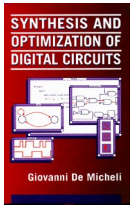
Scheduling

Giovanni De Micheli
Integrated Systems Centre
EPF Lausanne



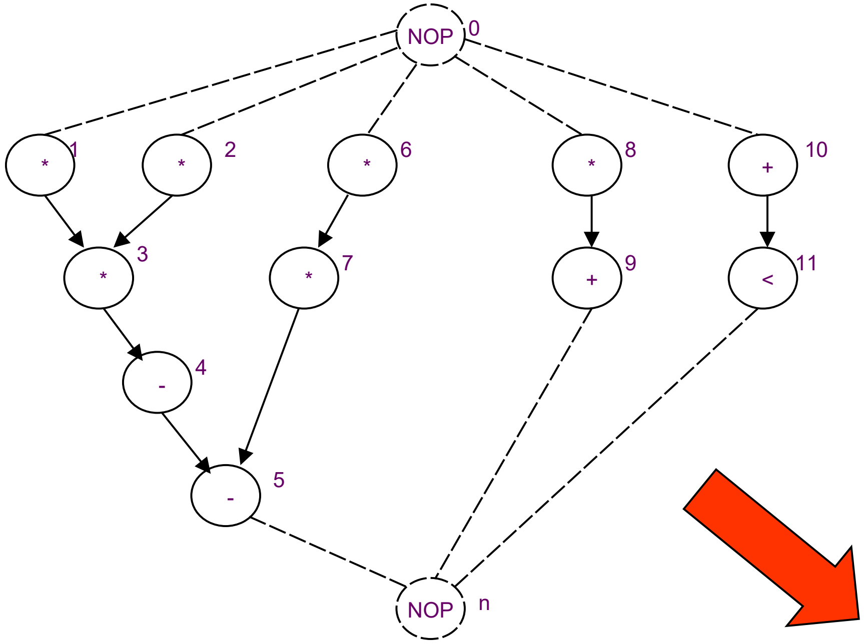
Module 1

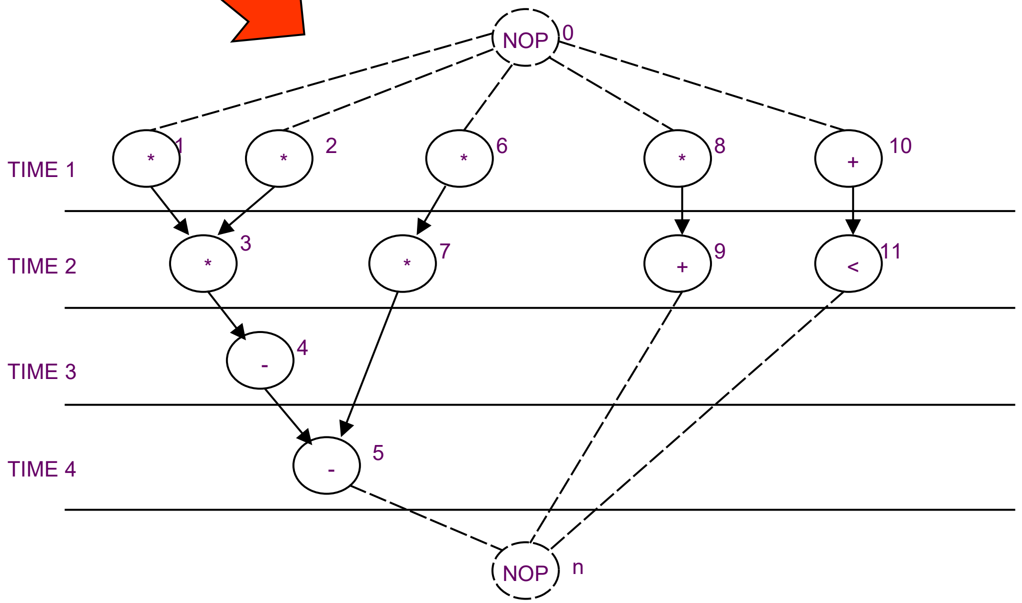
- **Objectives:**
  - **The scheduling problem**
    - **Case analysis**
  - **Scheduling without constraints**
  - **Scheduling with timing constraints**

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


NOP 0
* 1
* 2
* 6
* 8
+ 10
* 3
* 7
+ 9
< 11
- 4
- 5
NOP n


NOP 0
TIME 1
* 1
* 2
* 6
* 8
+ 10
TIME 2
* 3
* 7
+ 9
< 11
TIME 3
- 4
TIME 4
- 5
NOP n

Taxonomy

- Unconstrained scheduling
- Scheduling with timing constraints:
  - Latency
  - Detailed timing constraints
- Scheduling with resource constraints
- Related problems:
  - Chaining
  - Synchronization
  - Pipeline scheduling

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

Minimum-latency unconstrained scheduling problem

- Given a set of ops *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:

$t_i = \varphi(v_i),$

$t_i \geq t_j + d_j \qquad \forall\, i, j \text{ s.t. } (v_j , v_i) \in E$

and t_n is minimum

ASAP scheduling algorithm

ASAP ($G_s(V,E)$) {

Schedule v_0 by setting $t_0 = 1$;

repeat {

Select a vertex v_i whose predecessors are all scheduled;

Schedule v_i by setting $t_i = \max_{j:(v_j,v_i) \in E} t_j + d_j$;

