Hardware/Software Codesign

Prof. Tajana Simunic Rosing
Department of Computer Science and Engineering
University of California, San Diego.
System Architecture: Yesterday

PCB design

Processor
Cache/DRAM Controller

Cache
DRAM

Audio

Motion Video

External Bus

Graphics
VRAM

PCI Bus

ISA/EISA

Add-in board

SCSI/IDE
LAN
I/O

DRAM

VRAM
A System Architecture: Today

HW/SW Codesign of a SoC
System Design Problem Areas

1. Design environment, co-simulation constraint analysis.

2. HDL Modeling
   Architectural synthesis
   Logic synthesis
   Physical synthesis


4. Test Issues
HW-centric view of a Platform

HW-SW Kernel + Reference Design

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

SW-Centric View of Platforms

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
Software or hardware?

Decision based on hardware/software partitioning
Hardware/software codesign

Specification

Mapping

Processor P1

Processor P2

Hardware

\[ T \leq 22765 \text{ns} \]

\[ \Delta t \geq 5000 \text{ns} \]
– 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
  - Calibration
  - Feature Points
  - Language Conversion

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 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
Codesign Verification

- Run SW on the CPU
- Simulate HW (Verilog)
SpecC model
Foresight Co-Design

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

State Machines → Minispecs → Library Elements

I/O Count → Number Up → Fab. Cost

Die Size → SCP Cost → Test Cost

Gate Count → Lines of Code

Input Output

Architecture Block Diagram → Resource Specification

Resource Specification → System Characteristics

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
Industry Initiatives

• Seamless Co-Verification Environment-CVE
• Proridium (Foresight)
  – Customers: Boeing, Microsoft, Raytheon, Oracle etc.
• CoWare (now in Synopsys)
  – Cosimulation and IP integration
  – One of founding members of SystemC (language)
• New FPGA synthesis tools incorporate CPUs
• Platform-based design
  – Platform: predesigned architecture that designers can use to build systems for a given range of applications
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
ILP for HW/SW Partitioning

Ingredients:

- Cost function
- Constraints

\[ C = \sum_{x_i \in X} a_i x_i \text{ with } a_i \in \mathbb{R}, x_i \in \mathbb{N} \] (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**.
FAQ on integer programming

- Integer programming is NP-complete.
  - Running times increase exponentially with problem size
  - Commercial solvers can solve for thousands of variables
  - Maximizing the cost can be done by setting $C'=-C$
- 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 CPU $k \in KP$. 

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

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

- For all types \( \ell \) of operations & 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:

\[ \forall \forall \ell \in L, \forall k \in KP: NY_{\ell,k} \leq 1. \]
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 \( (\Sigma_{k \in KH}) \) of HW components should not exceed its maximum.
- The total cost of data memories \( (\Sigma_{k \in KP}) \) should not exceed its maximum.
- The total cost instruction memories \( (\Sigma_{k \in KP}) \) should not exceed its maximum.
Scheduling

Processor $p_1$

FIR$_1$

FIR$_2$

ASIC $h_1$

Communication channel $c_1$

$\vdots v_3 \ldots v_4$

or

$\vdots v_4 \ldots v_3$

$t$

$\vdots v_7 \ldots v_8$

or

$\vdots v_8 \ldots v_7$

$t$

$\vdots e_3 \ldots e_4$

or

$\vdots e_4 \ldots e_3$

$t$
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>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>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>
Operation assignment constraint

∀i ∈ 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></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
<td></td>
<td>100</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></td>
<td></td>
<td>100</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 $\ell = 1, 2, 3, 3, \text{ and } 1$.

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

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

- $NY_{1,1} \geq Y_{1,1}$
- $NY_{2,1} \geq Y_{2,1}$
- $NY_{3,1} \geq Y_{3,1}$
- $NY_{3,1} \geq Y_{4,1}$
- $NY_{1,1} \geq Y_{5,1}$
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>20</td>
<td>12</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>10</td>
<td>10</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>

Cost function:
\[ C = 20 \#(H1) + 25 \#(H2) + 30 \#(H3) + \text{cost}(\text{processor}) + \text{cost}(\text{memory}) \]
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></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>

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

- Combined scheduling/partitioning is 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](image)
Summary

• HW/SW codesign is complicated and limited by performance estimates
• Algorithms are in research and development,
  – much of the work is still done by expert designers
Sources and References

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