Silicon Compilation
a.k.a.
High-Level Synthesis
a.k.a.
Behavioral Synthesis
a.k.a.
Algorithmic Level Synthesis

Peeter Ellervere
Tallinn Technical University
Department of Computer Engineering
lrv@cc.ttu.ee
http://www.ttu.ee/users/lrv/

Example - differential equation
\[ \frac{d^2 y}{dx^2} + 5 \frac{dy}{dx} + 3y = 0 \]

```
sc_fix<6,10> a,dx,y,x,x1,x2,y1;
while ( true ) {
    wait(); a=inport.read(); wait(); dx=inport.read(); wait(); y=inport.read(); wait(); x=inport.read();
    wait(); u=inport.read(); while ( true ) {
        for (int i=0; i<7; i++) wait();
        x1 = x + dx;  y1 = y + (u*dx); u = u - 5*x*(u*dx) - 3*y*dx; x = x1; y = y1;
        if (!(x1<a)) break;
    }
    outport.write(y);
}
```

Software vs. Hardware
The Target Architecture

<table>
<thead>
<tr>
<th>CPU</th>
<th>Processing Unit</th>
<th>Custom HW</th>
</tr>
</thead>
<tbody>
<tr>
<td>FI</td>
<td>LD</td>
<td>EX</td>
</tr>
</tbody>
</table>

D - Data part
C - Control part

High-Level Synthesis Tasks

- Front-end:
  - Deriving an internal graph-based representation equivalent to the algorithmic description of both the data flow and the control flow
  - Compiler optimizations

- Back-end:
  - Behavioral transformations (control and/or data flow graph transformations)
  - Transforming data and control flow into register-transfer level structure - essential subtasks
  - Netlist extraction, state machine table generation

- Essential Subtasks
  - Resource allocation
    - Number and types of functional units, storage elements, and interconnections (buses)
  - Scheduling
    - Assignment of operations to time steps (subject to constraints)
  - Binding (resource assignment)
    - Operations to functional units, variables to storage elements, and data transfers to buses
Internal Representation

- Control flow model - CFG(V,E)
  - nodes - basic blocks
  - edges - flow of control
- Data flow model - DFG(V,E)
  - nodes - actors, representing operations
  - edges - links, representing data conveying paths
- CDFG (Control and Data Flow Graph)
  - control and data operations
  - control and data dependencies

Differential Equation -- CDFG

\[
\begin{align*}
\text{read} & \quad \text{read} \\
\frac{dx}{dt} & = \text{read} \\
\text{write} & \quad \text{write} \\
\end{align*}
\]

Synthesis and Scheduling

- Synthesizing an appropriate RT level structure implies meeting hardware constraints such as area, clocking frequency, delay, power consumption, etc.
- Physical parameters, however, can be estimated from the physical parameters of the hardware components in the library.
- Time is abstracted to the number of needed time steps. Time is estimated using the hardware component delays and rough estimates of the contributions by storage and interconnects.
- Depending on whether the time constraint or the area constraint is more difficult to meet, resource constrained scheduling or time constrained scheduling should be selected.

Scheduling Task

Definition

Given a set \( T \) of tasks of equal length, a partial order \( \prec \), a number of \( m \) processors, and an overall deadline \( D \), precedence constrained scheduling is defined as the following problem:

Is there a schedule \( \sigma : T \rightarrow \{0, 1, \ldots, D\} \) such that:

\( \forall t_i, t_j \in T, t_i \prec t_j \implies \sigma(t_i) < \sigma(t_j) \)?

