Embedded Systems
Example – Accelerator

Lothar Thiele
(contains slides from W. Wolf)

Purpose
- Show some issues in embedded system design concerning
  - performance analysis
  - hardware-software codesign
  - modeling
  - scheduling and allocation
- Motivate further parts of the lecture concerning
  - models
  - hardware architecture synthesis
  - system software

Accelerated Systems
- Use additional computational unit dedicated to some functions?
  - hardwired logic
  - extra CPU
  - coprocessor
- Applications
  - graphics and multimedia (streaming data)
  - encryption and compression
  - communication devices (signal processing)
  - supercomputing, numerical computations

System Design
- Hardware/software co-design
  - joint design of hardware and software architectures.
- Design a heterogeneous multiprocessor architecture
  - Communication (bus, network, …)
  - Memory architecture
  - Interfaces and I/O
  - Processing elements (CPU, application-specific integrated circuit, FPGA)
- Program the system
Why Accelerators?

- Better cost/performance.
  - Custom logic may be able to perform operation faster than a CPU of equivalent cost.
  - CPU cost is a non-linear function of performance.
- Better real-time performance.
  - Put time-critical functions on less-loaded processing elements.
- Less energy

Role of Performance Estimation

- First, determine that the system really needs to be accelerated.
  - How much faster is the accelerator on the core function (speed up)?
  - How much data transfer overhead?
  - How much is the overhead for synchronization with CPU?
- Performance estimation must be done on all levels of abstraction
  - Simulation based methods (only average case, need to find test patterns that stress the system)
  - Analytic methods (only limited accuracy, quick, used mainly on system level)

Accelerator Execution Time

- Total accelerator execution time: $t_{\text{accel}} = t_{\text{in}} + t_x + t_{\text{out}}$
  - Input/output time (bus transactions) include:
    - flushing register/cache values to main memory;
    - time required for CPU to set up transaction;
    - overhead of data transfers by bus packets, handshaking, etc.

Accelerator Gain

- Optimistic scenario:
  - application consists of one task only, repeated $n$ times
  - task can be executed completely on the accelerator
  - $\text{Gain} = n(t_{\text{CPU}} - t_{\text{accel}}) = n[t_{\text{CPU}} - (t_{\text{in}} + t_x + t_{\text{out}})]$
- In general:
  - not all tasks can be executed on the accelerator
  - CPU has also other obligations
  - possibilities:
    - single-threaded/blocking: CPU waits for accelerator;
    - multithreaded/non-blocking: CPU continues to execute along with accelerator, CPU must have useful work to do, software must support multi-threading
Total Execution Time

- **Single-threaded:**
  - P1
  - P2
  - A1
  - add execution time of all component processes (tasks)

- **Multi-threaded:**
  - P1
  - P2
  - A1
  - P3
  - P4
  - control flow
  - dependencies
  - more complicated … .

Sources of Parallelism

- **Overlap I/O and accelerator computation.**
  - Collect blocks (batches) of data before transmitting to and from the accelerator (amortize set up of I/O).
  - Perform operations in batches, read in second batch of data while computing on first batch.

- **Find other work to do on the CPU.**
  - May reschedule operations to move work after accelerator initiation.

Accelerator/CPU Interface Issues

- **Synchronization**
  - via interrupts
  - via special data and control registers at accelerator

- **Data transfer to main memory**
  - assisted by DMA or special logic within the accelerator
  - caching problems as CPU works on cache and accelerator works on main memory (declare data area as non-cacheable, invalidate cache after transfer, write through cache, …)

Interfacing Issues – Read and Write
Interfacing Issues – Coherency

Allocation, Mapping and Scheduling

There are four activities necessary:
- Divide functional specification into tasks.
  - Determine proper level of parallelism
  - Tasks may become processes
  - Determines granularity
- Allocate processing resources
- Map tasks onto processing elements.
- Schedule operations in time;
- All 4 activities strongly interact!

Example Communication Model

Communication:
- Assume communication within processing element is free.
- Cost of communication from P1 to P3 is d1 = 2;
  cost from P2 to P3 communication is d2 = 4.
- First design: Map P1, P2 -> M1; P3 -> M2

Example: Scheduling and Mapping

<table>
<thead>
<tr>
<th>Task graph</th>
<th>Hardware platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
</tr>
<tr>
<td>P3</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware platform</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>P3</td>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>
Example: Scheduling and Mapping

Second Design: Map P1 -> M1; P2, P3 -> M2:

\[
\begin{array}{c}
\text{M1} \\
\text{P1} \\
\text{M2} \\
\text{P2} \\
\text{P3} \\
\text{d2} \\
\text{network} \\
5 & 10 & 15 & 20 \\
\end{array}
\]

Time = 18