# ECE 667

**Spring 2011**

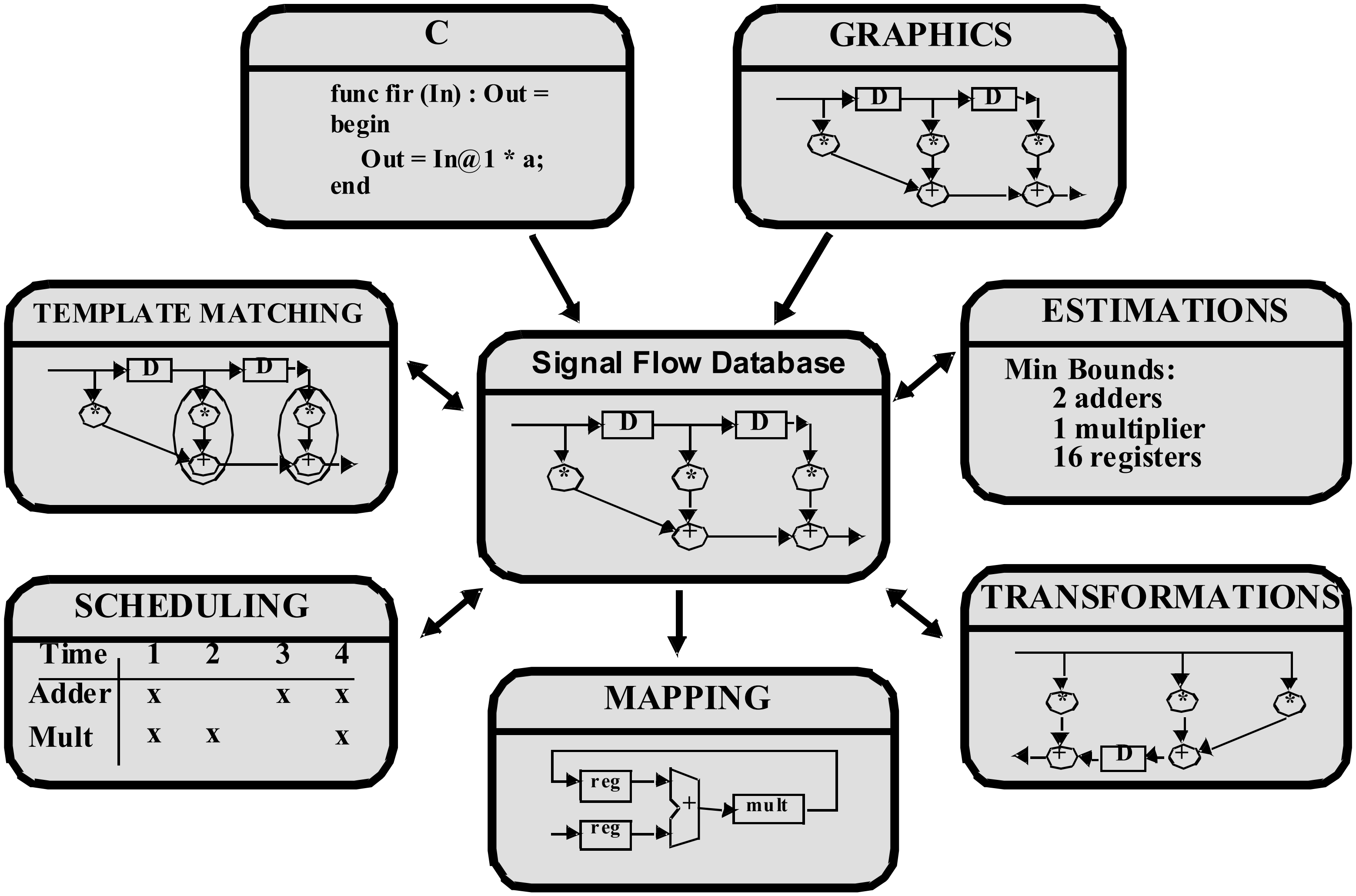
## Synthesis and Verification of Digital Circuits

## *High-Level (Architectural) Synthesis*

## High Level Synthesis (HLS)

- The process of converting a high-level description of a design to RTL
  - Input:
    - High-level languages (C, system C, system Verilog)
    - Behavioral hardware description languages (Verilog, VHDL)
    - Structural HDLs (VHDL, Verilog)
    - State diagrams / logic networks
  - Tools:
    - Parser
    - Library of modules
  - Constraints:
    - Resource constraints (no. of modules of a certain type)
    - Timing constraints (Latency, delay, clock cycle)
  - Output:
    - Operation scheduling (time) and binding (resource)
    - Control generation and RTL architecture

# High Level Synthesis



```
func fir (In) : Out =
begin
    Out = In@1 * a;
end
```

Min Bounds:
2 adders
1 multiplier
16 registers

| Time | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Adder | x |  | x | x |
| Mult | x | x |  | x |

# Example – Digital Filter design

- A second-order digital filter

Algorithm:

$$y_1(kh+h) = c.(r_1 + r_2)$$

$$r_1 = x_1(kh) + t_2(kh)$$

$$r_2 = r_1.a_{11} + t_2(kh)$$

$$t_1(kh+h) = r_3$$

$$r_3 = r_4.a_{21} + t_1(kh)$$

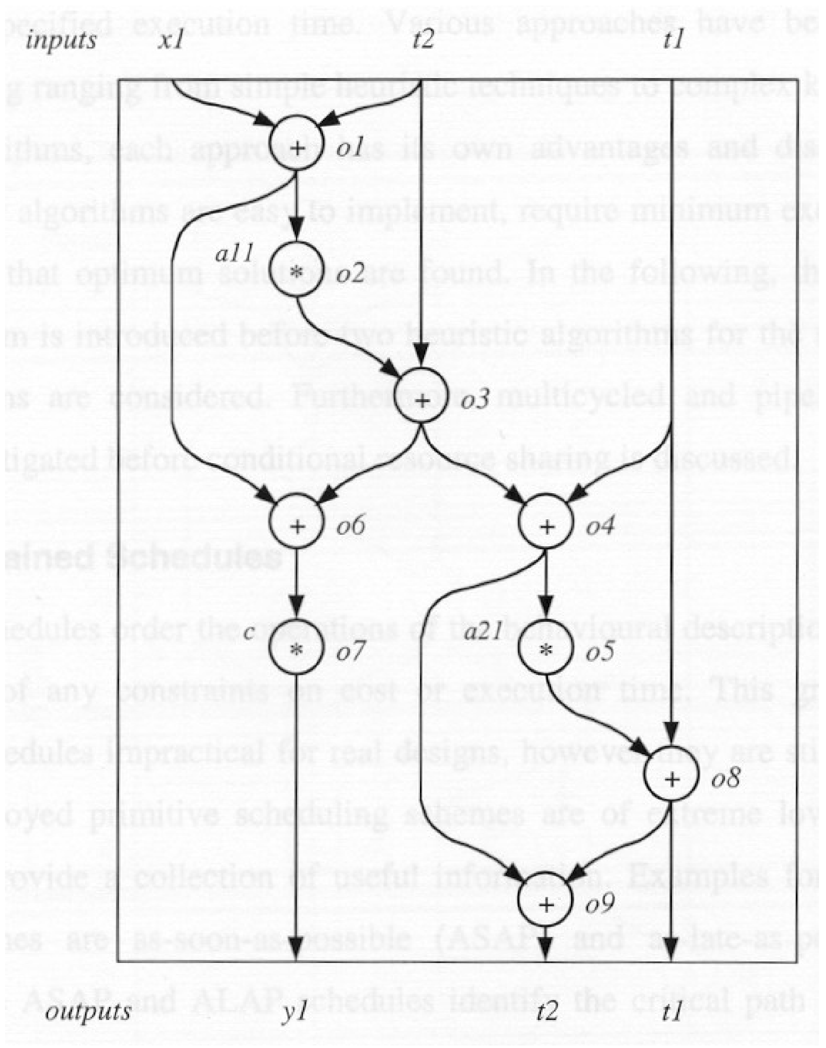
$$r_4 = r_2 + t_1(kh)$$

$$t_2(kh+h) = r_3 + r_4$$

Verilog code:

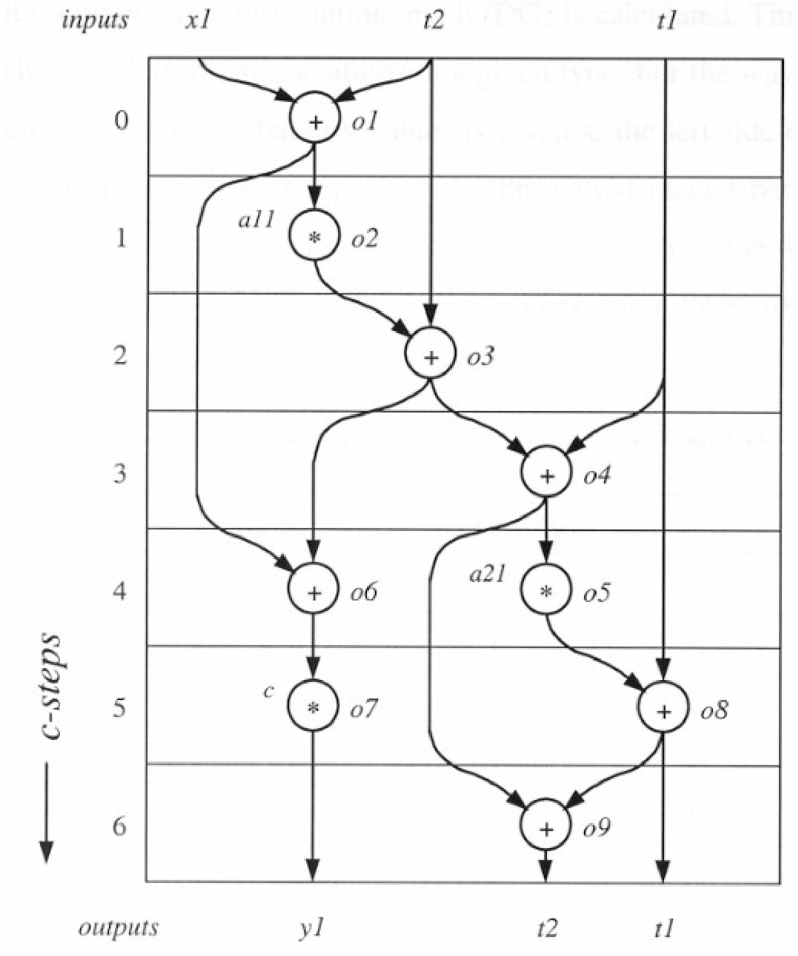
```
/* A behavioral description of a digital filter

module digital_filter(x1,y1);

input x1;
output y1;
wire [7:0] r1,r2,r3,r4,t1,t2,c,a11,a21;

      assign r1 = x1 + t2;
      assign r2 = r1 * a11 + t2;
      assign r4 = r2 + t1;
      assign r3 = r4 + a21 + t1;

      assign y1 = c* (r1 + r2);
      assign t1 = r3;
      assign t2 = r3 + r4;

endmodule
```

