CSE 237A
Hardware/Software Codesign

Tajana Simunic Rosing
Department of Computer Science and Engineering
University of California, San Diego.
Outline & Announcements

- Mid project report due now
- Topic research due next week

Today
- HW/SW Codesign
Topic research

- PPT slides due Monday, 3/3 at 8pm via email
- Report (3 pages max in pdf) due Tuesday 3/4 at 3:30pm
- Presentations on 3/4 starting at 3:30pm; 5 min each
  □ 30% of the grade, includes peer scores
- The quality of your scoring is a part of class participation grade

<table>
<thead>
<tr>
<th>Speaker:</th>
<th>Agree</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The speaker was well organized, interesting, and spoke clearly.</td>
<td>5</td>
<td>4 3 2 1</td>
</tr>
<tr>
<td>The topic is very relevant and merits further interest.</td>
<td>5</td>
<td>4 3 2 1</td>
</tr>
<tr>
<td>Motivations and challenges related to this topic are clear to me.</td>
<td>5</td>
<td>4 3 2 1</td>
</tr>
<tr>
<td>I see what are the solutions to the issues brought up.</td>
<td>5</td>
<td>4 3 2 1</td>
</tr>
<tr>
<td>Future directions have been clearly outlined.</td>
<td>5</td>
<td>4 3 2 1</td>
</tr>
<tr>
<td>Overall, this is an excellent topic and a great presentation.</td>
<td>5</td>
<td>4 3 2 1</td>
</tr>
</tbody>
</table>

Total score (please sum all the values)

List at least one good aspect of the presentation

Suggest at least one way to improve
ES Design
ES Application Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Application</th>
<th>Processor</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data flow</td>
<td>laser printers, X-terminals, routers, bridges, image processing</td>
<td>R4600, I960, 29k, Coldfire, PPC (403, 605)</td>
<td>Processes data and passes it on. High memory bw, high throughput.</td>
</tr>
<tr>
<td>Interactive video &amp; portable</td>
<td>set-top boxes, video games, PDAs, portable info appliances</td>
<td>R3900, R4100/ 4300/ 4600, ARM 6xx/ 7xx, V851, SH1/ 2/ 3</td>
<td>Interactive, low cost, low power, high throughput.</td>
</tr>
<tr>
<td>Classic embedded</td>
<td>controllers, disk controllers, automotive, industrial control</td>
<td>Piranha, ARM, MIPS, Cores</td>
<td>mix of CPU power, low cost, low power, peripherals</td>
</tr>
</tbody>
</table>

Time-constrained computing systems.
System Design Problem Areas

1. Design environment, co-simulation constraint analysis.

2. HDL Modeling
   - Architectural synthesis
   - Logic synthesis
   - Physical synthesis


4. Test Issues
System Architecture: Yesterday

PCB design

Processor
Cache/DRAM
Controller

Cache

VRAM

DRAM

PCI Bus

SCSI/IDE
LAN

I/O

External Bus

Graphics

Add-in board

Motion
Video

VRAM

DRAM

ISA/EISA
A System Architecture: Today
HW/SW Codesign of a SoC

[Diagram of a SoC system architecture with various components labeled: Processor Core, VRAM, DSP Processor Core, Glue, Graphics, Motion, Video, SCSI, Memory, Cache/SRAM, Encryption/Decryption, LAN Interface, PCI Interface, I/O Interface, EISA Interface.]
HW-centric view of a Platform

Pre-Qualified/Verified Foundation-IP*

Hardware IP

Programmable

HW-SW Kernel + Reference Design

Scaleable bus, test, power, IO, clock, timing architectures

Processor(s), RTOS(es) and SW architecture

Reconfigurable Hardware Region (FPGA, FPGA, …)

IP can be:
- HW or SW
- hard, soft or ‘firm’ (HW)
- source or object (SW)

Foundry-Specific HW Qualification

SW architecture characterisation

SW-Centric View of Platforms

CMOS VLSI Trends

Yesterday (1980s)
- memory
- processors
- gate arrays
- ASICs

Today
- memory
- processors
- SoC
- reconfigurable
- struc. ASIC

Tomorrow
- memory
- processors
- platform SoC
- struc. SoC
- custom SoC
- reconfigurable (no processor)
- struc. ASIC (no processor)
- ASICs
Increasing Customization Cost

