

## ECE382M.20: System-on-Chip (SoC) Design

---

### Lecture 9 – HLS Operation Scheduling

Source: G. De Micheli, *Integrated Systems Center, EPFL*  
“*Synthesis and Optimization of Digital Circuits*”, McGraw Hill, 2001.

Additional sources:

Notes by Kia Bazargan, <http://www.ece.umn.edu/users/kia/Courses/EE5301>

Notes by Rajesh Gupta, UCSD, <http://www.cecs.uci.edu/~rgupta/ics280.html>

Andreas Gerstlauer

Electrical and Computer Engineering  
The University of Texas at Austin  
gerstl@ece.utexas.edu



The University of Texas at Austin  
Chandra Department of Electrical  
and Computer Engineering  
Cockrell School of Engineering

---

## Lecture 9: Outline

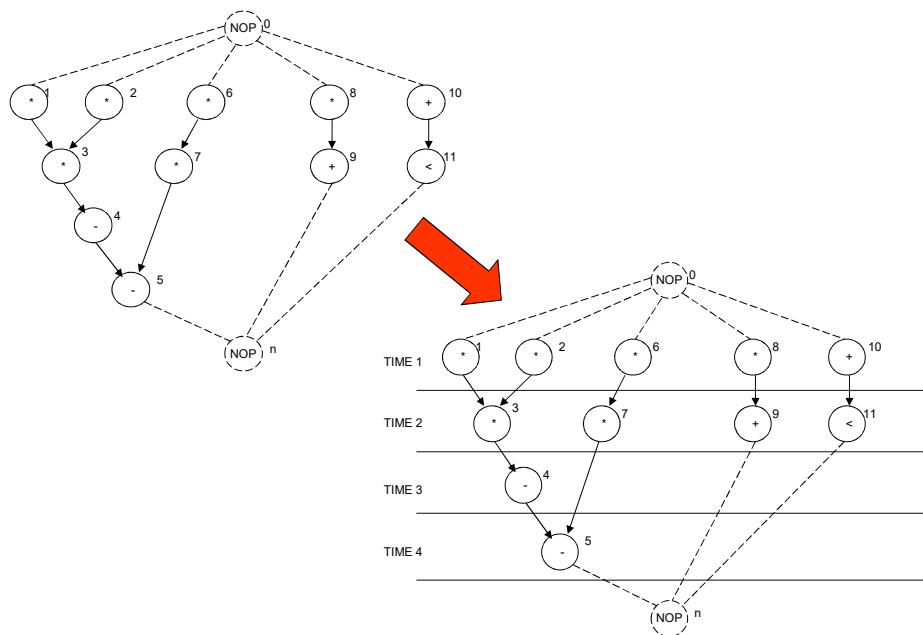
---

- **The scheduling problem**
  - Case analysis
- **Unconstrained scheduling**
  - ASAP and ALAP schedules
- **Resource constrained (RC) scheduling**
  - List scheduling
- **Time constrained (TC) scheduling**
  - Force-directed scheduling
- **Advanced scheduling problems**
  - Chaining
  - Pipelining

## Scheduling

- **Circuit model:**
  - Sequencing graph
  - Cycle-time is given
  - Operation delays expressed in cycles
- **Scheduling:**
  - Determine the start times for the operations
  - Satisfying all the sequencing (timing and resource) constraint
- **Goal:**
  - Determine *area/latency* trade-off

## Example



## Operation Scheduling

---

- **Input:**
  - Sequencing graph  $G(V, E)$ , with  $n$  vertices
  - Cycle time  $\tau$
  - Operation delays  $D = \{d_i; i=0..n\}$
- **Output:**
  - Schedule  $\phi$  determines start time  $t_i$  of operation  $v_i$ .
  - Latency  $\lambda = t_n - t_0$ .
- **Goal: determine area / latency tradeoff**
- **Classes:**
  - Non-hierarchical and unconstrained
  - Latency constrained
  - Resource constrained
  - Hierarchical

## Simplest Method

---

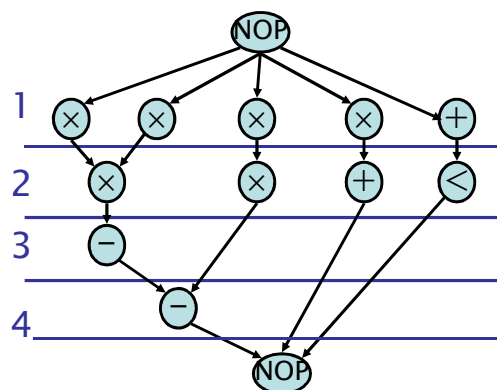
- **All operations have bounded delays**
- **All delays are in cycles:**
  - Cycle-time is given
- **No constraints – no bounds on area**
- **Goal:**
  - Minimize latency

## Min Latency Unconstrained Scheduling

- **Simplest case: no constraints, find min latency**
- **Given set of vertices  $V$ , delays  $D$  and a partial order  $>$  on operations  $E$ ,**
- **find an integer labeling of operations  $\phi: V \rightarrow \mathbb{Z}^+$  such that:**
  - $t_i = \phi(v_i)$
  - $t_i \geq t_j + d_j \quad \forall (v_j, v_i) \in E$
  - and  $\lambda = t_n - t_0$  is minimum
- **Solvable in polynomial time**
  - Bounds on latency for resource constrained problems
  - ASAP algorithm used: topological order

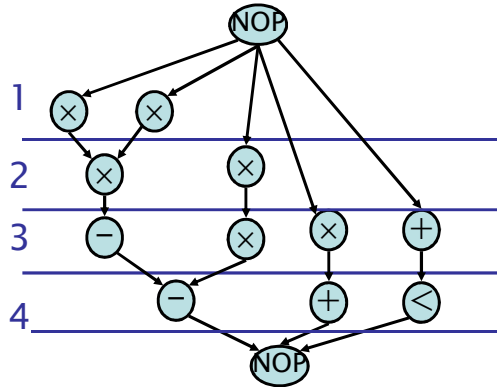
## ASAP Schedules

- **Schedule  $v_0$  at  $t_0=0$**
- **While ( $v_n$  not scheduled)**
  - Select  $v_i$  with all scheduled predecessors
  - Schedule  $v_i$  at  $t_i = \max \{t_j + d_j\}$ ,  $v_j$  being a predecessor of  $v_i$
- **Return  $t_n$**