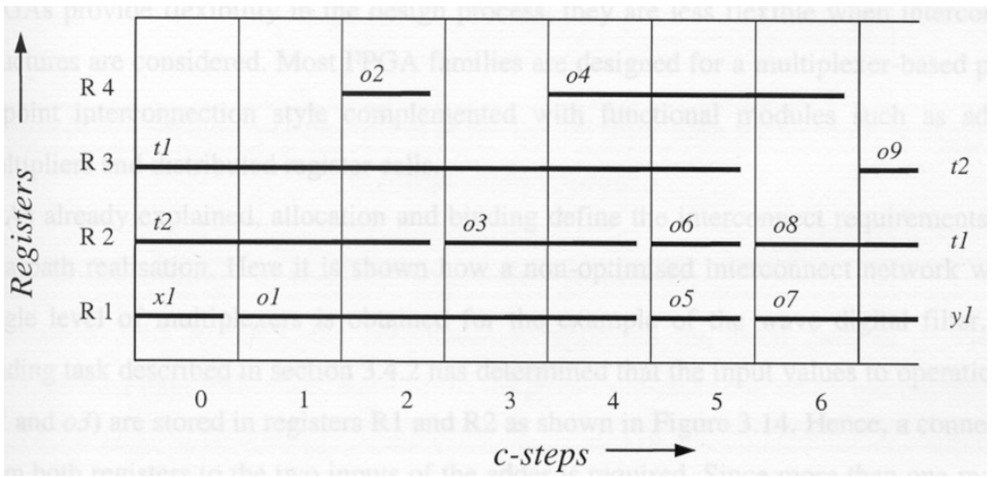
# Example – Unscheduled DFG



$$y_1(kh+h) = c.(r_1 + r_2)$$
$$r_1 = x_1(kh) + t_2(kh)$$
$$r_2 = r_1.a_{11} + t_2(kh)$$
$$t_1(kh+h) = r_3$$
$$r_3 = r_4.a_{21} + t_1(kh)$$
$$r_4 = r_2 + t_1(kh)$$
$$t_2(kh+h) = r_3 + r_4$$

# Example – Scheduling and Regs mapping

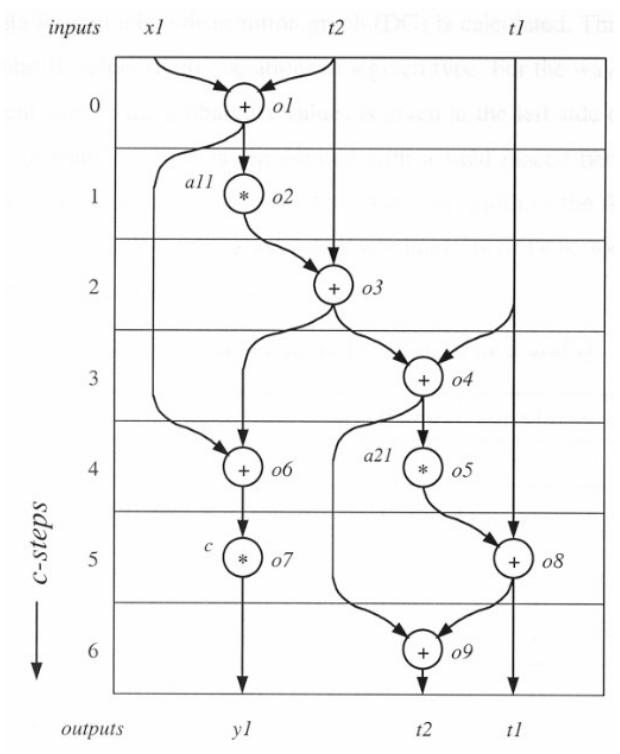


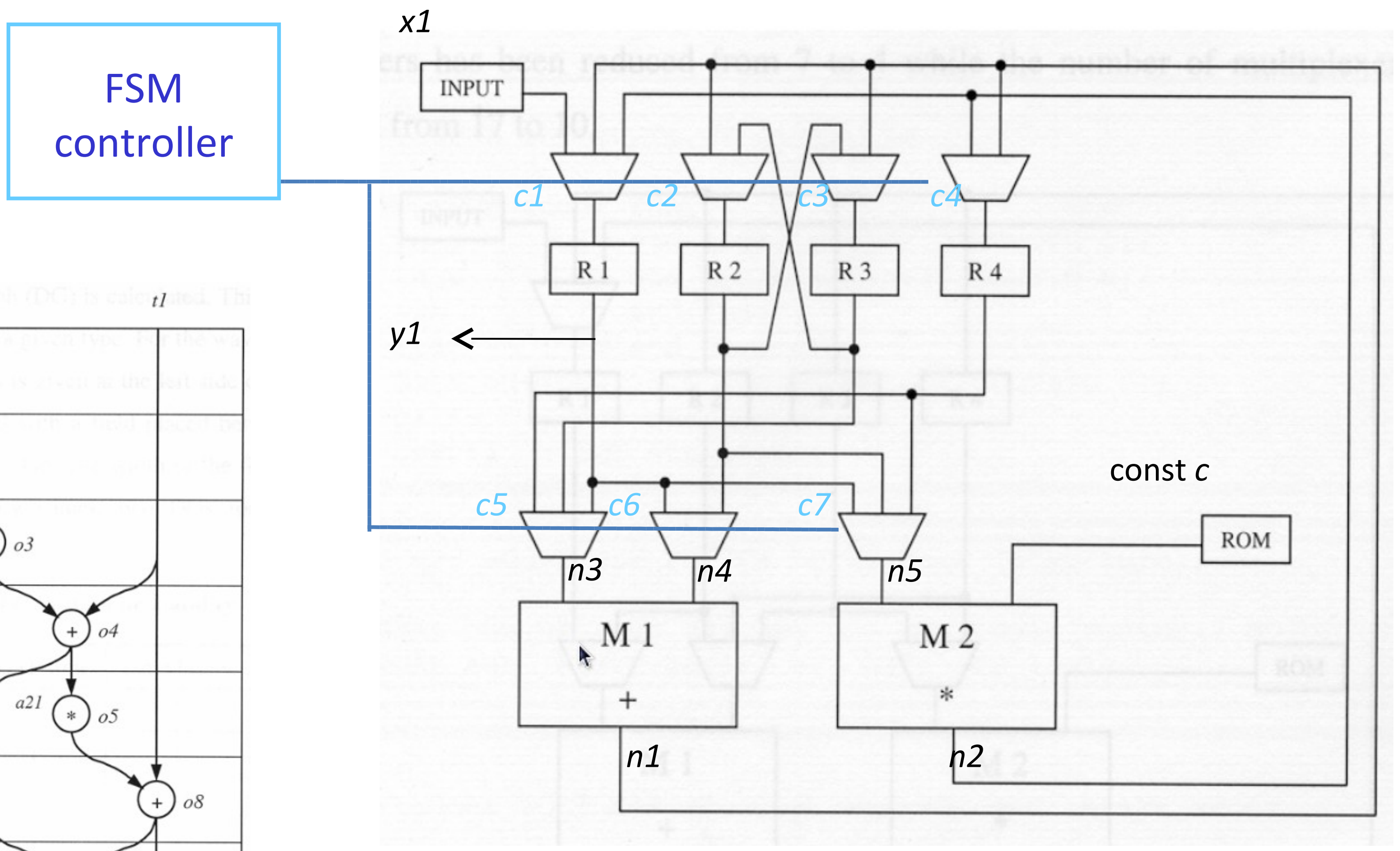


Register mapping (left-edge algorithm)

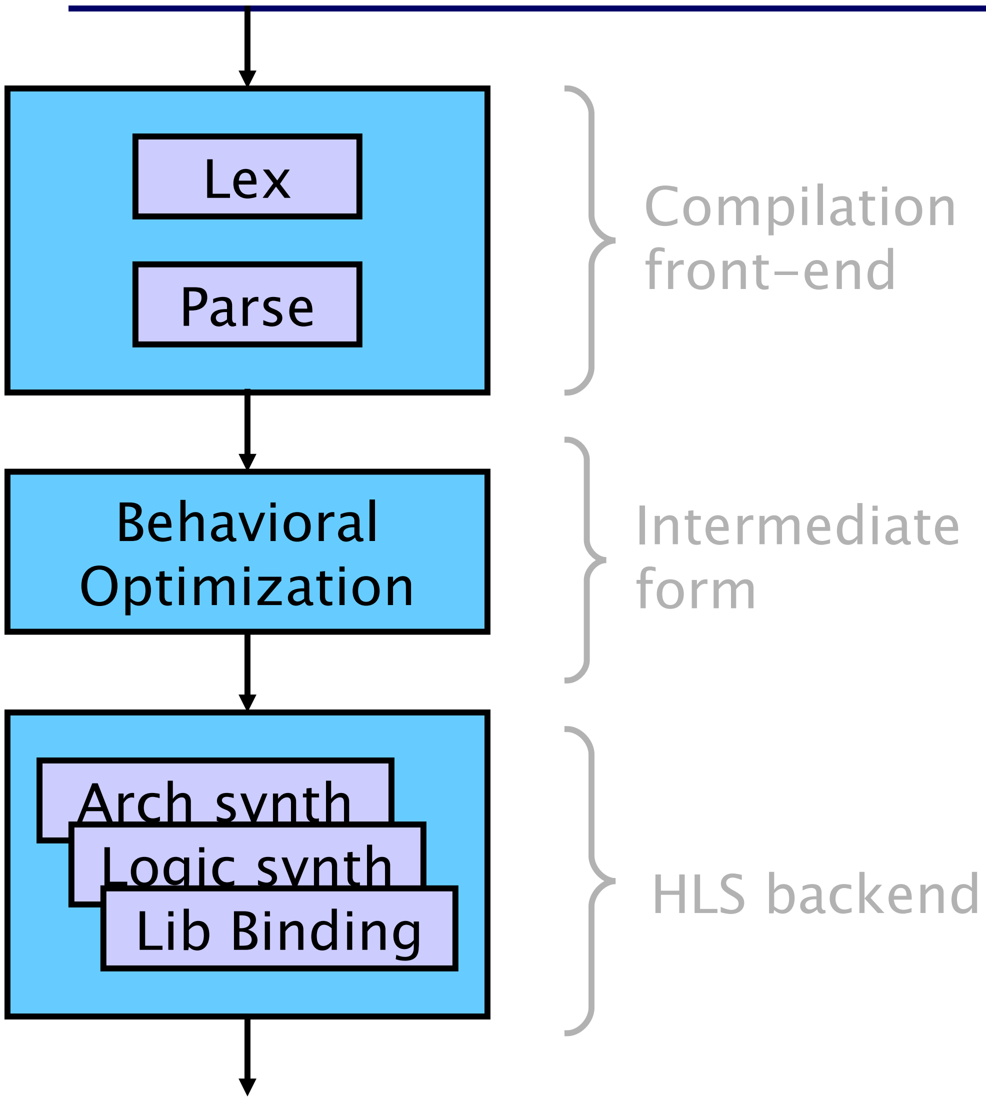
Resource-constraint scheduling ( 1 adder , 1 multiplier)