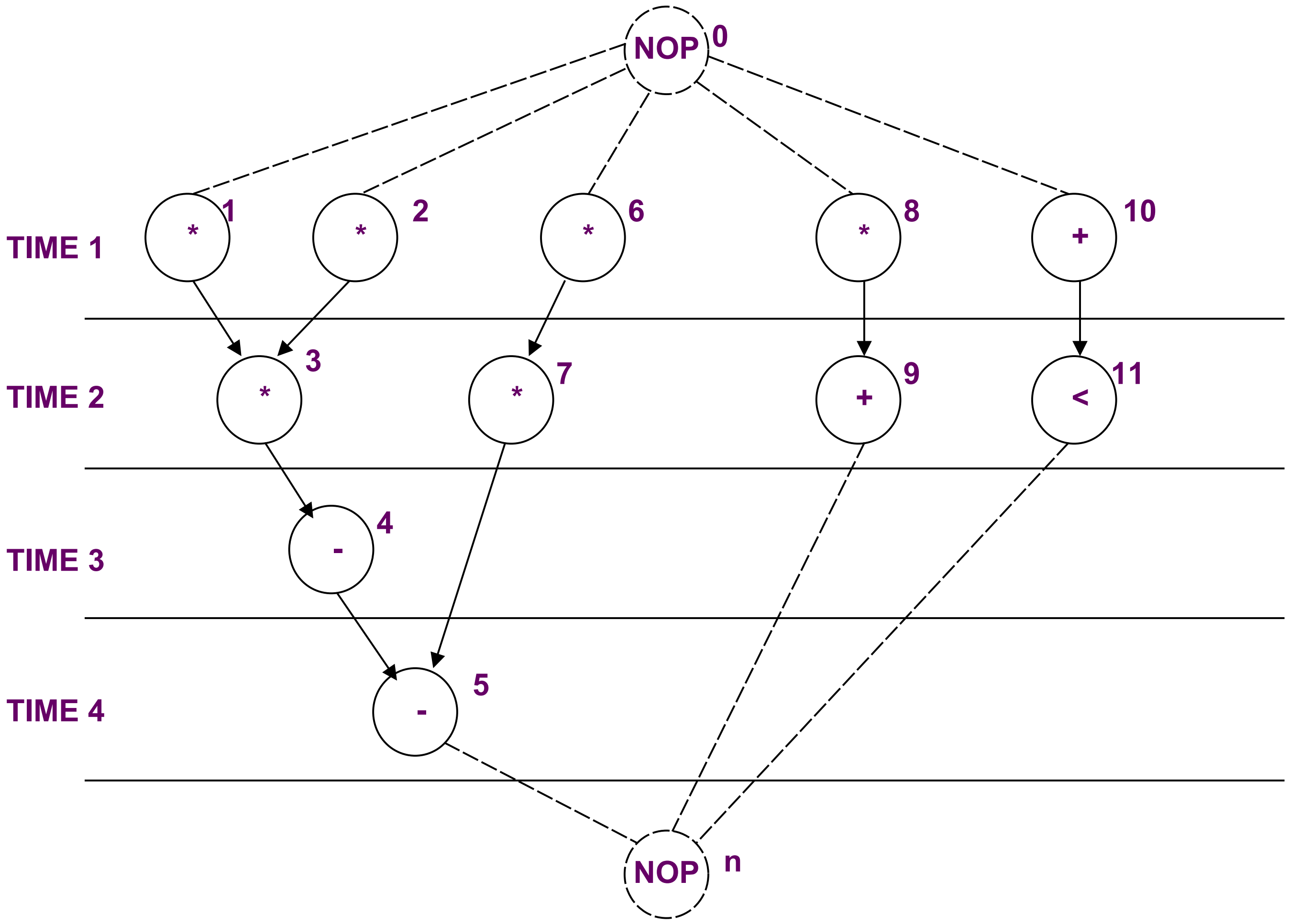
}

until (v_n is scheduled);

return (t);

}

Example



ALAP scheduling algorithm

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

Schedule v_n by setting $t_n = \overline{\lambda} + 1$;

repeat {

Select a vertex v_i whose successors are all scheduled;

Schedule v_i by setting $t_i = \min_{j:(v_i,v_j) \in E} t_j - d_i$;

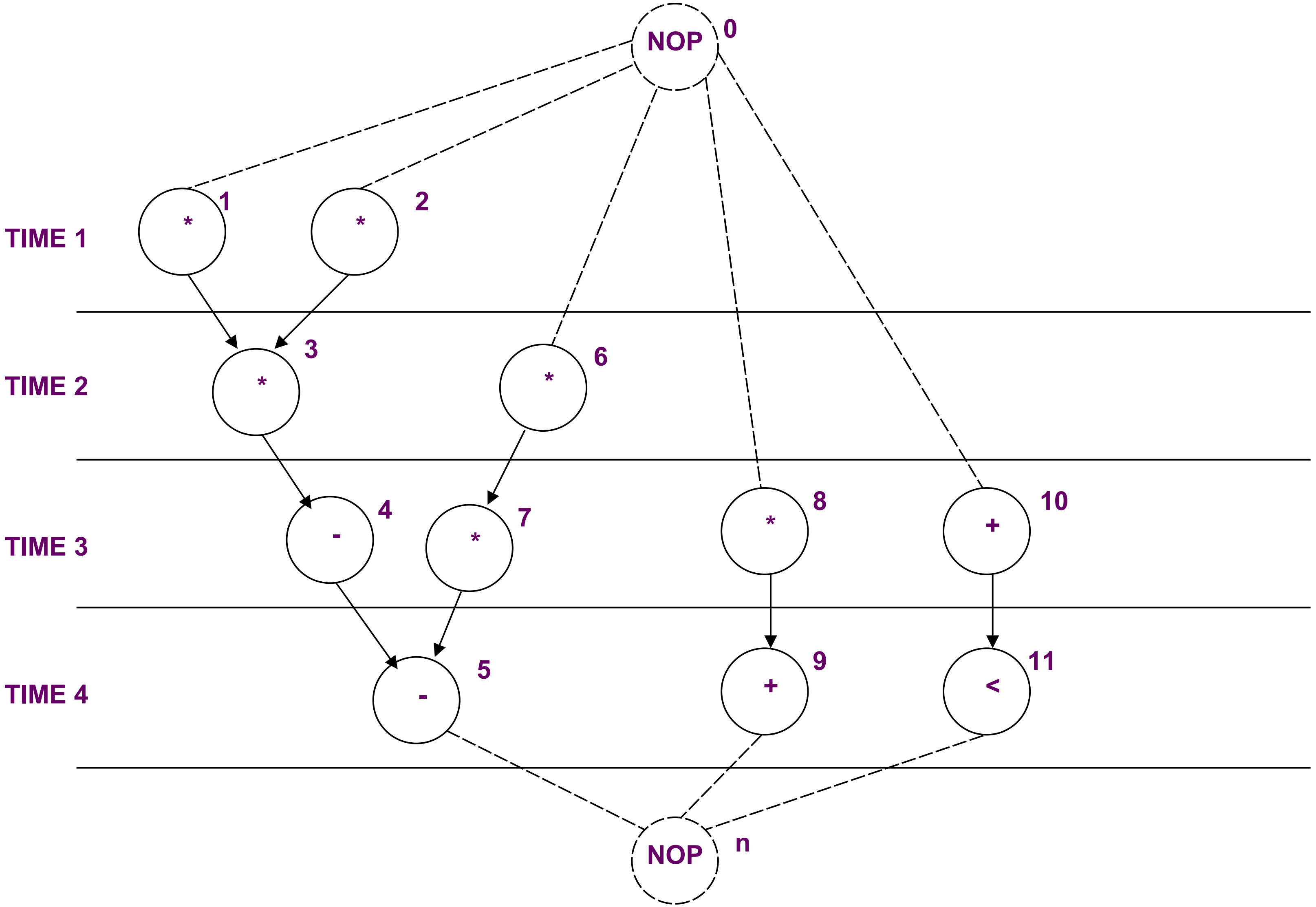
}

until (v_0 is scheduled);

return (t);

