COE 561
Digital System Design & Synthesis
Scheduling

Dr. Aiman H. El-Maleh

Computer Engineering Department

King Fahd University of Petroleum & Minerals

[Adapted from slides of Prof. G. De Micheli: Synthesis & Optimization of Digital Circuits]

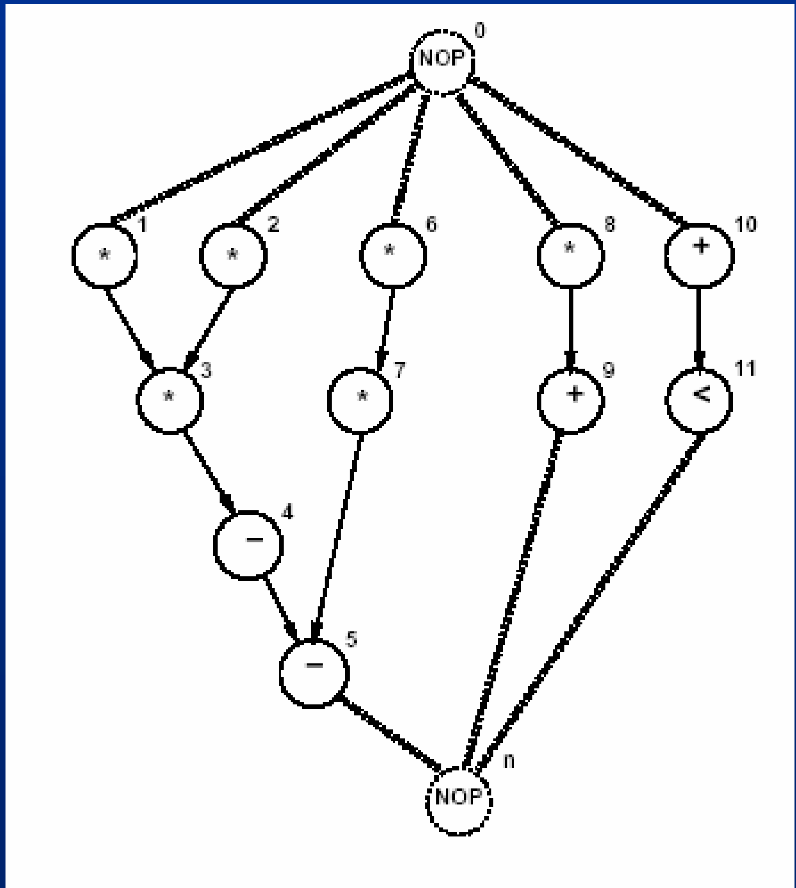
Outline

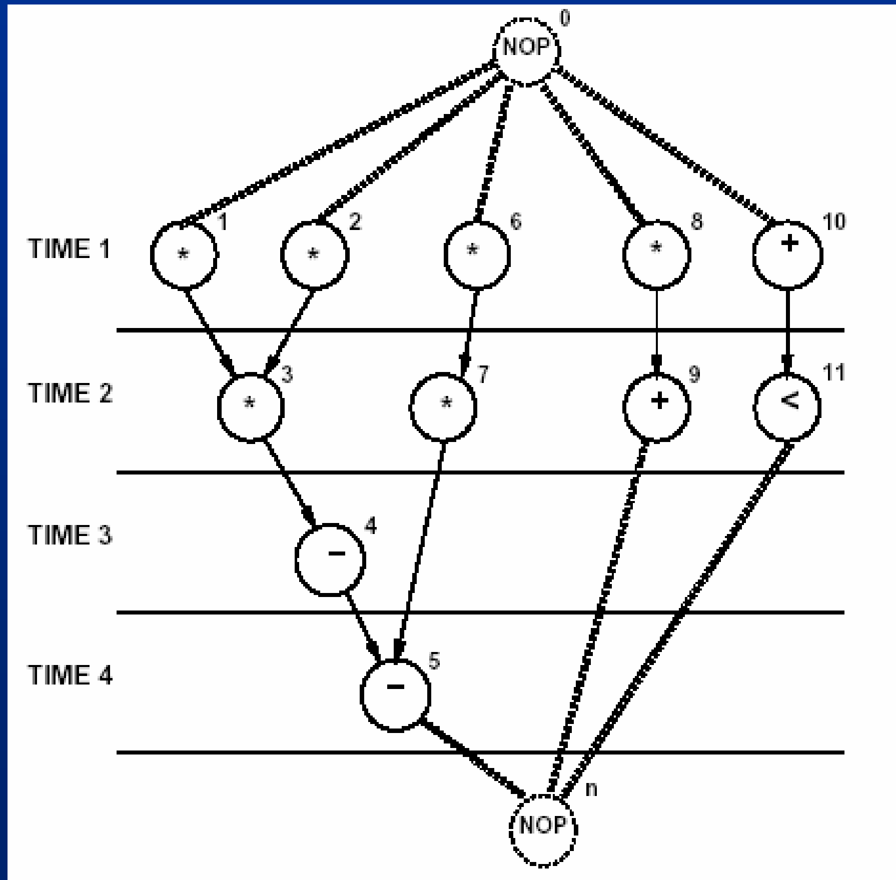
- **The scheduling problem.**
- **Scheduling without constraints.**
- **Scheduling under timing constraints.**
  - Relative scheduling.
- **Scheduling under resource constraints.**
  - The ILP model.
  - Heuristic methods
    - List scheduling
    - Force-directed scheduling

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.
- **Scheduling affects**
  - Area: maximum number of concurrent operations of same type is a lower bound on required hardware resources.
  - Performance: concurrency of resulting implementation.

Scheduling Example





Scheduling Models

- **Unconstrained scheduling.**
- **Scheduling with timing constraints**
  - Latency.
  - Detailed timing constraints.
- **Scheduling with resource constraints.**
- **Simplest scheduling model**
  - All operations have bounded delays.
  - All delays are in cycles.
    - Cycle-time is given.
  - No constraints - no bounds on area.
  - Goal
    - Minimize latency.

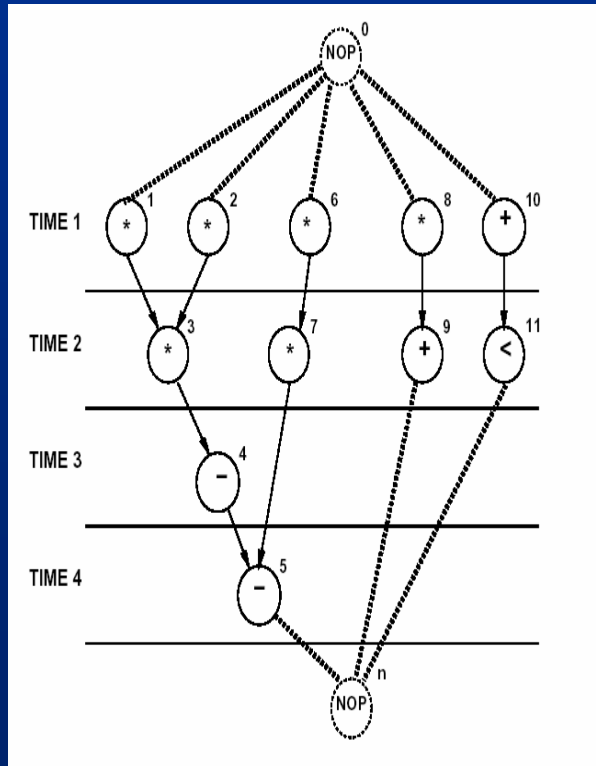
Minimum-Latency Unconstrained Scheduling Problem

- **Given a set of operations V with integer delays D and a partial order on the operations E**
- **Find an integer labeling of the operations $\varphi : V \rightarrow Z^+$, such that**
  - $t_i = \varphi(v_i)$,
  - $t_i \geq t_j + d_j \quad \forall$ i, j s.t. $(v_j, v_i) \in E$
  - and t_n is *minimum*.
- **Unconstrained scheduling used when**
  - Dedicated resources are used.
  - Operations differ in type.
  - Operations cost is marginal when compared to that of steering logic, registers, wiring, and control logic.
  - Binding is done before scheduling: resource conflicts solved by serializing operations sharing same resource.
  - Deriving bounds on latency for constrained problems.