# Example – Final Architecture

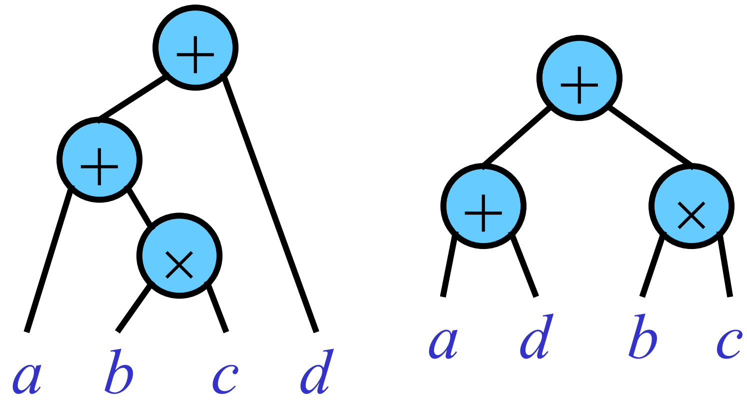




# High-Level Synthesis Compilation Flow

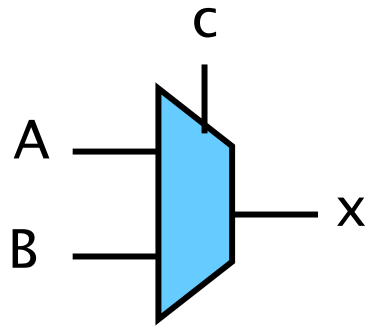


$$x = a + b \times c + d$$



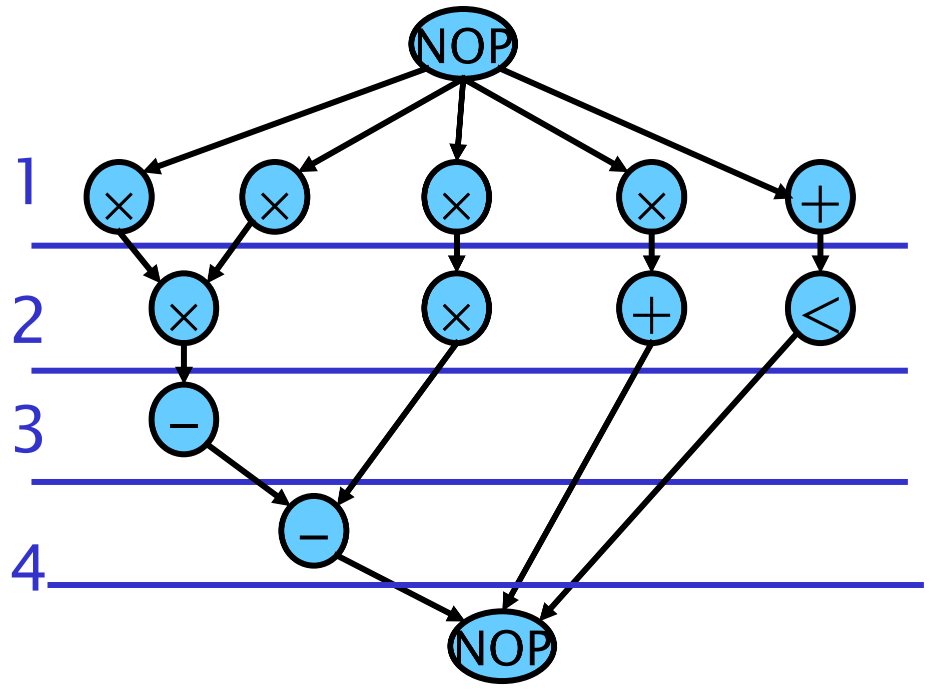
## Behavioral Optimization

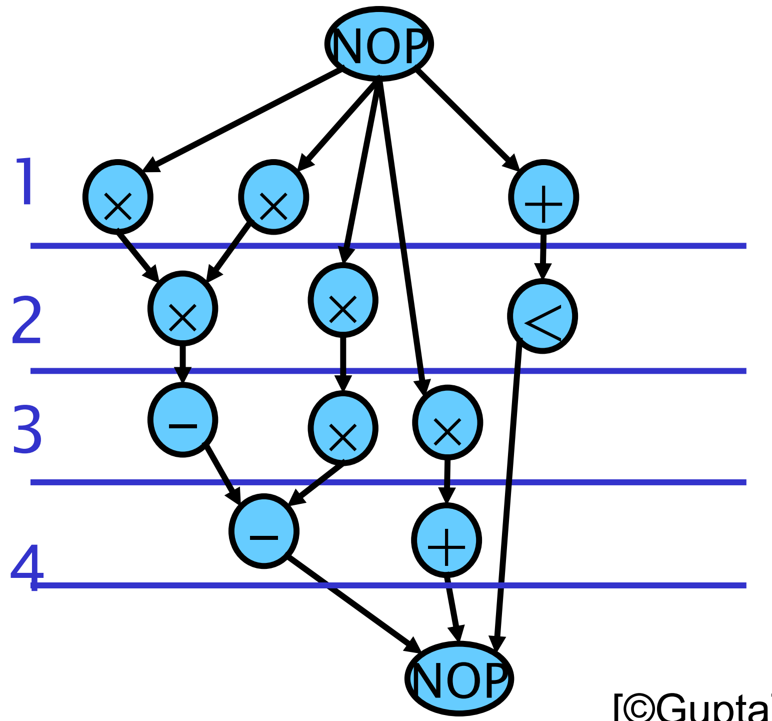
- Techniques used in software compilation
  - Expression tree height reduction
  - Constant and variable propagation
  - Common sub-expression elimination
  - Dead-code elimination
  - Operator strength reduction (e.g., *4 → << 2)
- Typical Hardware transformations
  - Conditional expansion
    - If (c) then x=A else x=B
      ➔ compute A and B in parallel, x=(C)?A:B
  - Loop unrolling
    - Instead of *k* iterations of a loop, replicate the loop body *k* times



## Optimization in Temporal Domain

- Scheduling and binding can be done in different orders or together
- Schedule:
  - Mapping of operations to time slots (cycles)
  - A scheduled sequencing graph is a labeled graph

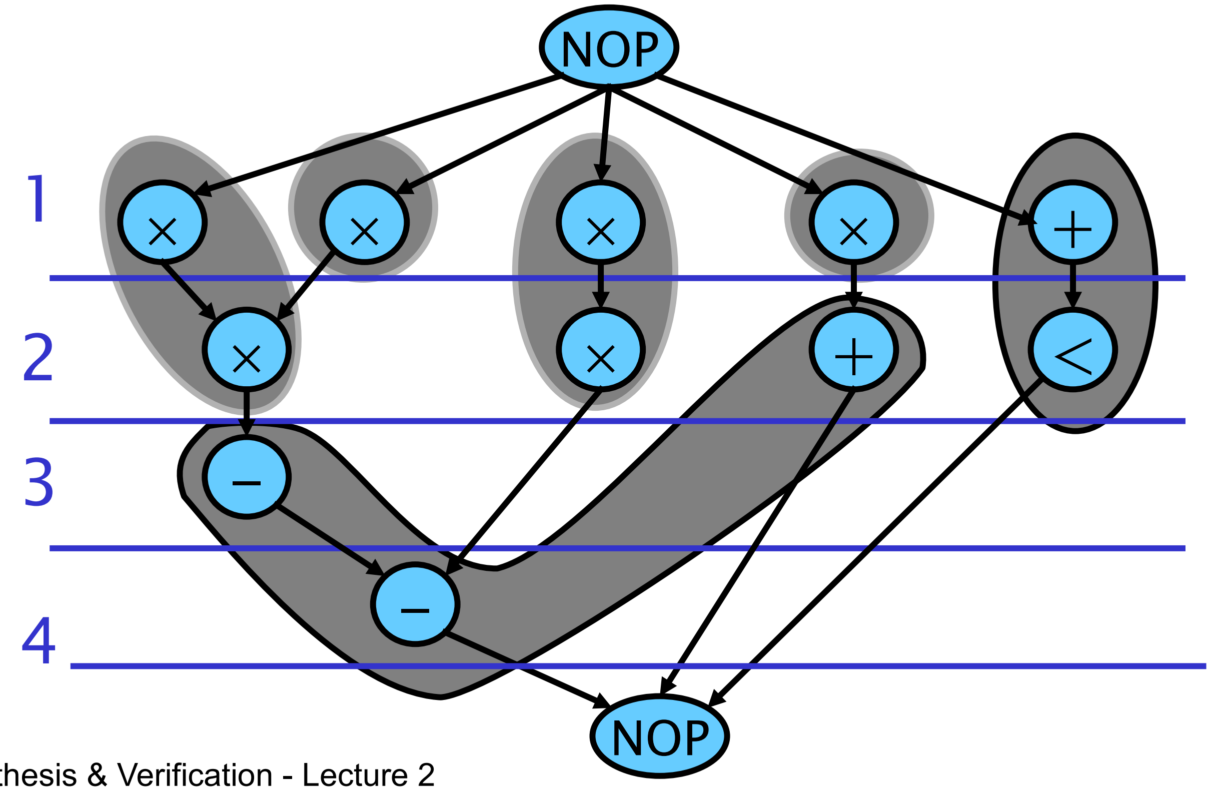




[©Gupta]