}

Example

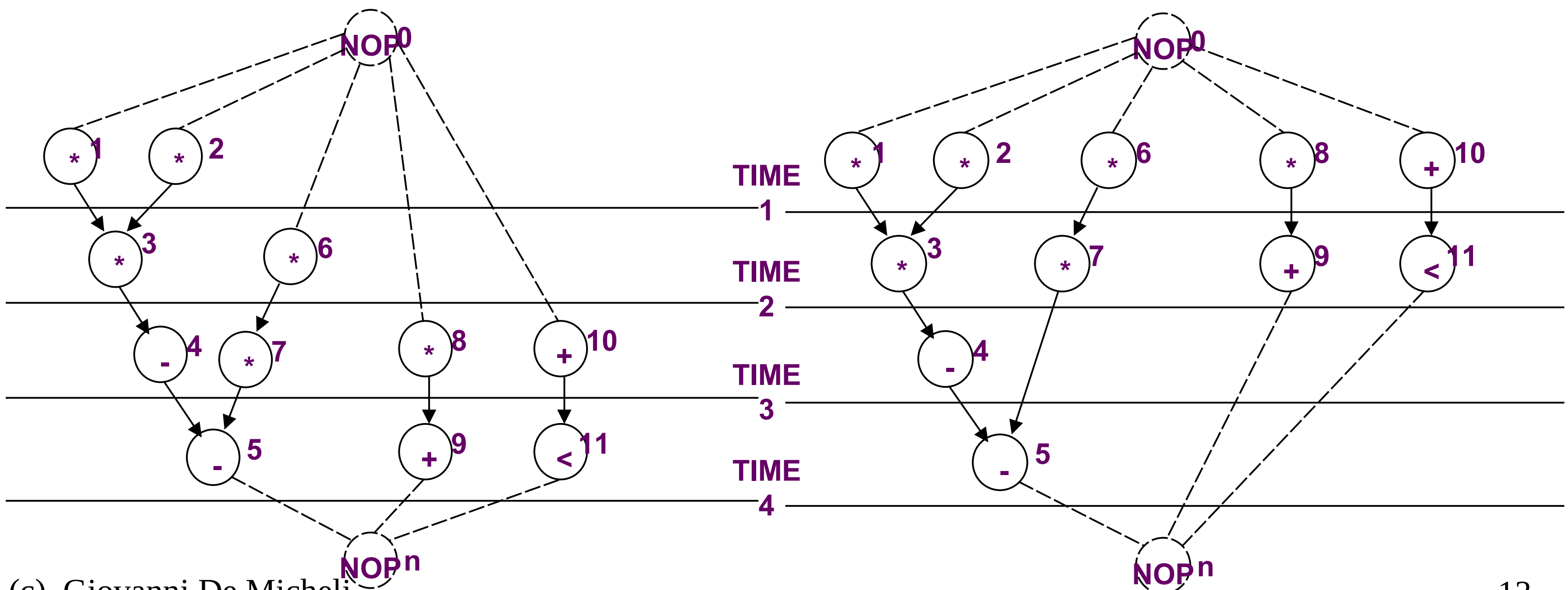


Remarks

- 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

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} }



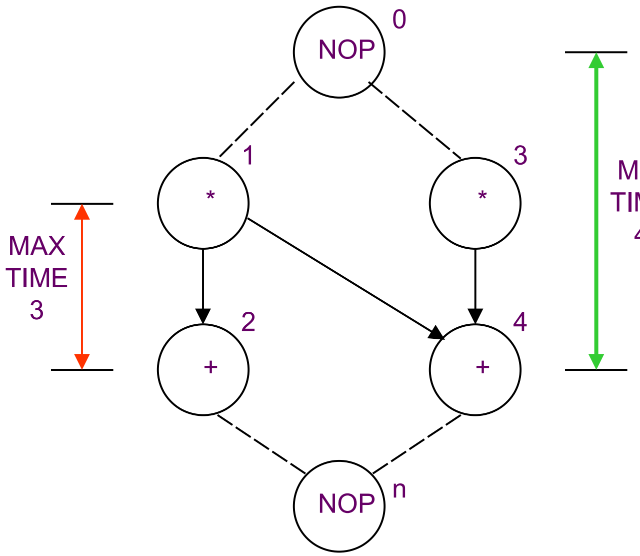
Scheduling under detailed timing constraints

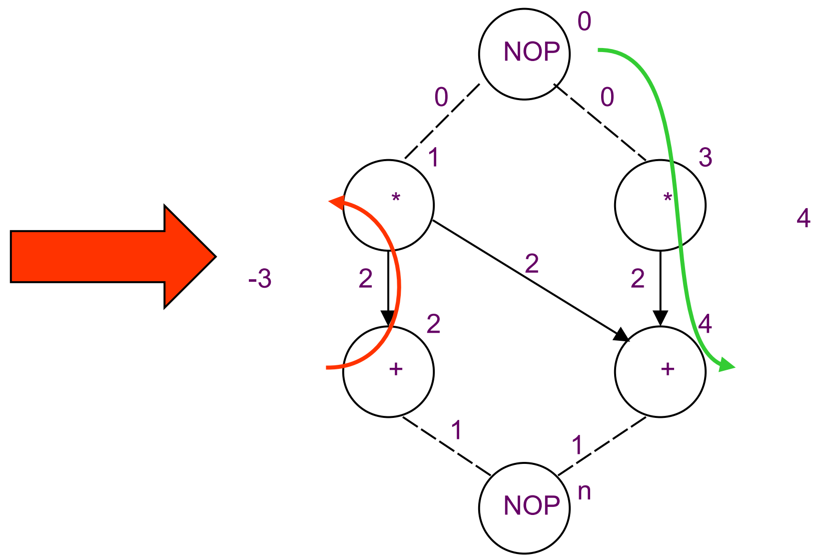
- **Motivation:**
  - Interface design
  - Control over operation start time
- **Constraints:**
  - Upper/lower bounds on start-time difference of any operation pair
- **Feasibility of a solution**

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:**
  - **That is, for constraint from v_i to v_j add backward edge (v_j, v_i) with weight: $-u_{ij}$**
    - **because $t_j \leq t_i + u_{ij} \rightarrow t_i \geq t_j - u_{ij}$**

Example





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

- **Assumption:**
  - **All delays are fixed and known**
- **Set of linear inequalities**
- **Longest *path* problem**
- **Algorithms:**
  - **Bellman-Ford, Liao-Wong**
- **Extensions:**
  - **Unbounded delays, relative scheduling**

Module 2

- **Objectives:**
  - **Scheduling with resource constraints**
  - **Exact formulation:**
    - **ILP**
    - **Hu's algorithm**
  - **Heuristic methods**
    - **List scheduling**
    - **Force-directed scheduling**

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 $\{ a_k;\ k = 1, 2, \ldots, n_{res} \}$ on resource usage:
- **Find an integer labeling of the operation** $\varphi : V \to z^{+}$
  such that :

  $t_i = \varphi(v_i),$

  $t_i \geq t_j + d_j \quad \forall\, i,j \text{ s.t. } (v_j, v_i) \in E,$

  $| \{v_i \mid T(v_i) = k$ **and** $t_i \leq l < t_j + d_j \} | \leq a_k$ **for all types** $k = 1,2,\ldots,n_{res}$
  and steps l

  and t_n ***is minimum***

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:**

  $X = \{ x_{il}, \quad i = 1,2,\ldots.\ n; \ \ l = 1,2,\ldots, \overline{\lambda} + 1 \}$

  x_{il} **is** TRUE **only when operation** v_i **starts in step** l **of the schedule (i.e.** $l = t_i$ **)**

  $\overline{\lambda}$ **is an upper bound on latency**

- **Start time of operation** v_i **:** $\quad \Sigma_l \ l \cdot x_{il}$

ILP formulation constraints

- **Operations start only once**

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

- **Sequencing relations must be satisfied**

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

  $\sum_l l \cdot x_{il} - \sum_l l \cdot x_{jl} - d_j \geq 0$ *for all* $(v_j, v_i) \in E$

- **Resource bounds must be satisfied**

  Simple case (unit delay)

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

ILP Formulation

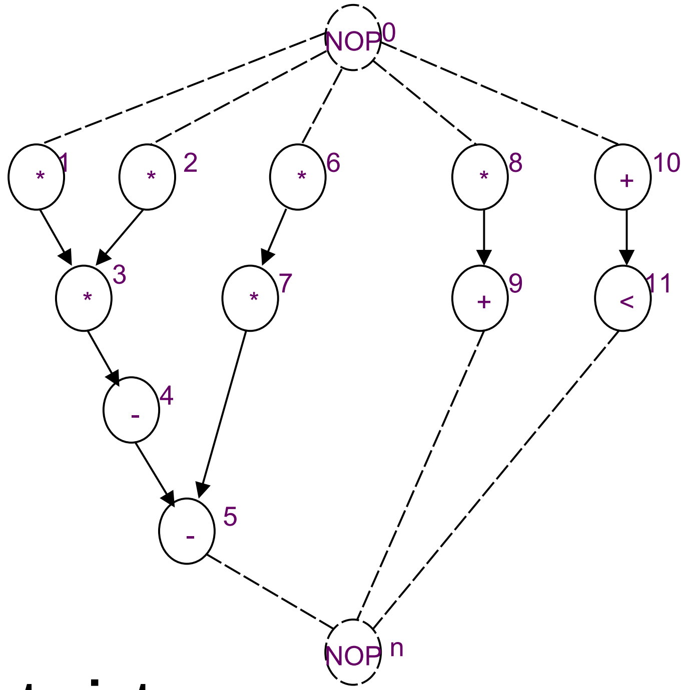
$$\min \|\mathbf{t}\| \text{ such that}$$

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

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

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

Example



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

Example

- **Operations start only once**

  $x_{11} = 1$

  $x_{61} + x_{62} = 1$

  ...

