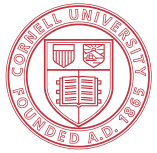
ECE 5997
Hardware Accelerator Design & Automation
Fall 2021

Scheduling



Cornell University

CSL

Announcements

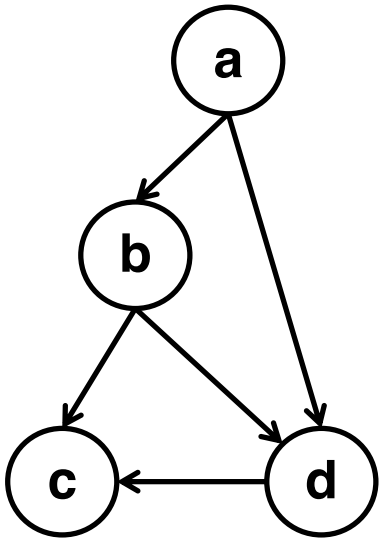
- Lab 2 due next Wed (Nov 10)

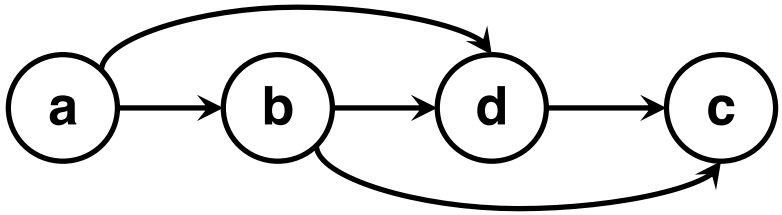
Outline

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

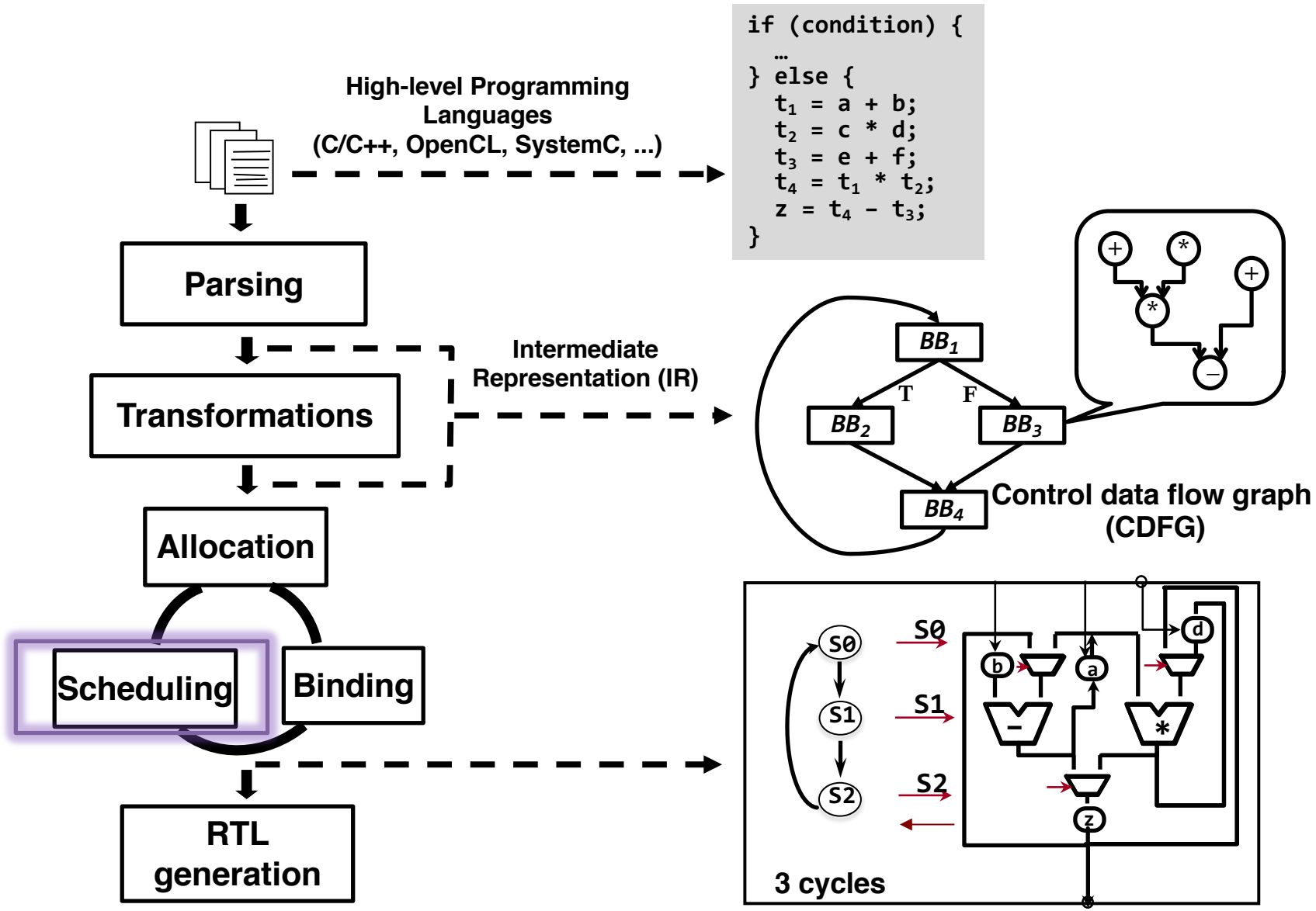
Recap: Topological Sort