## ALAP Schedules

- **Schedule  $v_n$  at  $t_n=l$**
- **While ( $v_0$  not scheduled)**
  - Select  $v_i$  with all scheduled successors
  - Schedule  $v_i$  at  $t_i = \min \{t_j - d_j\}$ ,  $v_j$  being a successor of  $v_i$



ECE382M.20: SoC Design, Lecture 9

© R. Gupta

9

## Remarks

- **ALAP solves a latency-constrained problem**
  - Latency bound can be set to latency computed by ASAP algorithm
- **Mobility**
  - Defined for each operation
  - Difference between ALAP and ASAP schedule
    - Slack on the start time

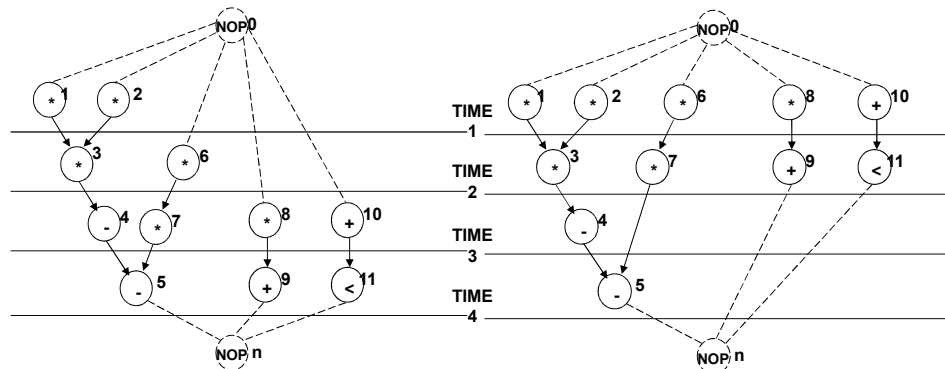
ECE382M.20: SoC Design, Lecture 9

© G. De Micheli

10

## Example

- **Operations with zero mobility:**
  - $\{v_1, v_2, v_3, v_4, v_5\}$
  - Critical path
- **Operations with mobility one:**
  - $\{v_6, v_7\}$
- **Operations with mobility two:**
  - $\{v_8, v_9, v_{10}, v_{11}\}$



ECE382M.20: SoC Design, Lecture 9

© G. De Micheli

11

## Lecture 9: Outline

- ✓ The scheduling problem
- ✓ Unconstrained scheduling
- **Resource constrained (RC) scheduling**
  - Exact formulations
    - ILP
    - Hu's algorithm
  - Heuristic methods
    - List scheduling
- Time constrained (TC) scheduling
- Advanced scheduling problems

ECE382M.20: SoC Design, Lecture 9

© G. De Micheli

12

## Scheduling under Resource Constraints

- **Classical scheduling problem**
  - Fix area bound – minimize latency (ML-RCS)
    - Minimum latency resource constrained scheduling
  - The amount of available resources affects the achievable latency
- **Dual problem:**
  - Fix latency bound – minimize resources (MR-LCS)
    - Minimum resources latency constrained scheduling
- **Assumption:**
  - All delays bounded and known

## ML-RCS

- **Given**
  - a set of ops  $V$  with integer delays  $D$
  - a partial order on the operations  $E$
  - upper bounds  $\{a_k; k = 1, 2, \dots, n_{res}\}$  on resource usage
- **Find an integer labeling  $\phi : V \rightarrow \mathbf{Z}^+$  such that:**
  - $t_i = \phi(v_i)$ ,
  - $t_i \geq t_j + d_j$  for all  $i, j$  s.t.  $(v_j, v_i) \in E$ ,
  - $|\{v_i | T(v_i) = k \text{ and } t_i \leq l < t_j + d_j\}| \leq a_k$ 
    - for all types  $k = 1, 2, \dots, n_{res}$  and steps  $l$
  - and  $t_n$  is minimum
- **Intractable problem**

## ILP Formulation

---

- **Binary decision variables**
  - $X = \{x_{il} \mid i = 1, 2, \dots, n; l = 1, 2, \dots, \bar{\lambda} + 1\}$
  - $x_{il}$  is **TRUE** only when operation  $v_i$  starts in step  $l$  of the schedule (i.e.  $l = t_i$ )
  - $\bar{\lambda}$  is an upper bound on latency
- **Start time of operation  $v_i$  :  $\sum_l l \cdot x_{il}$**

## ILP Constraints

---

- **Operations start only once**  

$$\sum_l x_{il} = 1 \quad i = 1, 2, \dots, n$$
- **Sequencing relations must be satisfied**  

$$t_i \geq t_j + d_j \rightarrow t_i - t_j - d_j \geq 0 \quad \text{for all } (v_j, v_i) \in E$$

$$\sum_l l \cdot x_{il} - \sum_l l \cdot x_{jl} - d_j \geq 0 \quad \text{for all } (v_j, v_i) \in E$$
- **Resource bounds must be satisfied**  
 Simple case (unit delay)  

$$\sum_{i:T(v_i)=k} x_{il} \leq a_k \quad k = 1, 2, \dots, n_{res}; \quad \text{for all } l$$



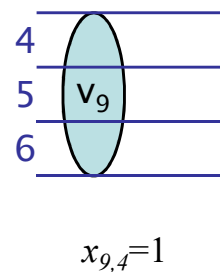
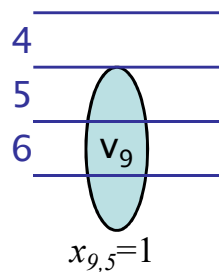
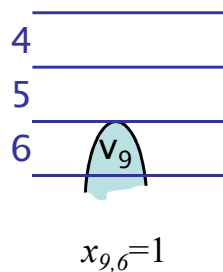
## Start Time vs. Execution Time

- For each operation  $v_i$ , only one start time
- If  $d_i=1$ , then the following questions are the same:
  - Does operation  $v_i$  start at step  $l$ ?
  - Is operation  $v_i$  running at step  $l$ ?
- But if  $d_i > 1$ , the two questions should be formulated as:
  - Does operation  $v_i$  start at step  $l$ ?
    - Does  $x_{il} = 1$  hold?
  - Is operation  $v_i$  running at step  $l$ ?
    - Does the following hold?

