ECE 5775
High-Level Digital Design Automation
Fall 2022

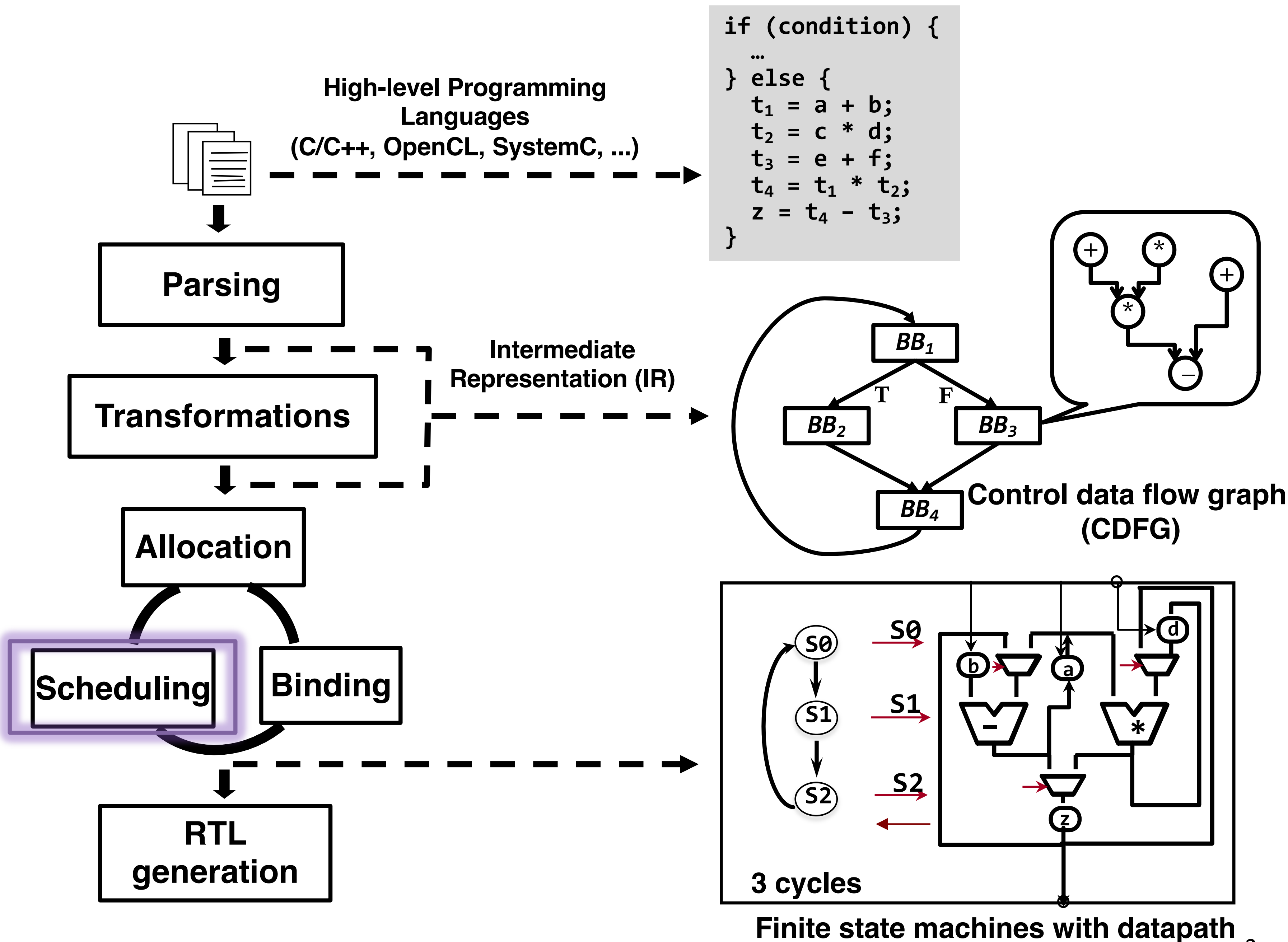
Scheduling

Cornell University

Agenda

- Unconstrained scheduling
  - ASAP and ALAP
- Constrained scheduling
  - Resource constrained scheduling (RCS)
  - Exact formulations with integer linear programming (ILP)

Recap: A Typical HLS Flow



Importance of Scheduling

- Scheduling is a central problem in HLS
  - Introduces clock boundaries to untimed (or partially timed) input specification