- A **topological sort** (or order) of a directed graph is an ordering of nodes where all edges go from an earlier vertex (left) to a later vertex (right)
  - Feasible if and only if the subject graph is a DAG





Recap: A Typical HLS Flow

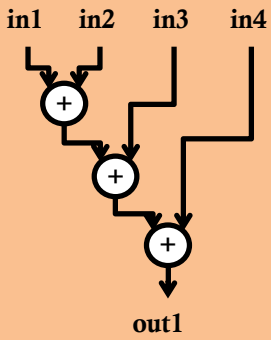
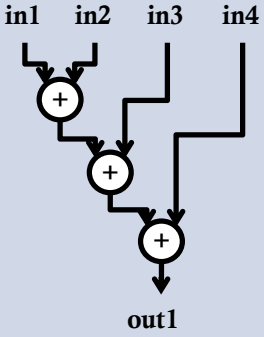
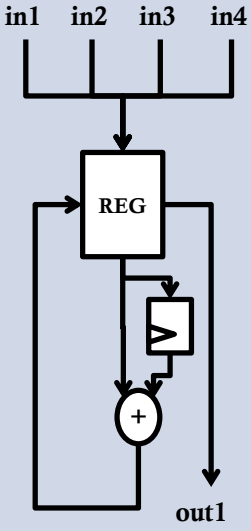
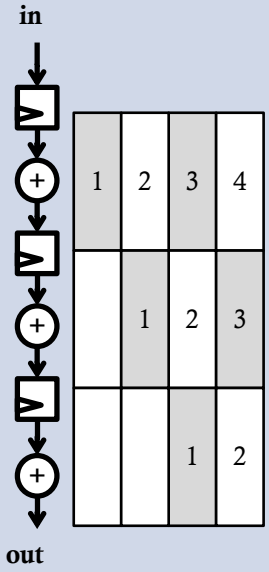