$$\sum_{m=l-d_i+1}^l x_{im} \stackrel{?}{=} 1$$

## Operation $v_i$ Still Running at Step $l$ ?

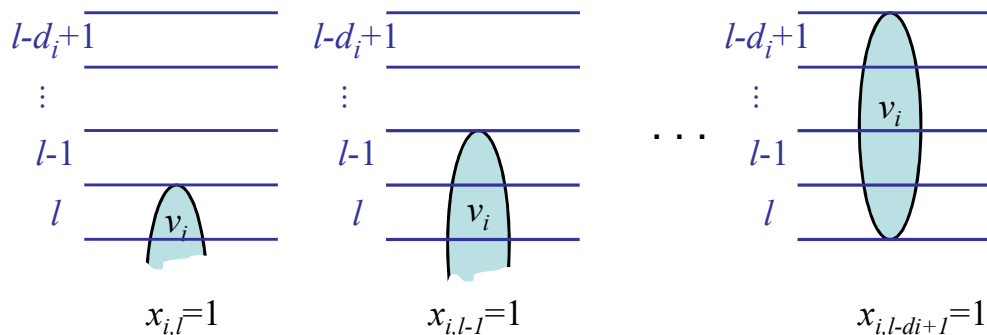
- Is  $v_9$  running at step 6?
  - Is  $x_{9,6} + x_{9,5} + x_{9,4} = 1$  ?



- 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

## Operation $v_i$ Still Running at Step $l$ ?

- Is  $v_i$  running at step  $l$  ?
  - Is  $x_{i,l} + x_{i,l-1} + \dots + x_{i,l-d_i+1} = 1$  ?



ECE382M.20: SoC Design, Lecture 9

© K. Bazargan

19

## ILP Formulation of ML-RCS

- **Constraints:**

- Unique start times:  $\sum_l x_{il} = 1, \quad i = 0, 1, \dots, n$

- Sequencing (dependency) relations must be satisfied

$$t_i \geq t_j + d_j \quad \forall (v_j, v_i) \in E \Rightarrow \sum_l l \cdot x_{il} \geq \sum_l l \cdot x_{jl} + d_j$$

- Resource constraints

$$\sum_{i:T(v_i)=k} \sum_{m=l-d_i+1}^l x_{im} \leq a_k, \quad k = 1, \dots, n_{res}, \quad l = 1, \dots, \bar{\lambda} + 1$$

- **Objective:**  $\min c^T t$

- $t$  = start times vector,  $c$  = cost weight (e.g.,  $[0 \ 0 \ \dots \ 1]$ )

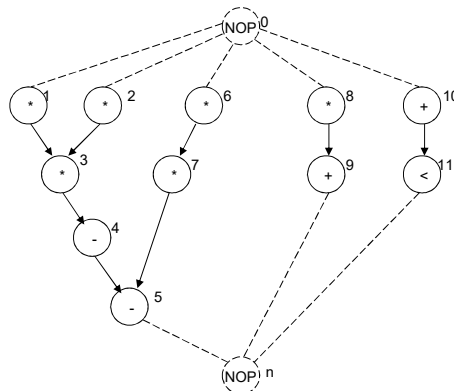
- When  $c = [0 \ 0 \ \dots \ 1]$ ,  $c^T t = \sum_l l \cdot x_{nl}$

ECE382M.20: SoC Design, Lecture 9

© K. Bazargan

20

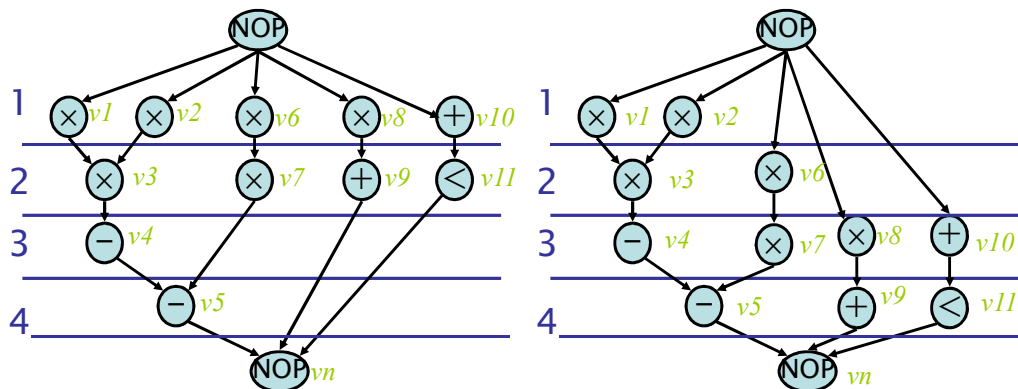
## ILP Example



- **Resource constraints**
  - 2 ALUs; 2 Multipliers
  - $a_1 = 2$ ;  $a_2 = 2$
- **Single-cycle operation**
  - $d_i = 1$  for all  $i$

## ILP Example

- Assume  $\bar{\lambda} = 4$
- **First, perform ASAP and ALAP**
  - (we can write the ILP without ASAP and ALAP, but using ASAP and ALAP will simplify the inequalities)



## ILP Example: Unique Start Times

- Without using ASAP and ALAP values:

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

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

...

...

...

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

- Using ASAP and ALAP:

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

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

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

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

$$x_{5,4} = 1$$

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

$$x_{7,2} + x_{7,3} = 1$$

$$x_{8,1} + x_{8,2} + x_{8,3} = 1$$

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

....

## ILP Example: Dependency Constraints

- Using ASAP and ALAP, the non-trivial inequalities are: (assuming unit delay for + and \*)

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

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

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

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

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

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

## ILP Example: Resource Constraints

- Resource constraints (assuming 2 adders and 2 multipliers)

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

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

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

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

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

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

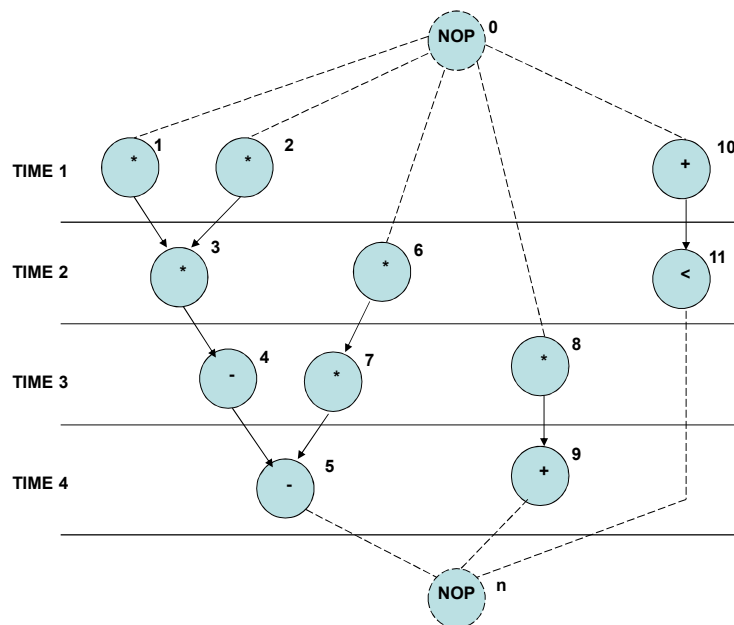
$$x_{5,4} + x_{9,4} + x_{11,4} \leq 2$$

- Objective:

- Since  $\lambda=4$  and sink has no mobility, any feasible solution is optimum, but we can use the following anyway:

$$\text{Min } x_{n,1} + 2 \cdot x_{n,2} + 3 \cdot x_{n,3} + 4 \cdot x_{n,4}$$

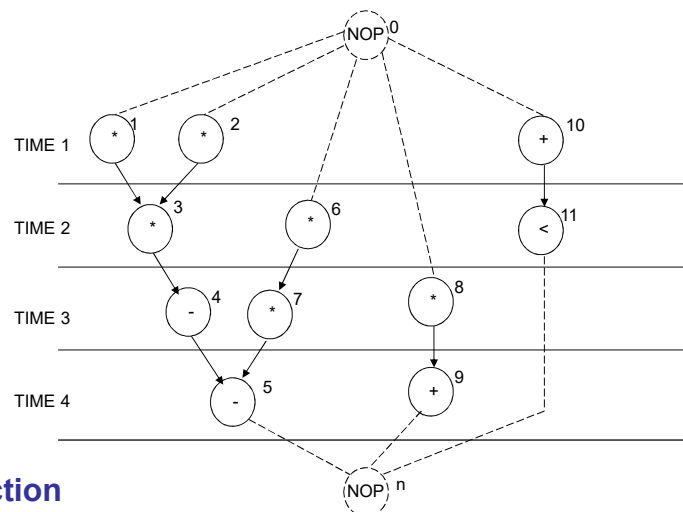
## ILP Example: Solution



## MR-LCS Dual ILP formulation

- **Minimize resource usage under latency constraint**
- **Additional constraint**
  - Latency bound must be satisfied
  - $\sum_l l x_{nl} \leq \lambda + 1$
- **Resource usage is unknown in the constraints**
  - Resource usage is the objective to minimize

## MR-LCS ILP Example



- **Cost function**
  - Multiplier area = 5
  - ALU area = 1
  - Objective function:  $5a_1 + a_2$

## ILP Solving

---

- **Use standard ILP packages**
- **Transform into LP problem**
- **Advantages**
  - Exact method
  - Others constraints can be incorporated
- **Disadvantages**
  - Works well up to few thousand variables

## Hu's Algorithm

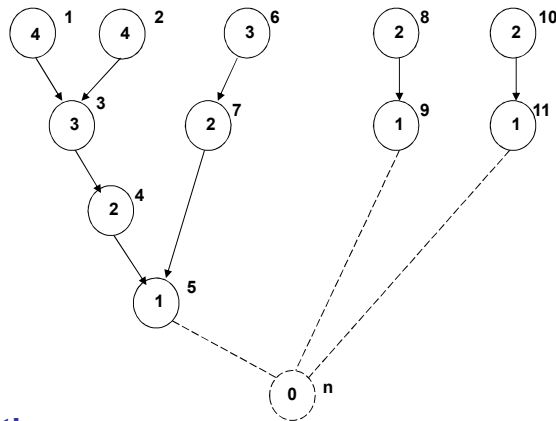
---

- **Simple case of the scheduling problem**
  - Operations of unit delay
  - Operations (and resources) of the same type
- **Hu's algorithm**
  - Greedy, polynomial *and* optimal (exact)
    - Computes lower bound on number of resources for given latency  
OR  
Computes lower bound on latency subject to resource constraints
- **Basic idea**
  - Label operations based on their distances from the sink
  - Try to schedule nodes with higher labels first (i.e., most “critical” operations have priority)

## Hu's Algorithm with $\bar{a}$ Resources

- Label operations with distance to sink
- Set step  $l = 1$
- Repeat until all ops are scheduled
  - $U =$  unscheduled vertices in  $V$ 
    - Predecessors have been scheduled (or no predecessors)
  - Select  $S \subseteq U$  resources with
    - $|S| \leq \bar{a}$
    - Maximal labels
  - Schedule the  $S$  operations at step  $l$
  - Increment step  $l = l + 1$

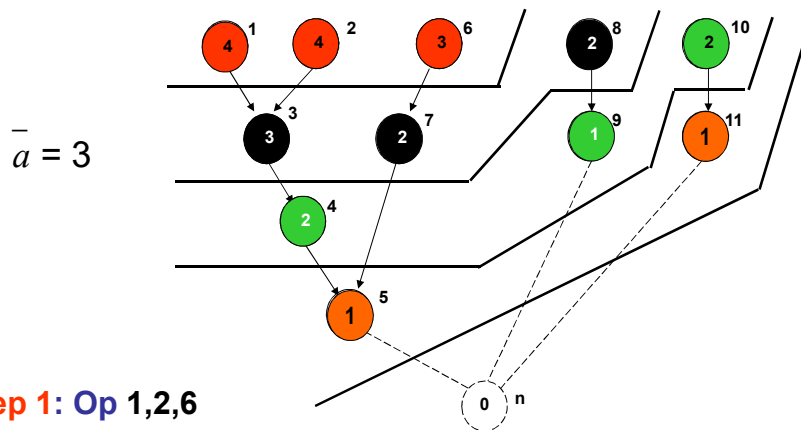
## Hu's Algorithm Example



- **Assumptions**
  - One resource type only
  - All operations have unit delay
- **Labels**
  - Distance to sink



## Hu's Algorithm Example



**Step 1:** Op 1,2,6

**Step 2:** Op 3,7,8

**Step 3:** Op 4,9,10

**Step 4:** Op 5,11

## List Scheduling

- **Heuristic method for:**
  - Min *latency* subject to *resource bound* (ML-RCS)
  - Min *resource* subject to *latency bound* (MR-LCS)
- **Greedy strategy (like Hu's)**
  - Does not guarantee optimality (unlike Hu's)
- **General graphs (unlike Hu's)**
  - Resource constraints on different resource types
  - Operations of arbitrary delay
- **Priority list heuristics**
  - Priority decided by criticality (similar to Hu's)
  - Longest path to sink, longest path to timing constraint
  - $O(n)$  time complexity