## Scheduling in Spatial Domain

- Resource sharing
  - More than one operation bound to same resource
  - Operations have to be serialized
  - Can be represented using hyperedges (define vertex partition)



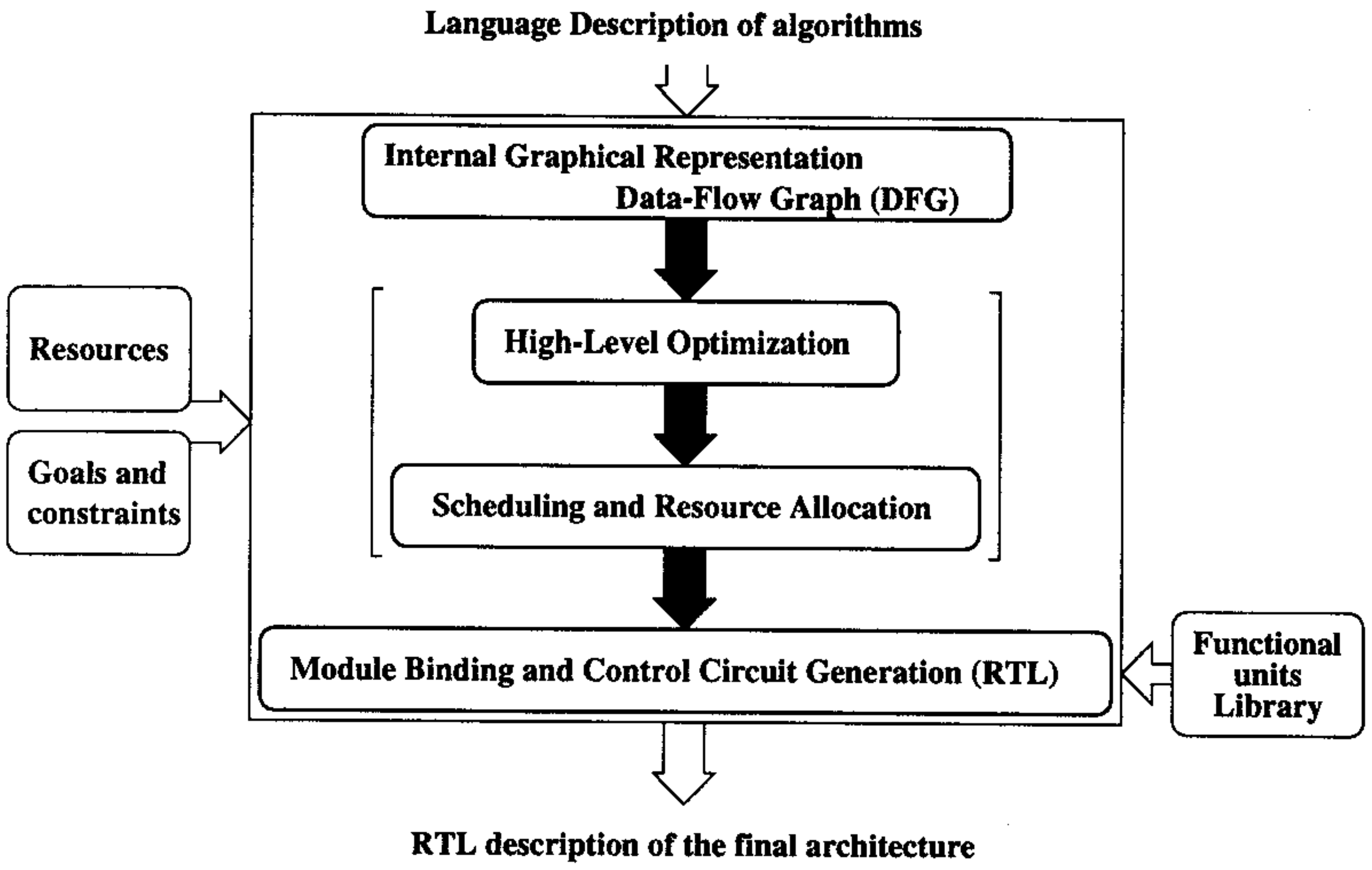
[©Gupta]

# Architectural Optimization

- Optimization in view of design space flexibility
- A multi-criteria optimization problem:
  - Determine schedule $\phi$ and binding $\beta$.
  - Under area $\mathrm{A}$, latency *L* and cycle time $\tau$ objectives
- Find non-dominated points in solution space
- Solution space tradeoff curves:
  - Non-linear, discontinuous
  - Area / latency / cycle time (more?)
- Evaluate (estimate) cost functions
- Unconstrained optimization problems for resource dominated circuits:
  - Min area: solve for minimal binding
  - Min latency: solve for minimum *L* scheduling

[©Gupta]

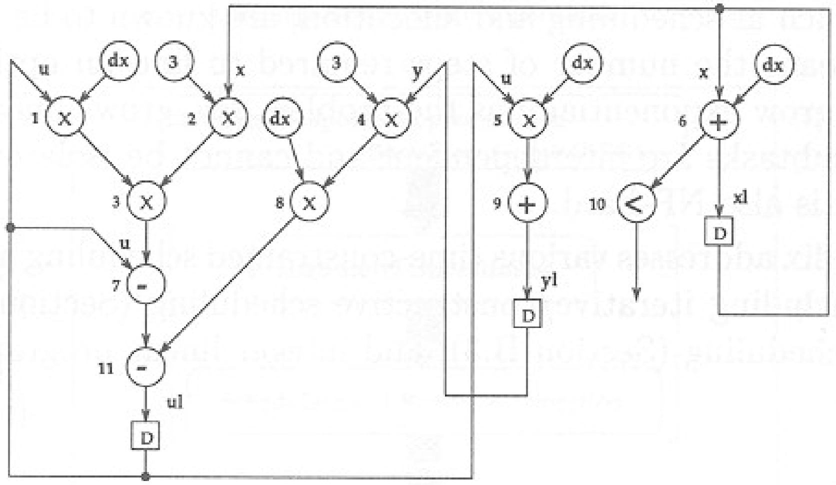
## Typical High-Level Synthesis System


Language Description of algorithms
Internal Graphical Representation
Data-Flow Graph (DFG)
Resources
Goals and constraints
High-Level Optimization
Scheduling and Resource Allocation
Module Binding and Control Circuit Generation (RTL)
Functional units Library
RTL description of the final architecture

## Algorithm Description

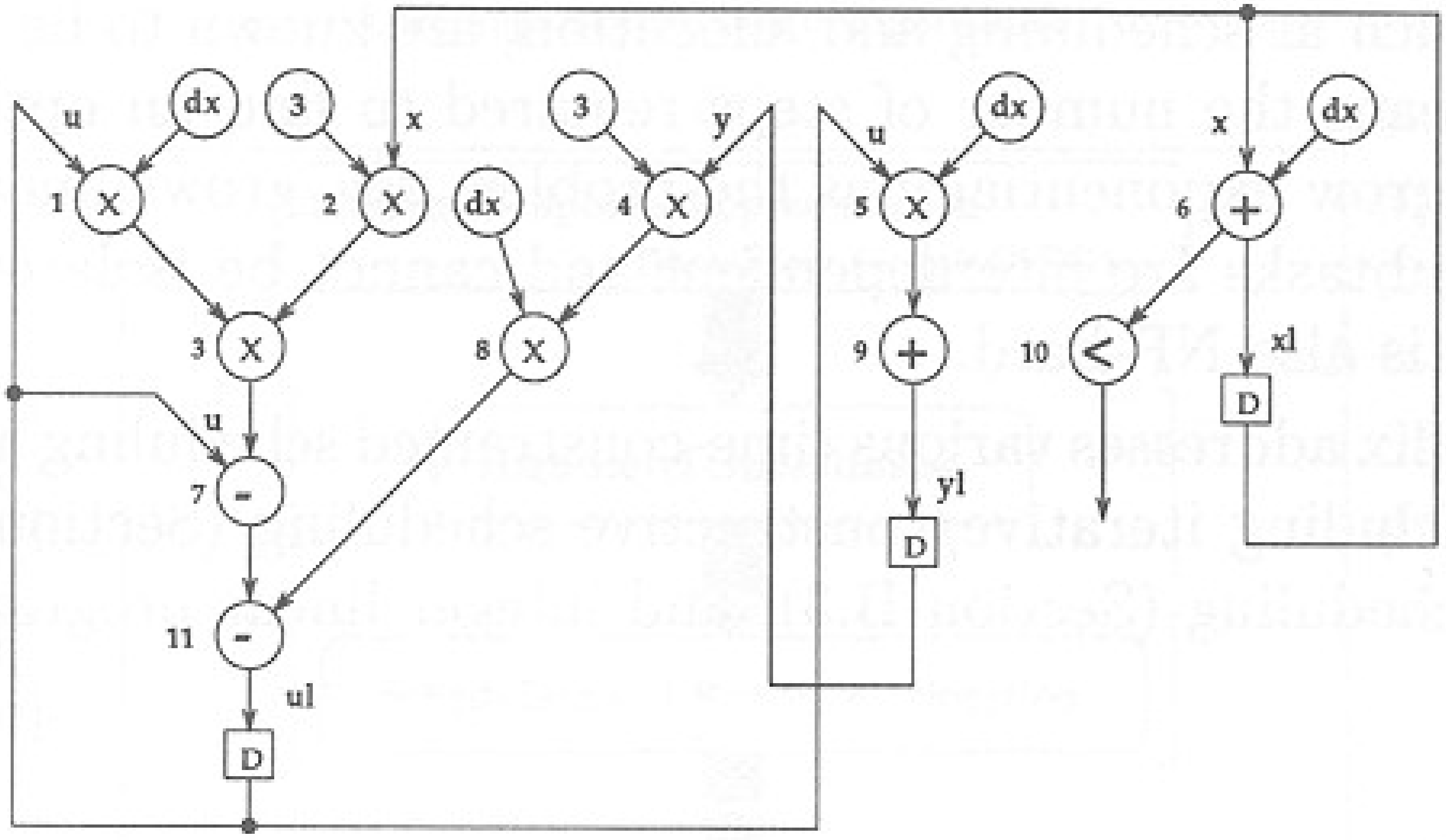
$$y'' + 3xy' + 3y = 0 \qquad u = y' = \frac{dy}{dx}$$