Importance of Scheduling

- 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

    …

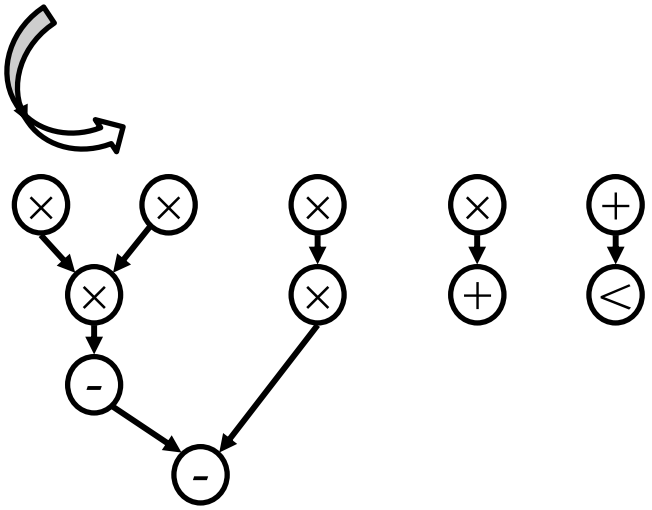
Scheduling: Untimed to Timed

	Latency	Area	Throughput
 **Control-Data Flow Graph**			
$out1 = f(in1, in2, in3, in4)$	$t_{clk} \approx 3 * d_{add}$ $T_1 = 1 / t_{clk}$ $A_1 = 3 * A_{add}$	$t_{clk} \approx d_{add} + d_{setup}$ $T_2 = 1 / (3 * t_{clk})$ $A_2 = A_{add} + 2 * A_{reg}$	$t_{clk} \approx d_{add} + d_{setup}$ $T_3 = 1 / t_{clk}$ $A_3 = 3 * A_{add} + 6 * A_{reg}$
Untimed	**Combinational**	**Sequential**	**Pipelined**

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

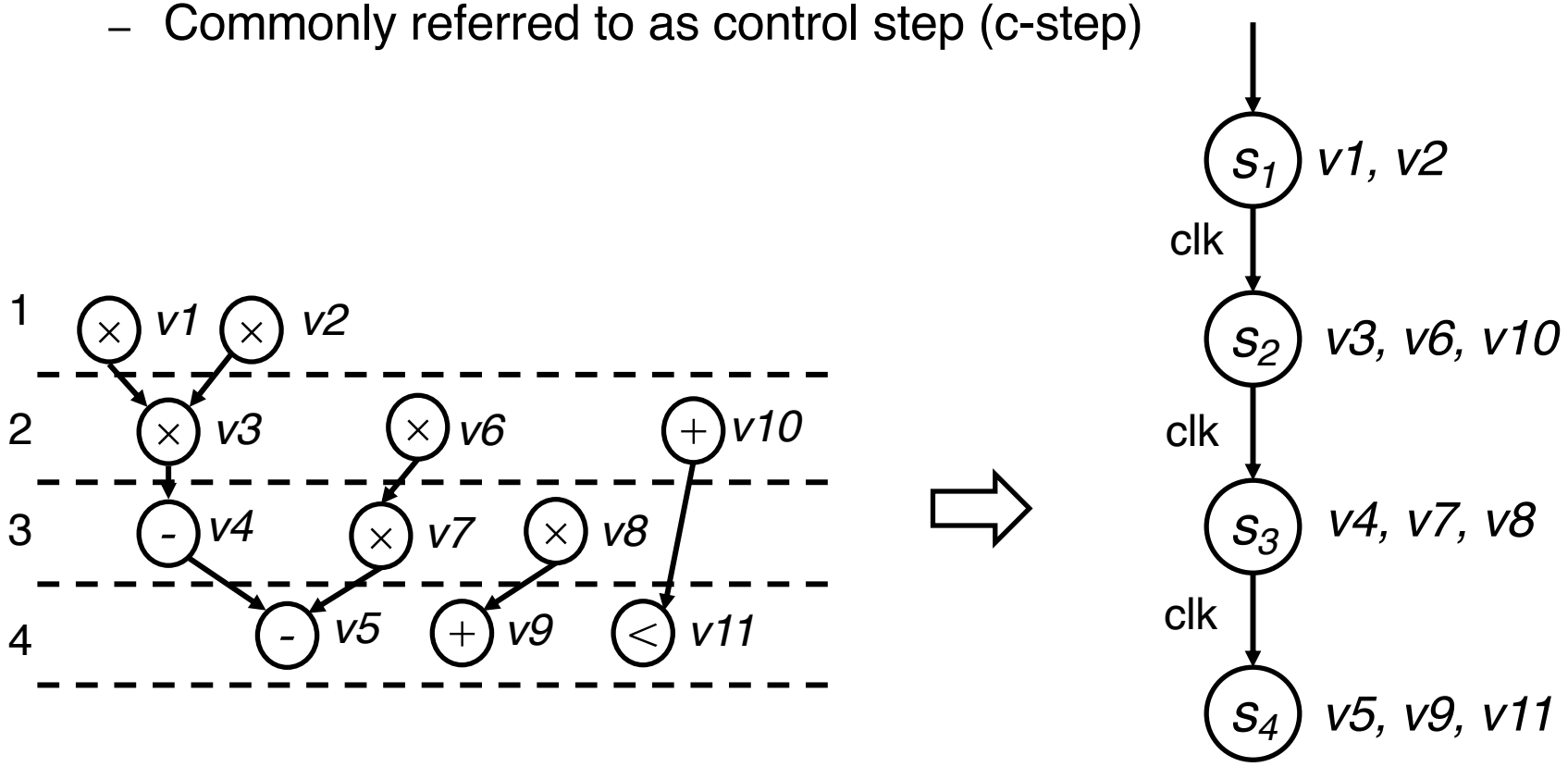
```
xl = x+dx;
ul = u-3*x*u*dx-3*y*dx
yl = y+u*dx
c = xl<a;
x = xl; u = ul; y = yl;
```