## List Scheduling for Minimum Latency

```

LIST_L( G(V, E), a ) {
  l = 1;
  repeat {
    for each resource type k = 1, 2, ..., n_res {
      Determine ready operations Ul,k;
      Determine unfinished operations Tl,k;
      Select Sk ⊆ Ul,k vertices, s.t. |Sk| + |Tl,k| ≤ ak;
      Schedule the Sk operations at step l;
    }
    l = l + 1;
  }
  until (vn is scheduled);
  return (t);
}

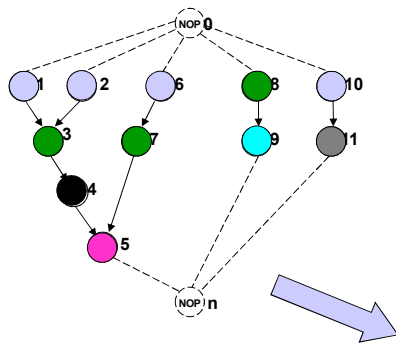
```

ECE382M.20: SoC Design, Lecture 9

© G. De Micheli

35

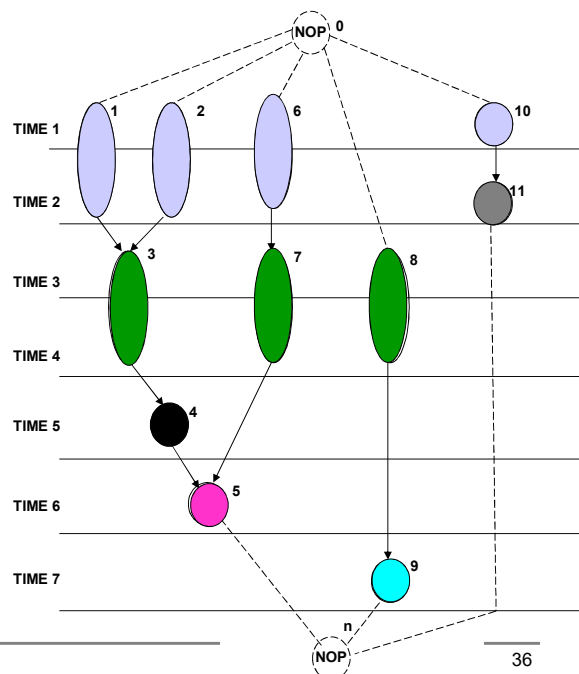
## List Scheduling Example



### Resource bounds

3 multipliers with delay 2

1 ALU with delay 1



ECE382M.20: SoC Design, Lecture 9

36

## Lecture 9: Outline

---

- ✓ The scheduling problem
- ✓ Unconstrained scheduling
- ✓ Resource constrained (RC) scheduling
- Time constrained (TC) scheduling
  - ✓ Exact methods
    - ✓ ILP formulations
    - ✓ Hu's algorithm
  - Heuristics
    - List scheduling
    - Force-directed scheduling
- Advanced scheduling problems

## List Scheduling for Minimum Resources

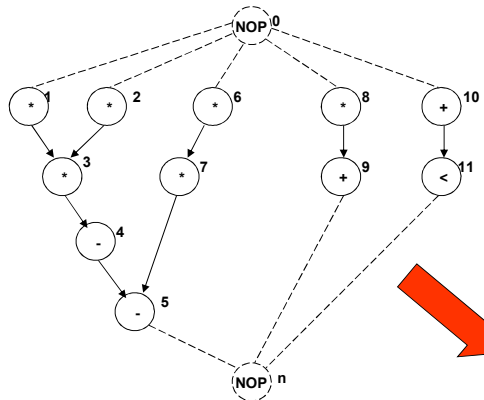
---

```

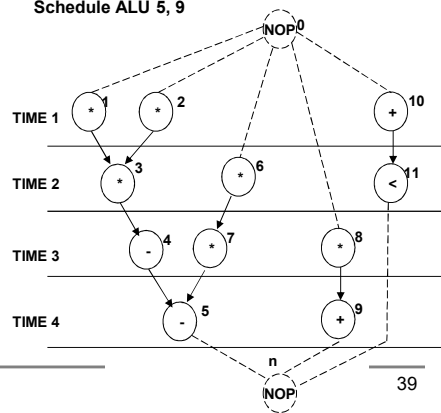
LIST_R(  $G(V, E), \bar{\lambda}$  ) {
   $a = 1$ ;
  Compute the latest possible start times  $t^L$  by ALAP (  $G(V, E), \bar{\lambda}$  );
  if ( $t_0 < 0$ )
    return ( $\emptyset$ );
   $l = 1$ ;
  repeat {
    for each resource type  $k = 1, 2, \dots, n_{res}$  {
      Determine ready operations  $U_{l,k}$ ;
      Compute the slacks  $\{s_i = t_i - l \text{ for all } v_i \in U_{l,k}\}$ ;
      Schedule candidate operations with zero slack and update  $a$ ;
      Schedule candidate operations not needing add'l resources;
    }
     $l = l + 1$ ;
  }
  until ( $y_n$  is scheduled) ;
  return ( $t, a$ );
}

```

## List Scheduling Example



- Step 1**  
Two multiplications on CP  
Set  $a_1 = 2$   
Schedule Mult 1,2  
Schedule ALU 10
- Step 2**  
Schedule Mult 3,6  
Schedule ALU 11
- Step 3**  
Schedule Mult 7,8  
Schedule ALU 4
- Step 4**  
Set  $a_2 = 2$   
Schedule ALU 5,9



- **Assumptions**
  - Unit-delay resources
  - Maximum latency = 4
- **Start with**
  - $a_1 = 1$  multiplier
  - $a_2 = 1$  ALUs

ECE382M.20: SoC Design, Lecture 9

39

## Force-Directed Scheduling (FDS)

- **Heuristic, similar to list scheduling**
  - Can handle ML-RCS and MR-LCS
  - For ML-RCS, schedules step-by-step
  - BUT, selection of the operations tries to find the *globally* best set of operations
- **Idea [Paulin]**
  - Find the mobility  $\mu_i = t_i^L - t_i^S$  of operations (ALAP-ASAP)
  - Look at the operation type probability distributions
  - Try to flatten the operation type distributions
- **Definition: operation probability density**
  - $p_i(l) = \Pr \{ v_i \text{ executes in step } l \}$
  - Assume uniform distribution:  $p_i(l) = \frac{1}{\mu_i + 1}$  for  $l \in [t_i^S, t_i^L]$