$$\frac{du}{dx} = y'' = \frac{d^2y}{dx^2} = -3xy' - 3y = -3xu - 3y$$

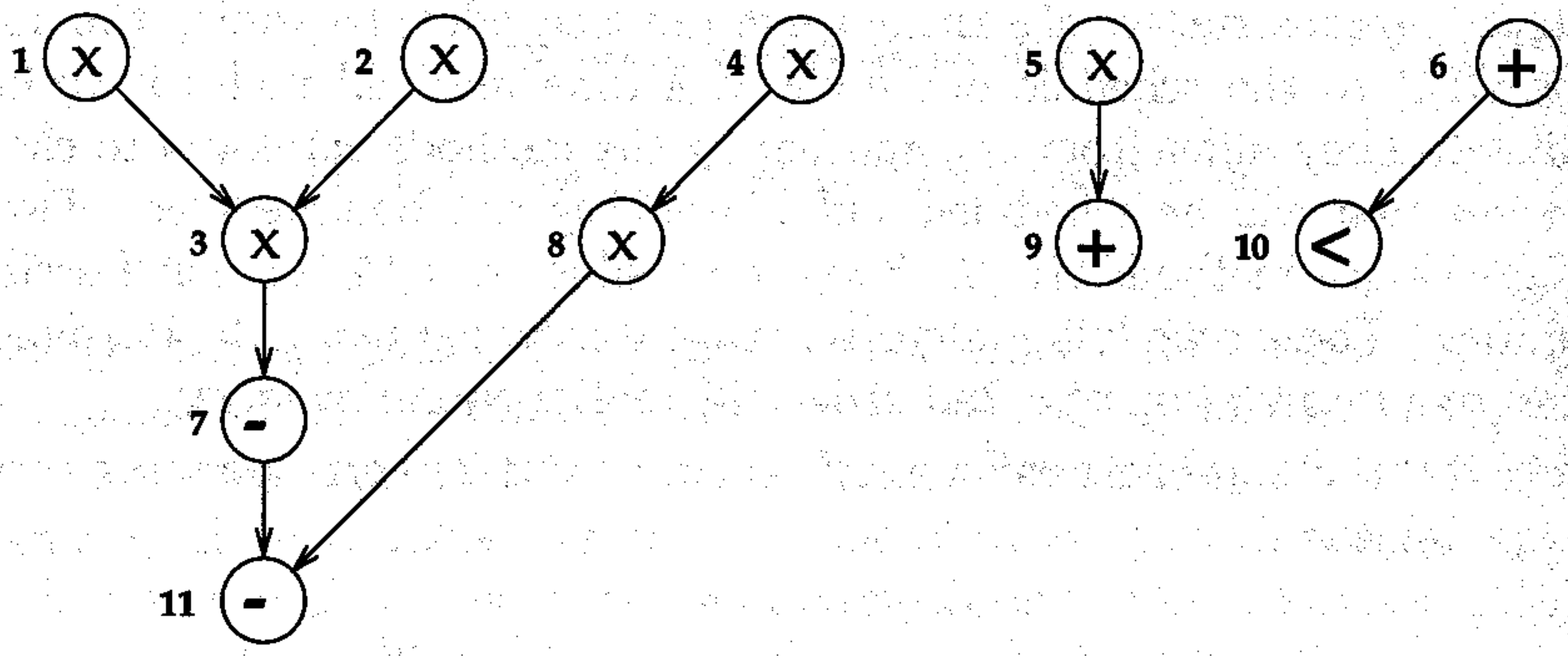


```
while ($x < a$) {
        $xl = x + dx;$
        $ul = u - (3 * x * u * dx) - (3 * y * dx);$
        $yl = y + u * dx;$
        $x = xl; y = yl; u = ul;$
}
```

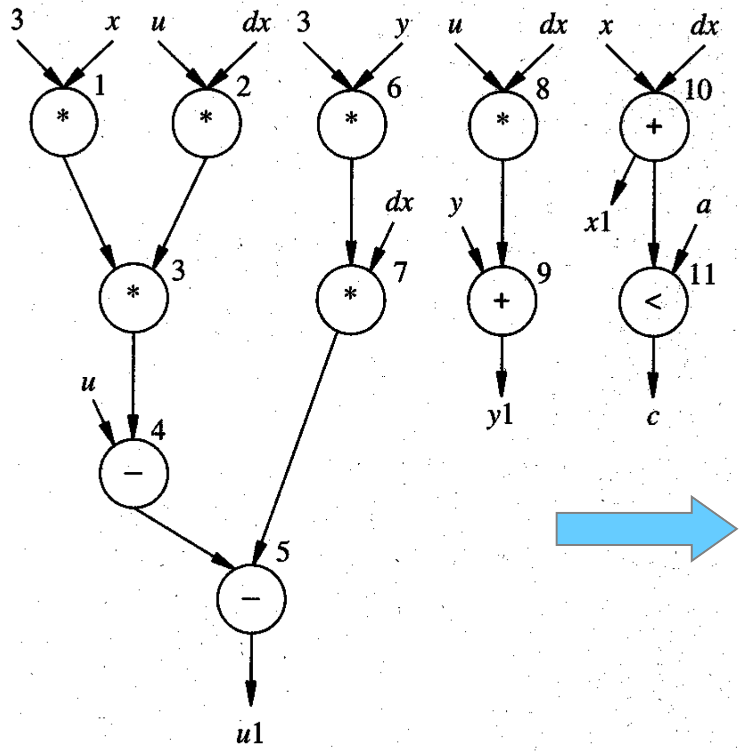
## Control Data Flow Graph (CDFG)