- 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)



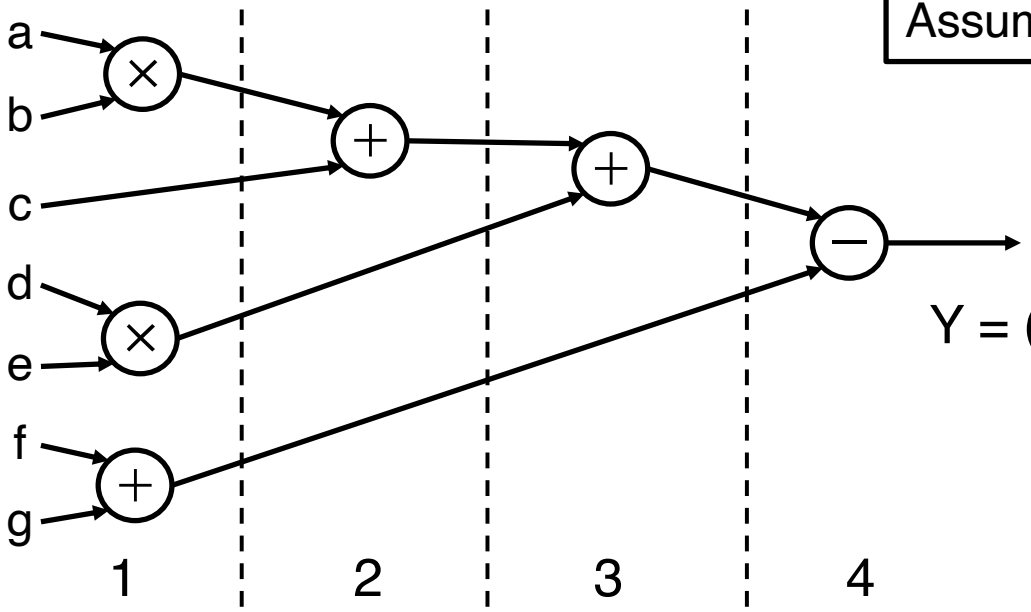
DFG

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



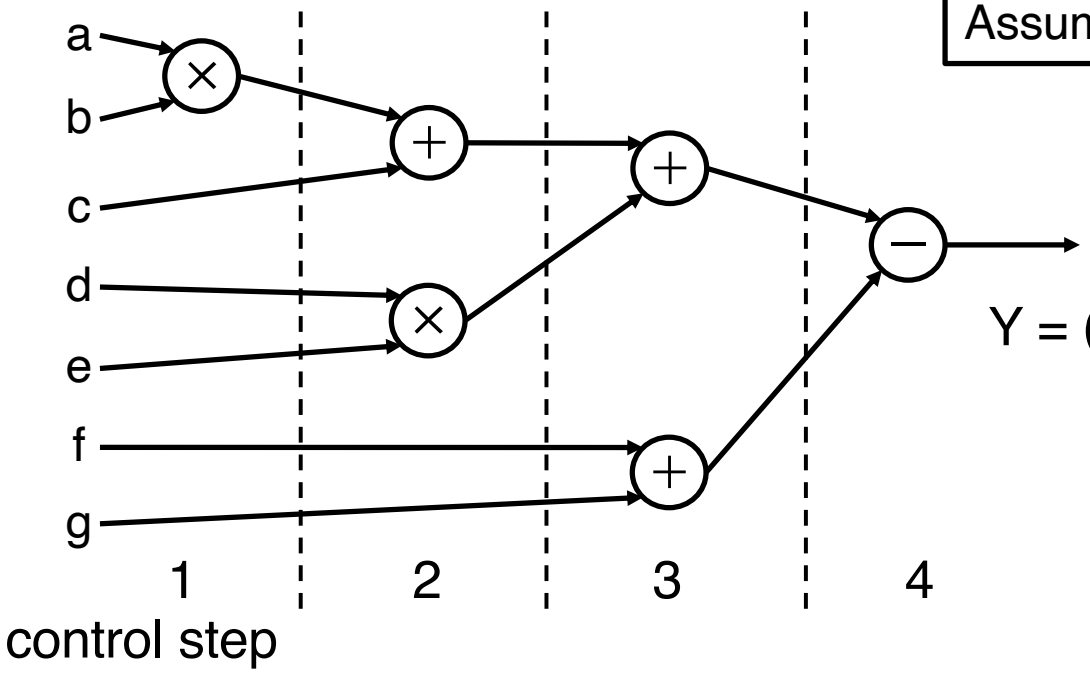
Assumption: Node delay = Cycle time

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 $v_i \in$ PIs: $t_i = 1$
    // Assume no chaining & single-cycle operations
    else: $t_i = \max(t_j + 1)$; // $(v_j, v_i) \in E$

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

ALAP Schedule



Assumption: Node delay = Cycle time

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 have to 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

$a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n \leq b_1$

$a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n \leq b_2$

....

$a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n \leq b_m$

$x_i \geq 0$

$x_i \in \mathbf{Z}$

Vector form

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

$Ax \leq b$ // A is a *mxn* matrix; $b = (b_1, b_2, \ldots, b_n)$

$x \geq 0$

$x_i \in \mathbf{Z}$

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. Big data analytics	MW 2:00-3:30pm	3	8 hrs
2. How to build 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

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