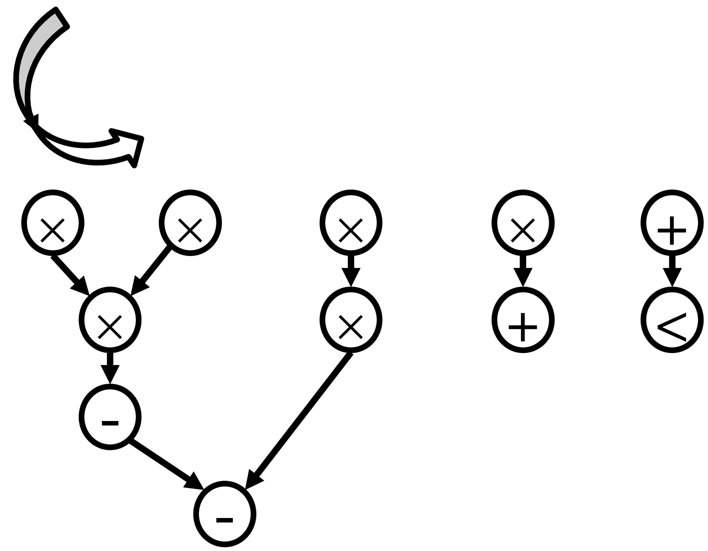
  - Has significant impact on the quality of results
    - Frequency
    - Latency
    - Throughput
    - Area
    - Power

    …

Scheduling Input

- Control data flow graph (CDFG)
  - Generated by a compiler front end from high-level description
  - Nodes
    - Operations (and pseudo operations)
  - Directed edges
    - Data edges, control edges, precedence edges

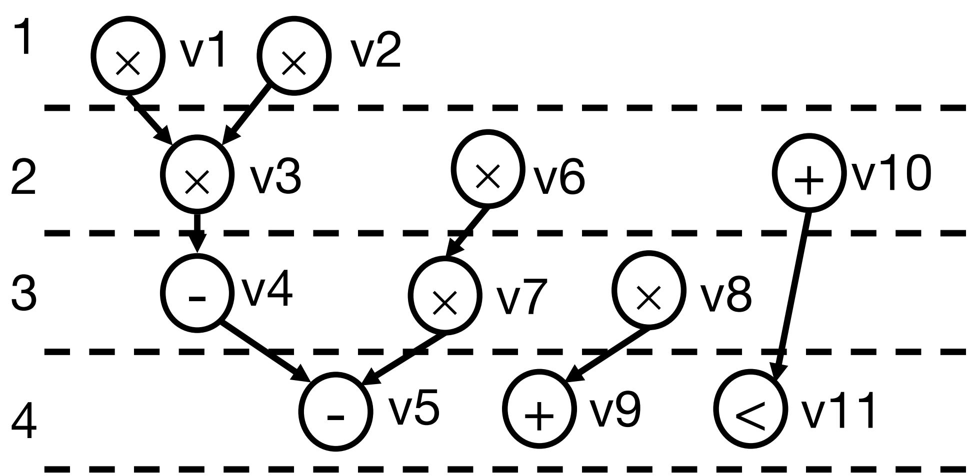
```
xl = x+dx;
ul = u-3*x*u*dx-3*y*dx
yl = y+u*dx
c = xl<a;
x = xl; u = ul; y = yl;
```



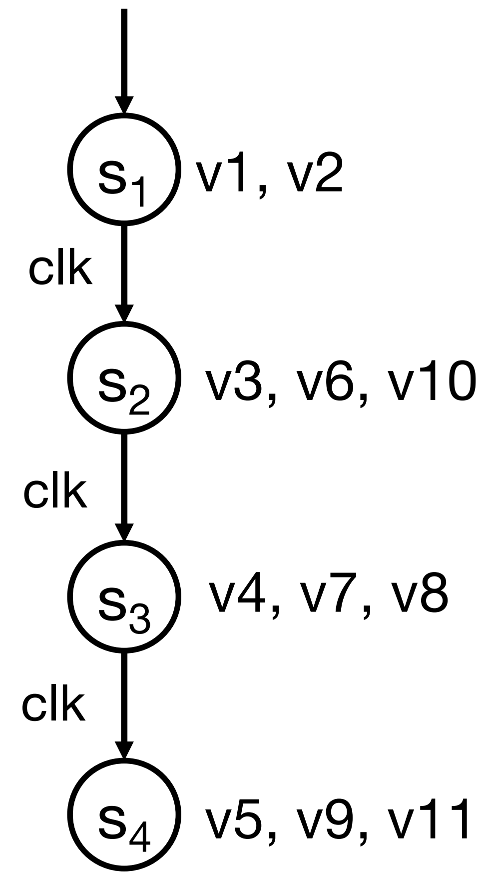
- Without control flow, the basic structure is a data flow graph (DFG)

Scheduling Output

- Scheduling: map operations to states
- Each clock cycle corresponds to a state in the FSM
  - Commonly referred to as control step (c-step)



DFG



FSM or State Transition Diagram (STG)

