Silicon Compilation

a.k.a.

High-Level Synthesis

a.k.a.

Behavioral Synthesis

a.k.a.

Algorithmic Level Synthesis

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

Compilation

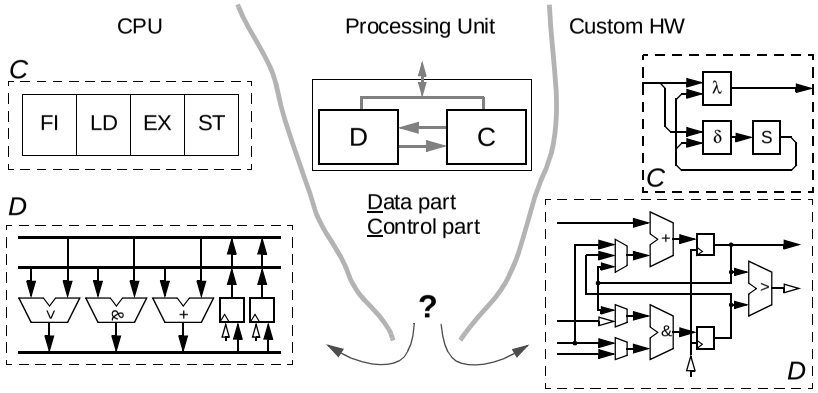
- **Example - differential equation** $\frac{d^2y}{dx^2} + 5\frac{dy}{dx}x + 3y = 0$

```
{
  sc_fixed<6,10> a,dx,y,x,u,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);
  };
}
```

```
# R1:a, R2:dx, R3:y, R4:x, R5:u,
# R6:x1, R7:x2, R8:y1, R9:tmp
     ...
_loop_$32:
     ADD.fx   R6, R4, R2      # x1=x+dx
     MUL.fx   R9, R5, R2      # tmp=u*dx
     ADD.fx   R8, R3, R9      # y1=y+tmp
     MUL.fx   R9, R4, R9      # tmp=x*tmp
     MUL.fx   R9, R9, $5      # tmp=5*tmp
     SUB.fx   R5, R5, R9      # u=u-tmp
     MUL.fx   R9, R3, R2      # tmp=y*dx
     MUL.fx   R9, R9, $3      # tmp=3*tmp
     SUB.fx   R5, R5, R9      # u=u-tmp
     ADD.fx   R4, R6, $0      # x=x1
     ADD.fx   R3, R8, $0      # y=y1
     SUB.fx   R9, R6, R1      # tmp=x1-a
     JMP.neg  _loop_$32       # ...break
     ...
```


Software vs. Hardware

The Target Architecture



High-Level Synthesis Tasks

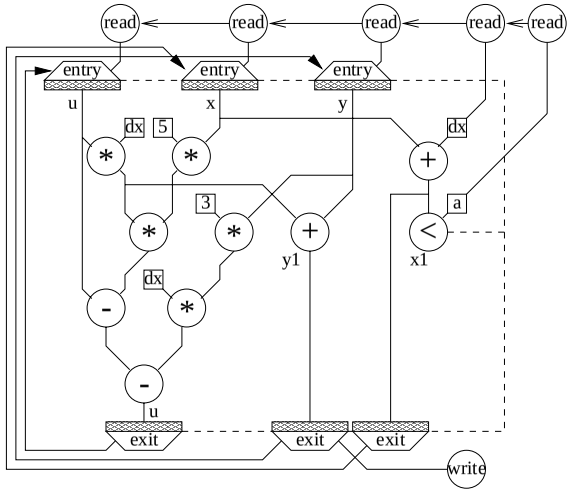
- **Front-end:**
  - Deriving an internal graph-based representation equivalent to the algorithmic description of both the data flow and the control flow
  - Compiler optimizations
- **Back-end:**
  - Behavioral transformations (control and/or data flow graph transformations)
  - Transforming data and 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
  - DFG specifies a partial order of operations
- Control flow and data flow can be combined in a single graph - CDFG (Control and Data Flow Graph)
  - control and data operations
  - control and data dependencies

Differential Equation -- CDFG



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.