- **Sequencing relations must be satisfied**

  $x_{61} + 2x_{62} - 2x_{72} - 3x_{73} + 1 \leq 0$

  $2x_{92} + 3x_{93} + 4x_{94} - 5x_{N5} + 1 \leq 0$

  ...

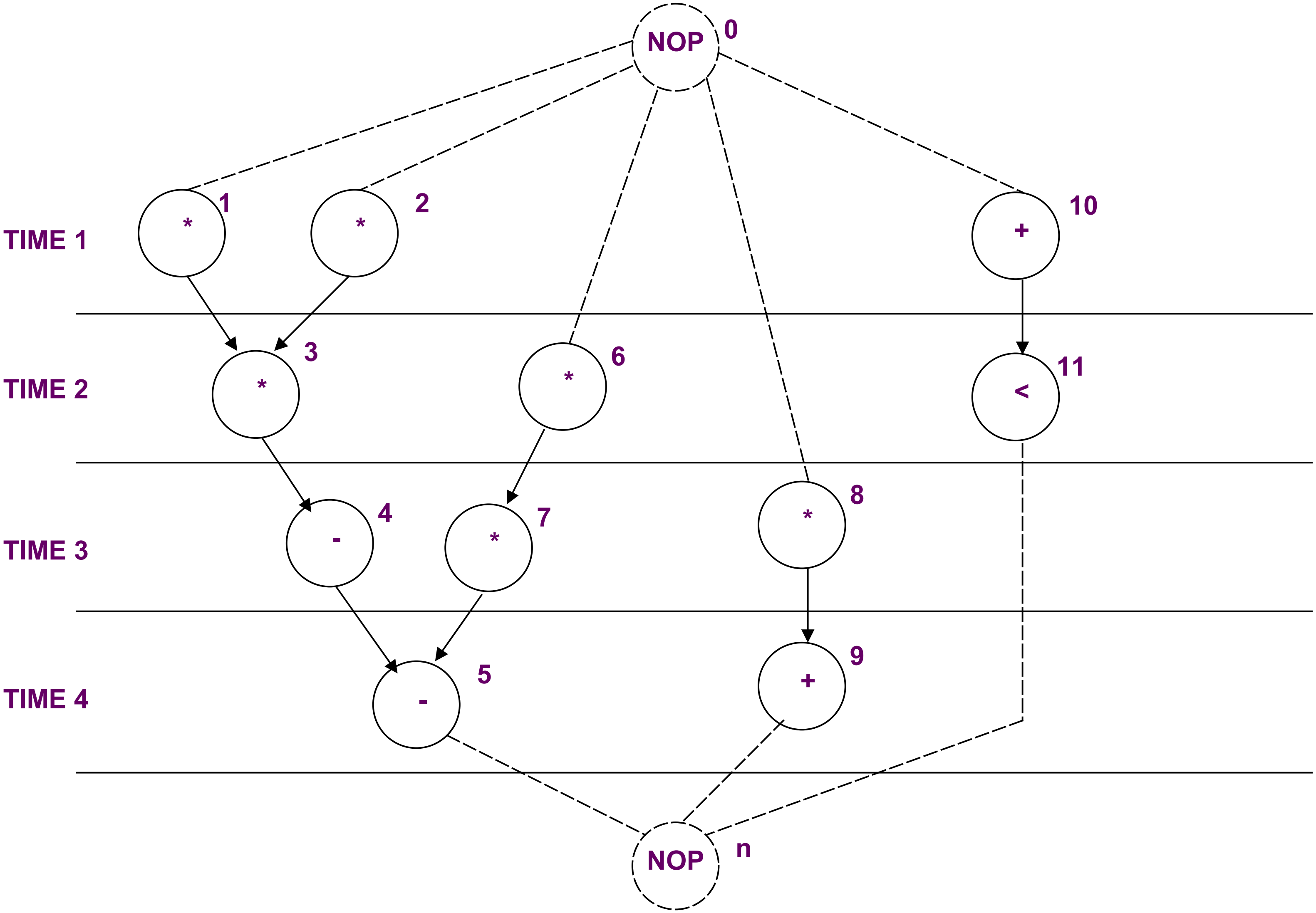
- **Resource bounds must be satisfied**

  $x_{11} + x_{21} + x_{61} + x_{81} \leq 2$

  $x_{32} + x_{62} + x_{72} + x_{81} \leq 2$

  ...