ECE382M.20: SoC Design, Lecture 9

© R. Gupta

40

## Force-Directed Scheduling: Definitions

- **Operation probabilities over control steps**

- $p_i = \{p_i(0), p_i(1), \dots, p_i(n)\}$

- **Operation-type distribution (sum of operation probabilities for each type)**

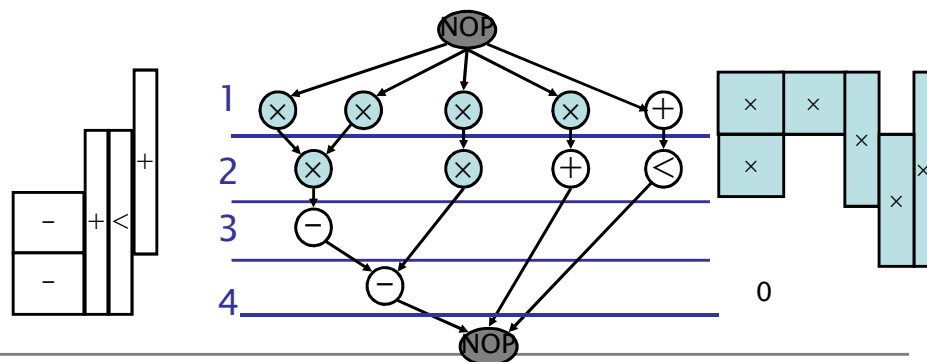
- $q_k(l) = \sum_{i:T(v_i)=k} p_i(l)$

- **Distribution graph of type  $k$  over all steps**

- $\{q_k(0), q_k(1), \dots, q_k(n)\}$

- $q_k(l)$  can be thought of as *expected* operator cost for implementing operations of type  $k$  at step  $l$

## Force-Directed Scheduling Example



## Force-Directed Scheduling Example

$$q_{add}(1) = \frac{1}{3} = 0.33$$

$$q_{add}(2) = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$$

$$q_{add}(3) = 1 + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 2$$

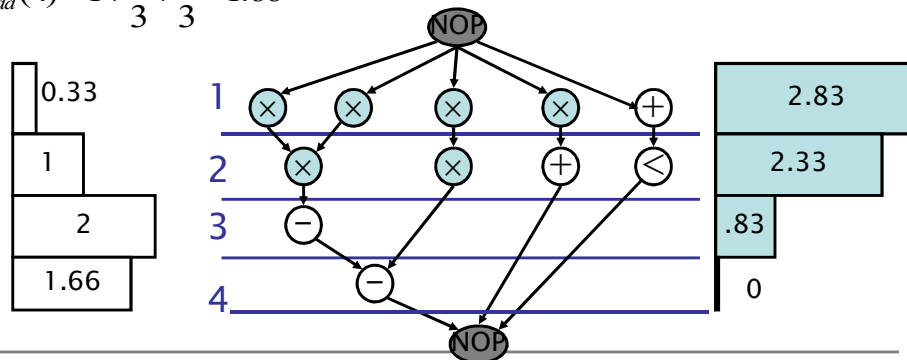
$$q_{add}(4) = 1 + \frac{1}{3} + \frac{1}{3} = 1.66$$

$$q_{mult}(1) = 1 + 1 + \frac{1}{2} + \frac{1}{3} = 2.83$$

$$q_{mult}(2) = 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{3} = 2.33$$

$$q_{mult}(3) = \frac{1}{2} + \frac{1}{3} = 0.83$$

$$q_{mult}(4) = 0$$



ECE382M.20: SoC Design, Lecture 9

© K. Bazargan

43

## Force-Directed Scheduling Algorithm

- **Very similar to LIST\_L( $G(V,E), a$ )**
  - Compute mobility of operations using ASAP and ALAP
  - Computer operation probabilities and type distributions
  - Select and schedule operation
  - Update operation probabilities and type distributions
  - Go to next step/operation
- **Difference with list scheduling in selecting operations**
  - Select operations with least force
  - $O(n^2)$  time complexity due to pair-wise force computations

ECE382M.20: SoC Design, Lecture 9

© R. Gupta

44

## Force

---

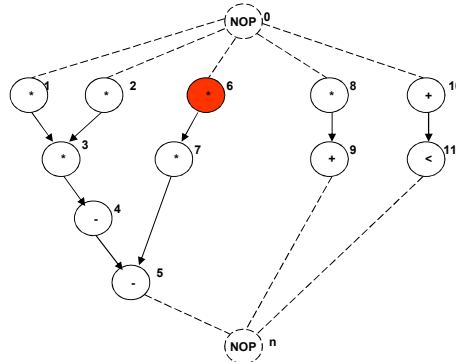
- **Used as *priority* function**
- **Force is related to concurrency**
  - Sort operations for least force
- **Mechanical analogy (spring)**
  - Force = constant × displacement
    - Constant = operation-type distribution
    - Displacement = change in probability

## Two Types of Forces

---

- **Self-force**
  - Sum of forces to feasible schedule steps
  - Self-force for operation  $v_i$  in step  $l$ 
    - Sum over type distribution × delta probability
    - $\sum_{m \text{ in interval}} q_k(m) (\delta_{lm} - p_i(m))$
    - Higher self-force indicates higher mobility
- **Predecessor/successor-force**
  - Related to the predecessors/successors
    - Fixing an operation timeframe restricts timeframe of predecessors/successors
    - Ex: Delaying an operation implies delaying its successors
    - Computed by changes in self-forces of neighbors

## Example: Schedule Operation $v_6$



- Distribution graphs for multiplier and ALU



- Operation  $v_6$  can be scheduled in step 1 or step 2

## Example: Operation $v_6$

- Op  $v_6$  can be scheduled in the first two steps
  - $p(1) = 0.5; p(2) = 0.5; p(3) = 0; p(4) = 0$
- Distribution
  - $q(1) = 2.8; q(2) = 2.3$
- Assign  $v_6$  to step 1
  - Variation in probability  $1 - 0.5 = 0.5$  for step 1
  - Variation in probability  $0 - 0.5 = -0.5$  for step 2
- Self-force
  - $2.8 * 0.5 - 2.3 * 0.5 = +0.25$
- No successor force
- Total force = 0.25