$$x_i = \begin{cases} 1 & \text{if course } i \text{ is taken} \\ 0 & \text{if not} \end{cases}$$

	Time	CRs	Work
1. Big data	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

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

$$t_i = \sum_{k=1}^{L} k x_{ik}$$

t_i indicates the start time of operation i

ILP Formulation of RCS: Constraints (1)

- Linear constraints:
  - Unique start times: $\sum_k x_{ik} = 1, \quad i = 1, 2, \ldots, N$
  - Dependence must be satisfied (no chaining)

$$t_j \geq t_i + d_i \quad : \forall (v_i, v_j) \in E \Rightarrow \sum_k k\ x_{jk} \geq \sum_k k\ x_{ik} + d_i$$

v_j must not start before v_i completes since v_j depends on v_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. Time(s) of Execution

- When $d_i = 1$, the following questions are the same:
  - Does operation i start at step k
  - Is operation i running at step k

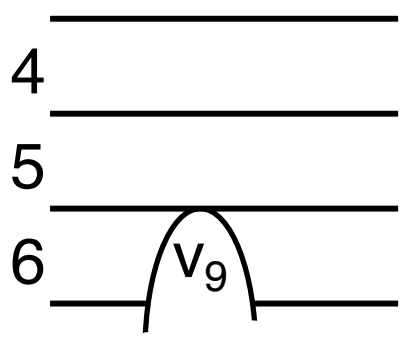
- But if $d_i > 1$, the two questions should be formulated as:
  - Does operation i **start** at step k
    - Check if x_{ik} is 1
  - Is operation i **running** at step k
    - Check if the following hold

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

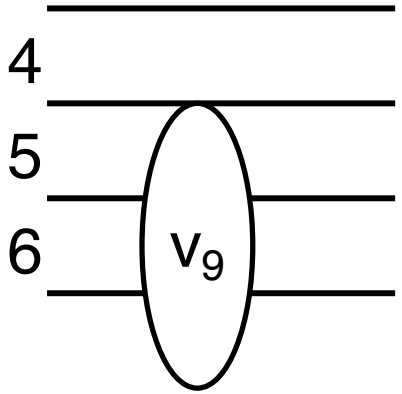
Operation v_i Still Running at Step k ?

- Is v_9 (d_9 = 3) running at step 6?