*Estimated Cost - $85 M - $90 M*

Example: Design with 80 M transistors in 100 nm technology

- **Top cost drivers**
  - Verification (40%)
  - Architecture Design (26%)
  - Embedded Design
    - 1400 man months (SW)
    - 1150 man months (HW)
  - HW/SW integration

12 – 18 months

Responses to Increasing Cost

- General purpose ISA
  - Universality $\rightarrow$ high volumes and reuse
  - Abstraction $\rightarrow$ compilation technologies and high application/development productivity

- Custom silicon for embedded platforms *in sufficiently high volumes*
  - Domain specific ISAs, e.g., DSPs
  - Application Specific Standard Products
  - Reconfigurable hardware

- HW/SW Codesign
HW/SW Codesign: Motivations

- Benefit from both HW and SW
  - HW:
    - Parallelism -> better performance, lower power
    - Higher implementation cost
  - SW
    - Sequential implementation -> great for some problems
    - Lower implementation cost, but often slower and higher power
Co-Design Methodology
HW/SW Codesign Issues

- **Task level concurrency management**
  Which tasks in the final system?

- **High level transformations**
  Transformation outside the scope of traditional compilers

- **Hardware/software partitioning**
  Which operation mapped to hardware, which to software?

- **Compilation**
  Hardware-aware compilation

- **Scheduling**
  Performed several times, with varying precision

- **Design space exploration**
  Set of possible designs, not just one.
Software or hardware?

Decision based on hardware/software partitioning,
Hardware/software codesign

Specification

Mapping

Processor P1  Processor P2  Hardware

\( T \leq 22765\text{ns} \)
Good partitioning mechanism:

1) Minimize communication across bus
2) Allows parallelism -> both HW & CPU operating concurrently
3) Near peak processor utilization at all times
Determining Communication Level

- Easier to program at application level
  - (send, receive, wait) but difficult to predict
- More difficult to specify at low level
  - Difficult to extract from program but timing and resources easier to predict
Partitioning Costs

- **Software Resources**
  - Performance and power consumption
  - Lines of code – development and testing cost
  - Cost of components

- **Hardware Resources**
  - Fixed number of gates, limited memory & I/O
  - Difficult to estimate timing for custom hardware
  - Recent design shift towards IP
    - Well-defined resource and timing characteristics
Software Cost Analysis Process

- Functional Blocks
- Feature Points
- Source Lines of Code (SLOC)
- Equivalent SLOC including reuse
  - Software development effort
  - Software maintenance effort
  - Software schedule
- Software Development and Testing Cost
Hardware Cost Analysis Process
HW & SW Foundries

- **HW1**
  - LSI Logic ASIC Wafer Foundry Data
    - 0.18 μm feature size
    - 8 inch wafers
    - 6 layers
  - TSMC 018 Wafer Processing

- **HW2**
  - Samsung Semiconductor ASIC Wafer Foundry Data
    - 0.35 μm feature size
    - 6 inch wafers
    - 4 layers
  - TSMC 035 Wafer Processing

- **SW1**
  - Nominal to High development effort

- **SW2**
  - Low to Nominal development effort
MIXED Implementation Using HW1 and SW1

Percentage of Total Cost:
- Software development
- Testing
- Packaging
- Fabrication
- Tooling
- Design
- Reuse of:
  - Gate-level IP
  - Code

Production Quantity and Level of Reuse:
- 1000, No
- 1000, 20%
- 1000, 40%
- 10000, No
- 10000, 20%
- 100000, No
- 100000, 20%
- 100000, 40%
- Recurring
Total Cost Per Chip

Percent Custom Hardware

Total Cost ($/chip)

10,000 Units
Partitioning Analysis

- Result of compilation is synthesizable HDL and assembly code for the processor
- Compiler & profiler determine dependence and rough performance estimates
Hardware/Software Partitioning

Simple architectural model: CPU + 1 or more ASICs on a bus

Properties of classic partitioning algorithms
- Single rate; Single-thread: CPU waits for ASIC
- Type of CPU is known; ASIC is synthesized
HW/SW Partitioning Styles

- **HW first approach**
  - start with all-ASIC solution which satisfies constraints
  - migrate functions to software to reduce cost

- **SW first approach**
  - start with all-software solution which does not satisfy constraints
  - migrate functions to hardware to meet constraints
Partitioning - ILP

Ingredients:
- **Cost function** \[ C = \sum_{x_i \in X} a_i x_i \text{ with } a_i \in \mathbb{R}, x_i \in \mathbb{N} \] (1)
- **Constraints** \[ \forall j \in J : \sum_{x_i \in X} b_{i,j} x_i \geq c_j \text{ with } b_{i,j}, c_j \in \mathbb{R} \] (2)

**Def.**: The problem of minimizing (1) subject to the constraints (2) is called an **integer programming (IP)** problem.

