Structure of this course

2: Specification

3: ES-hardware

4: system software (RTOS, middleware, …)

Design repository

5: Evaluation & validation (energy, cost, performance, …)

6: Application mapping

7: Optimization

8: Test

Numbers denote sequence of chapters
Hardware/software partitioning

No need to consider special hardware in the future?

Correct for fixed functionality, but wrong in general: “By the time MPEG-\(n\) can be implemented in software, MPEG-\(n+1\) has been invented” [de Man]

Functionality to be implemented in software or in hardware?
Functionality to be implemented in software or in hardware?

Decision based on hardware/software partitioning, a special case of hardware/software codesign.
Codesign Tool (COOL) as an example of HW/SW partitioning

Inputs to COOL:

1. Target technology
2. Design constraints
3. Required behavior
Hardware/software codesign: approach

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

Specification

Mapping

Steps of the COOL partitioning algorithm (1)

1. Translation of the behavior into an internal graph model
2. Translation of the behavior of each node from VHDL into C
3. Compilation
   • All C programs compiled for the target processor,
   • Computation of the resulting program size,
   • Estimation of the resulting execution time (simulation input data might be required)
4. Synthesis of hardware components:
   ∀ leaf nodes, application-specific hardware is synthesized.
   High-level synthesis sufficiently fast.
Steps of the COOL partitioning algorithm (2)

5. Flattening of the hierarchy:
   • Granularity used by the designer is maintained.
   • Cost and performance information added to the nodes.
   • Precise information required for partitioning is pre-computed

6. Generating and solving a mathematical model of the optimization problem:
   • Integer linear programming ILP model for optimization. Optimal with respect to the cost function (approximates communication time)
7. **Iterative improvements:**
Adjacent nodes mapped to the same hardware component are now merged.
Steps of the COOL partitioning algorithm (4)

8. **Interface synthesis:**
   After partitioning, the glue logic required for interfacing processors, application-specific hardware and memories is created.
An integer linear programming model for HW/SW partitioning

Notation:

- Index set $V$ denotes task graph nodes.
- Index set $L$ denotes task graph node types e.g. square root, DCT or FFT
- Index set $M$ 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.
An ILP model for HW/SW partitioning

- $X_{v,m} = 1$ if node $v$ is mapped to hardware component type $m \in M$ and 0 otherwise.
- $Y_{v,k} = 1$ if node $v$ is mapped to processor $k \in KP$ and 0 otherwise.
- $NY_{l,k} = 1$ if at least one node of type $l$ is mapped to processor $k \in KP$ and 0 otherwise.
- $Type$ is a mapping from task graph nodes to their types: $Type : V \rightarrow L$
- The cost function accumulates the cost of hardware units:

$$C = \text{cost(processors)} + \text{cost(memories)} + \text{cost(application specific hardware)}$$
Constraints

Operation assignment constraints

\[ \forall v \in V : \sum_{m \in M} X_{v,m} + \sum_{k \in KP} Y_{v,k} = 1 \]

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:

\[ \forall v \in V : \forall m \in M : X_{v,m} \leq 1 \]

\[ \forall v \in V : \forall k \in KP : Y_{v,k} \leq 1 \]
∀ l ∈ L, ∀ v: Type(v) = c_l, ∀ k ∈ KP : NY_{l,k} ≥ Y_{v,k}

For all types $l$ of operations and for all nodes $v$ of this type: if $v$ is mapped to some processor $k$, then that processor must implement the functionality of $l$.

Decision variables must also be 0/1 variables:
∀ l ∈ L, ∀ k ∈ KP : NY_{l,k} ≤ 1.
Resource & design constraints

- $\forall m \in M$, the cost (area) for components of type $m$ is the sum of the costs of the components of that type. This cost 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 ($\sum_{m \in M}$) 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

Communication channel $c_1$

$\text{FIR}_2$ on $h_1$  $\text{FIR}_1$  $\text{FIR}_2$

Processor $p_1$

$\text{ASIC} \ h_1$

$e_3$  $e_4$

$\begin{align*}
v_1 & \rightarrow v_2 \\
v_3 & \rightarrow v_4 \\
v_5 & \rightarrow v_6 \\
v_7 & \rightarrow v_8 \\
v_9 & \rightarrow v_{10} \\
v_{11} & \rightarrow \text{FIR}_2 \text{ on } h_1
\end{align*}$

$\begin{align*}
p_1 & \rightarrow \text{ASIC} \ h_1 \\
& \rightarrow \text{FIR}_1
\end{align*}$

$\begin{align*}
c_1 & \rightarrow e_3 \\
e_4 & \rightarrow c_1
\end{align*}$
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.