ASAP Scheduling Algorithm

- **Denote by *t^s* the start times computed by the *as soon as possible* (ASAP) algorithm.**
- **Yields *minimum* values of start times.**

ASAP ($G_s(V,E)$) {
    Schedule v_0 by setting $t_0^S = 1$;
    repeat {
        Select a vertex v_i whose pred. are all scheduled;
        Schedule v_i by setting $t_i^S = \max\limits_{j:(v_j,v_i)\in E} t_j^S + d_j$;
    }
    until (v_n is scheduled) ;
    return (t^S);
}



ALAP Scheduling Algorithm

- **Denote by t^L the start times computed by the *as late as possible* (ALAP) algorithm.**
- **Yields *maximum* values of start times.**
- **Latency upper bound $\bar{\lambda}$**

$ALAP(\, G_s(V,E), \bar{\lambda}) \,\{$

$\quad$ Schedule v_n by setting $t_n^L = \bar{\lambda} + 1$;

$\quad$ **repeat** $\{$

$\qquad$ Select vertex v_i whose succ. are all scheduled;

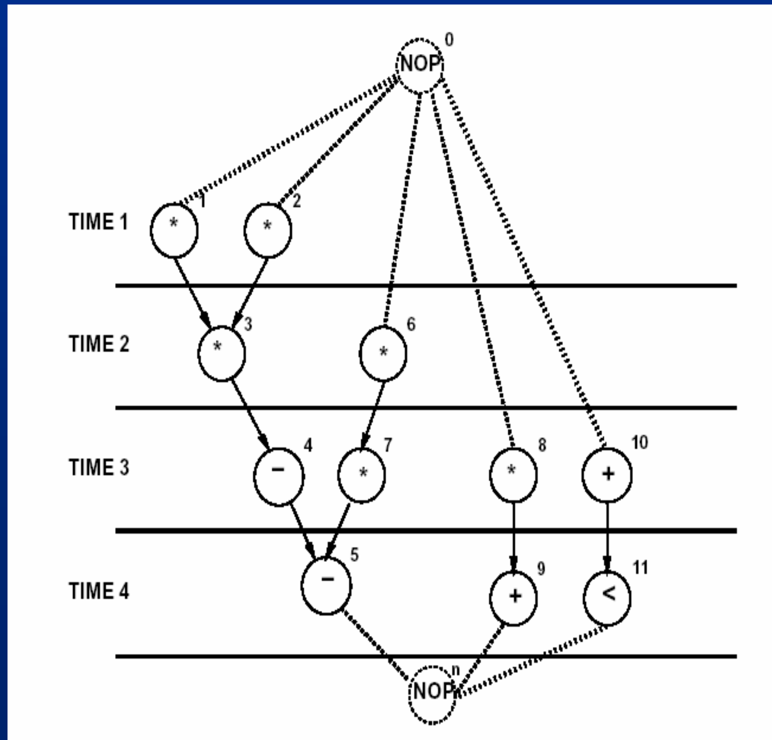
$\qquad$ Schedule v_i by setting $t_i^L = \min\limits_{j:(v_i,v_j)\in E} t_j^L - d_i$;

$\quad \}$

$\quad$ **until** (v_0 is scheduled) ;

$\quad$ **return** (t^L);

$\}$

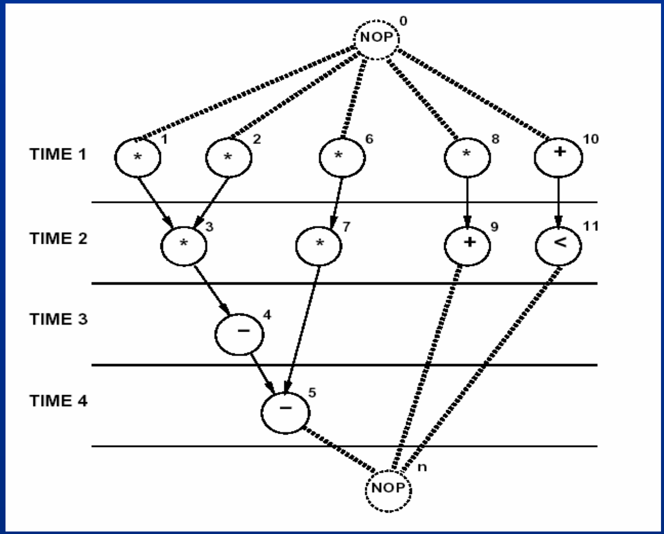


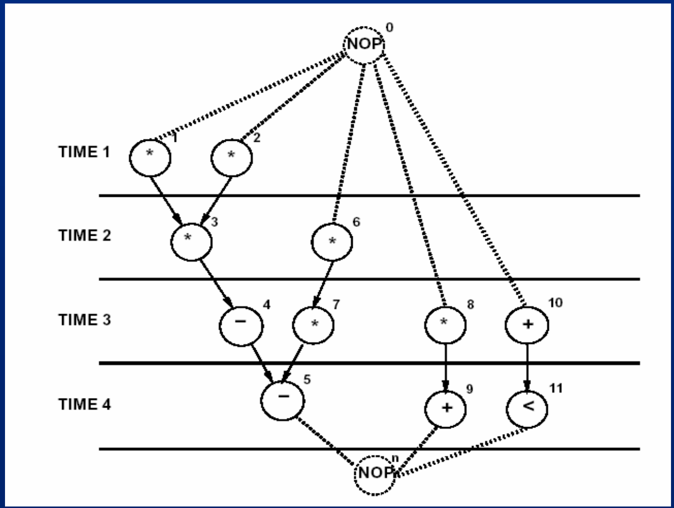
Latency-Constrained Scheduling

- **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.
  - Zero mobility implies that an operation can be started only at one given time step.
  - Mobility greater than 0 measures span of time interval in which an operation may start.
- **Slack on the start time.**

Example

- Operations with zero mobility
  - **{v1, v2, v3, v4, v5}.**
  - **Critical path.**
- Operations with mobility one
  - **{v6, v7}.**
- Operations with mobility two
  - **{v8, v9, v10, v11}**





Scheduling under Detailed Timing Constraints …

- **Motivation**
  - Interface design.
  - Control over operation start time.
- **Constraints**
  - Upper/lower bounds on start-time difference of any operation pair.
- **Minimum timing constraints between two operations**
  - An operation follows another by *at least* a number of prescribed time steps
  - $l_{ij} \geq 0$ requires $t_j \geq t_i + l_{ij}$
- **Maximum timing constraints between two operations**
  - An operation follows another by *at most* a number of prescribed time steps
  - $u_{ij} \geq 0$ requires $t_j \leq t_i + u_{ij}$

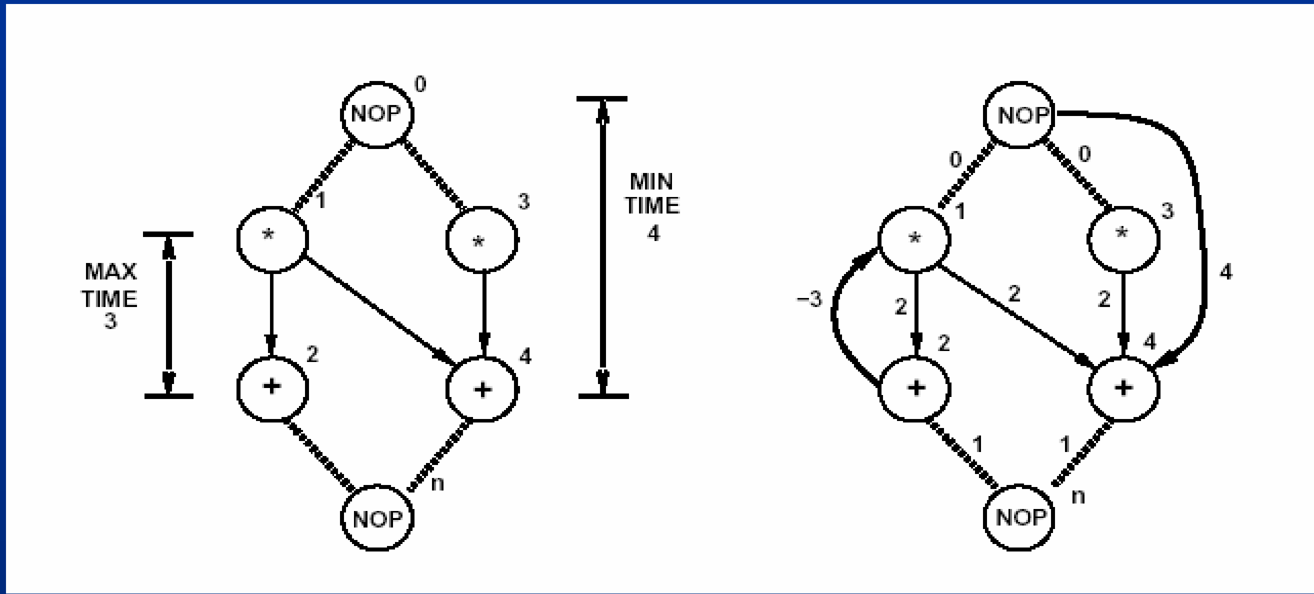
… Scheduling under Detailed Timing Constraints

- **Example**
  - Circuit reads data from a bus, performs computation, writes result back on the bus.
  - Bus interface constraint: data written three cycles after read.
  - Minimum and maximum constraint of 3 cycles between read and write operations.
- **Example**
  - Two circuits required to communicate simultaneously to external circuits.
  - Cycle in which data available is irrelevant.
  - Minimum and maximum timing constraint of zero cycles between two write operations.