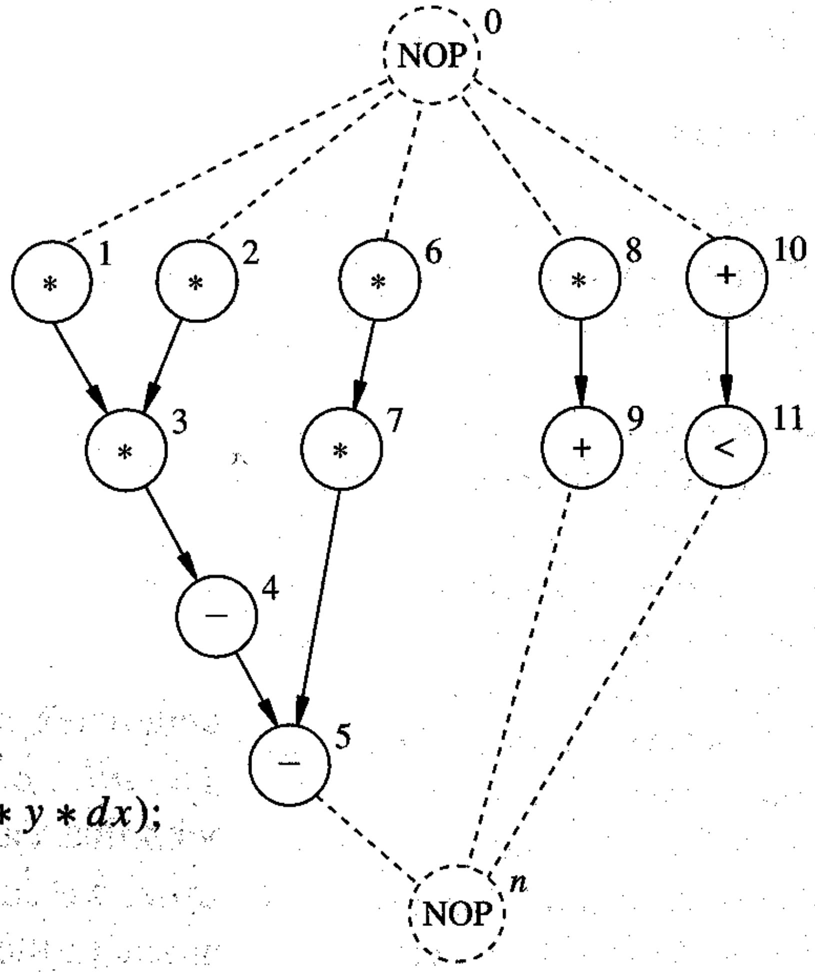


# Precedence Graph



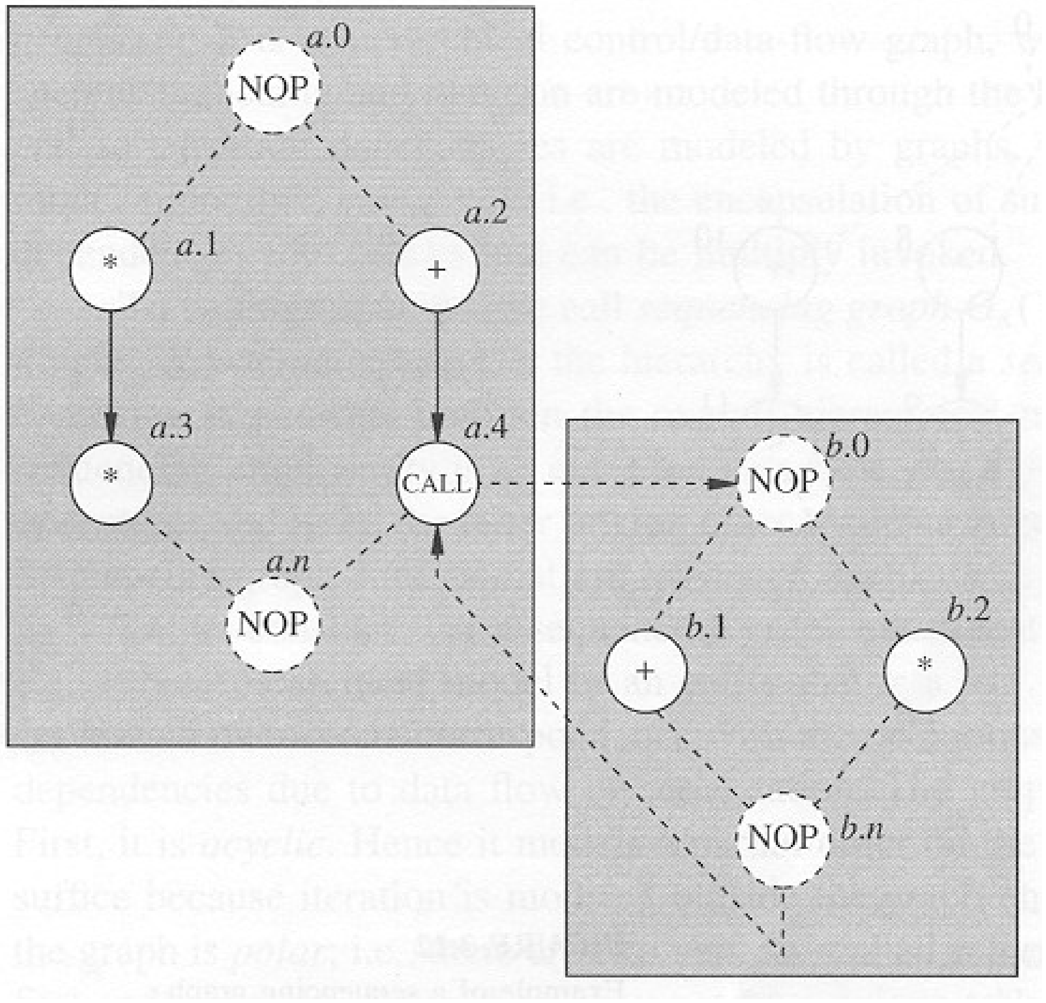
# Sequence Graph: Start and End Nodes



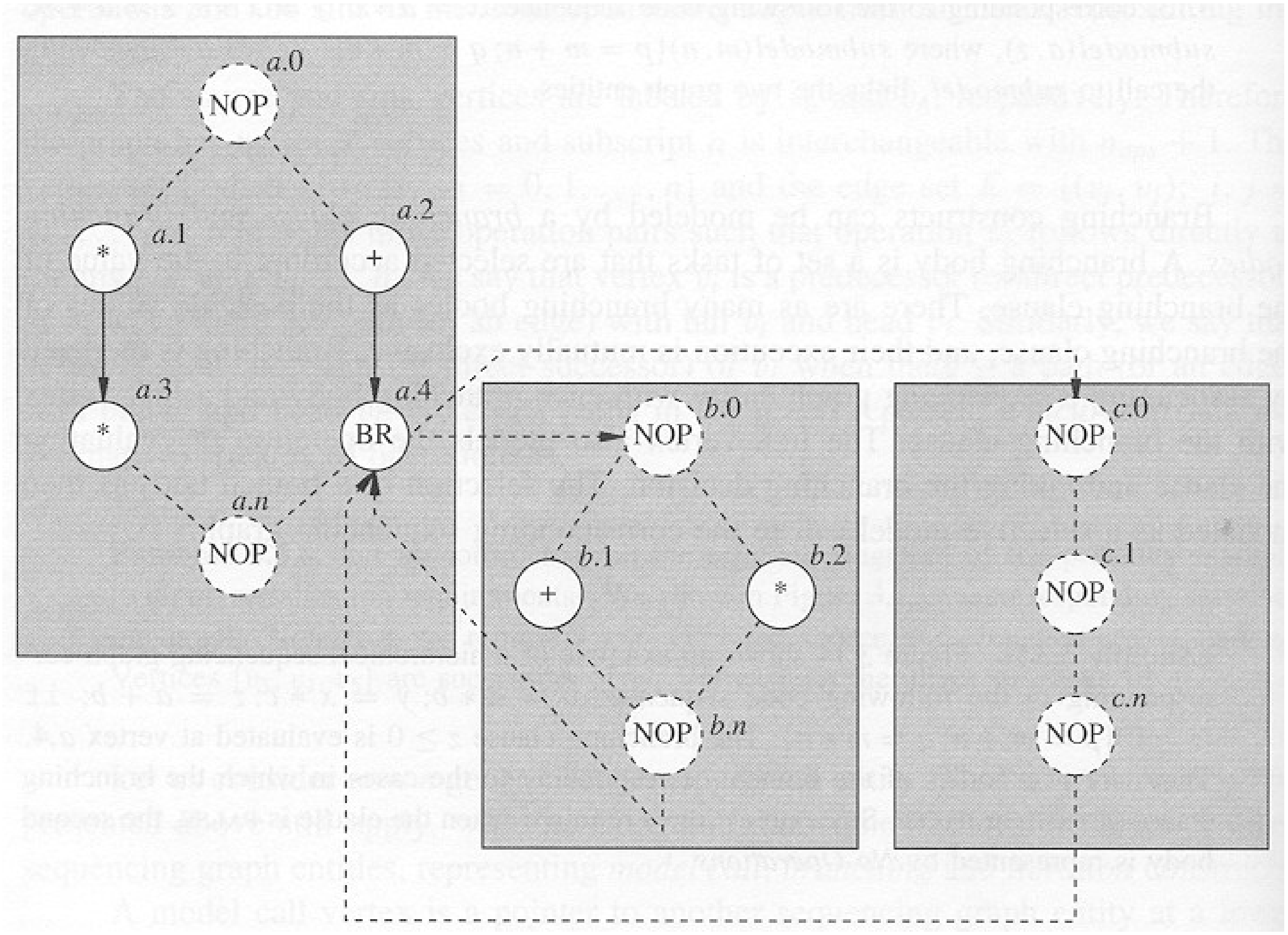


$$
\begin{aligned}
xl &= x + dx; \\
ul &= u - (3 * x * u * dx) - (3 * y * dx); \\
yl &= y + u * dx; \\
c &= xl < a;
\end{aligned}
$$

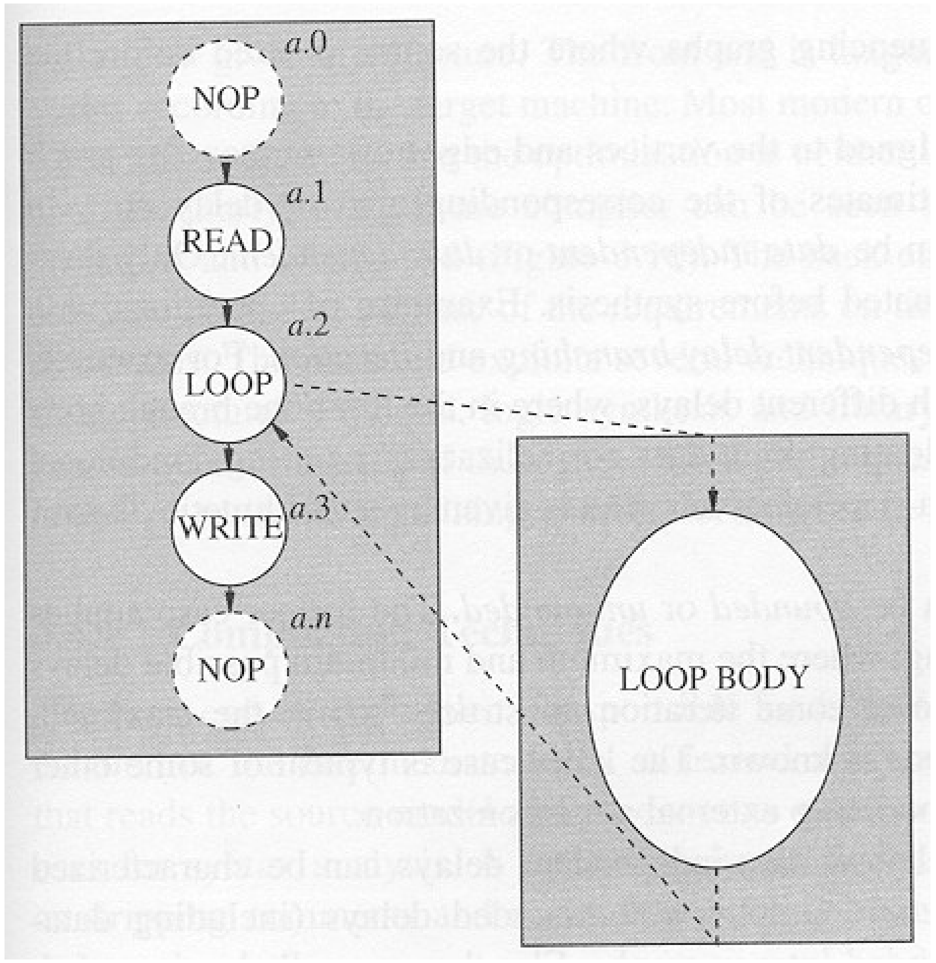
## Hierarchy in Sequence Graphs



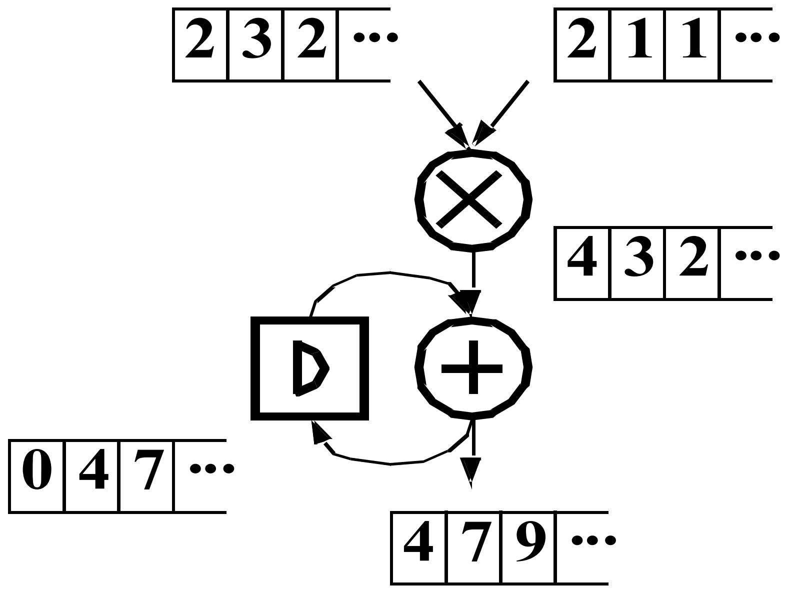
## Hierarchy in Sequence Graphs (contd.)