Precedence constrained scheduling is NP-complete task.
Minimal Hardware

\[
\frac{d^2y}{dx^2} + \frac{5}{dx} x + 3y = 0
\]

**“unlimited execution time”**

Minimal Time

\[
\frac{d^2y}{dx^2} + \frac{5}{dx} x + 3y = 0
\]

**“unlimited number of execution units”**

Scheduling Approaches

- ASAP & ALAP
- Resource constrained scheduling (RCS)
  - The number of resources is limited, the goal is to minimize the number of clock steps (time)
  - List scheduling
- Time constrained scheduling (TCS)
  - The number of clock steps (time) is limited, the goal is to minimize the hardware to be allocated (the number of resources)
  - Force directed scheduling
- Special cases
  - Iterative schedulers, e.g., neural net based
  - Control flow oriented, e.g., path-based scheduler

Allocation and Binding

- Allocation and binding is the assignment of operations to hardware, possibly according to a given schedule, constraints and cost function(s)
  - Allocation -- selecting the type of the functional / storage / interconnection unit
  - Allocation -- selecting the type of the functional / storage / interconnection unit
  - Allocation of operations to FU instances (if not assignment before scheduling)
  - Assignment of values to storage elements
  - Assignment of busses (if buses are required and not allocated in advance)
  - Assignment of data to be transferred to buses (if busses are used)
- Subtasks after scheduling
  - Allocation of FUs (if not allocated before scheduling)
  - Assignment of operations to FU instances (if not assignment before scheduling)
  - Allocation of storage (if not allocated before scheduling)
  - Assignment of values to storage elements
  - Assignment of busses (if buses are required and not allocated in advance)
  - Assignment of data to be transferred to buses (if busses are used)
- Allocation and Assignment Approaches
  - rule based schemes, greedy, iterative, branch-and-bound, ILP
  - graph theoretical - clique partitioning & node coloring
Differential Equation -- Schedule and Lifetimes

Functional unit assignment

<table>
<thead>
<tr>
<th>Functional unit</th>
<th>Multiplier M1</th>
<th>Multiplier M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>1, 3, 5</td>
<td>2, 4</td>
</tr>
</tbody>
</table>

Final result for register assignment

<table>
<thead>
<tr>
<th>Register</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>u</td>
<td>x</td>
<td>y1</td>
<td>R1</td>
<td>h1</td>
<td>h3</td>
</tr>
<tr>
<td></td>
<td>h5</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>h6</td>
</tr>
</tbody>
</table>

Multiplier optimization

Control step

<table>
<thead>
<tr>
<th>Control step</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register, left input of M1</td>
<td>R1</td>
<td>R1</td>
<td>R5</td>
<td>R5</td>
<td>Rdx</td>
<td>Rdx</td>
<td>-</td>
</tr>
<tr>
<td>Register, right input of M1</td>
<td>Rdx</td>
<td>Rdx</td>
<td>R6</td>
<td>R6</td>
<td>R6</td>
<td>R6</td>
<td>-</td>
</tr>
</tbody>
</table>

Inputs may be swapped in control steps 1 and 2 because the multiply operation is commutative.

Bidirectional Bus Architecture

Differential Equation example

Bus allocation and assignment tasks

<table>
<thead>
<tr>
<th>Bus</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Transfer</td>
<td>d1</td>
<td>d2</td>
<td>d3</td>
<td>d6</td>
</tr>
<tr>
<td></td>
<td>d4</td>
<td>d7</td>
<td>d5</td>
<td>d6</td>
</tr>
<tr>
<td></td>
<td>d9</td>
<td>d8</td>
<td>d10</td>
<td>d12</td>
</tr>
<tr>
<td></td>
<td>d11</td>
<td>d12</td>
<td>d13</td>
<td>d18</td>
</tr>
</tbody>
</table>

Conclusion

Scheduling

- ... affects the final quality of the design
- ... decisions can be made at rather high abstraction levels
- Many good techniques exits, specialized for certain applications
- Data and control dependencies treated in different manner

There are more tasks...

- Memory management: deals with the allocation of memories, with the assignment of data to memories, and with the generation of address calculation units.
- High-level data path mapping: partitions the data part into application specific units and defines their functionality.
- Encoding data types and control signals.
- High-level synthesis is usually integrated with the register-transfer level synthesis. Also, a tighter connection with early floor-planning is important.