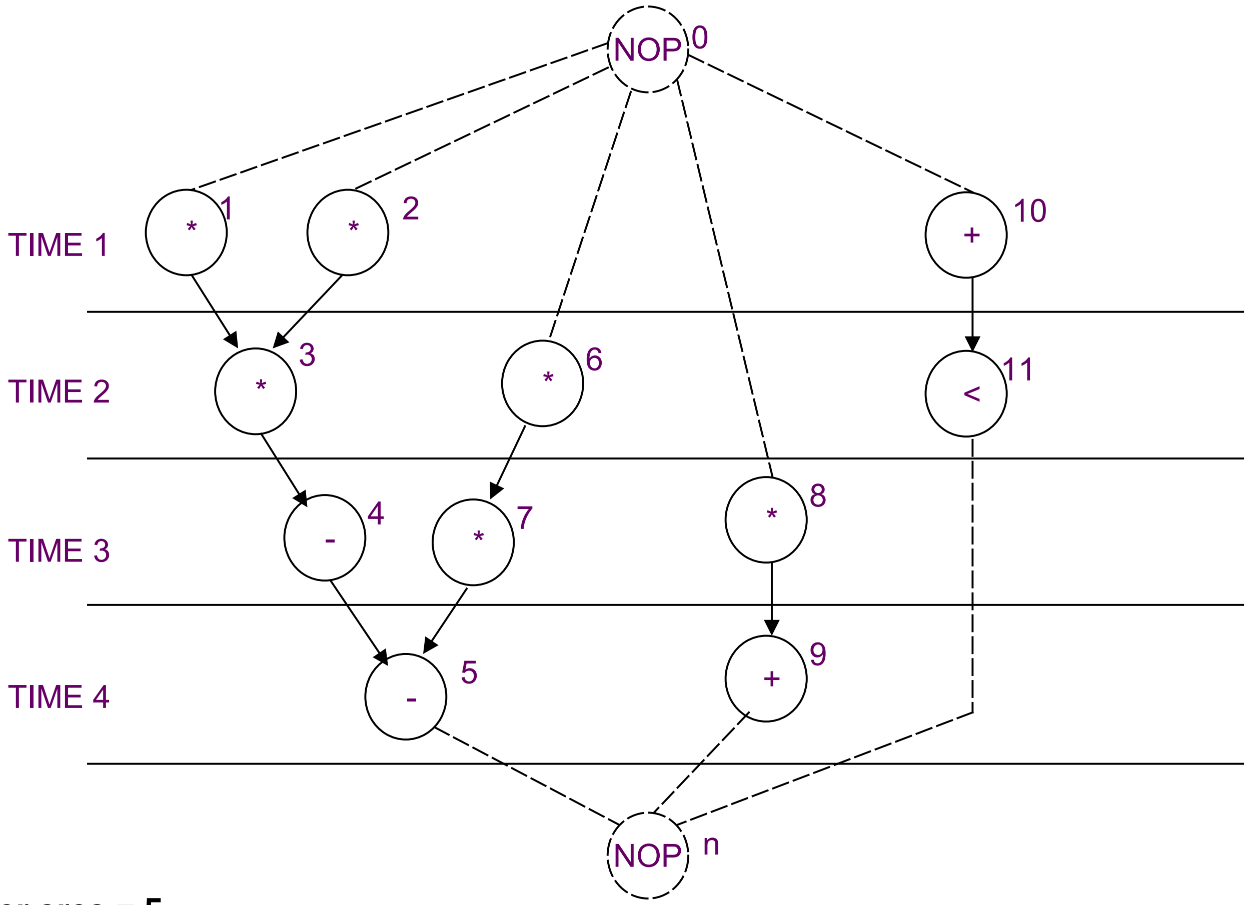
Example



Dual ILP formulation

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

Example



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

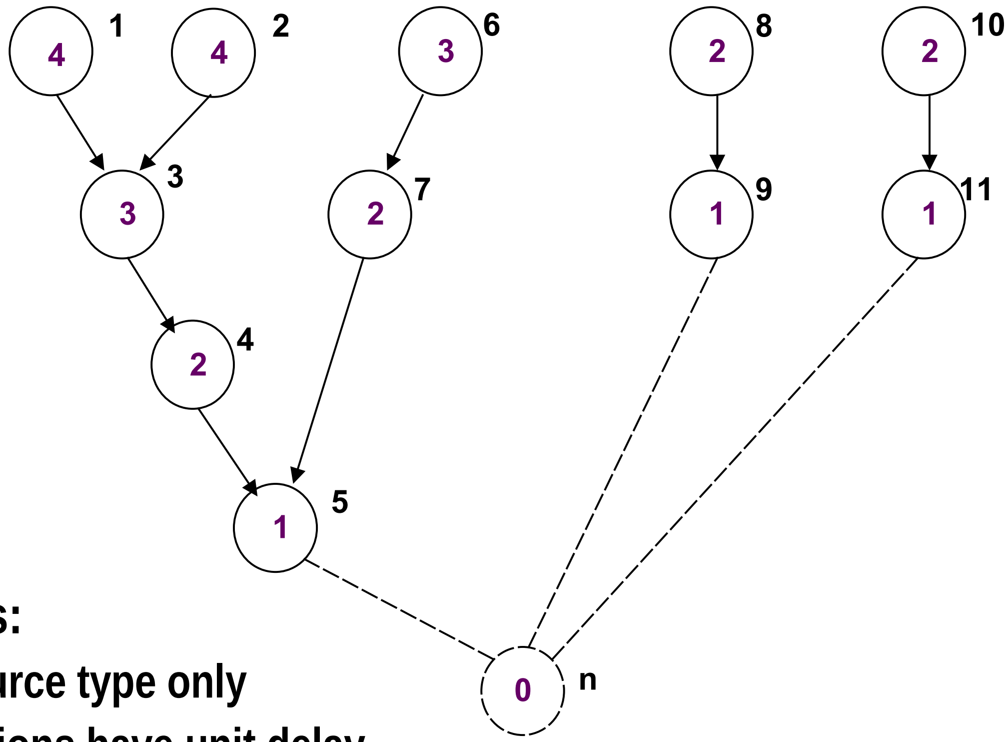
ILP Solution

- 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

- **Assumptions:**
  - Graph is a forest
  - All operations have unit delay
  - All operations have the same type
- **Algorithm:**
  - Greedy strategy
  - Exact solution

Example



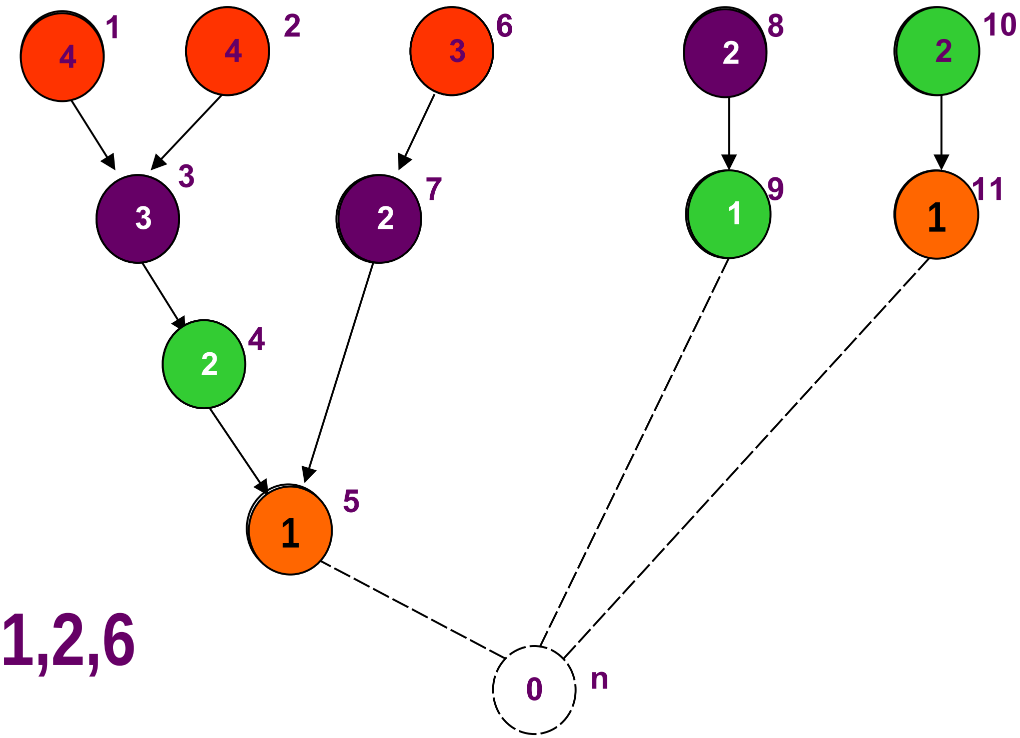
- **Assumptions:**
  - One resource type only
  - All operations have unit delay
- **Labels:**
  - Distance to sink

Algorithm
Hu's schedule with $\bar{a}$ resources

- Label operations with distance to sink
- Set step $l = 1$
- Repeat until all ops are scheduled:
  - Select $s \leq \bar{a}$ resources with
    - All predecessors scheduled
    - Maximal labels
  - Schedule the s operations at step l
  - Increment step $l = l + 1$