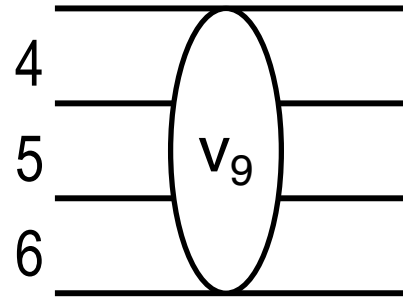
  If and only if $x_{9,6} + x_{9,5} + x_{9,4}$ equals 1



$x_{9,6}=1$



$x_{9,5}=1$



$x_{9,4}=1$

- Note:
  - Only one (if any) of the above three cases can happen
  - To meet resource constraints, we have to ask the same question for ALL steps, and ALL operations of that type

ILP Formulation of RCS: Constraints (2)

- Linear constraints:
  - Unique start times: $\sum_k x_{ik} = 1, \quad i = 1, 2, \ldots, N$
  - Dependence must be satisfied (no chaining)

$$t_j \geq t_i + d_i \quad : \forall (v_i, v_j) \in E \Rightarrow \sum_k k\ x_{jk} \geq \sum_k k\ x_{ik} + d_i$$

  - Resource constraints

$$\sum_{i:RT(v_i)=r} \sum_{l=k-d_i+1}^{k} x_{il} \leq a_r, \quad r = 1, \ldots, n_{res}, \quad k = 1, \ldots, L$$

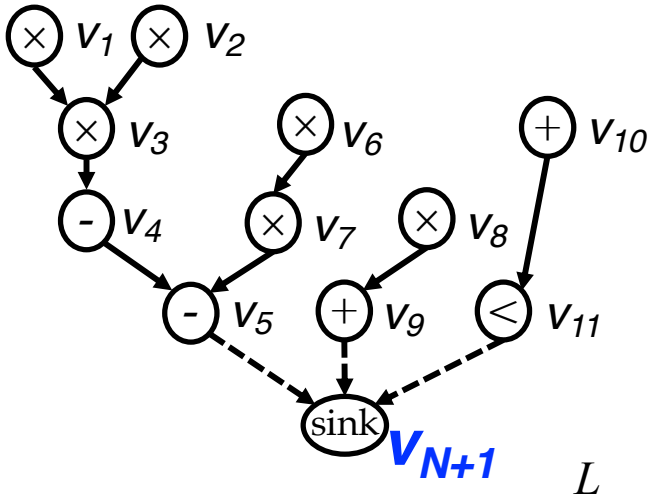
$RT(v_i)$: resource type ID of operation v_i (between 1~n_{res})

a_r is the number of available resources for resource of type r

ILP Formulation of RCS: Objective

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

- To minimize the overall latency, we can introduce a pseudo node to serve as a unique sink (v_{N+1})
  - This sink depends on the original primary output nodes
  - We then minimize the start time of the sink node

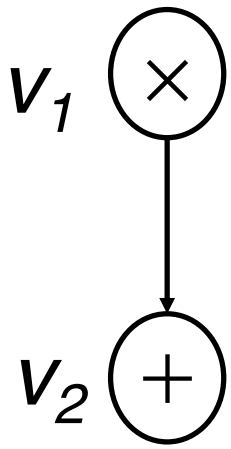


$$\min t_{N+1} \quad \Rightarrow \quad \min \sum_{k=1}^{L} k \bullet x_{N+1,k}$$

Exercise: ILP for ASAP Scheduling

- Two operations: v_1 and v_2
  - Each operate has a full-cycle delay
  - No operation chaining allowed
  - L = 3, i.e., a three-cycle scheduling window

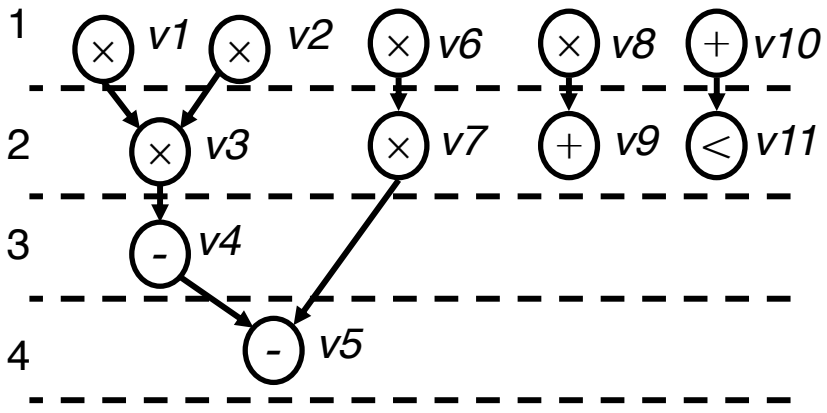
- Objective: ASAP (no resource constraints here)
- Write down the ILP formulation