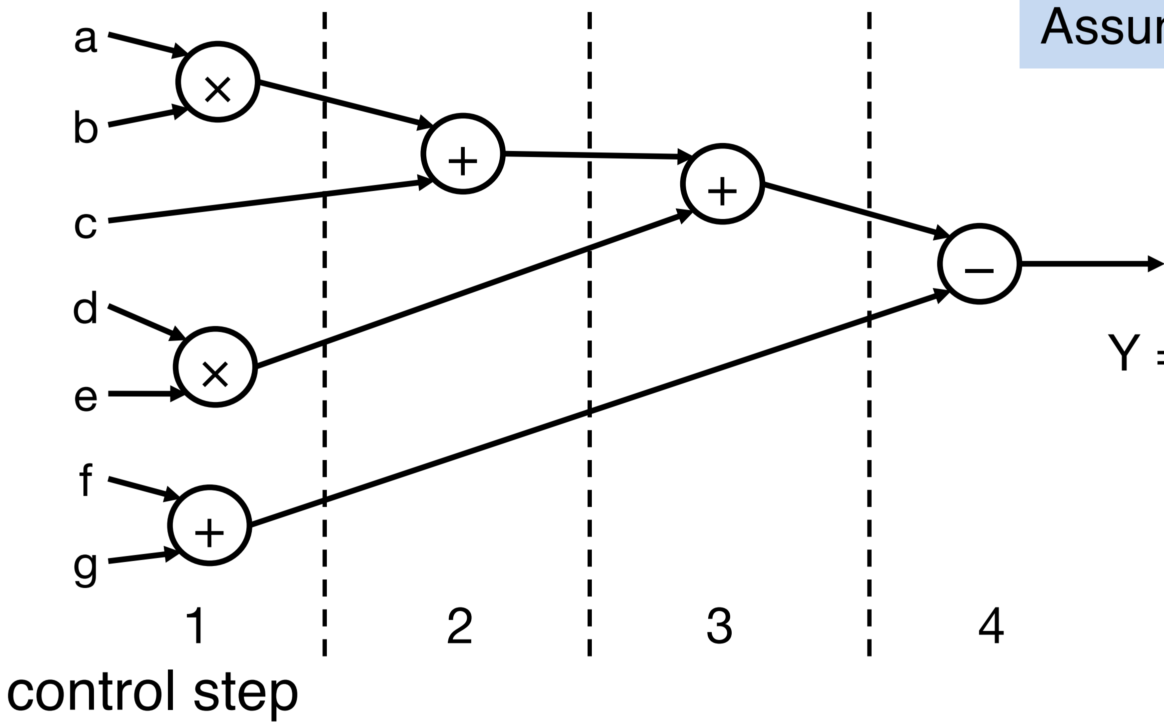
Unconstrained Scheduling

- Only consideration: dependence

- As soon as possible (ASAP)
  - Schedule an operation to the earliest possible step

- As late as possible (ALAP)
  - Schedule an operation to the earliest possible step, without increasing the total latency