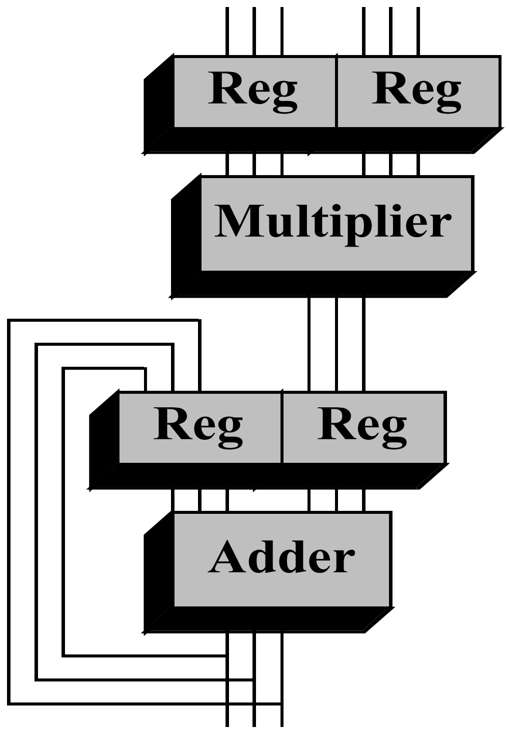


## Hierarchy in Sequence Graphs (contd.)

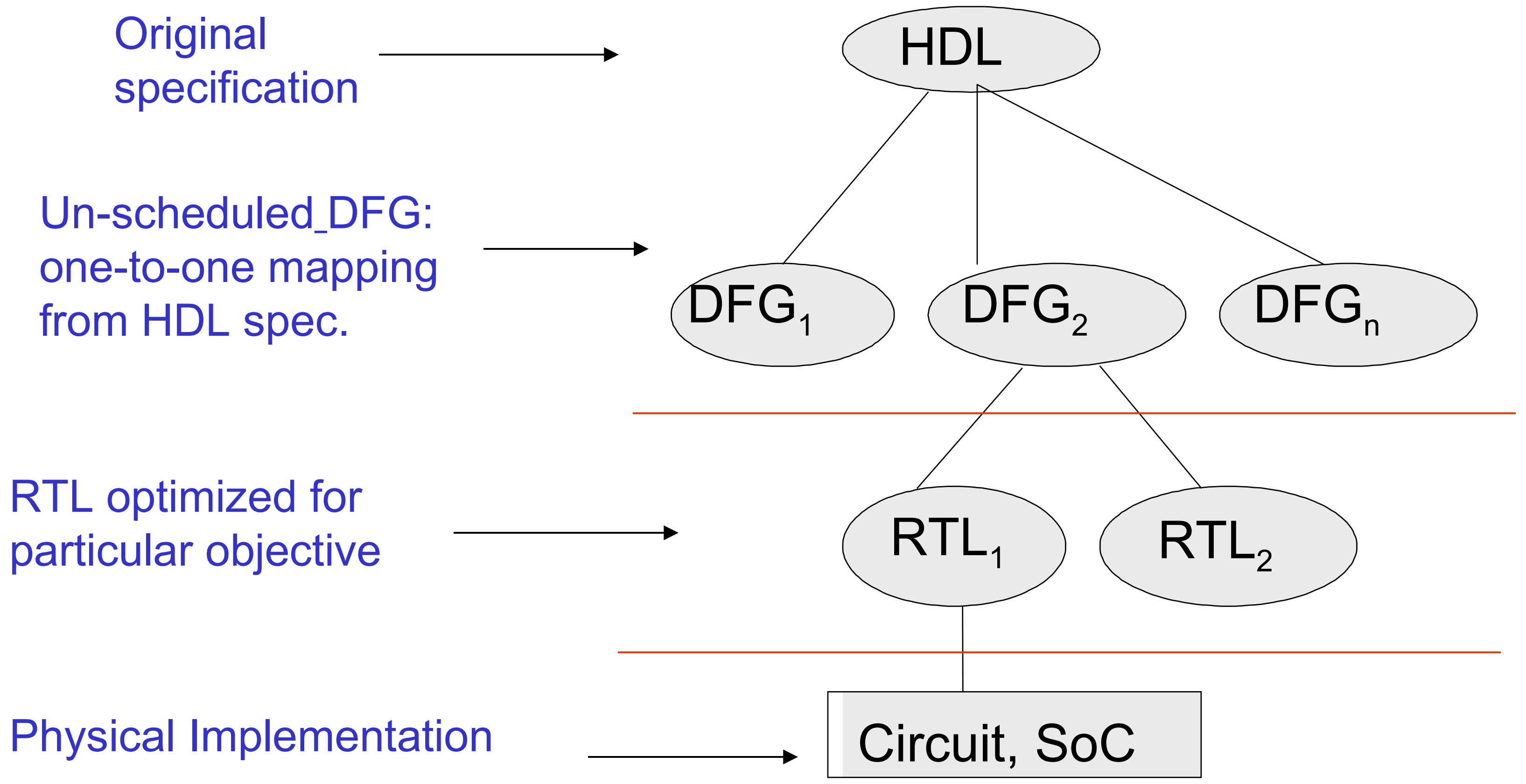


## Implementation





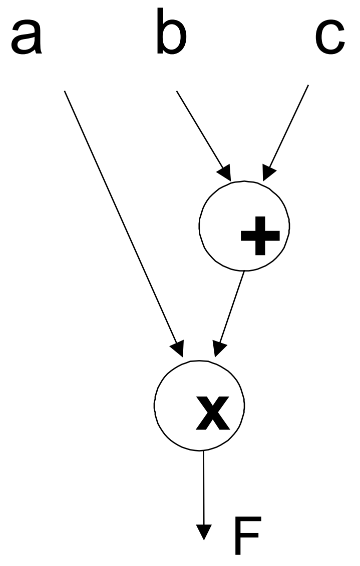
## Synthesis Flow – mapping & optimization steps