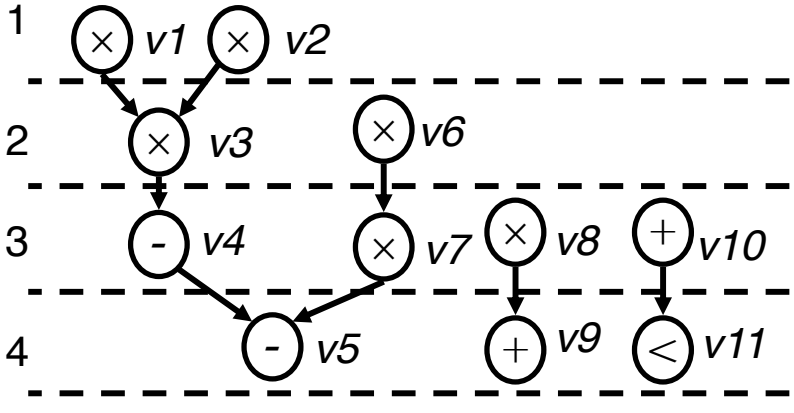
Use of ASAP and ALAP

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

- We can write the ILP without ASAP and ALAP, but using ASAP and ALAP will simplify the inequalities



ASAP schedule



ALAP schedule

ILP Formulation: Unique Start Time Constraints

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

- Without using ASAP and ALAP

$$x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} = 1$$

$$x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} = 1$$

$$\ldots$$

$$x_{11,1} + x_{11,2} + x_{11,3} + x_{11,4} = 1$$

- Using ASAP and ALAP

$$x_{1,1} = 1$$

$$x_{2,1} = 1$$

$$x_{3,2} = 1$$

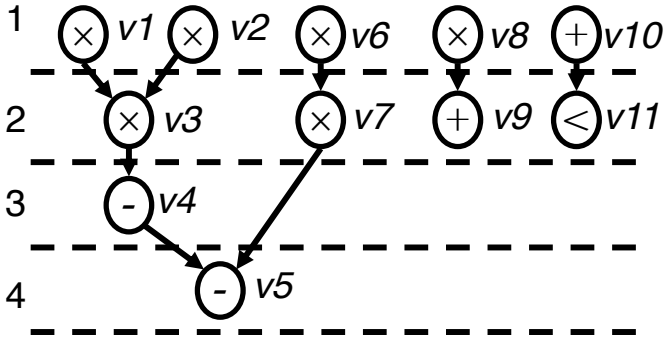
$$\ldots$$

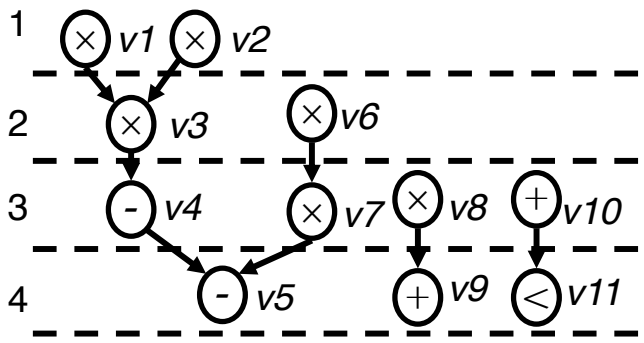
$$x_{6,1} + x_{6,2} = 1$$

$$\ldots$$

$$x_{9,2} + x_{9,3} + x_{9,4} = 1$$

$$\ldots$$





assume L=4

ILP Formulation: Dependence Constraints

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

$$2x_{7,2} + 3x_{7,3} - x_{6,1} - 2x_{6,2} - 1 \geq 0$$