Constraint Graph Model

- **Start from sequencing graph.**
- **Model delays as weights on edges.**
- **Add forward edges for minimum constraints.**
  - Edge (v_i, v_j) with weight $l_{ij} \Rightarrow t_j \geq t_i + l_{ij}$
- **Add backward edges for maximum constraints.**
  - Edge (v_j, v_i) with weight $-u_{ij} \Rightarrow t_j \leq t_i + u_{ij}$
  - because $t_j \leq t_i + u_{ij} \Rightarrow t_i \geq t_j - u_{ij}$

… Constraint Graph Model

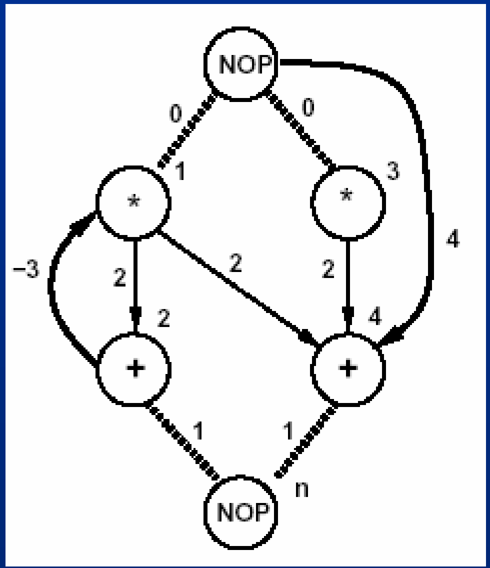


Mul delay = 2
ADD delay =1

Vertex	Start time
v_0	1
v_1	1
v_2	3
v_3	1
v_4	5
v_n	6

Methods for Scheduling under Detailed Timing Constraints …

- **Presence of maximum timing constraints may prevent existence of a consistent schedule.**
- **Required upper bound on time distance between operations may be inconsistent with first operation execution time.**
- **Minimum timing constraints may conflict with maximum timing constraints.**
- **A criterion to determine existence of a schedule:**
  - For each maximum timing constraint u_{ij}
  - Longest weighted path between v_i and v_j must be $\leq u_{ij}$

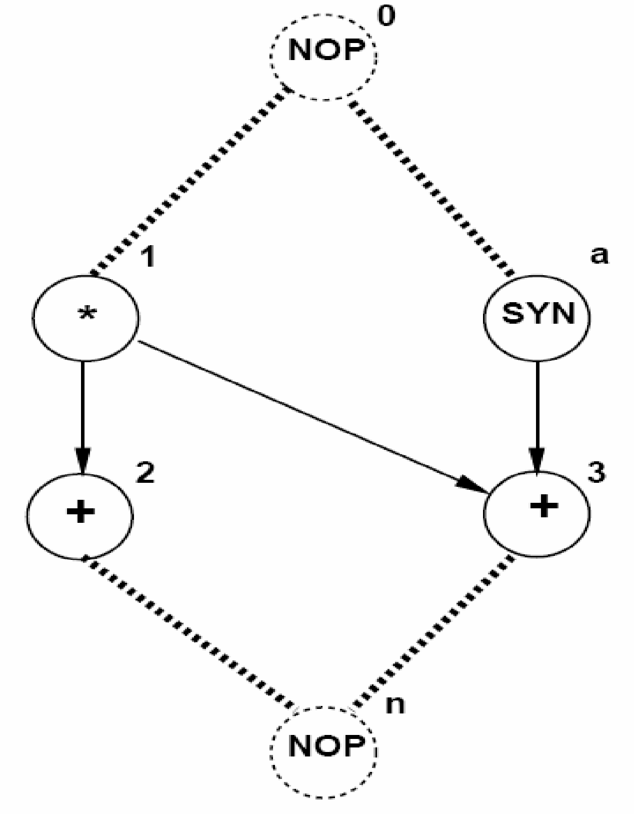


… Methods for Scheduling under Detailed Timing Constraints

- Weight of longest path from source to a vertex is the minimum start time of a vertex.
- Bellman-Ford or Lia-Wong algorithm provides the schedule.
- A necessary condition for existence of a schedule is constraint graph has no positive cycles.

Method for Scheduling with Unbounded-Delay Operations

- **Unbounded delays**
  - Synchronization.
  - Unbounded-delay operations (e.g. loops).
- **Anchors.**
  - Unbounded-delay operations.
- **Relative scheduling**
  - Schedule ops w.r.t. to the anchors.
  - Combine schedules.



$$t_3 = \max\{t_1 + d_1; t_a + d_a\}$$

Relative Scheduling Method

- For each vertex
  - **Determine *relevant* anchor set R(.).**
  - **Anchors affecting start time.**
  - **Determine time offset from anchors.**
- Start-time
  - **Expressed by:**

$$t_i = \max_{a \in R(v_i)} \{t_a + d_a + t_i^a\}$$

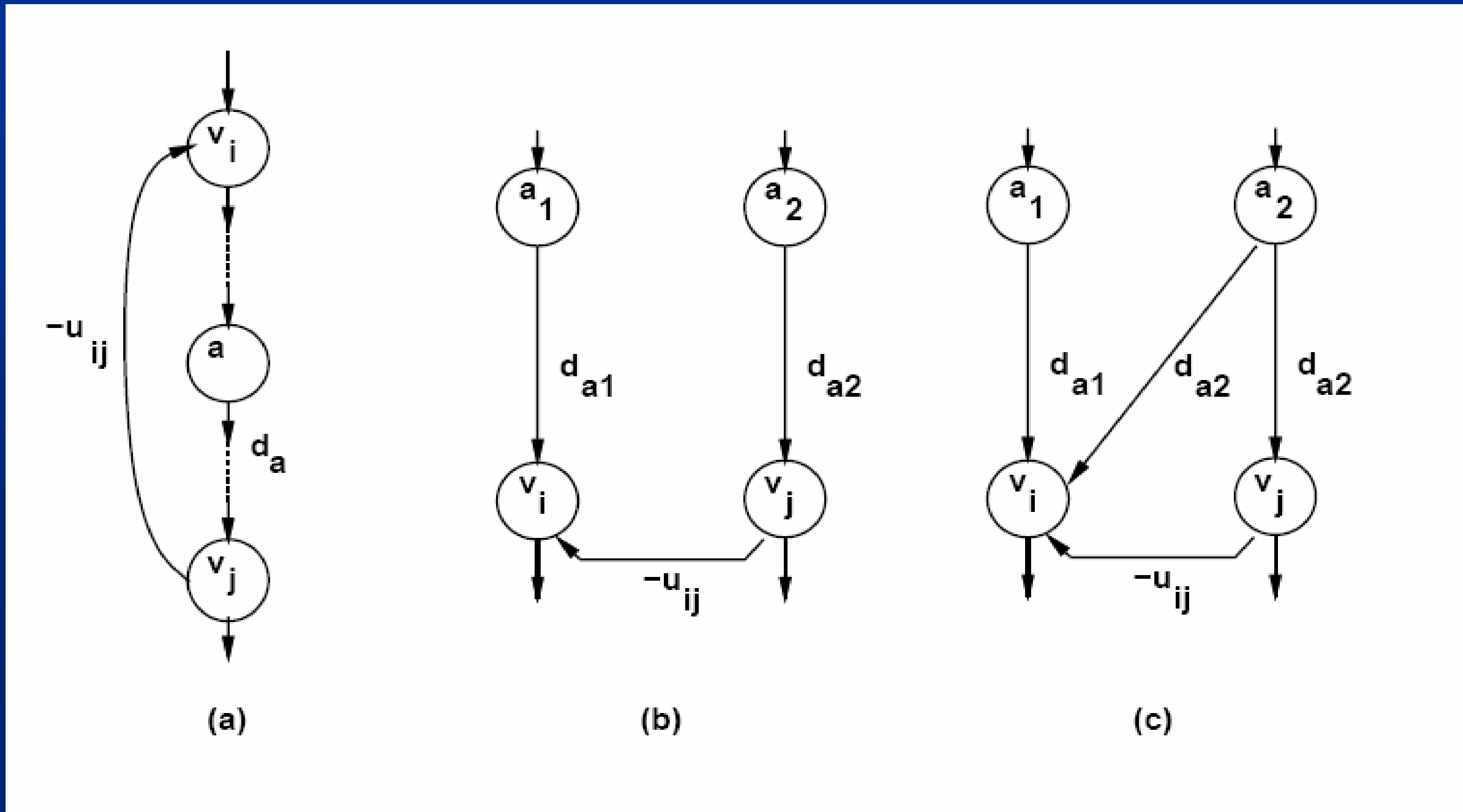
  - **Computed only at run-time because delays of anchors are unknown.**

Relative Scheduling under Timing Constraints

- **Problem definition**
  - Detailed timing constraints.
  - Unbounded delay operations.
- **Solution**
  - May or may not exist.
  - Problem may be ill-specified.
- **Feasible problem**
  - A solution exists when unknown delays are zero.
- **Well-posed problem**
  - A solution exists for any value of the unknown delays.
- **Theorem**
  - A constraint graph can be made well-posed if there are no cycles with unbounded weights.