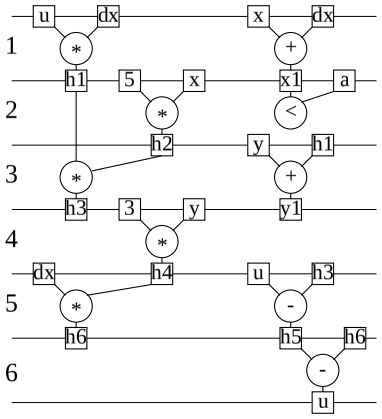
Component (LSI-10K, 16 bits)	Delay	Area
ALU(+,-,<)	24 ns	208
Adder	18 ns	125
Parallel Multiplier	49 ns	2284
2:1 Multiplexer || Tristate driver	2 ns	48
Register	1 ns	112

- 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

- Scheduling - assignment of operations to time (control steps), possibly within given constraints and minimizing a cost function
  - transformational and constructive algorithms
  - use potential parallelism, alternation and loops
  - many good algorithms exist, well understood
- Definition
  - Given a set T of tasks of equal length 1, a partial order $\prec$ on T, a number of $m \in Z^+$ processors, and an overall deadline $D \in Z^+$.
  - *Precedence constrained scheduling* is defined as the following problem:
    Is there a schedule $\sigma : T \rightarrow \{0,1,...,D\}$ such that
    $|\{ t \in T : \sigma(t)=s \}| \leq m \; \forall s \in \{0,1,...,D\}$ and $t_i \prec t_j \Rightarrow \sigma(t_i) < \sigma(t_j)$?
  - Precedence constrained scheduling is NP-complete task.

Minimal Hardware



$$\frac{ddy}{dx^2} + 5\frac{dy}{dx}\,x + 3y = 0$$

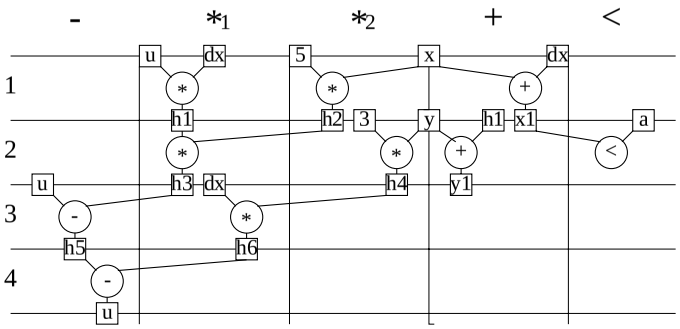
* FU1

+/-/< FU2

"unlimited execution time"

Minimal Time

$$\frac{ddy}{dx^2} + 5\frac{dy}{dx}\,x + 3y = 0$$



"unlimited number of execution units"

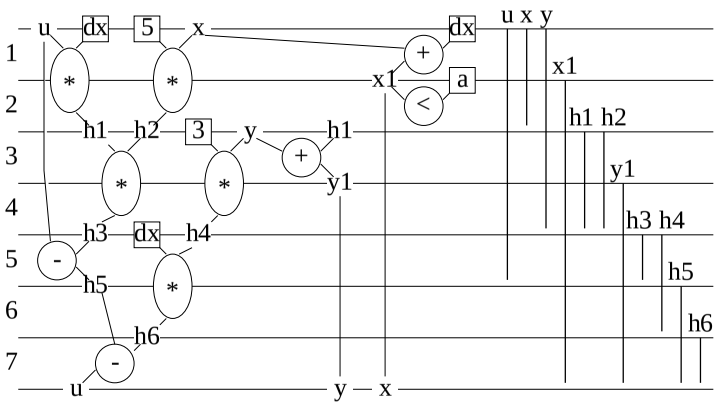
Scheduling Approaches

- 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
    - module selection -- select among several ones
  - binding -- mapping an operation to a particular hardware unit
- 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**

Functional unit	Multiplier M1	Multiplier M2
Operation	1, 3, 5	2, 4

- **Final result for register assignment**

Register	R1	R2	R3	R4	R5	R6
Value	u h5	x y1	y	x1	h1 h3 h6	h2 h4

- **Multiplexer optimization**

Control step	1	2	3	4	5	6	7
Register, left input of M1	R1	R1	R5	R5	Rdx	Rdx	-
Register, right input of M1	Rdx	Rdx	R6	R6	R6	R6	-

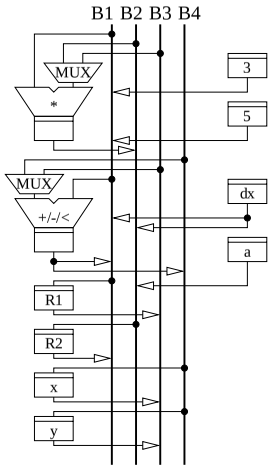
  - **Inputs may be swapped in control steps 1 and 2 because the multiply operation is commutative**

Bidirectional Bus Architecture

Differential Equation example

Bus allocation and assignment tasks

Bus	B1	B2	B3	B4
Data Transfer	d1 d4 d9 d11 d14	d2 d7 d8 d12 d15 d17 d19	d3 d5 d10 d16	d6 d13 d18



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.**