## Example: Operation $v_6$

---

- **Assign  $v_6$  to step 2**
  - variation in probability  $0 - 0.5 = -0.5$  for step 1
  - variation in probability  $1 - 0.5 = 0.5$  for step 2
- **Self-force**
  - $-2.8 * 0.5 + 2.3 * 0.5 = -0.25$
- **Successor-force**
  - Operation  $v_7$  assigned to step 3
  - Succ. force is  $2.3 ( 0 - 0.5 ) + 0.8 ( 1 - 0.5 ) = - .75$
- **Total force = -1**

## Example: Operation $v_6$

---

- **Total force in step 1 = + 0.25**
- **Total force in step 2 = -1**
- **Conclusion:**
  - Least force is for step 2
  - Assigning  $v_6$  to step 2 reduces concurrency

## FDS for Minimum Resources

---

```

FDS (  $G(V, E)$ ,  $\overline{\lambda}$  )
{
  repeat {
    Compute/update the time-frames;
    Compute the operation and type probabilities;
    Compute the self-forces, p/s-forces and total forces;
    Schedule the op. with least force;
  }
  until (all operations are scheduled)
  return (t);
}

```

## Scheduling Generalizations

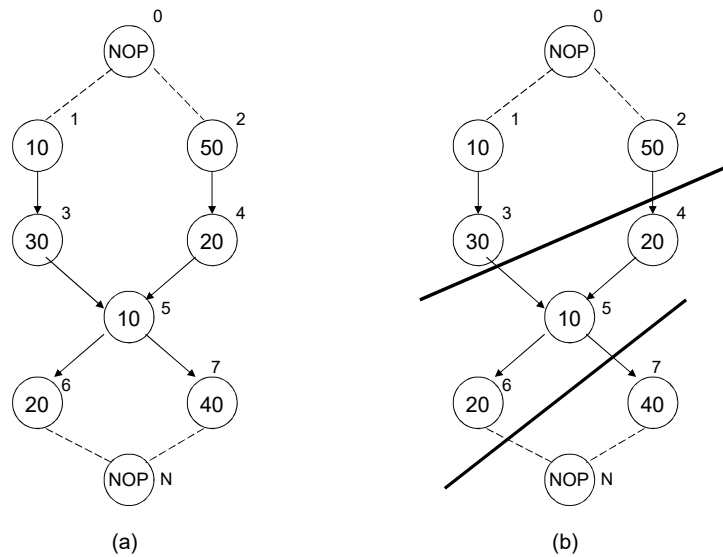
---

- **Detailed timing constraints**
  - Protocol and interface synthesis
    - Bounds on start time differences
    - Forward & backward edges for min/max constraints
- **Operation generalizations**
  - Unbounded delay operations (e.g. synchronization)
    - Relative scheduling w.r. to anchors and combine
  - Conditional operations
- **Resource generalizations**
  - Multi-cycling and chaining
  - Pipelined resources
- **Model generalizations**
  - Hierarchy
  - Pipelining
  - Loops

## Multi-Cycling and Chaining

- Consider delays of resources not in terms of cycles
  - Use scheduling to *chain* multiple operations in the same control step
  - Use scheduling to *multi-cycle* an operation across more than one control step
- Useful techniques to explore effect of *cycle-time* on area/latency trade-off
- Algorithms
  - ILP
  - ALAP/ASAP
  - List scheduling

## Chaining Example



- **Cycle-time: 50**

## Pipelining

---

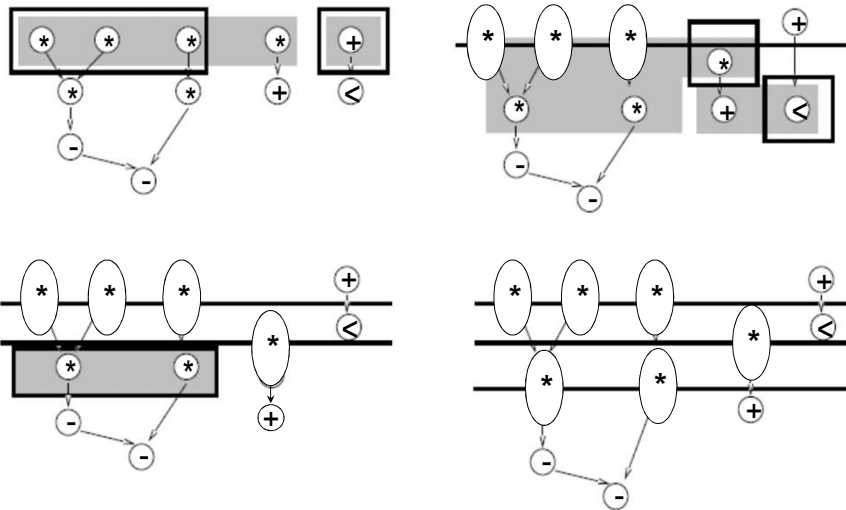
- **Two levels of data pipelining**
  - Structural pipelining
    - Pipelined resources or datapath
    - Non-pipelined model
  - Functional pipelining
    - Non-pipelined resources
    - Pipelined model
- **Control pipelining**
  - Pipelined control logic

## Structural Pipelining

---

- **Non-pipelined model using pipelined resources**
- **Resources characterized by**
  - Execution delay
  - Data introduction interval:  $DII$
- **Implications**
  - Operations sharing a pipelined resource are serialized (always)
  - Operations do not have data dependency
- **Solution using list scheduling**
  - Relax criteria for selection of vertices