Example

(a) & (b) Ill-posed constraint *(c) well-posed constraint*

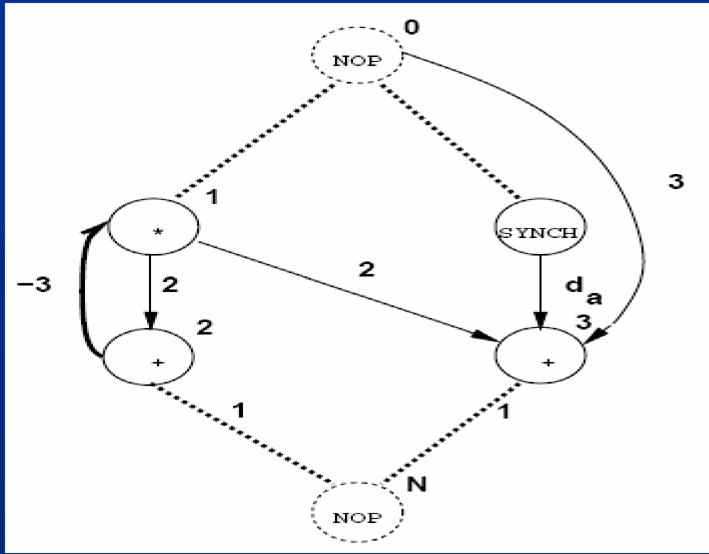


Relative Scheduling Approach

- **Analyze graph**
  - Detect anchors.
  - Well-posedness test.
  - Determine dependencies from anchors.
- **Schedule ops with respect to relevant anchors**
  - Bellman-Ford, Liao-Wong, Ku algorithms.
- **Combine schedules to determine start times:**

$$t_i = \max_{a \in R(v_i)} \{t_a + d_a + t_i^a\} \quad \forall i$$

Example



Vertex v_i	Relevant Anchor Set $R(v_i)$	Offsets t_0	t_a
a	$\{v_0\}$	0	-
v_1	$\{v_0\}$	0	-
v_2	$\{v_0\}$	2	-
v_3	$\{v_0, a\}$	3	0

Scheduling under Resource Constraints

- **Classical scheduling problem.**
  - Fix area bound - minimize latency.
- **The amount of available resources affects the achievable latency.**
- **Dual problem**
  - Fix latency bound - minimize resources.
- **Assumption**
  - All delays bounded and known.

Minimum Latency Resource-Constrained Scheduling Problem

- Given a set of ops V with integer delays D, a partial order on the operations E, and upper bounds $\{a_k; k = 1, 2, \ldots, n_{res}\}$
- Find an integer labeling of the operations $\varphi : V \rightarrow Z^+$**, such that**
  - $t_i = \varphi(v_i)$,
  - $t_i \geq t_j + d_j \quad \forall\, i, j$ s.t. $(v_j, v_i) \in E$

$$|\{v_i | \mathcal{T}(v_i) = k \text{ and } t_i \leq l < t_i + d_i\}| \leq a_k$$
$$\forall \text{types } k = 1, 2, \ldots, n_{res} \text{ and } \forall \text{ steps } l$$

$$\mathcal{T} : V \rightarrow \{1, 2, \ldots n_{res}\}$$

  - and t_n is *minimum*.
- **Number of operations of any given type in any schedule step does not exceed bound.**

Scheduling under Resource Constraints

- **Intractable problem.**
- **Algorithms**
  - Exact
    - Integer linear program.
    - Hu (restrictive assumptions).
  - Approximate
    - List scheduling.
    - Force-directed scheduling.

ILP Formulation …

- Binary decision variables
  - **X = { x_{il}; i = 1, 2, … , n; l = 1, 2, … , $\bar{\lambda}$+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

$$t_i = \sum_l l \cdot x_{il}$$

- Operations start only once

$$\sum_l x_{il} = 1 \quad i = 1, 2, \ldots, n$$

… ILP Formulation …

- Sequencing relations must be satisfied

  - $t_i \geq t_j + d_j \qquad \forall (v_j, v_i) \in E$
  - $\sum_l l \cdot x_{il} - \sum_l l \cdot x_{jl} - d_j \geq 0 \quad \forall (v_j, v_i) \in E$

- Resource bounds must be satisfied

  - Simple case (unit delay)
  - $\sum_{i:\mathcal{T}(v_i)=k} x_{il} \leq a_k \quad k = 1, 2, \ldots, n_{res}; \ \forall l$

… ILP Formulation

- Minimize $c^T t$ such that

$$\sum_j x_{ij} = 1 \quad i = 1, 2, \ldots, n$$

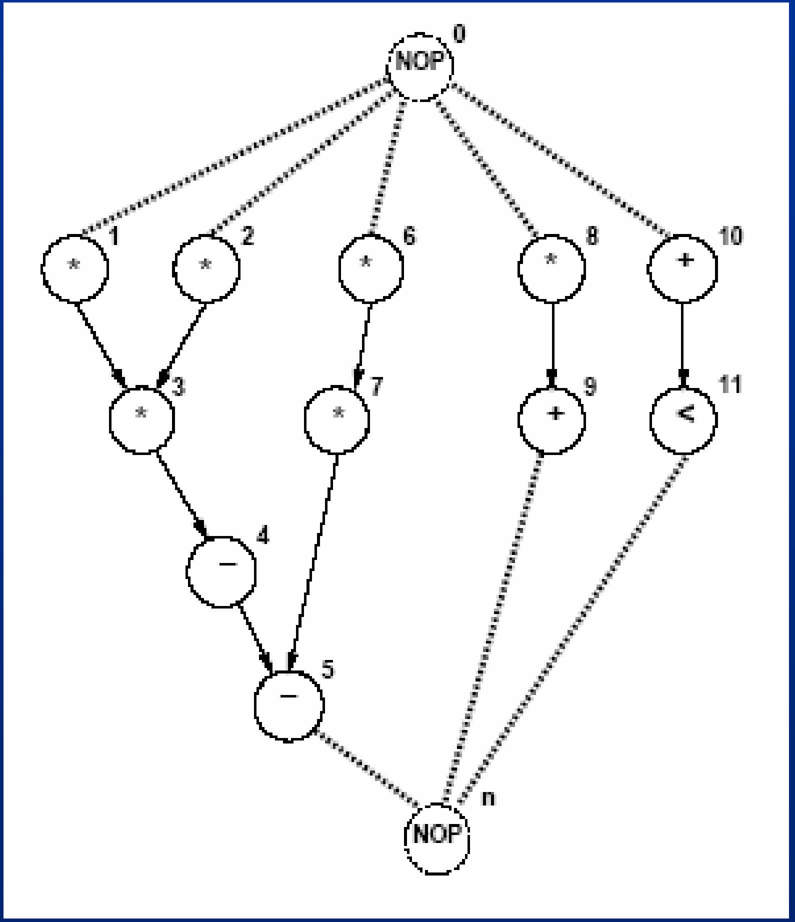
$$\sum_l l \cdot x_{il} - \sum_l l \cdot x_{jl} - d_j \geq 0 \quad i, j = 1, 2, \ldots, n, (v_j, v_i) \in E$$

$$\sum_{i:\mathcal{T}(v_i)=k} \sum_{m=l-d_i+1}^{l} x_{im} \leq a_k \quad k = 1, 2, \ldots, n_{res}; l = 0, 1, \ldots, t_n$$

- $c^T=[0,0,\ldots,0,1]^T$ corresponds to minimizing the latency of the schedule.
- $c^T=[1,1,\ldots,1,1]^T$ corresponds to finding the earliest start times of all operations under the given constraints.