Example

$\bar{a} = 3$



Step 1: Op 1,2,6

Step 2: Op 3,7,8

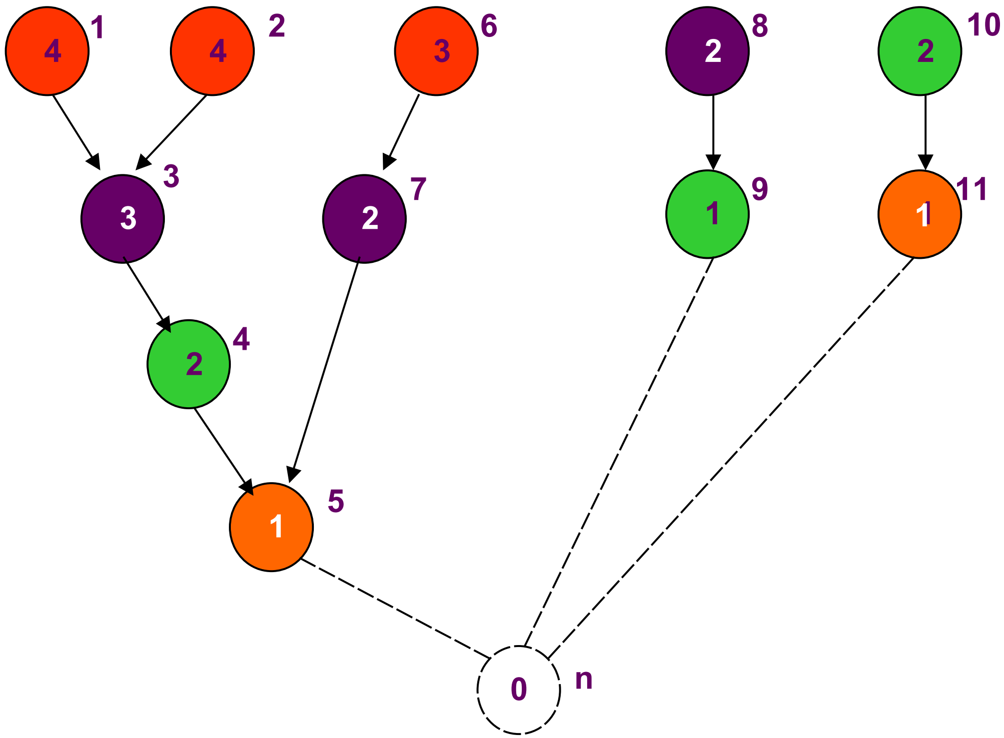
Step 3: Op 4,9,10

Step 4: Op 5,11

Exactness of Hu's algorithm

- **Definitions:**
  - Label of vertex v_i is called α_i
  - Maximal label is called α
  - Number of vertices with label b is called $p(b)$
  - Latency is called λ
  - A lower bound on the number of resources to complete a schedule with latency λ is called $\bar{a}$

Example



$\alpha = 4$

p (4) = 2

p (3) = 2

p (2) =4

p (1) =3

Exactness of Hu's algorithm$_{\gamma}$

- **Theorem1:**
  - **Given a dag with operations of the same type**
  - $\bar{a}$ **= max** $\left\lceil \dfrac{\sum_{j=1}^{\gamma} p(\alpha + 1 - j)}{\gamma + \lambda - \alpha} \right\rceil$
  - $\bar{a}$ **is a lower bound on the number of resources to complete a schedule with latency** λ
  - γ **is a positive integer**
- **Theorem2:**
  - **Hu's algorithm applied to a tree with** $\bar{a}$ **unit-cycle resources achieves latency** λ
- **Corollary:**
  - **Since** $\bar{a}$ **is a lower bound on the number of resources for achieving** λ**, then** λ **is minimum**

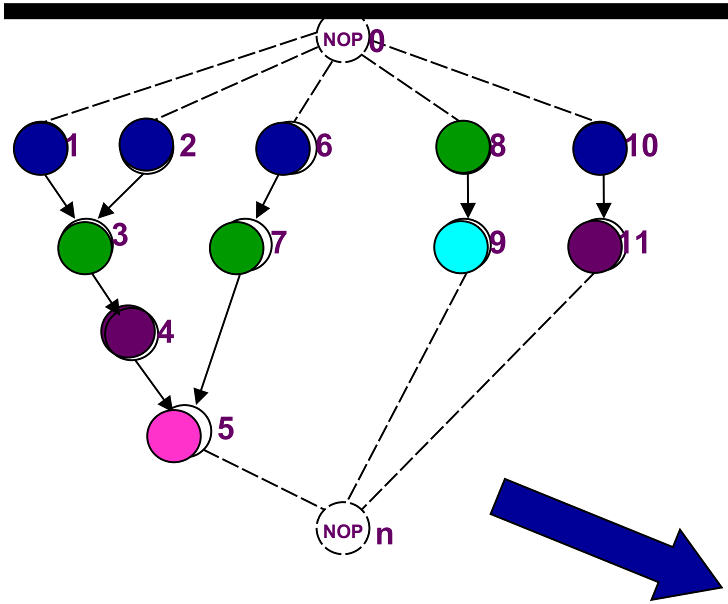
List scheduling algorithms

- **Heuristic method for:**
  - **Min *latency* subject to *resource bound***
  - **Min *resource* subject to *latency bound***
- **Greedy strategy (like Hu's)**
- **General graphs (unlike Hu's)**
- **Priority list heuristics**
  - **Longest path to sink**
  - **Longest path to timing constraint**

