HW/SW Co-design

Design of Embedded Systems
Jaap Hofstede
Version 3, September 1999
Embedded Systems

Embedded system

- is a computer system (combination of hardware and software)
- is part of a larger system (that may or may not be a computer)
- performs a fixed function for that system
## SW or HW?

<table>
<thead>
<tr>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>✍️ Flexibility</td>
<td>✍️ Performance</td>
</tr>
<tr>
<td>✍️ Late design changes</td>
<td>✍️ Low power</td>
</tr>
<tr>
<td>✍️ Features</td>
<td>(Security)</td>
</tr>
<tr>
<td>✍️ Reuse</td>
<td></td>
</tr>
<tr>
<td>✍️ Reduced time to market</td>
<td></td>
</tr>
</tbody>
</table>
# Applications

<table>
<thead>
<tr>
<th>Relatively small systems</th>
<th>Relatively large systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer electronics</td>
<td>Airplanes</td>
</tr>
<tr>
<td>Consumer products</td>
<td>Industry</td>
</tr>
<tr>
<td>Automobiles</td>
<td></td>
</tr>
<tr>
<td>Telecommunication</td>
<td></td>
</tr>
<tr>
<td>Computer peripherals</td>
<td></td>
</tr>
<tr>
<td>Copiers, FAX machines</td>
<td></td>
</tr>
<tr>
<td>Multimedia systems</td>
<td></td>
</tr>
</tbody>
</table>
Classes of embedded systems

Reactive
(Control Dominated)

- State
- Real-time

Transforming
(Data Dominated)

- Digital Signal Processing (DSP)
- Limited functionality, but very fast
Cost

Main parts of cost

- Initial cost
- Product cost
- Time to market

Design is a trade-off

- Cost
- Design time
- Performance
Technological developments

- Number of transistors: 35%
- Clock speed: 25%
- Processor performance: 50%
- Design Capacity: only about 20%

What to do with all those transistors?

- Integrate several functions
- Programmable processors
- ROM for Programs for these processors

Gap is growing
Classes of embedded systems

- Standard microprocessor or DSP with separate RAM, ROM and interfaces
- Standard microprocessor or DSP with ASIC or FPGA
- Processor core integrated on ASIC or FPGA
- ASIP
Traditional design

To manage complexity

✍ SW and HW partitioning at an early stage
✍ HW and SW designed separately
HW/SW Co-design

- Specify, explore, refine
- Flexible design strategy
- Hierarchy of models at different abstraction levels
- HW and SW designed with interaction and feedback
- Final partitioning after evaluation of trade-offs

Requires
- Co-specification
- Co-development (Co-synthesis)
- Co-verification
Goals

- Reduced time to market
- Better products
- Lower development costs
- Lower number of design cycles
- Higher level of abstraction
- Volume specific targets
Tools

- Design framework
- Common specifications
- Silicon compilers
- Software compilers
- Real-time kernels
- Simulators
- Analysis tools
Disciplines

- Application domain
- Specification and programming languages
- Compilers (!)
- VLSI design
- Parallel/distributed systems
- Real-time systems
- Formal methods
- (Performance) analysis
Specification

- Provide clear and unambiguous description
- Allow CAD tools
- Should not constrain implementation in either SW or HW
- To be compiled to other languages for HW (VHDL) or SW (C)
- Executable to verify if requirements are met
Computation models

Control dominated

- Finite State Machines (FSM)
- Communicating Sequential Processes (CSP)
- Discrete event models (VHDL)
- Petri nets

Data dominated

- Control flow graphs
- Data flow graphs
- Control/Data flow graphs
Co-simulation

Simulation of systems with a mix of different kinds of SW and HW components

Levels
- High abstraction
- Instructions
- ASIC models
- Gate
- Switch
- Analogue
Compilers

Front-end
(target independent)

- Scanning
- Parsing
- Static semantic analysis
- Optimization

Back-end
(source independent)

- Optimization
- Code generation
Code generation

1. Instruction selection
2. Register allocation
3. Instruction scheduling

Currently often separate
Characteristics of DSP’s and ASIPs make these three subtasks mutually dependent
Coupling necessary for generation of optimal code
Characteristics of DSP’s

- Harvard
- Multiple busses with restricted connectivity
- High I/O-rate
- Irregular register sets of different sizes
- Special purpose registers
- Special registers for zero-overhead loops
- Multiply-add-accumulate in one instruction
- Fixed-point operations
- Hardware supported addressing modes
Example: Data path ADSP210x

∀i: 0 ≤ i ≤ n: \( y_i[j] = y_{i-1}[j] + x[j-i]a[i] \)
Compiler back-ends for DSP’s

Very hard to generate sufficiently efficient code

- Intensive optimization required
- Coupling of three subtasks
- Long compilation time acceptable
Compiler back-ends for ASIP’s

- ASIP’s often have characteristics of DSP’s
- Specific code generator required
- Retargetable code generators
- Code generator must be generated automatically from description of processor
  - From register description and instruction set (behavior)
  - From structure of ASIP data path as generated by high level synthesis (structure)