If all \( x_i \) are constrained to be either 0 or 1, the IP problem said to be a **0/1 integer programming problem**.
Equation stored as image in order to protect against font problems

Peter Marwedel, 1/8/2004
Maximizing the cost done by setting $C' = -C$

Integer programming is NP-complete.
  - Running times increase exponentially with problem size
  - Commercial solvers can solve for thousands of variables

IP models are a good starting point for modelling even if in the end heuristics have to be used to solve them.
IP model for HW/SW partitioning

**Notation:**
- Index set $I$ denotes task graph nodes.
- Index set $L$ denotes task graph node types
  - e.g. square root, DCT or FFT
- Index set $KH$ denotes hardware component types.
  - e.g. hardware components for the DCT or the FFT.
- Index set $J$ of hardware component instances
- Index set $KP$ denotes processors.
  - All processors are assumed to be of the same type
- $T$ is a mapping from task graph nodes to their types
  
  $T: I \rightarrow L$

**Therefore:**
- $X_{i,k} = 1$ if node $v_i$ is mapped to HW component type $k \in KH$
- $Y_{i,k} = 1$ if node $v_i$ is mapped to processor $k \in KP$
- $NY_{\ell,k} = 1$ if at least one node of type $\ell$ is mapped to processor $k \in KP$
Constraints

- Operation assignment constraints

\[ \forall i \in I : \sum_{k \in KH} X_{i,k} + \sum_{k \in KP} Y_{i,k} = 1 \]

All task graph nodes have to be mapped either in software or in hardware.
Variables are assumed to be integers.
Additional constraints to guarantee they are either 0 or 1:

\[ \forall i \in I : \forall k \in KH : X_{i,k} \leq 1 \]
\[ \forall i \in I : \forall k \in KP : Y_{i,k} \leq 1 \]
Operation assignment constraints

\[ \forall \ell \in L, \forall i: T(v_i) = c_{\ell}, \forall k \in KP: NY_{\ell,k} \geq Y_{i,k} \]

- For all types \( \ell \) of operations and for all nodes \( i \) of this type:
  - if \( i \) is mapped to some processor \( k \), then that processor must implement the functionality of \( \ell \).

- Decision variables must also be 0/1 variables:

\[ \forall \ell \in L, \forall k \in KP: NY_{\ell,k} \leq 1. \]
Resource & design constraints

- \( \forall k \in KH \), the cost for components of that type should not exceed its maximum.
- \( \forall k \in KP \), the cost for associated data storage area should not exceed its maximum.
- \( \forall k \in KP \) the cost for storing instructions should not exceed its maximum.
- The total cost \( (\sum_{k \in KH}) \) of HW components should not exceed its maximum.
- The total cost of data memories \( (\sum_{k \in KP}) \) should not exceed its maximum.
- The total cost instruction memories \( (\sum_{k \in KP}) \) should not exceed its maximum.
Scheduling

Processor $p_1$

FIR$_1$  
FIR$_2$

ASIC $h_1$

Communication channel $c_1$

FIR$_2$ on $h_1$

... $v_3$  $v_4$

or

... $v_4$  $v_3$

t

... $v_7$  $v_8$

or

... $v_8$  $v_7$

t

... $e_3$  $e_4$

or

... $e_4$  $e_3$

t
Scheduling / precedence constraints

- For all nodes $v_{i1}$ and $v_{i2}$ that are potentially mapped to the same processor or hardware component instance, introduce a binary decision variable $b_{i1,i2}$ with $b_{i1,i2}=1$ if $v_{i1}$ is executed before $v_{i2}$ and $=0$ otherwise.

Define constraints of the type

(end-time of $v_{i1}$) $\leq$ (start time of $v_{i2}$) if $b_{i1,i2}=1$ and

(end-time of $v_{i2}$) $\leq$ (start time of $v_{i1}$) if $b_{i1,i2}=0$

- Ensure that the schedule for executing operations is consistent with the precedence constraints in the task graph.