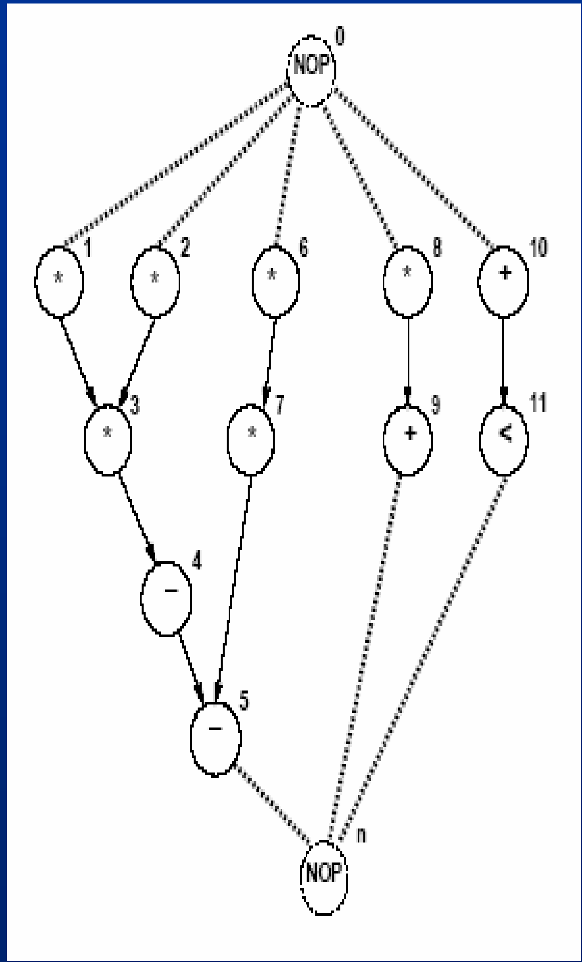
Example …

- **Resource constraints**
  - 2 ALUs; 2 Multipliers.
  - $a_1 = 2; a_2 = 2.$
- **Single-cycle operation.**
  - $d_i = 1 \; \forall i.$
- **Operations start only once**
  - $x_{0,1}=1; x_{1,1}=1; x_{2,1}=1; x_{3,2}=1$
  - $x_{4,3}=1; x_{5,4}=1$
  - $x_{6,1}+ x_{6,2}=1$
  - $x_{7,2}+ x_{7,3}=1$
  - $x_{8,1}+ x_{8,2}+x_{8,3}=1$
  - $x_{9,2}+ x_{9,3}+x_{9,4}=1$
  - $x_{10,1}+ x_{10,2}+x_{10,3}=1$
  - $x_{11,2}+ x_{11,3}+x_{11,4}=1$
  - $x_{n,5}=1$



… Example …

- **Sequencing relations must be satisfied**
  - $2x_{3,2}-x_{1,1} \geq 1$
  - $2x_{3,2}-x_{2,1} \geq 1$
  - $2x_{7,2}+3x_{7,3}-x_{6,1}-2x_{6,2} \geq 1$
  - $2x_{9,2}+3x_{9,3}+4x_{9,4}-x_{8,1}-2x_{8,2}-3x_{8,3} \geq 1$
  - $2x_{11,2}+3x_{11,3}+4x_{11,4}-x_{10,1}-2x_{10,2}-3x_{10,3} \geq 1$
  - $4x_{5,4}-2x_{7,2}-3x_{7,3} \geq 1$
  - $4x_{5,4}-3x_{4,3} \geq 1$
  - $5x_{n,5}-2x_{9,2}-3x_{9,3}-4x_{9,4} \geq 1$
  - $5x_{n,5}-2x_{11,2}-3x_{11,3}-4x_{11,4} \geq 1$
  - $5x_{n,5}-4x_{5,4} \geq 1$

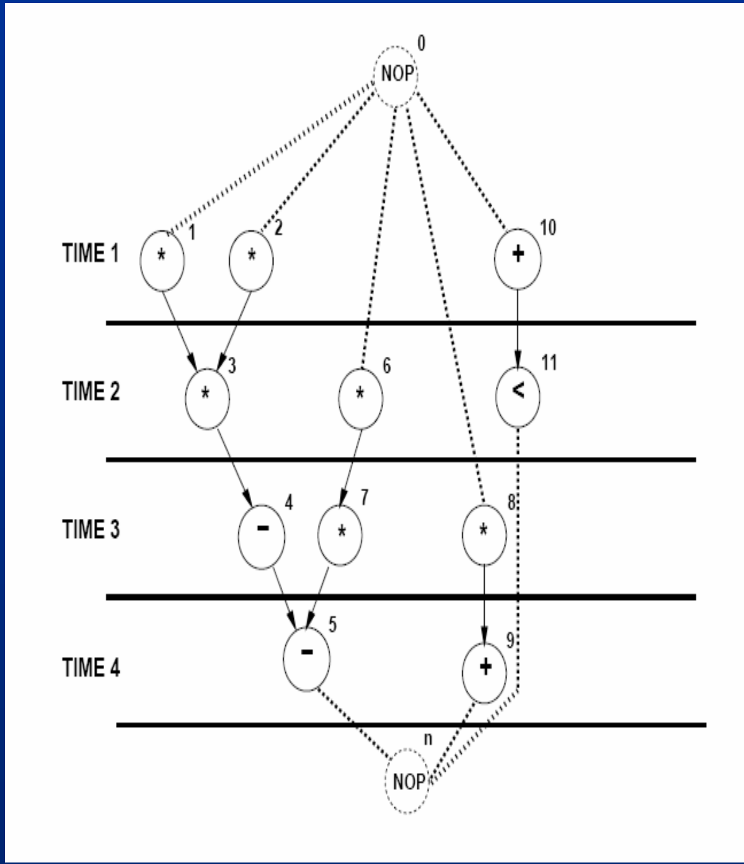


… Example

- Resource bounds must be satisfied:

$$
\begin{aligned}
&- \quad x_{11} + x_{21} + x_{61} + x_{81} \leq 2 \\
&- \quad x_{32} + x_{62} + x_{72} + x_{82} \leq 2 \\
&- \quad \ldots
\end{aligned}
$$

- Any set of start times satisfying constraints provides a feasible solution.
- Any feasible solution is optimum since sink ($x_{n,5}=1$) mobility is 0.



Dual ILP Formulation

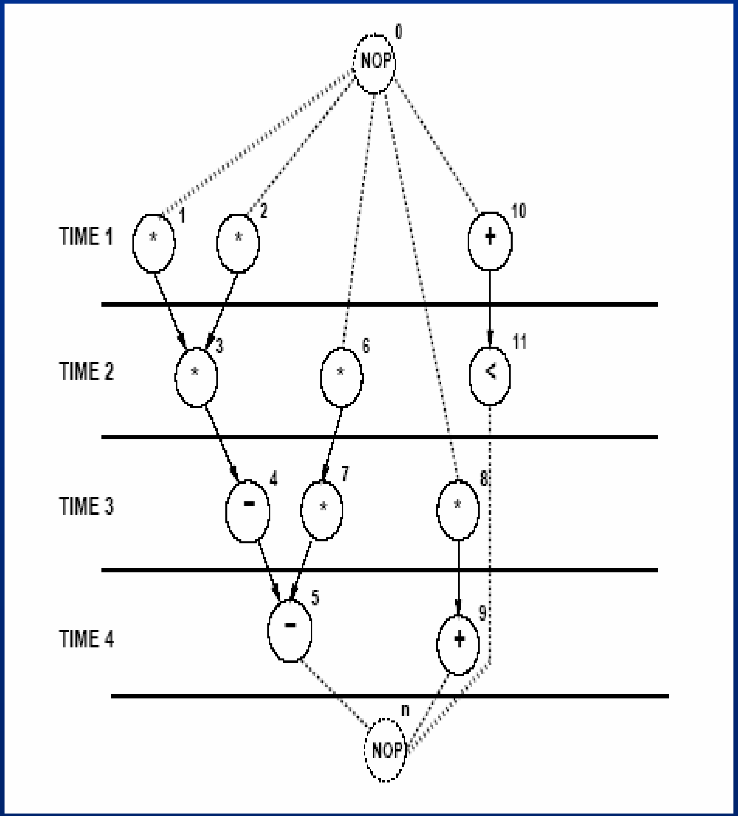
- **Minimize resource usage under latency constraint.**
- **Same constraints as previous formulation.**
- **Additional constraint**
  - Latency bound must be satisfied.

$$\sum_{l} l \; x_{nl} \leq \overline{\lambda} + 1$$

- **Resource usage is unknown in the constraints.**
- **Resource usage is the objective to minimize.**
  - Minimize $\mathbf{c}^T \mathbf{a}$
    - ***a* vector represents resource usage**
    - ***c*^{T} vector represents resource costs**

Example

- Multiplier area = 5; ALU area = 1.
- Objective function: $5a_1 + a_2$. $\bar{\lambda} = 4$
- Start time constraints same.
- Sequencing dependency constraints same.
- Resource constraints
  - $x_{1,1}+x_{2,1}+x_{6,1}+x_{8,1} - a_1 \leq 0$
  - $x_{3,2}+x_{6,2}+x_{7,2}+x_{8,2} - a_1 \leq 0$
  - $x_{7,3}+x_{8,3} - a_1 \leq 0$
  - $x_{10,1} - a_2 \leq 0$
  - $x_{9,2}+x_{10,2}+x_{11,2} - a_2 \leq 0$
  - $x_{4,3}+x_{9,3}+x_{10,3}+x_{11,3} - a_2 \leq 0$
  - $x_{5,4}+x_{9,4}+x_{11,4} - a_2 \leq 0$



ILP Solution

- **Use standard ILP packages.**
- **Transform into LP problem [Gebotys].**
- **Advantages**
  - Exact method.
  - Other constraints can be incorporated easily
    - Maximum and minimum timing constraints
- **Disadvantages**
  - Works well up to few thousand variables.

