr>
<td>LMS (16 taps)</td>
<td>1810</td>
<td>25</td>
<td>275</td>
</tr>
<tr>
<td>Matrix mult (4x4)</td>
<td>3813</td>
<td>36</td>
<td>1715</td>
</tr>
<tr>
<td>N real updates (N=16)</td>
<td>1950</td>
<td>12</td>
<td>1250</td>
</tr>
<tr>
<td>Average (geometric mean)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Experiments

• **JPEG decoder result**

<table>
<thead>
<tr>
<th></th>
<th>SW</th>
<th>Conventional</th>
<th>CRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td></td>
<td>355,947</td>
<td>355,947</td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td>4,793,885</td>
<td>547,501</td>
</tr>
<tr>
<td>Management</td>
<td></td>
<td></td>
<td>189,996</td>
</tr>
<tr>
<td>Total</td>
<td>7,511,315</td>
<td>5,149,832</td>
<td>1,093,444</td>
</tr>
<tr>
<td>Kernels Speedup</td>
<td>1.00</td>
<td>1.46</td>
<td>6.87</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Array</th>
<th>Size (Bytes)</th>
<th>Scope</th>
<th>Address</th>
<th>in/out</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>256</td>
<td>global</td>
<td>fixed</td>
<td>in</td>
</tr>
<tr>
<td>B</td>
<td>64</td>
<td>global</td>
<td>sweeping</td>
<td>in</td>
</tr>
<tr>
<td>C</td>
<td>128</td>
<td>global</td>
<td>sweeping</td>
<td>in+out</td>
</tr>
<tr>
<td>D</td>
<td>128</td>
<td>local</td>
<td>fixed</td>
<td>in+out</td>
</tr>
<tr>
<td>E</td>
<td>128</td>
<td>local</td>
<td>fixed</td>
<td>in+out</td>
</tr>
<tr>
<td>F</td>
<td>64</td>
<td>local</td>
<td>fixed</td>
<td>in+out</td>
</tr>
</tbody>
</table>
Experiments

• MPEG-2 encoder result

<table>
<thead>
<tr>
<th></th>
<th>SW</th>
<th>Conventional</th>
<th>CRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>-</td>
<td>90M</td>
<td>90M</td>
</tr>
<tr>
<td>Overhead</td>
<td>-</td>
<td>547M</td>
<td>267M</td>
</tr>
<tr>
<td>Total</td>
<td>3,322M</td>
<td>638M</td>
<td>357M</td>
</tr>
<tr>
<td>Speedup</td>
<td>1.00</td>
<td>5.21</td>
<td>9.29</td>
</tr>
<tr>
<td>Others Total</td>
<td>523M</td>
<td>523M</td>
<td>523M</td>
</tr>
<tr>
<td>MPEG-2 Cycles</td>
<td>3,846M</td>
<td>1,161M</td>
<td>881M</td>
</tr>
<tr>
<td>Speedup</td>
<td>1.00</td>
<td>3.31</td>
<td>4.36</td>
</tr>
</tbody>
</table>

Conclusion

• Binary acceleration with FloRA
  – Tried to make it transparent with CRM architecture
  – Less communication overhead between the processor and the accelerator

• Future work
  – Improving CRM architecture for efficient access from RCM
  – Automatic code generation
  – Application to other kinds of accelerators
Thank you!