- Timing constraints need to be met
Example

- HW types H1, H2 and H3 with costs of 20, 25, and 30.
- Processors of type P.
- Tasks T1 to T5.
- Execution times:

<table>
<thead>
<tr>
<th>T</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>20</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>
Operation assignment constraint

\[
\forall i \in I: \sum_{k \in KH} X_{i,k} + \sum_{k \in KP} Y_{i,k} = 1
\]

<table>
<thead>
<tr>
<th>T</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(X_{1,1} + Y_{1,1} = 1\) (task 1 mapped to H1 or to P)
\(X_{2,2} + Y_{2,1} = 1\)
\(X_{3,3} + Y_{3,1} = 1\)
\(X_{4,3} + Y_{4,1} = 1\)
\(X_{5,1} + Y_{5,1} = 1\)
Operation assignment constraint

Assume types of tasks are $\ell = 1, 2, 3, 3,$ and $1$.

$\forall \ell \in L, \ \forall i: T(v_i) = c_\ell, \ \forall k \in KP: NY_{\ell,k} \geq Y_{i,k}$

Functionality 3 to be implemented on processor if node 4 is mapped to it.
Other equations

**Time constraint:** Application specific hardware required for time constraints under 100 time units.

<table>
<thead>
<tr>
<th>T</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>20</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>

**Cost function:**

\[ C = 20 \#(H1) + 25 \#(H2) + 30 \#(H3) + \text{cost(processor)} + \text{cost(memory)} \]
Result

For a time constraint of 100 time units and cost(P)<cost(H3):

<table>
<thead>
<tr>
<th>T</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solution (educated guessing):
T1 → H1
T2 → H2
T3 → P
T4 → P
T5 → H1
Separation of scheduling and partitioning

- Combined scheduling/partitioning very complex;
  - Heuristic: Compute estimated schedule
- Perform partitioning for estimated schedule
- Perform final scheduling
- If final schedule does not meet time constraint, go to 1 using a reduced overall timing constraint.

![Diagram showing iterations and execution times](image-url)
Codesign Verification

- Run SW on the native processor
- Simulate HW (Verilog)
Co-simulation for HW & SW

- Transistor-level accurate
  - post layout SPICE model
- Gate-level accurate
  - precise HDL gate delay model
- Cycle accurate
  - correct transitions at clock edges
  - timing information between edges is thrown away
- Bus accurate
  - cycle accurate bus model
  - behavioral model of processor, hardware
- Instruction set accurate
  - instruction set simulator used for processors
  - used for early design space exploration
SpecC model

Synthesis flow
- Capture
  - Specification model
  - Architecture exploration
    - Architecture model
      - Communication synthesis
        - Communication model
          - Back-end
            - Implementation
              - Software compilation
              - Hardware compilation
                - Implementation model
                  - Manufacturing

Validation flow
- Compilation
  - Validation Analysis Estimation
    - Compilation
      - Validation Analysis Estimation
        - Compilation
          - Validation Analysis Estimation
            - Compilation
              - Validation Analysis Estimation
                - Compilation
                  - Validation Analysis Estimation
                    - Simulation model
Foresight Co-Design

System Requirements Capture → Functional Behavior Block Diagram → User-defined Reusables → Integrated Toolset

Data Flow Monitors → Architecture Block Diagram → Resource Specification → System Characteristics

Gate Count → Lines of Code → Derived from Foresight

Cost Analysis (Ghost)

HW
- I/O Count
- Number Up
- Die Size
- Fab. Cost
- SCP Cost
- Test Cost

SW
- Dev. Cost
- Dev. Schedule
- Maintenance Cost

Outputs

System Performance Metrics → System Cost → TSR

System Metrics

System Cost

Cost Analysis (Ghost)
Industry Initiatives

- Seamless Co-Verification Environment-CVE
- SystemC (language)
  - v.2.0 incorporated advantages of SpecC
- CoWare
  - Cosimulation and IP integration
  - Refine specifications (e.g., SystemC)
- New FPGA synthesis tools
  - Programmable logic + CPUs
- Platform-based design
Summary

- HW/SW codesign is complicated and limited by performance estimates
- Algorithms not as good as human partitioning
- Other interesting topics: MPSoCs
  - HW/SW codesign issues
  - Multithreading, parallelizing, scheduling
Sources and References

- Giovanni De Micheli @ EPFL
- Vincent Mooney @ Gatech
- Nikil Dutt @ UCI