List Scheduling Algorithms

- **Heuristic method for**
  - Minimum latency subject to resource bound.
  - Minimum resource subject to latency bound.
- **Greedy strategy.**
- **Priority list heuristics.**
  - Assign a weight to each vertex indicating its scheduling priority
    - Longest path to sink.
    - Longest path to timing constraint.

List Scheduling Algorithm for Minimum Latency …

$LIST\_L(\ G(V,E), \mathbf{a}\ )$ {
    $l = 1$;
    repeat {
        for each resource type $k = 1, 2, \ldots n_{res}$ {
            Determine candidate operations $U_{l,k}$;
            Determine unfinished operations $T_{l,k}$;
            Select $S_k \subseteq U_{l,k}$ vertices, s.t. $|S_k| + |T_{l,k}| \leq a_k$;
            Schedule the S_k operations at step l;
            }
        $l = l + 1$;
    }
    until (v_n is scheduled) ;
    return ($\mathbf{t}$);
}

… List Scheduling Algorithm for Minimum Latency

- **Candidate Operations $U_{l,k}$**
  - Operations of type *k* whose predecessors are scheduled and completed at time step before *l*

$$U_{l,k} = \{v_i \in V : Type(v_i) = k \textbf{ and } t_j + d_j \leq l \; \forall j : (v_j, v_i) \in E\}$$

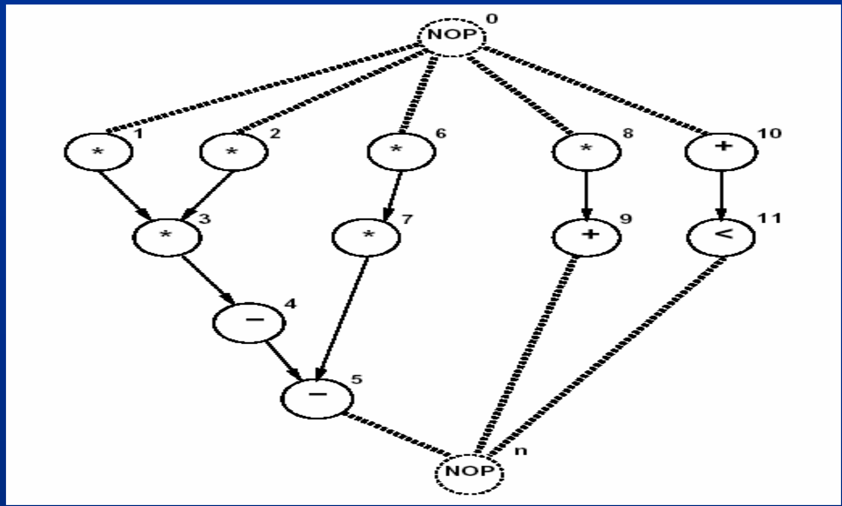
- **Unfinished operations $T_{l,k}$ are operations of type *k* that started at earlier cycles and whose execution is not finished at time *l***

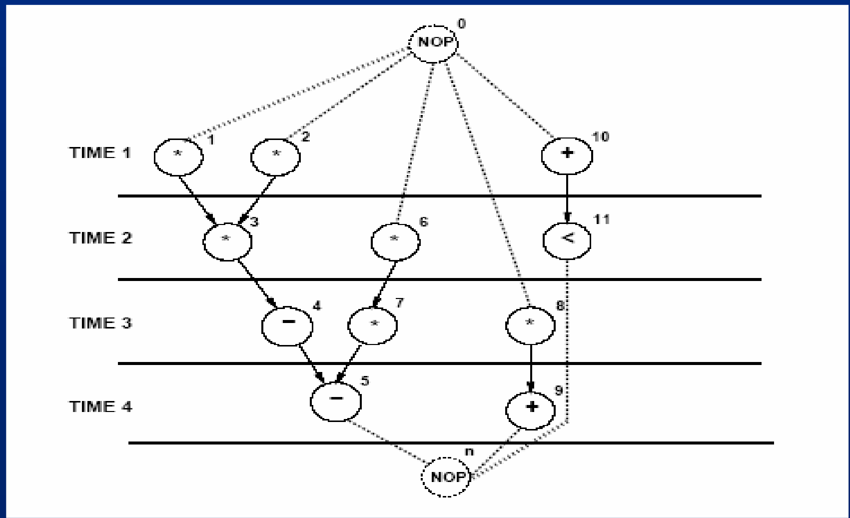
$$T_{l,k} = \{v_i \in V : Type(v_i) = k \textbf{ and } t_j + d_j > l \; \forall j : (v_j, v_i) \in E\}$$

  - Note that when execution delays are 1, $T_{l,k}$ is empty.

Example

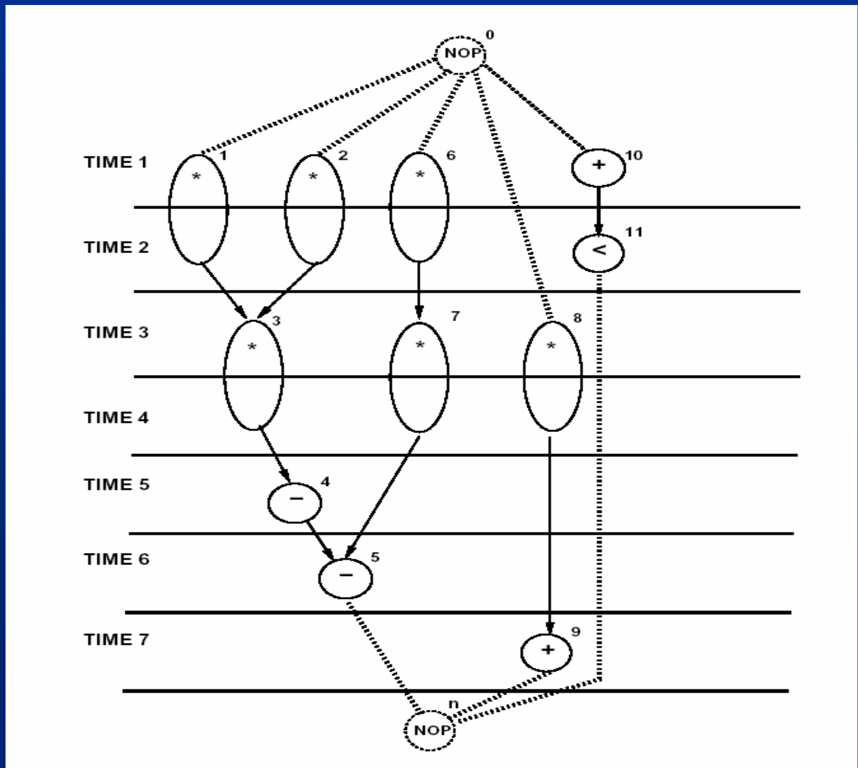
- Assumptions
  - **a_1 = 2 multipliers with delay 1.**
  - **a_2 = 2 ALUs with delay 1.**
- **First Step**
  - $U_{1,1} = \{v_1, v_2, v_6, v_8\}$
  - Select $\{v_1, v_2\}$
  - $U_{1,2} = \{v_{10}\}$; selected
- **Second step**
  - $U_{2,1} = \{v_3, v_6, v_8\}$
  - select $\{v_3, v_6\}$
  - $U_{2,2} = \{v_{11}\}$; selected
- **Third step**
  - $U_{3,1} = \{v_7, v_8\}$
  - Select $\{v_7, v_8\}$
  - $U_{3,2} = \{v_4\}$; selected
- **Fourth step**
  - $U_{4,2} = \{v_5, v_9\}$; selected

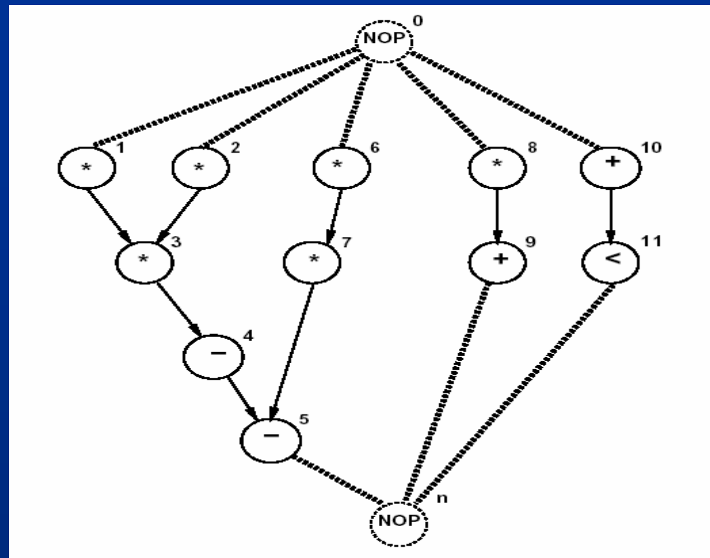




Example

- Assumptions
  - a_1 = 3 multipliers with delay 2.
  - a_2 = 1 ALU with delay 1.