- Approach fixes the order of execution
Other constraints

- **Timing constraints**
  These constraints can be used to guarantee that certain time constraints are met.
- Some less important constraints omitted..
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 constraints (1)

<table>
<thead>
<tr>
<th>T</th>
<th>$H_1$</th>
<th>$H_2$</th>
<th>$H_3$</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>10</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>10</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>10</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

$\forall v \in V : \sum_{m \in KM} X_{v,m} + \sum_{k \in KP} Y_{v,k} = 1$

$X_{1,1} + Y_{1,1} = 1$ (task 1 mapped to $H_1$ or to $P$)
$X_{2,2} + Y_{2,1} = 1$
$X_{3,3} + Y_{3,1} = 1$
$X_{4,3} + Y_{4,1} = 1$
$X_{5,1} + Y_{5,1} = 1$
Assume types of tasks are \( l = 1, 2, 3, 3, \) and 1.
\[ \forall l \in L, \forall v: \text{Type}(v) = c_l, \forall k \in KP : NY_{l,k} \geq Y_{v,k} \]

\[
egin{align*}
NY_{1,1} & \geq Y_{1,1} \\
NY_{2,1} & \geq Y_{2,1} \\
NY_{3,1} & \geq Y_{3,1} \\
NY_{3,1} & \geq Y_{4,1} \\
NY_{1,1} & \geq Y_{5,1} \\
\end{align*}
\]

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

Time constraints leading to: Application specific hardware required for time constraints $\leq 100$ 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:

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

For a time constraint of 100 time units and \( \text{cost}(P) < \text{cost}(H3) \):

<table>
<thead>
<tr>
<th></th>
<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>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>20</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solution (educated guessing) :
\( T1 \rightarrow H1 \)
\( T2 \rightarrow H2 \)
\( T3 \rightarrow P \)
\( T4 \rightarrow P \)
\( T5 \rightarrow H1 \)
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.

<table>
<thead>
<tr>
<th></th>
<th>Actual execution time</th>
<th>Approx. execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Iteration</td>
<td><em>specification</em></td>
<td>New specification</td>
</tr>
<tr>
<td>2nd Iteration</td>
<td>Approx. execution time</td>
<td></td>
</tr>
</tbody>
</table>
Application example

Audio lab (mixer, fader, echo, equalizer, balance units); slow SPARC processor
1µ ASIC library
Allowable delay of 22.675 µs (~ 44.1 kHz)

Outdated technology; just a proof of concept.
Running time for COOL optimization

- Only simple models can be solved optimally.
Deviation from optimal design

Hardly any loss in design quality.
Running time for heuristic
Design space for audio lab

- Everything in software: 72.9 µs, 0 $\lambda^2$
- Everything in hardware: 3.06 µs, 457.9x10$^6$ $\lambda^2$
- Lowest cost for given sample rate: 18.6 µs, 78.4x10$^6$ $\lambda^2$
Positioning of COOL

COOL approach:
- shows that a formal model of hardware/SW codesign is beneficial; IP modeling can lead to useful implementation even if optimal result is available only for small designs.

Other approaches for HW/SW partitioning:
- starting with everything mapped to hardware; gradually moving to software as long as timing constraint is met.
- starting with everything mapped to software; gradually moving to hardware until timing constraint is met.
- Binary search.
HW/SW partitioning in the context of mapping applications to processors

- Handling of heterogeneous systems
- Handling of task dependencies
- Considers of communication (at least in COOL)
- Considers memory sizes etc (at least in COOL)
- For COOL: just homogeneous processors
- No link to scheduling theory