Multi-Mode Pipelined MPSoCs for Streaming Applications

Haris Javaid    Daniel Witono    Sri Parameswaran

January 23, 2013

School of Computer Science and Engineering
The University of New South Wales
AUSTRALIA
Multimedia platforms are increasingly becoming heterogeneous
- Single-/Multi-processor host system
- Graphics and streaming accelerators
- Examples: Tegra, OMAP, etc.

Streaming accelerators
- Optimized at design-time for performance and energy efficiency
- Based upon pipelined MPSoCs rather than ASICs in this work
Pipelined MPSoCs

- Pipelined MPSoCs
  - Task- and pipeline-level parallelisms (data-flow structure)
  - Data- and instruction-level parallelisms (SIMD and VLIW instructions)
Research Aim

- All the streaming accelerators may not be active at all times
  - Either browse pictures or watch videos (H.264 and JPEG not used simultaneously)
  - Differing standards (H.264, MPEG, VC1 not used simultaneously)

- Multi-mode accelerator
  - Combine mutually exclusive accelerators to reduce area footprint
  - Simultaneously active accelerators are not combined

- Our proposal
  - Multi-mode Pipelined MPSoCs – a mode refers to execution of one streaming application
  - Combine application graphs, and then derive a multi-mode pipelined MPSoC
An Example

Assumptions

- An application graph represents the corresponding pipelined MPSoC
- Notation \( m.n_x \) means \( n-th \) node in \( m-th \) stage of \( x-th \) application

Application 1

Application 2

Merged Graph

Multi-mode Pipelined MPSoC
Problem Statement

- Given X application graphs, the goal is to combine them into one graph such that the pipelined MPSoC derived from it has minimal area.

- In the merged graph:
  - Minimize the number of nodes: cost of processors
  - Minimize the number of edges: cost of processor/FIFO ports
  - Minimize the capacity of edges: cost of FIFO sizes

- Merging applications graph is an **NP complete problem** [reference in paper]

- Our Solution:
  - Near-optimal but fast heuristics
    - MaxS
    - MaxN
  - Optimal heuristic
    - MaxC
Related Work

- Data-path merging in digital design
  - Bipartite graph matching [DATE’01]
  - Subsequence/Substring matching [DAC’04]
  - Finding maximum clique [IEEE TCAD’05] [IEEE TCAD’09]

- Typical multi-mode systems [references in paper]
  - Fixed platform
  - Involves selection of processing elements, and mapping and scheduling

- Application graphs merging
  - Merging multiple uses-cases of applications [ACM TODAES’08]
  - Merging based upon subsequence/substring matching [JRC’12]

- Our Contribution
  - Multi-mode pipelined MPSoCs
  - Use of maximum clique approach to find optimal merging
  - Three heuristics to tradeoff accuracy with running time
Design Flows

**Application**

- **Application-level Balancing**
  - merge/split tasks

- **One-to-one Mapping**

- **System-level Balancing**
  - customize processors

- **Pipelined MPSoC**

**Applications**

- **Application-level Balancing**
  - merge/split tasks of each application separately

- **Application Graphs Merging**

- **One-to-one Mapping**

- **System-level Balancing**
  - customize processors of each application separately

- **Multi-mode Pipelined MPSoC**
Heuristics MaxS (Max. Stages)

- Works on application graphs’ topologies
- Combines nodes on a stage by stage basis
Heuristics MaxS (Max. Stages)

- Works on application graphs’ topologies
- Combines nodes on a stage by stage basis

The higher of the two capacities is used
Heuristic MaxS (Max. Stages)

- Works on application graphs’ topologies
- Combines nodes on a stage by stage basis

Application 1

Application 2

Merged Graph
Heuristic MaxN (Max. Nodes)

- Nodes in the merged graph should not exceed the application with maximum number of nodes
- Combines nodes in a breadth-first manner
- Exhausts all permutations of merging graphs
Heuristics MaxN (Max. Nodes)

- Nodes in the merged graph should not exceed the application with maximum number of nodes
- Combines nodes in a breadth-first manner
- Exhausts all permutations of merging graphs

![Diagram of merged graphs with nodes and arrows indicating combinations and permutations]
Heuristic MaxN (Max. Nodes)

- Nodes in the merged graph should not exceed the application with maximum number of nodes
- Combines nodes in a breadth-first manner
- Exhausts all permutations of merging graphs
Heuristic MaxC (Max. Weight Clique)

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph
Heuristics MaxC (Max. Weight Clique)

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph

Application 1

Application 2

Partial Compatibility Graph
**Heuristic MaxC (Max. Weight Clique)**

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph

---

**Application 1**

**Application 2**

**Partial Compatibility Graph**
Heuristic MaxC (Max. Weight Clique)

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph

1. Each node has a **weight which indicates area saving** for that particular merging
2. Area saving is calculated using the **cost functions provided by the designer**
Heuristic MaxC (Max. Weight Clique)