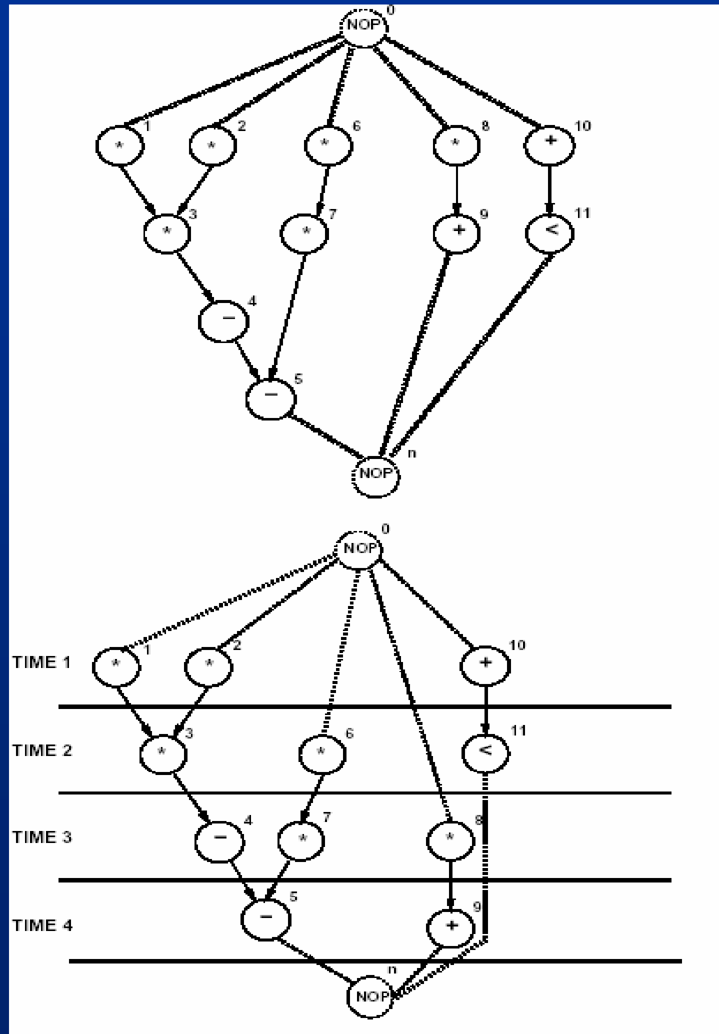
Operation		
Multiply	**ALU**	**Start time**
$\{v_1, v_2, v_6\}$	v_{10}	1
—	v_{11}	2
$\{v_3, v_7, v_8\}$	—	3
—	—	4
—	v_4	5
—	v_5	6
—	v_9	7

List Scheduling Algorithm for Minimum Resource Usage

```
LIST_R( G(V,E), λ̄ ) {
    a = 1;
    Compute the latest possible start times t^L
    by ALAP ( G(V,E), λ̄ );
    if ( t_0^L < 0 )
        return (∅);
    l = 1;
    repeat {
        for each resource type k = 1,2,...n_res {
            Determine candidate operations U_lk;
            Compute the slacks {s_i = t_i^L − l ∀v_i ∈ U_lk};
            Schedule the candidate operations
            with zero slack and update a;
            Schedule the candidate operations
            that do not require additional resources;
        }
        l = l + 1;
    }
    until (v_n is scheduled) ;
    return (t, a);
}
```

Example

- **Assume λ=4**
- **Let a = $[1, 1]^T$**
- **First Step**
  - $U_{1,1} = \{v_1, v_2, v_6, v_8\}$
  - Operations with zero slack $\{v_1, v_2\}$
  - a = $[2, 1]^T$
  - $U_{1,2} = \{v_{10}\}$
- **Second step**
  - $U_{2,1} = \{v_3, v_6, v_8\}$
  - Operations with zero slack $\{v_3, v_6\}$
  - $U_{2,2} = \{v_{11}\}$
- **Third step**
  - $U_{3,1} = \{v_7, v_8\}$
  - Operations with zero slack $\{v_7, v_8\}$
  - $U_{3,2} = \{v_4\}$
- **Fourth step**
  - $U_{4,2} = \{v_5, v_9\}$
  - Both have zero slack; a = $[2, 2]^T$

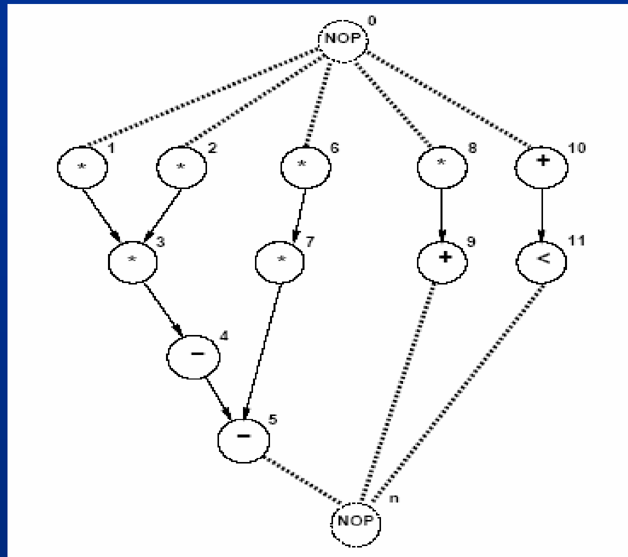


Force-Directed Scheduling …

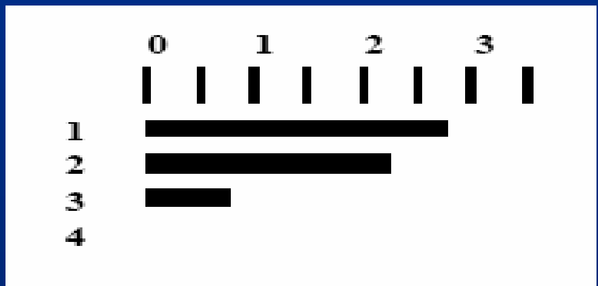
- **Heuristic scheduling methods [Paulin]**
  - Min latency subject to resource bound.
    - Variation of list scheduling: FDLS.
  - Min resource subject to latency bound.
    - Schedule one operation at a time.
- **Rationale**
  - Reward uniform distribution of operations across schedule steps.
- **Operation interval: mobility plus one (μ_i+1).**
  - Computed by ASAP and ALAP scheduling $[t_i^S, t_i^L]$
- **Operation probability $p_i(l)$**
  - Probability of executing in a given step.
  - 1/(μ_i+1) inside interval; 0 elsewhere.

… Force-Directed Scheduling

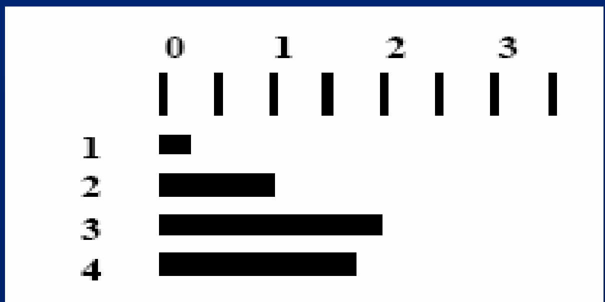
- **Operation-type distribution $q_k(l)$**
  - Sum of the op. prob. for each type.
  - Shows likelihood that a resource is used at each schedule step.



- **Distribution graph for multiplier**



- **Distribution graph for adder**



$p_1(1)=1$, $p_1(2)=p_1(3)=p_1(4)=0$
$p_2(1)=1$, $p_2(2)=p_2(3)=p_2(4)=0$
$\mu_6=1$; time frame [1,2]
$p_6(1)=0.5$, $p_6(2)=0.5$, $p_6(3)=p_6(4)=0$
$\mu_8=2$; time frame [1,3]
$p_8(1)=p_8(2)=p_8(3)=0.3$, $p_8(4)=0$
$q_{mul}(1)=1+1+0.5+0.3=2.8$

Force

- **Used as priority function.**
- **Selection of operation to be scheduled in a time step is based on force.**
- **Forces attract (repel) operations into (from) specific schedule steps.**
- **Force is related to concurrency.**
  - The larger the force the larger the concurrency
- **Mechanical analogy**
  - Force exerted by elastic spring is proportional to displacement between its end points.
  - Force = constant $\times$ displacement.
    - constant = operation-type distribution.
    - displacement = change in probability.