ASAP Schedule



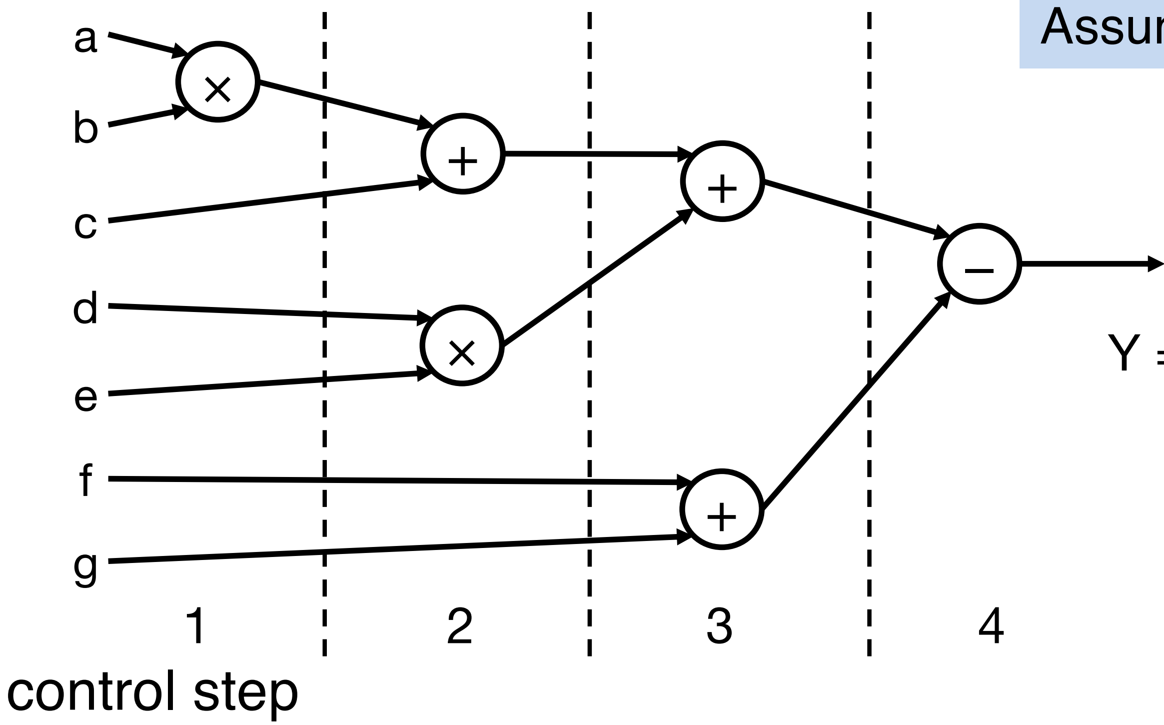
Assumption: No operation chaining

Y = ((a*b)+c)+(d*e)-(f+g)

```
ASAP(G(V, E)):
  V' = Topological_Sort(G)
  foreach v_i in V' :
    // Primary inputs (PIs) to first cycle
    if v_i ∈ PIs: t_i = 1
    // Assume no chaining & single-cycle operations
    else: t_i = max(t_j + 1); // (v_j , v_i ) ∈ E
```

The start time for each operation is the least one allowed by the dependencies

ALAP Schedule



Assumption: No operation chaining

Y = ((a*b)+c)+(d*e)-(f+g)

ALAP(G(V, E), L): // L is the latency bound
  V' = Reverse_Topological_Sort(G)
  foreach v_i in V' :
    // Primary outputs (POs) to last cycle
    if $v_i \in$ POs: $t_i = L$
    // Assume no chaining & single-cycle operations
    else: $t_i = \min(t_j) - 1$; // $(v_i, v_j) \in E$

The end time of each operation is the latest one allowed by the dependencies and the latency constraint

Constrained Scheduling

- Constrained scheduling
  - General case NP-hard
  - Resource-constrained scheduling (RCS)
    - Minimize latency given constraints on area or resources
  - Time-constrained scheduling (TCS)
    - Minimize resources subject to bound on latency

- Exact methods
  - Integer linear programming (ILP)
  - Hu's algorithm for a very restricted problem

- Heuristics
  - List scheduling
  - Force-directed scheduling
  - SDC-based scheduling

  …

Linear Programming

- **Linear programming (LP)** solves the problem of maximizing or minimizing a linear objective function subject to linear constraints
  - Efficiently solvable both in theory and in practice

- **Integer linear programming (ILP)**: in addition to linear constraints and objective, the values for the variables must be integer
  - NP-Hard in general (A special case, 0-1 ILP)
  - Modern ILP solvers can handle problems with nontrivial size

- Enormous number of problems can be expressed in LP or ILP

Canonical Form of ILP

maximize $c_1x_1+c_2x_2+\ldots+c_nx_n$ // objective function
subject to // linear constraints

$a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n \le b_1$
$a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n \le b_2$
....
$a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n \le b_m$
$x_i \ge 0$
$x_i \in \mathbf{Z}$

<u>Vector form</u>

maximize c^Tx // $c = (c_1, c_2, \ldots, c_n)$
subject to

$Ax \le b$ // A is a *mxn* matrix; $b = (b_1, b_2, \ldots, b_n)$
$x \ge 0$
$x_i \in \mathbf{Z}$

Example: Course Selection Problem

- A student is about to finalize course selection for the coming semester, given the following information:
  - Minimum credits per semester: 8

	Schedule	Credits	Est. workload (per week)
1. Metaverse	MW 2:00-3:30pm	3	8 hrs
2. How to start a start-up	TT 2:00-3:00pm	2	4 hrs
3. Linear programming	MW 9:00-11:00am	4	10 hrs
4. Analog circuits	TT 1:00-3:00pm	4	12 hrs

Question: Which courses to take to minimize the amount of work?

ILP Formulation for Course Selection

	Time	CRs	Work
1. Metaverse	MW 2-3:30pm	3	8 hrs
2. Start-up	TT 2-3pm	2	4 hrs
3. Linear prog.	MW 9-11am	4	10 hrs
4. Analog	TT 1-3pm	4	12 hrs

- Define decision variables (i = 1, 2, 3, 4):

$$x_i \begin{cases} 1 & \text{if course i is taken} \\ 0 & \text{otherwise} \end{cases}$$

- The total expected work hours: $8x_1+4x_2+10x_3+12x_4$
- The total credits taken: $3x_1+2x_2+4x_3+4x_4$
- Account for the schedule conflict: $x_2+x_4 \leq 1$

- Complete ILP formulation (in canonical form):

minimize $8x_1+4x_2+10x_3+12x_4$

s.t. $3x_1+2x_2+4x_3+4x_4 \geq 8$

$x_2+x_4 \leq 1$

$x_i \in \{0,1\}$

Resource Constrained Scheduling (RCS)