## Structural Pipelining Example

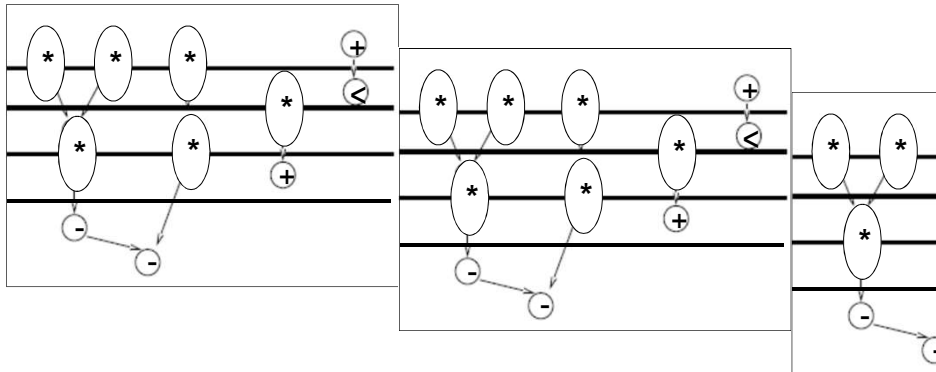


- 3 multipliers w/ 2 cycle delay and  $DII = 1$

## Functional (Loop) Pipelining

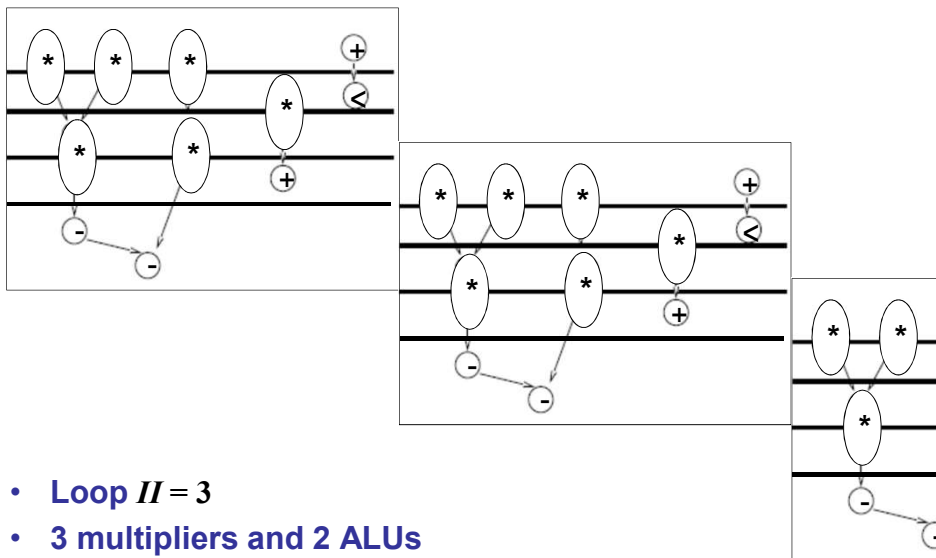
- **Pipelined model, non-pipelined resources**
- **Assume non-hierarchical graphs**
- **Model characterized by**
  - Latency
  - Initiation interval,  $II$
- **Restart source before completing sink**
  - Implicit loop
  - Limited by loop-carried dependencies
- **Solutions using ILP or heuristics**
  - ILP resource constraints modified to include increased concurrency
  - List or force-directed methods

## Loop Pipelining Example



- **Loop  $II = 1$**
- **6 multipliers and 3 ALUs (in this example)**
  - Trade off latency for resources under equal throughput ( $II$ )

## Loop Pipelining Example



- **Loop  $II = 3$**
- **3 multipliers and 2 ALUs**

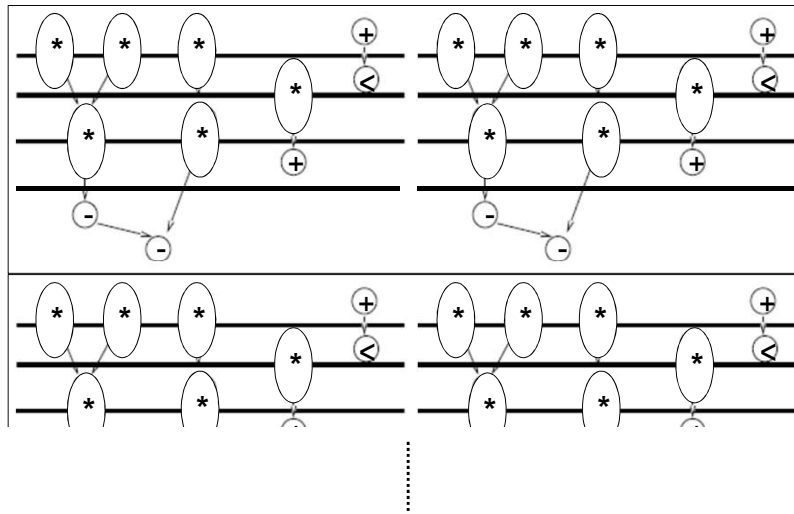
## Loop Pipelining and Concurrency

- **$II$  determines resource usage**
  - Smaller  $II$  leads to larger overlaps, higher resource requirements
 
$$\min\{a_k\} = n_k, \text{ for } II=1 \text{ (all } n_k \text{ operations are concurrent)}$$
  - In general,  $\bar{a}_k = \left\lceil \frac{n_k}{II} \right\rceil$
- **Concurrent operations**
  - Operations  $v_i$  and  $v_j$  are executing concurrently at control step  $l$ , if
 
$$\text{rem}\{t_i/II\} = \text{rem}\{t_j/II\} = l$$
  - Affects the design of the controller circuitry

## Loop Scheduling

- **Exploit potential parallelism across loop invocations**
- **Single loops**
  - Sequential execution
  - Loop unrolling (known iteration count)
    - Merge multiple iterations into one to provide scheduling opportunities
  - Loop pipelining (iteration count might be unknown)
    - Start next iteration while current one is still running
    - Depends on dependencies across iterations
      - Functional pipelining
- **Merging of multiple loops**
  - Run different loops in parallel (no dependencies)

## Loop Unrolling Example



- Loop unroll factor of 2

## Loop Scheduling Summary

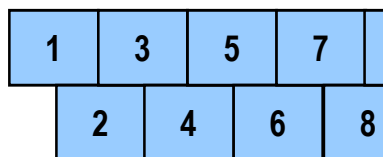
- Sequential



- Unrolled



- Pipelined



- Iteration count =  $N$
- Loop latency =  $N \cdot \lambda$
- Pipeline loop iterations with  $II < \lambda$
- Latency of the pipelined loop
  - $N \cdot II + \text{overhead}$
  - Overhead =  $\lceil \frac{\lambda}{II} \rceil - 1$



## Lecture 9: Summary

---

- **Scheduling determines *area/latency* trade-off**
- **Intractable problem in general**
  - Heuristic algorithms
  - ILP formulation (small-case problems)
- **Several heuristic formulations**
  - List scheduling is the fastest and most used
  - Force-directed scheduling tends to yield good results
- **Several extensions**
  - Chaining and multi-cycling
  - Pipelining