Forces Related to the Assignment of an Operation to a Control Step

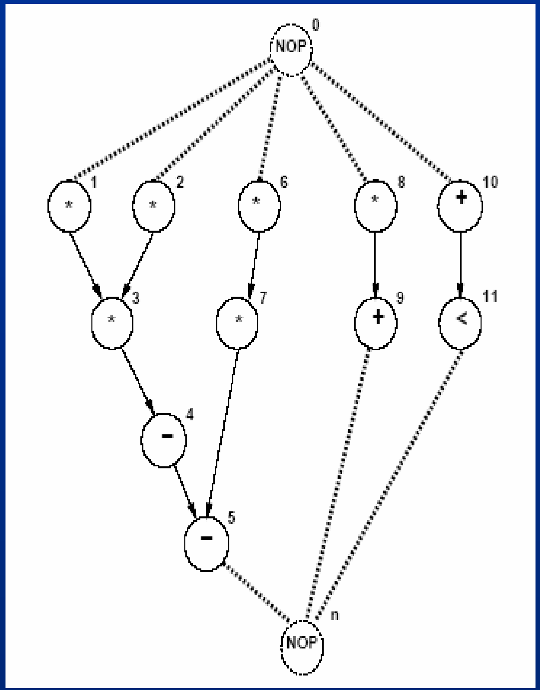
- Self-force
  - **Sum of forces relating operation to all schedule steps in its time frame.**
  - **Self-force for scheduling operation v_i in step *l***

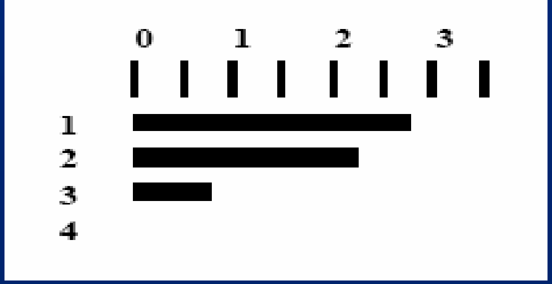
$$self-force(i,l)=\sum_{m=t_i^s}^{t_i^L} q_k(m)(\delta_{lm}-p_i(m))=q_k(l)-\frac{1}{\mu_i+1}\sum_{m=t_i^s}^{t_i^L} q_k(m)$$

  - ***δ_{lm} denotes a Kronecker delta function; equal 1 when m=l.***
- Successor-force
  - **Related to the successors.**
  - **Delaying an operation implies delaying its successors.**

Example: Operation v_6 …

- **It 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; q(3)=0.8.**
- **Assign v_6 to step 1**
  - variation in probability 1 – 0.5 = 0.5 for step1
  - variation in probability 0 – 0.5 = -0.5 for step2
  - Self-force: 2.8 * 0.5 + 2.3 * -0.5 = +0.25
- **Assign v_6 to step 2**
  - variation in probability 0 – 0.5 = -0.5 for step1
  - variation in probability 1 – 0.5 = 0.5 for step2
  - Self-force: 2.8 * -0.5 + 2.3 * 0.5 = -0.25





… Example: Operation v6 …

- **Successor-force**
  - Assigning v_6 to step 2 implies operation v_7 assigned to step 3.
  - 2.3 (0-0.5) + 0.8 (1 -0.5) = -.75
  - Total-force on v_6 = (-0.25)+(-0.75)=-1.
- **Conclusion**
  - Least force is for step 2.
  - Assigning v_6 to step 2 reduces concurrency (i.e. resources).
- **Total force on an operation related to a schedule step**
  - = self force + predecessor/successor forces with affected time frame

$$ps\text{-}force(i,l)_j = \frac{1}{\tilde{\mu}_j + 1} \sum_{m=\tilde{t}_j^S}^{m=t_j^{\tilde{L}}} q_k(m) - \frac{1}{\mu_j + 1} \sum_{m=t_j^S}^{m=t_j^L} q_k(m)$$

… Example: Operation v6

- **Assignment of v_6 to step 2 makes v_7 assigned at step 3**
  - Time frame change from [2, 3] to [3, 3]
  - Variation on force of v_7 = 1*q(3) – ½ * (q(2)+q(3)) = 0.8-0.5(2.3+0.8)= -0.75
- **Assignment of v_8 to step 2 makes v_9 assigned to step 3 or 4**
  - Time frame change from [2, 3, 4] to [3, 4]
  - Variation on force of v_9 = 1/2*(q(3)+q(4)) – 1/3 * (q(2)+q(3)+q(4)) = 0.5*(2+1.6)-0.3*(1+2+1.6)=0.3

Force-Directed List Scheduling: Minimum Latency under Resource Constraints

- **Outer structure of algorithm same as LIST-L.**
- **Selected candidates determined by**
  - Reducing iteratively candidate set $U_{l,k}$.
  - Operations with least force are deferred.
  - Maximize local concurrency by selecting operations with large force.
  - At each outer iteration of loop, time frames updated.

```
LIST_L( G(V,E),a ) {
    l = 1;
    repeat {
        for each resource type k = 1,2,...,n_res {
            Determine candidate operations U_l,k;
            Determine unfinished operations T_l,k;
            Select S_k ⊆ U_l,k vertices, s.t. |S_k| + |T_l,k| ≤ a_k;
            Schedule the S_k operations at step l;
        }
        l = l + 1;
    }
    until (v_n is scheduled) ;
    return (t);
}
```

Force-Directed Scheduling Algorithm for Minimum Resources

- **Operations considered one a time for scheduling**
- **For each iteration**
  - Time frames, probabilities and forces computed
  - Operation with least force scheduled

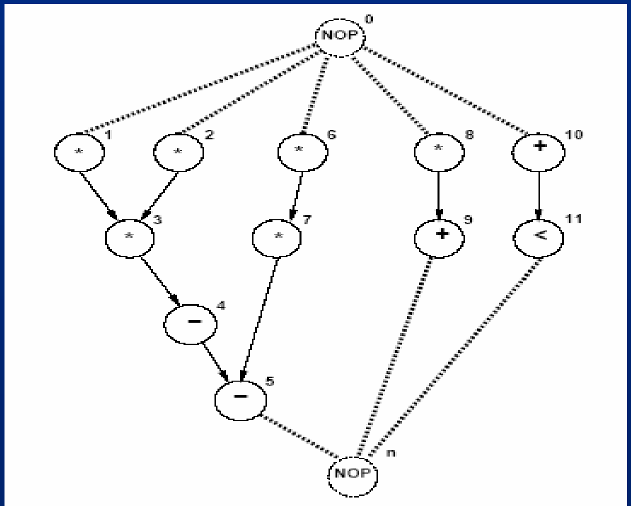
```
FDS( G(V,E), λ̄ ) {
    repeat {
        Compute the time-frames;
        Compute the operation and type probabilities;
        Compute the self-forces, p/s-forces and total forces;
        Schedule the op. with least force, update time-frame;
    } until (all operations are scheduled)
    return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

Scheduling Algorithms for Extended Sequencing Models

- **For hierarchical sequencing graphs, scheduling performed bottom up.**
- **Computed start times are relative to source vertices in corresponding graph entities.**
- **Timing and resource-constrained scheduling is not straightforward.**
- **Simplifying assumptions**
  - No resource can be shared across different graph entities in hierarchy.
  - Timing and resource constraints apply within each graph entity.
  - Schedule each graph entity independently.

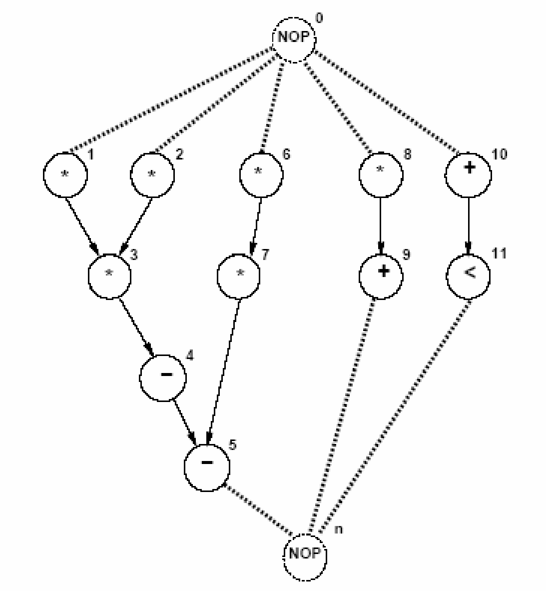
Scheduling Graphs with Alternative Paths

- **Assume sequencing graph has alternative paths related to branching constructs.**
  - Obtained by expanding branch entities
- **ILP formulation**
  - Resource constraints need to express that operations in alternative paths can be scheduled in same time step without affecting resource usage.
- **Example**
  - Assume that path (v_0, v_8, v_9, v_n) is mutually exclusive with other operations.



Scheduling Graphs with Alternative Paths

- **Resource constraints**
  - $x_{1,1}+x_{2,1}+x_{6,1} - a_1 \leq 0$
  - $x_{3,2}+x_{6,2}+x_{7,2} - a_1 \leq 0$
  - $x_{10,2}+x_{11,2} - a_2 \leq 0$
  - $x_{4,3}+x_{10,3}+x_{11,3} - a_2 \leq 0$
  - $x_{5,4}+x_{11,4} - a_2 \leq 0$



- **List scheduling and force-directed scheduling algorithms can support mutually exclusive operations**
  - By modifying way resource usage computed.