- When functional units are limited
  - Each functional unit can only perform one operation at each clock cycle
    - e.g., if there are only K adders, no more than K additions can be executed in the same c-step
- A typical resource-constrained scheduling problem for DFG
  - Given the number of functional units of each type, minimize latency (in cycles)
  - *NP-hard*

ILP Formulation of RCS

- Use binary decision variables
  - $x_{i,k} = 1$ if operation i starts at step k, otherwise = 0
    - $1 \leq i \leq N, 1 \leq k \leq L$
    - N is the **total number of operations**
    - L is the given **upper bound on latency**

$$t_i = \sum_{k=1}^{L} k \cdot x_{i,k}$$

t_i indicates the **start time** of operation i

ILP Formulation of RCS: Constraints (1)

- Unique start times

$$\forall i \in [1, N] : \sum_k x_{i,k} = 1$$

- Dependence must be satisfied (assuming no chaining)

$$\forall (i, j) \in E : t_j \geq t_i + d_i \rightarrow \sum_k k \cdot x_{j,k} \geq \sum_k k \cdot x_{i,k} + d_i$$

Operation j must not start before i completes if j depends on i

d_i : latency of operation i

- $d_i = 1$ means a single-cycle operation
- $d_i > 1$ indicates a multi-cycle operation

Start Time vs. Active Time(s)

- When $d_i = 1$, the following are the same
  - Does operation i start at step k?
  - Is operation i actively running at step k?
  - Same equality check: if x_{ik} is 1

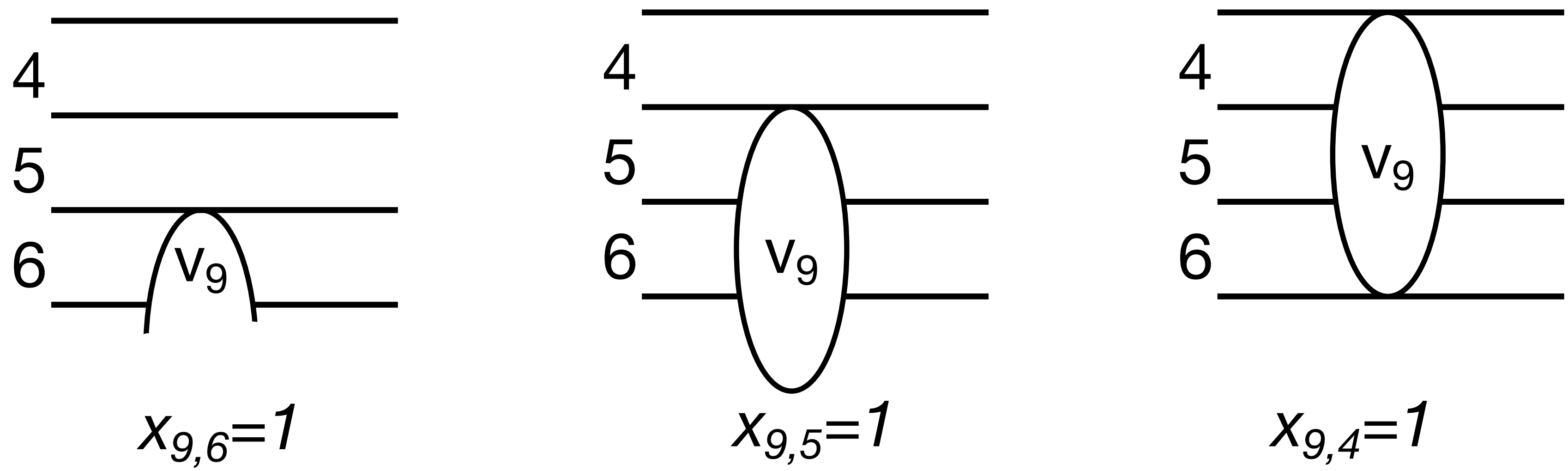
- But if $d_i > 1$, the two questions are different
  - Does operation i **start** at step k? Check if x_{ik} is 1
  - Is operation i **active** at step k? Check if the following holds

$$\sum_{l=k-d_i+1}^{k} x_{i,l} \stackrel{?}{=} 1$$

Is Operation i Still Active at Step k ?

- Is operation 9 active (running) at step 6? assuming d_9 = 3

if and only if $\boldsymbol{x_{9,6}} + \boldsymbol{x_{9,5}} + \boldsymbol{x_{9,4}}$ equals 1 $\Rightarrow \sum_{l=6-3+1}^{6} x_{9,l} \stackrel{?}{=} 1$



- Notes
  - Only one (if any) of the above three cases can happen
  - To meet resource constraints, we must check the same equality for ALL steps, and ALL operations of that type

ILP Formulation of RCS: Constraints (2)

- Physical resource limits
  - R denotes the total number of different resource types (RT)
  - $RT(i) \in [1, R]$ is the resource type of operation i
  - a_r is the number of physically available resources of type r