List scheduling algorithm for minimum latency

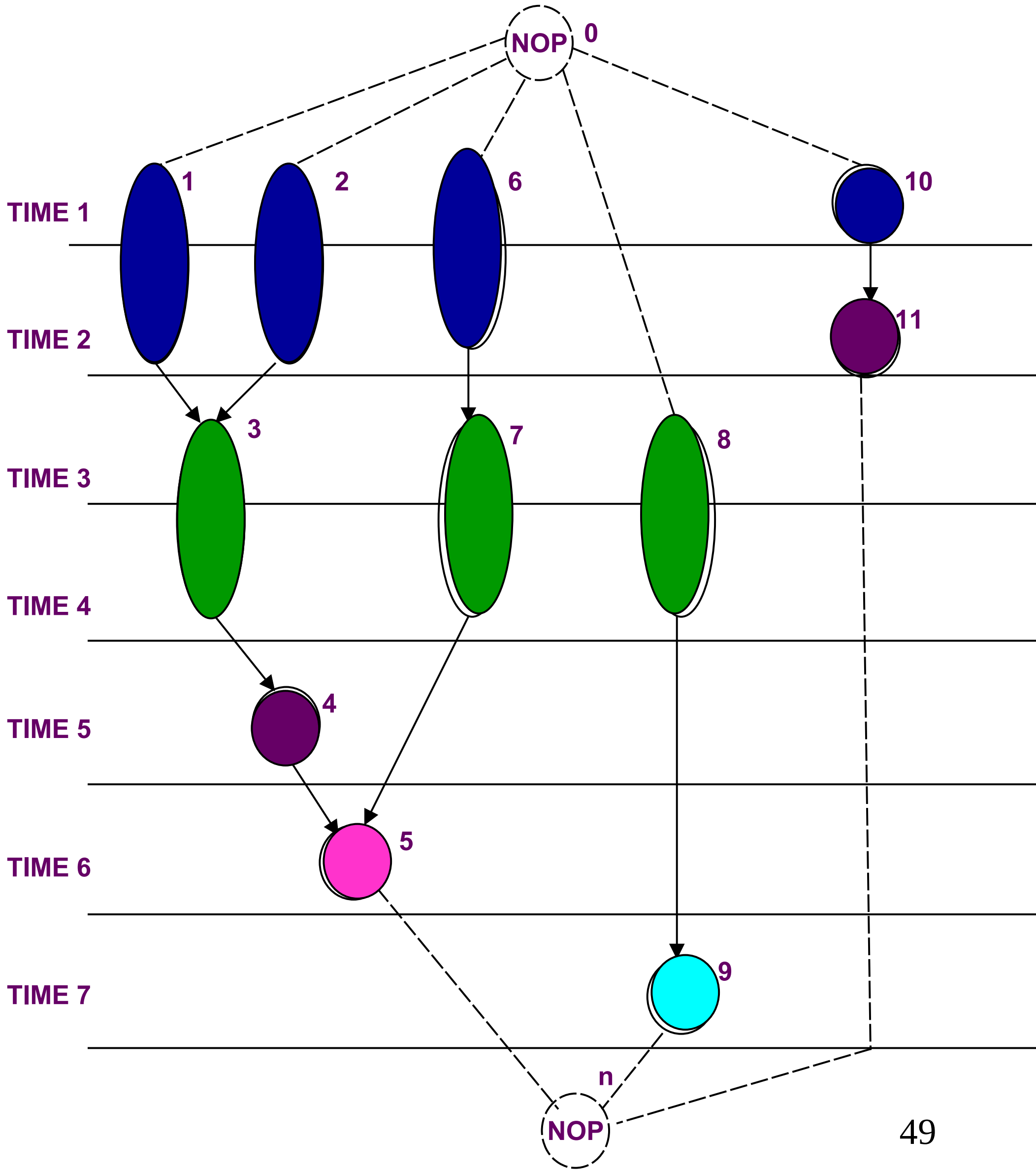
```
LIST_L( G(V, E), a) {
    l = 1;
    repeat {
    for each resource type k = 1, 2, …, n_res {
       Determine ready operations U_l,k;
       Determine unfinished operations T_l,k;
       Select S_k ⊆ U_l,k vertices, s.t. |S_k| + |T_l,k| ≤ a_k;
       Schedule the S_k operations at step l;
     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

Example



Resource bounds:

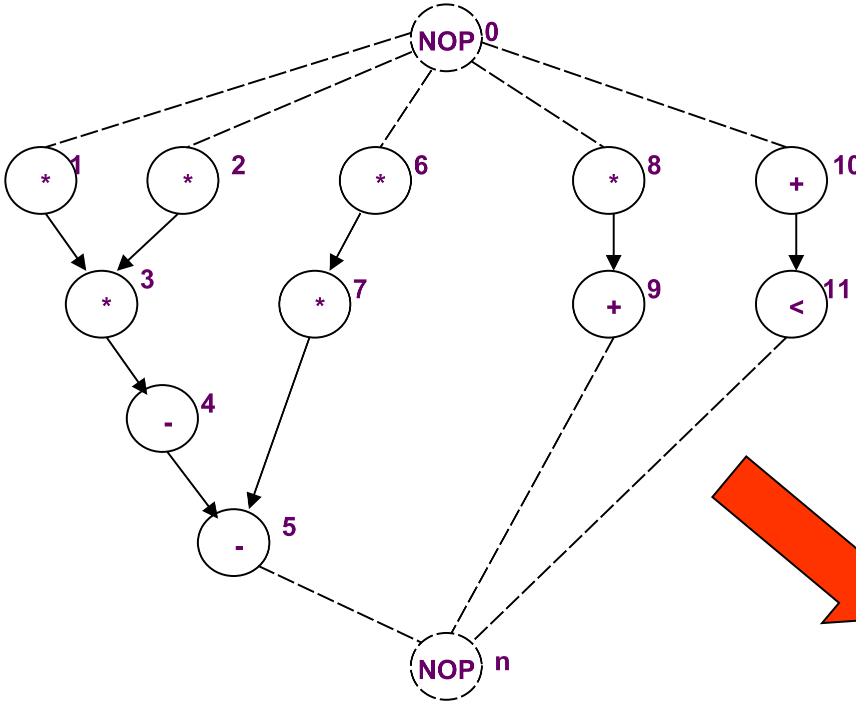
- **3 multipliers with delay 2**
- **1 ALU with delay 1**



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 < 0)
      return (∅);
    l = 1;
    repeat {
            for each resource type k = 1, 2, …, n_res {
              Determine ready operations U_l,k;
              Compute the slacks { s_i = t_i^L – l    for all v_i ∊ U_lk};
              Schedule the candidate operations with zero slack and update a;
              Schedule the candidate operations not needing additional resources;
              }
            l = l + 1;
    }
    until (v_n is scheduled) ;
    return (t, a);
}
```

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

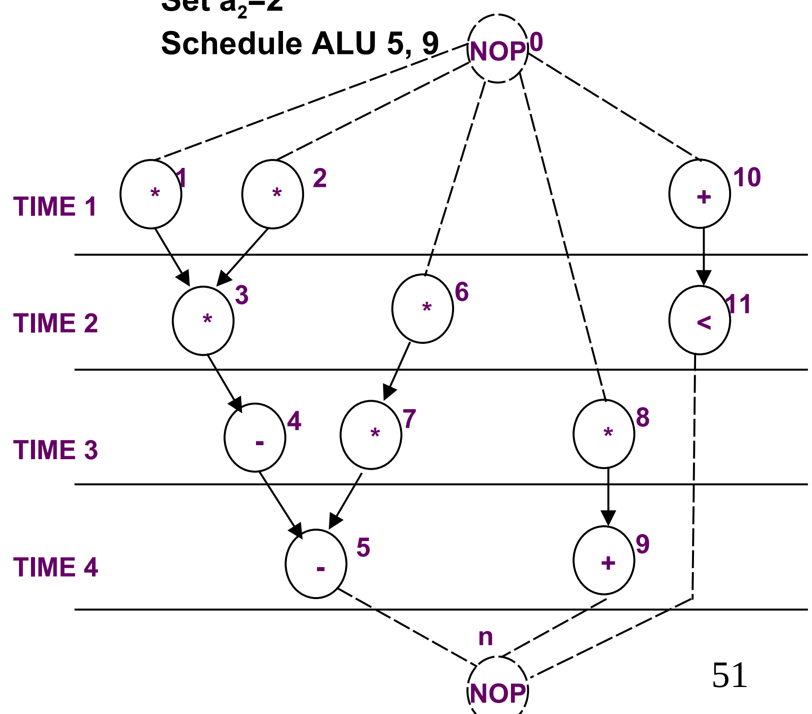
Schedule Mult 7,8

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

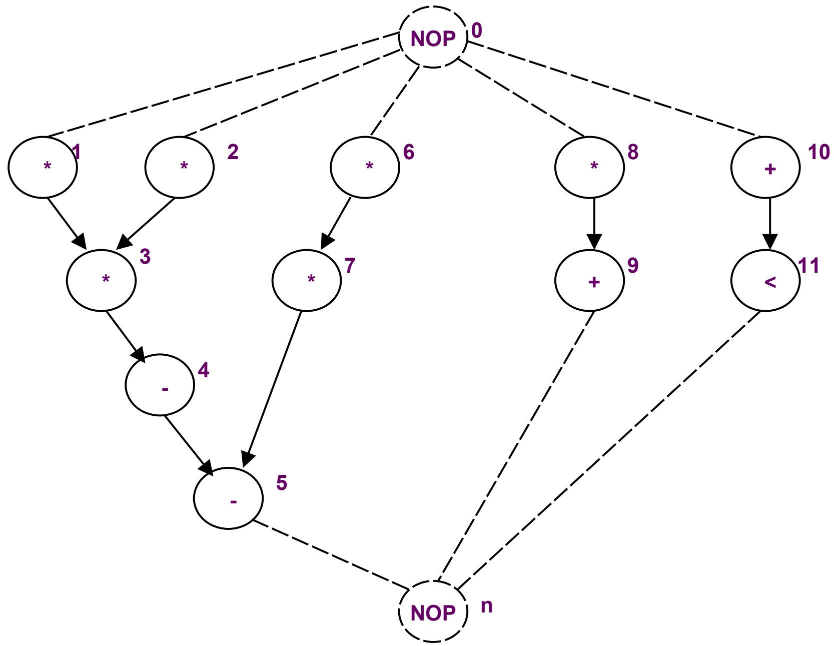
Force-directed scheduling

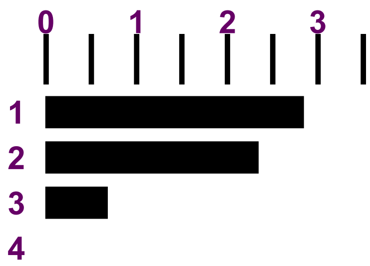
- **Heuristic scheduling methods [Paulin]:**
  - **Min *latency* subject to *resource bound***
    - ***Variation* of list scheduling : FDLS**
  - **Min *resource* subject to *latency bound***
    - **Schedule one operation at a time**
- **Rationale:**
  - **Reward *uniform distribution* of operations across schedule steps**

