Module 2: HW/SW Partitioning
Lecture 2.3: Target Architectures and HW Accelerator Design

Anand Raghunathan
raghunathan@purdue.edu

Fall 2014, ME 1052, T Th 12:00PM-1:15PM
Outline

• Objective of HW/SW partitioning
• Why is application-specific HW more efficient?
• Target architectures for HW/SW partitioning
• Key steps in HW accelerator design
Architectural Templates for HW/SW Partitioning

• Differ primarily in how the custom HW is integrated with the processor
  – Co-processor (a.k.a HW accelerator)
  – Custom instruction
Co-processors / HW Accelerators

- Minimal changes to the processor itself
- Accelerator is connected to the system bus or a dedicated co-processor interface
- The instruction set and pipeline structure do not change
Custom Instruction Units

- Require fine-grained integration into the processor
  - Part of the processor’s pipeline
- Typically need to be synthesized together with the processor
- ISA changes

Example: Xtensa processor from Tensilica
## Architectural Templates for HW/SW Partitioning

<table>
<thead>
<tr>
<th>Consideration</th>
<th>HW accelerator</th>
<th>Custom instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>How is custom HW interfaced?</td>
<td>System bus</td>
<td>Custom instruction interface</td>
</tr>
<tr>
<td></td>
<td>Dedicated co-proc. I/F</td>
<td>Directly into processor pipeline</td>
</tr>
<tr>
<td>How does SW access the HW</td>
<td>Memory-mapped I/O. Driver abstracts the operation performed by the accelerator as API for SW</td>
<td>Compiler maps operations to custom instructions. Intrinsics (macros).</td>
</tr>
<tr>
<td>Ability to access memory hierarchy</td>
<td>Direct access to main memory, cannot access cache / registers</td>
<td>Access typically limited to register file, same view of memory as processor</td>
</tr>
<tr>
<td>Parallel execution with the processor</td>
<td>Yes, assuming SW does not block</td>
<td>Pipelining custom instruction allows other instructions to execute concurrently</td>
</tr>
<tr>
<td>Granularity of computations targeted</td>
<td>Coarse (100s – millions of cycles)</td>
<td>Fine (few – tens of cycles)</td>
</tr>
</tbody>
</table>
Outline

• Objective of HW/SW partitioning
• Why is application-specific HW more efficient?
• Target architectures for HW/SW partitioning
• Key steps in HW accelerator design
Key steps in designing accelerators

1. Select Function(s) to Accelerate
2. Implement HW for Selected Function(s)
3. Interface to HW
4. Integrate with SW
5. Evaluate (performance, power, area)
Selecting functions to accelerate

• The four “C”s
  – Is it a Computational bottleneck?
  – What is the Communication overhead?
  – Is the HW implementation Cost-efficient?
  – Is it a Commonly used function with a Clean interface?

Amdahl’s law applied to HW acceleration

\[
\text{Speedup} = \frac{T_{\text{orig}}}{T_{\text{unaccel}} + T_{\text{accelerated}} + T_{\text{communication}}} \\
\leq \frac{1}{\frac{T_{\text{unaccel}}}{T_{\text{orig}}} + \frac{T_{\text{communication}}}{T_{\text{orig}}}}
\]