At any step k, the total number of active operations of the same type r must not exceed a_r, the number of physically available resources for type r

$$\forall k \in [1, L], \forall r \in [1, R] : \boxed{\sum_{i:RT(i)=r}} \boxed{\sum_{l=k-d_i+1}^{k} x_{i,l}} \leq a_r$$

Operations whose resource type is r

If operation i is active at step k

ILP Formulation of RCS: Putting It Together

- Unique start times

$$\forall i \in [1, N] : \sum_{k} x_{i,k} = 1$$

- Dependence must be satisfied (assuming no chaining)

$$\forall (i, j) \in E : t_j \geq t_i + d_i \rightarrow \sum_{k} k \cdot x_{j,k} \geq \sum_{k} k \cdot x_{i,k} + d_i$$

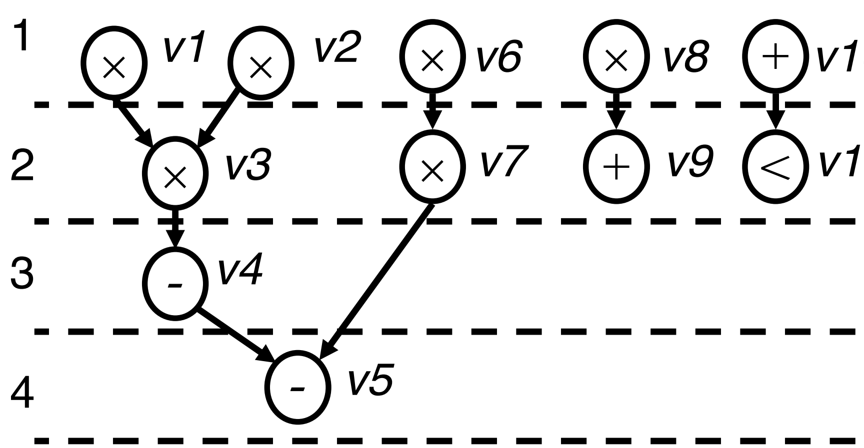
- Physical resource limits

$$\forall k \in [1, L], \forall r \in [1, R] : \sum_{i:RT(i)=r} \sum_{l=k-d_i+1}^{k} x_{i,l} \leq a_r$$

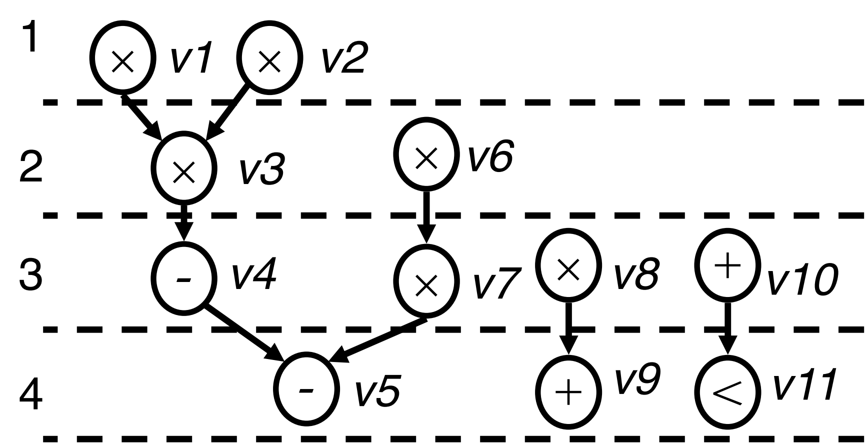
Use of ASAP and ALAP

- In general, the following helps the ILP solver run faster
  - Minimize # of variables and constraints
  - Simplify the constraints

- We can write the ILP without ASAP/ALAP, but using ASAP and ALAP can simplify the constraints



ASAP schedule



ALAP schedule

ILP Formulation: Unique Start Time Constraints

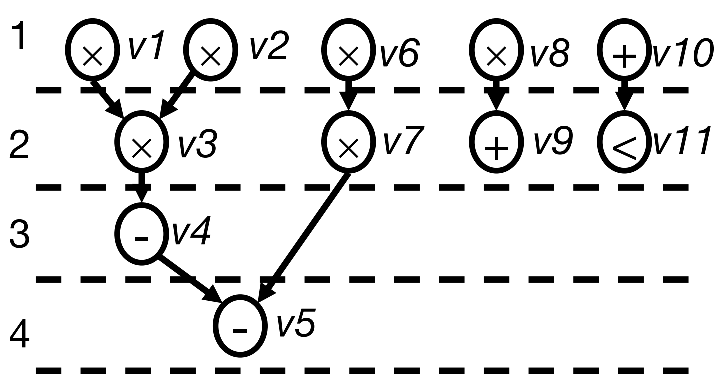
$$\boxed{\begin{array}{l} x_{il} = 0 \quad for \quad l < t_i^S \quad and \quad l > t_i^L \\ (t_i^S = ASAP(v_i), t_i^L = ALAP(v_i)) \end{array}}$$