Force-directed scheduling definitions

- Operation *interval*:
  - Mobility plus one $(\mu_i + 1)$
  - Computed by ASAP and ALAP scheduling $[\, t^S , t^L]$
- Operation *probability* $p_i\,(l)$:
  - Probability of executing in a given step
  - $1/(\mu_i + 1)$ inside interval; 0 elsewhere
- *Operation-type* distribution $q_k\,(l)$:
  - Sum of the operation probabilities for each type

Example





- Distribution graphs for multiplier and ALU

Force

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

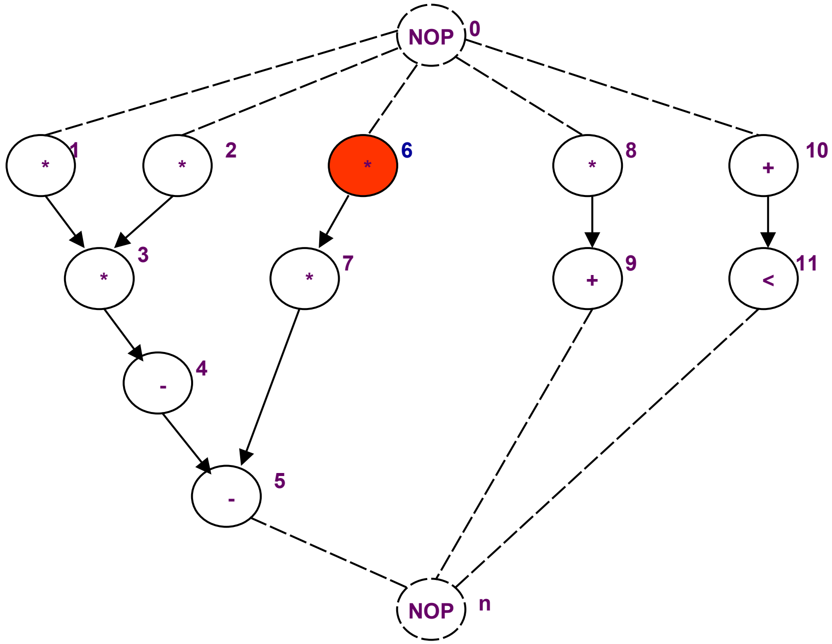
Forces related to the assignment of an operation to a control step

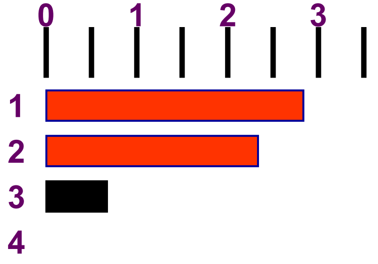
- **Self-force:**
  - Sum of forces to feasible schedule steps
  - Self-force for operation v_i in step l

$$\sum_{m \text{ in interval}} q_k(m)\,(\delta_{lm} - p_i(m))$$

- **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

Example
Schedule operation v_6





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

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**

Force-directed scheduling algorithm 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)**

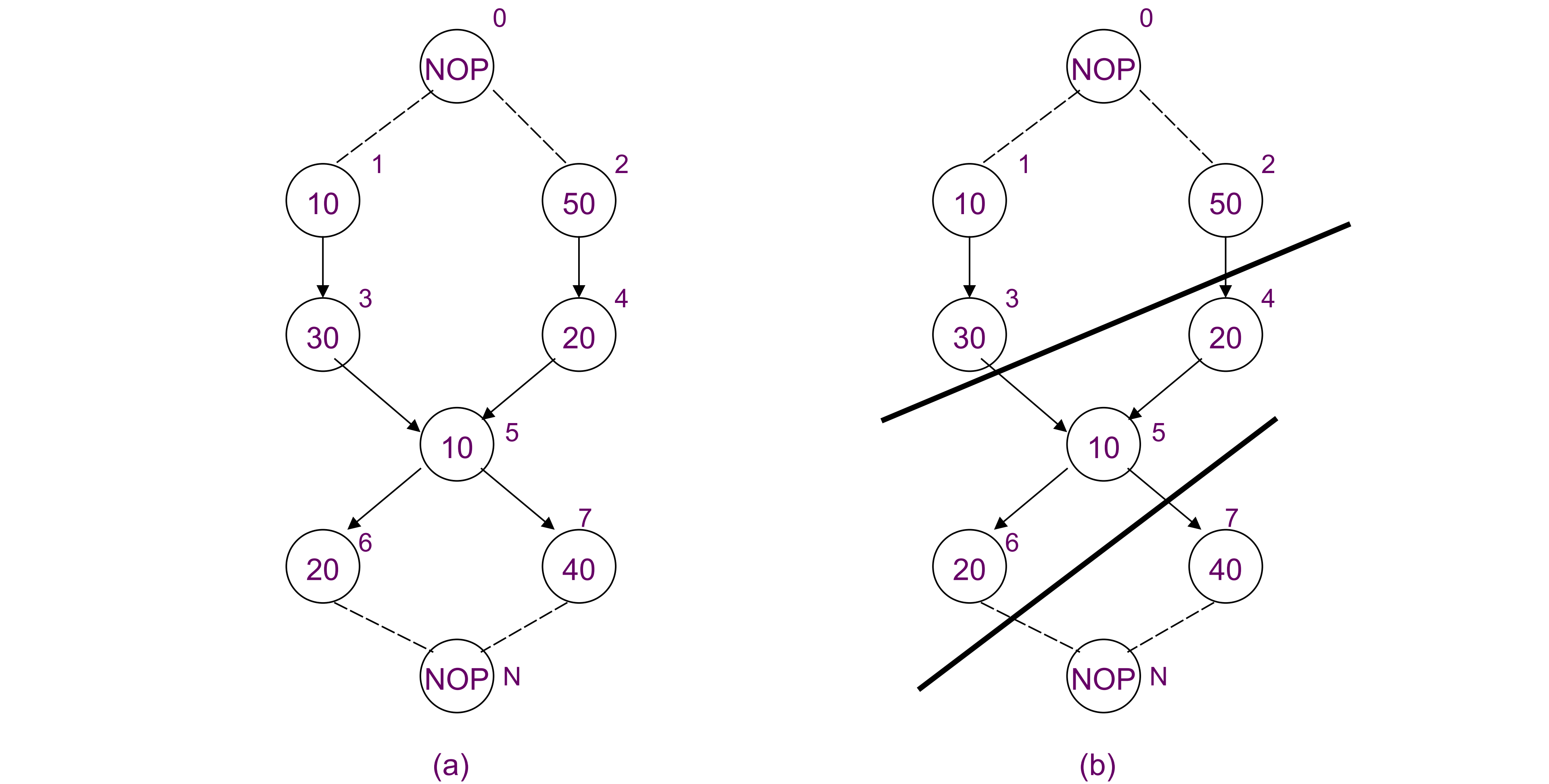
return (t)**;**

}

Scheduling and chaining

- Consider propagation delays of resources not in terms of cycles
- Use scheduling to *chain* multiple operations in the same control step
- Useful technique to explore effect of *cycle-time* on area/latency trade-off
- Algorithms:
  - ILP, ALAP/ASAP, list scheduling

Example



- **Cycle-time: 60**

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 extensisons
  - Chaining