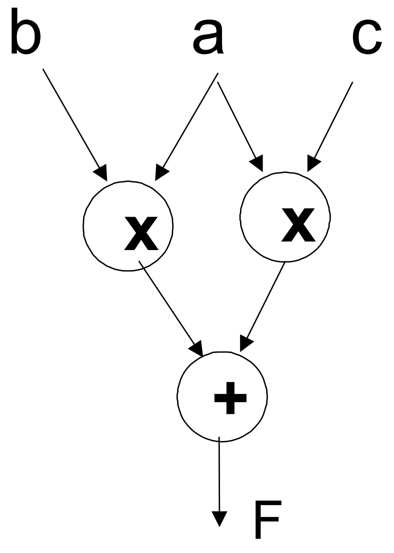


# Data Flow Graph (DFG)

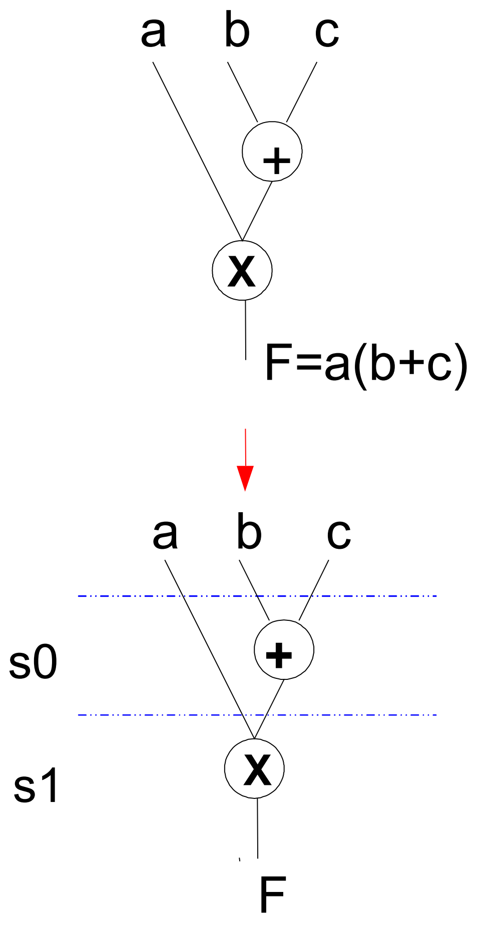
Precedence graph for computation *F*

$$F = a*(b + c) \xleftrightarrow{\textit{Transformation}} F = a*b + a*c$$

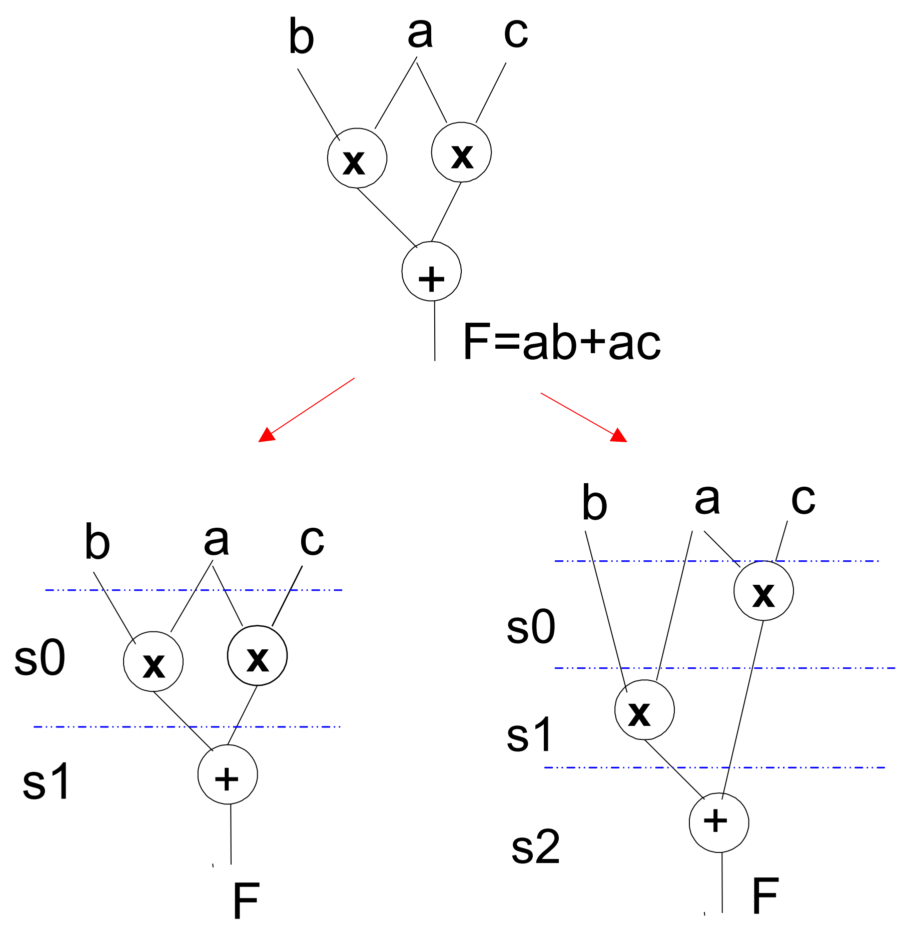




# DFG Scheduling



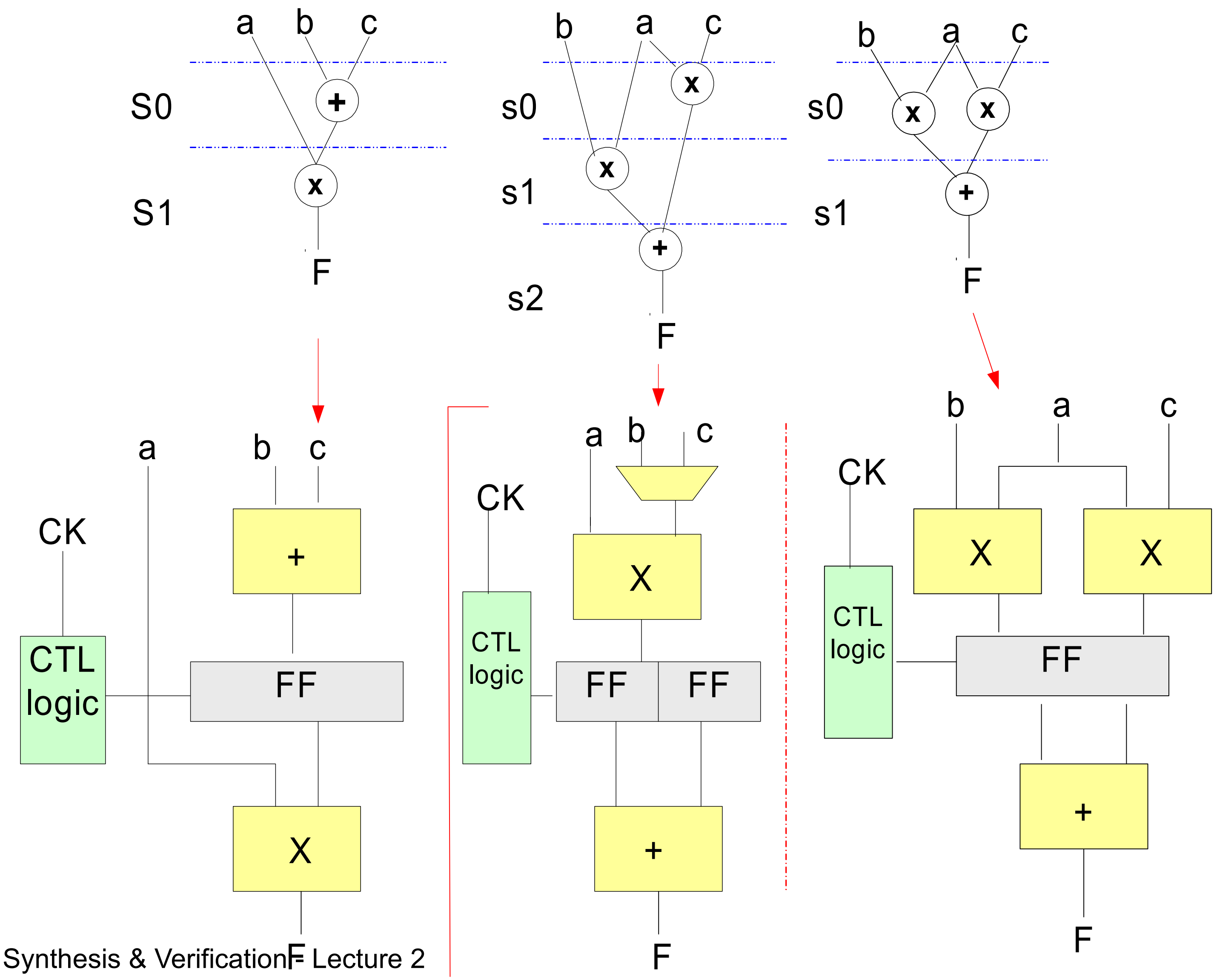
L=2, 1 Add, 1 Mult



L=2, 1 Add, 2 Mult

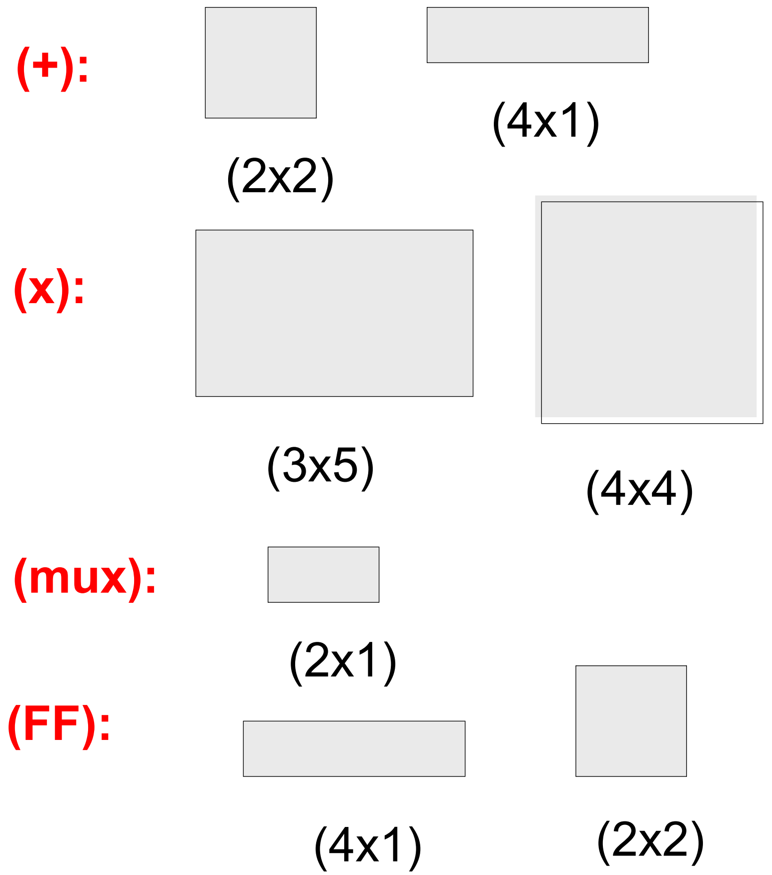
L=3, 1 Add, 1 Mult

# Scheduling affects RTL

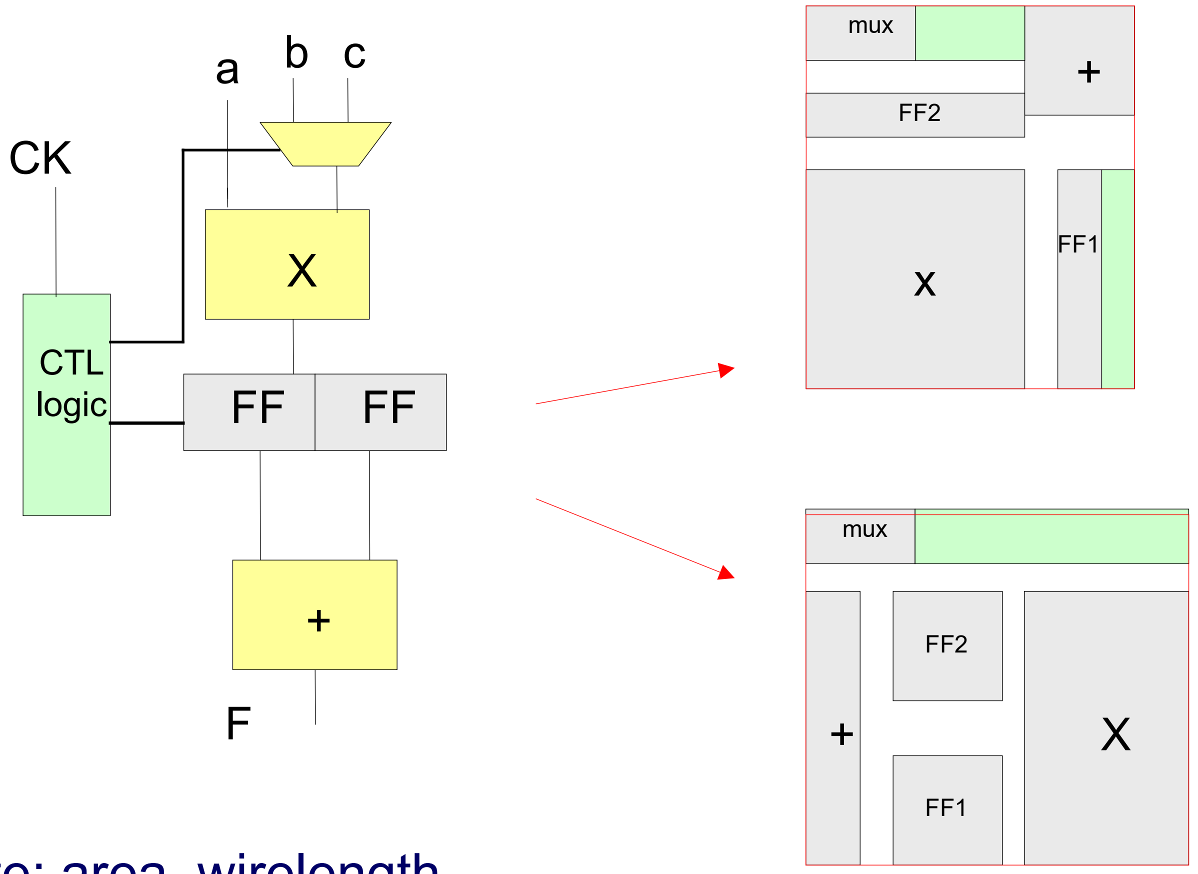


# Resources: Datapath Library

- Operators available from datapath library
  - pre-synthesized, pre-characterized
  - different layout, etc



# Floorplan Estimation



Evaluate: area, wirelength, congestion, interconnect, …