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph

Partial Compatibility Graph → Maximum Clique → Merged Graph
Experimental Setup

- **Benchmarks**

<table>
<thead>
<tr>
<th>Application</th>
<th># Nodes</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPEG Enc</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>JPEG Dec</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>MP3 Enc</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>FFT</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>BF</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>TDE</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>Syn1</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>Syn2</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>Syn3</td>
<td>17</td>
<td>20</td>
</tr>
</tbody>
</table>

- **Pipelined MPSoCs**
  - Used LX3 processors with queue interface from Tensilica

- **Cliquer tool to find maximum weight clique**
## Results (Nodes)

<table>
<thead>
<tr>
<th>Merge</th>
<th># Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Merge</td>
</tr>
<tr>
<td>JPEGenc/dec</td>
<td>12</td>
</tr>
<tr>
<td>JPEGenc/MP3enc</td>
<td>12</td>
</tr>
<tr>
<td>JPEGdec/MP3enc</td>
<td>10</td>
</tr>
<tr>
<td>JPEGenc/dec/MP3enc</td>
<td>17</td>
</tr>
<tr>
<td>FFT/BF</td>
<td>24</td>
</tr>
<tr>
<td>FFT/TDE</td>
<td>26</td>
</tr>
<tr>
<td>BF/TDE</td>
<td>26</td>
</tr>
<tr>
<td>FFT/BF/TDE</td>
<td>38</td>
</tr>
<tr>
<td>Syn1/Syn2</td>
<td>28</td>
</tr>
<tr>
<td>Syn/Syn3</td>
<td>31</td>
</tr>
<tr>
<td>Syn2/Syn3</td>
<td>31</td>
</tr>
</tbody>
</table>
## Results (Edges)

<table>
<thead>
<tr>
<th>Merge</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Merge</td>
</tr>
<tr>
<td>JPEGenc/dec</td>
<td>14</td>
</tr>
<tr>
<td>JPEGenc/MP3enc</td>
<td>13</td>
</tr>
<tr>
<td>JPEGdec/MP3enc</td>
<td>11</td>
</tr>
<tr>
<td>JPEGenc/dec/MP3enc</td>
<td>19</td>
</tr>
<tr>
<td>FFT/BF</td>
<td>24</td>
</tr>
<tr>
<td>FFT/TDE</td>
<td>25</td>
</tr>
<tr>
<td>BF/TDE</td>
<td>25</td>
</tr>
<tr>
<td>FFT/BF/TDE</td>
<td>37</td>
</tr>
<tr>
<td>Syn1/Syn2</td>
<td>30</td>
</tr>
<tr>
<td>Syn/Syn3</td>
<td>35</td>
</tr>
<tr>
<td>Syn2/Syn3</td>
<td>35</td>
</tr>
<tr>
<td>Merge</td>
<td>Time</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>No Merge</td>
</tr>
<tr>
<td>JPEGenc/dec</td>
<td>-</td>
</tr>
<tr>
<td>JPEGenc/MP3enc</td>
<td>-</td>
</tr>
<tr>
<td>JPEGdec/MP3enc</td>
<td>-</td>
</tr>
<tr>
<td>JPEGenc/dec/MP3enc</td>
<td>-</td>
</tr>
<tr>
<td>FFT/BF</td>
<td>-</td>
</tr>
<tr>
<td>FFT/TDE</td>
<td>-</td>
</tr>
<tr>
<td>BF/TDE</td>
<td>-</td>
</tr>
<tr>
<td>FFT/BF/TDE</td>
<td>-</td>
</tr>
<tr>
<td>Syn1/Syn2</td>
<td>-</td>
</tr>
<tr>
<td>Syn/Syn3</td>
<td>-</td>
</tr>
<tr>
<td>Syn2/Syn3</td>
<td>-</td>
</tr>
</tbody>
</table>
Results (Area Saving)

- Merging cost functions
  - Two nodes saves a processor
  - Two edges saves two FIFO ports + size of smaller FIFO

62% Processor Area
57% FIFO Area
44 Processor/FIFO Ports

Throughput degradation: 1%
Latency degradation: 2%
Increase in energy/iteration: 3%
Conclusions

- Multi-mode pipelined MPSoCs can be designed by merging of application graphs
- The proposed heuristics saved up to
  - 62% processor area
  - 57% FIFO area
  - 44 processor/FIFO ports
- Miniscule degradation in performance and energy efficiency
- Future work
  - Consider memories
  - Consider code size
  - Consider simultaneously executing accelerators
Thank You!
Heuristic MaxC (Max. Weight Clique)

Partial Compatibility Graph

Maximum Clique

Merged Graph
Multimedia platforms are increasingly becoming heterogeneous:

- Single-/Multi-processor host system
- Graphics and **streaming accelerators**

**Streaming accelerators**

- Optimized at design-time for performance and energy efficiency
- Based upon **pipelined MPSoCs** rather than ASICs in this work
Pipelined MPSoCs

- Task- and pipeline-level parallelisms (data-flow structure)
- Data- and instruction-level parallelisms (SIMD and VLIW instructions)