- Without using ASAP and ALAP

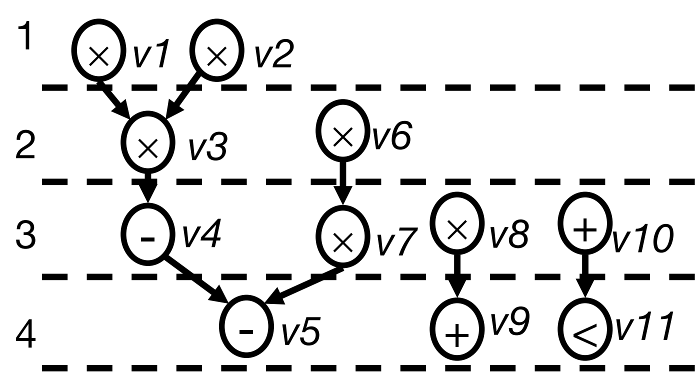
$$\begin{aligned} x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} &= 1 \\ x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} &= 1 \\ &\dots \\ x_{6,1} + x_{6,2} + x_{6,3} + x_{6,4} &= 1 \\ &\dots \\ x_{9,1} + x_{9,2} + x_{9,3} + x_{9,4} &= 1 \\ &\dots \end{aligned}$$

- Using ASAP and ALAP

$$\begin{aligned} x_{1,1} &= 1 \\ x_{2,1} &= 1 \\ &\dots \\ x_{6,1} + x_{6,2} &= 1 \\ &\dots \\ x_{9,2} + x_{9,3} + x_{9,4} &= 1 \\ &\dots \end{aligned}$$



ASAP schedule



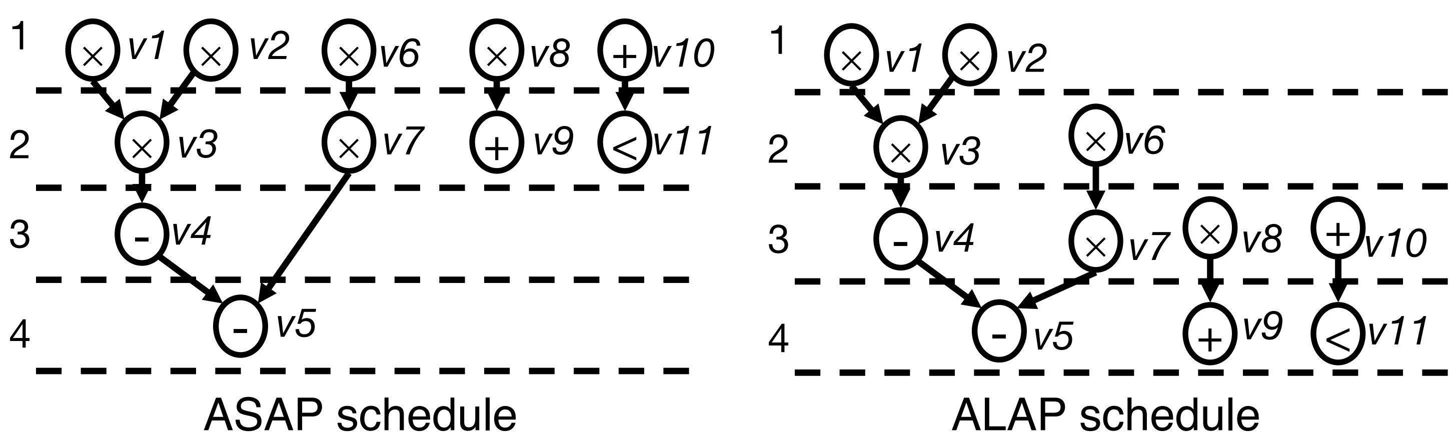
ALAP schedule

assume L=4

ILP Formulation: Dependence Constraints

- Using ASAP and ALAP, the non-trivial inequalities are: (assuming no chaining and single-cycle ops)

$$
\begin{aligned}
2x_{7,2} + 3x_{7,3} - x_{6,1} - 2x_{6,2} &\geq 1 \\
4x_{5,4} - 2x_{7,2} - 3x_{7,3} &\geq 1 \\
2x_{9,2} + 3x_{9,3} + 4x_{9,4} - x_{8,1} - 2x_{8,2} - 3x_{8,3} &\geq 1 \\
2x_{11,2} + 3x_{11,3} + 4x_{11,4} - x_{10,1} - 2x_{10,2} - 3x_{10,3} &\geq 1
\end{aligned}
$$