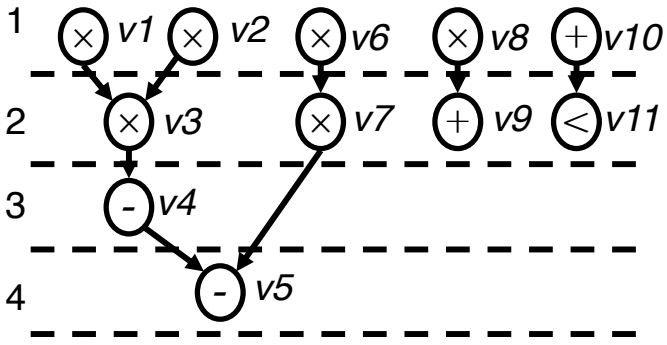
$$2x_{9,2} + 3x_{9,3} + 4x_{9,4} - x_{8,1} - 2x_{8,2} - 3x_{8,3} - 1 \geq 0$$

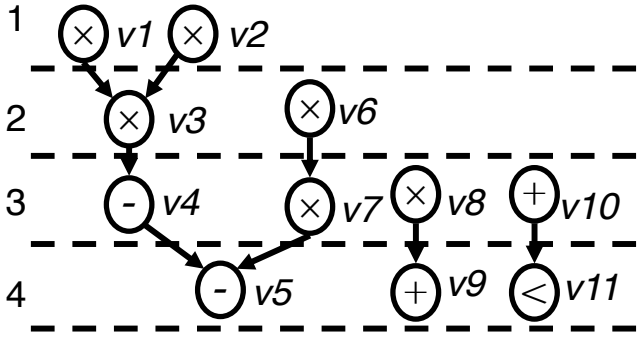
$$2x_{11,2} + 3x_{11,3} + 4x_{11,4} - x_{10,1} - 2x_{10,2} - 3x_{10,3} - 1 \geq 0$$

$$4x_{5,4} - 2x_{7,2} - 3x_{7,3} - 1 \geq 0$$

$$5x_{n,5} - 2x_{9,2} - 3x_{9,3} - 4x_{9,4} - 1 \geq 0$$

$$5x_{n,5} - 2x_{11,2} - 3x_{11,3} - 4x_{11,4} - 1 \geq 0$$





assume L=4

ILP Formulation: Resource Constraints

- Resource constraints (assuming 2 ALUs and 2 multipliers)

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

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

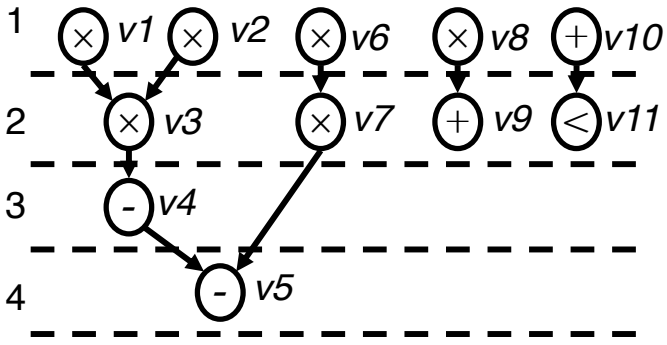
$$x_{7,3} + x_{8,3} \leq 2$$

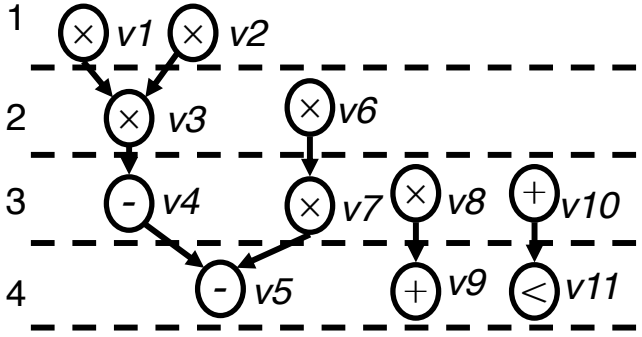
$$x_{10,1} \leq 2$$

$$x_{9,2} + x_{10,2} + x_{11,2} \leq 2$$

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

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





assume L=4

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)