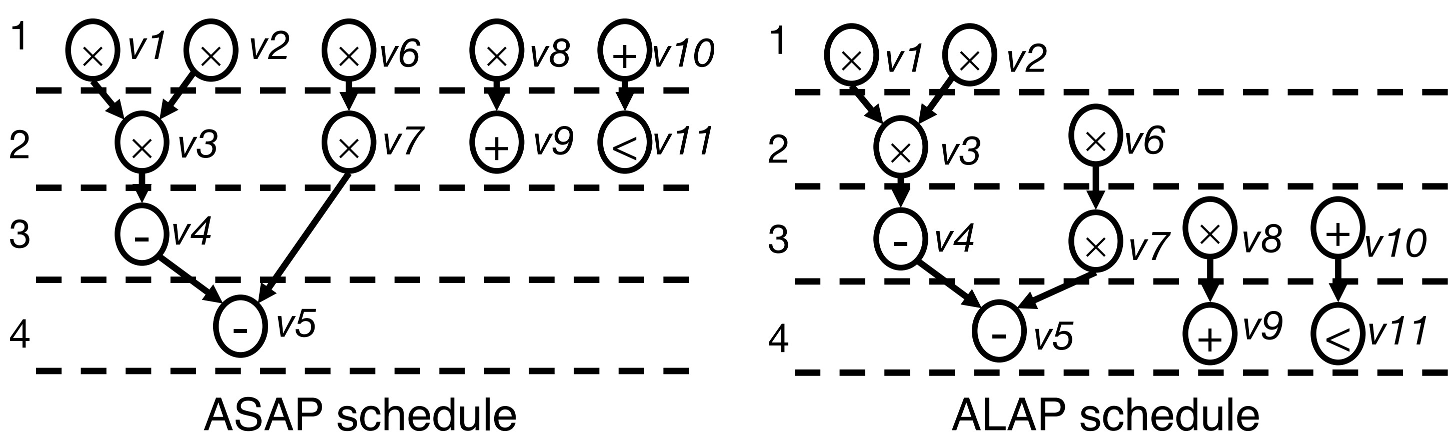
assume L=4

ILP Formulation: Resource Constraints

- Resource constraints (assuming 2 multipliers and 1 ALU)

$$x_{1,1} + x_{2,1} + x_{6,1} + x_{8,1} \leq 2$$
$$x_{3,2} + x_{6,2} + x_{7,2} + x_{8,2} \leq 2$$
$$x_{7,3} + x_{8,3} \leq 2$$

$$x_{10,1} \leq 1$$
$$x_{9,2} + x_{10,2} + x_{11,2} \leq 1$$
$$x_{4,3} + x_{9,3} + x_{10,3} + x_{11,3} \leq 1$$
$$x_{5,4} + x_{9,4} + x_{11,4} \leq 1$$



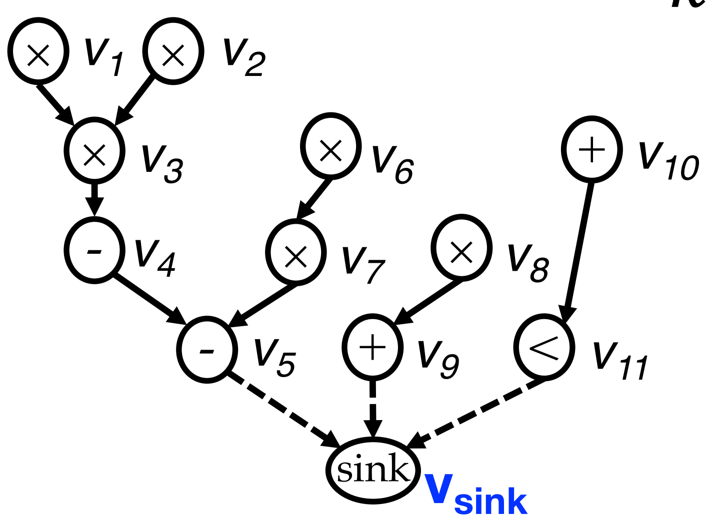
assume L=4

ILP Formulation of RCS: Objective Function

- Objective: min $\mathbf{c}^T\mathbf{t}$
  - $\mathbf{t}$ = start times vector, $\mathbf{c}$ = cost weight (e.g., [0 ... 0 1])

- For simplicity, we introduce a pseudo node to serve as a unique sink (v_{sink}) of the DFG
  - This sink depends on all the original primary output nodes
  - To minimize the overall latency, we simply minimize the start time of the sink node

$$min\ t_{sink} \quad \Rightarrow \quad min \sum_{k=1}^{L} k \cdot x_{sink,k}$$



ILP Summary

- Pros: versatile modeling ability
  - Can be extended to handle almost every design aspects
    - Resource allocation
    - Module selection
    - Area, power, etc.

- Cons: computationally expensive
  - #variables = O(#nodes * #c-steps)
  - 0-1 assignment variables: need extensive search to find optimal solution

Next Lecture

- More scheduling algorithms

Acknowledgements

- These slides contain/adapt materials developed by
  - Ryan Kastner (UCSD)