SCHEDULING II

Stanford University

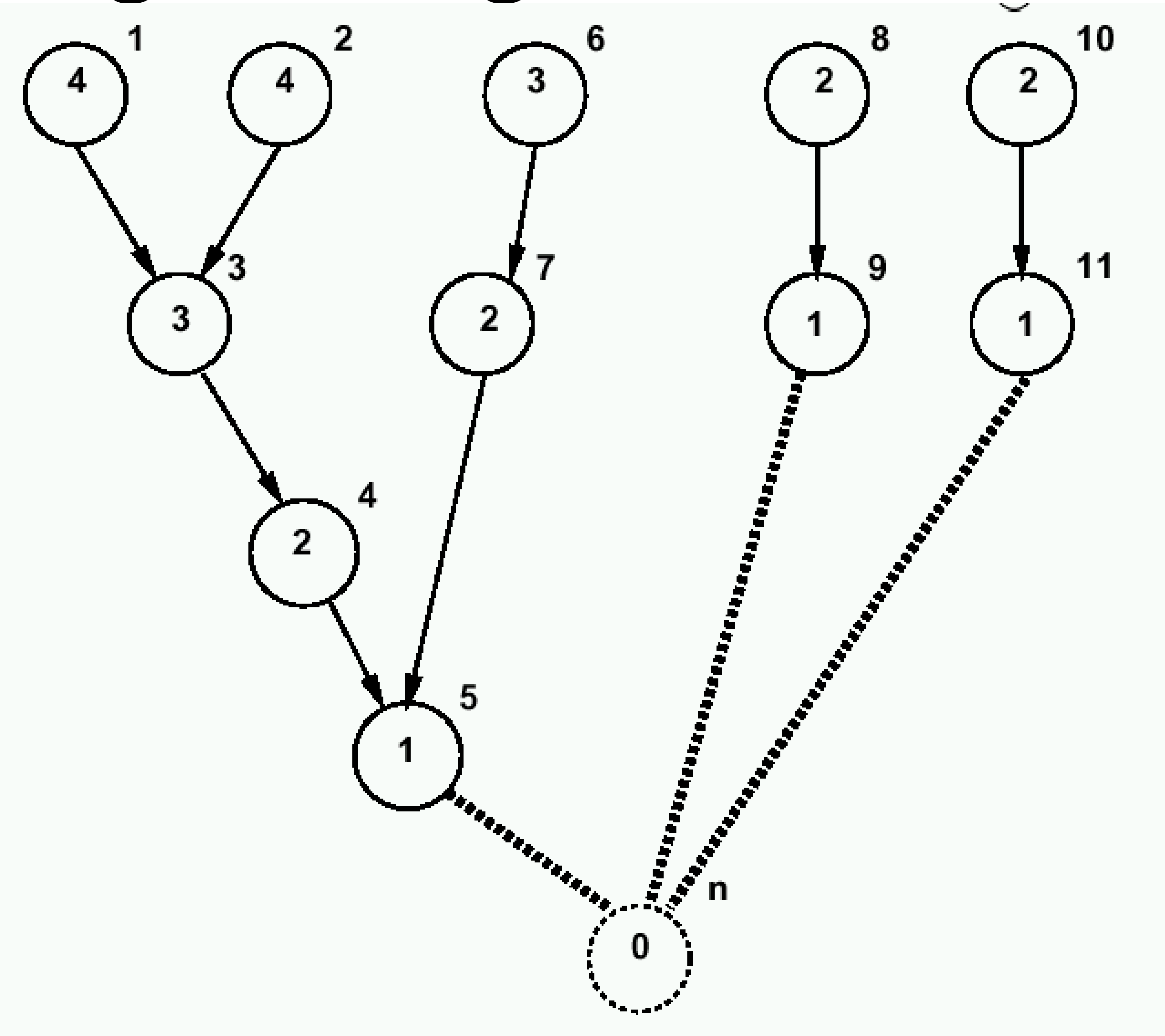
Scheduling under resource constraints

- **Simplified models**:
  - Hu's algorithm.
- **Heuristic algorithms**:
  - List scheduling.
  - Force-directed scheduling.

Hu's algorithm

- **Assumptions:**
  - Graph is a forest.
  - All operations have unit delay.
  - All operations have the same type.
- **Algorithm:**
  - Label vertices with distance from sink.
  - Greedy strategy.
  - Exact solution.

Example of using Hu's algorithm



- <u>Assumptions:</u>
  - **One resource type** only.
  - All operations have **unit delay**.

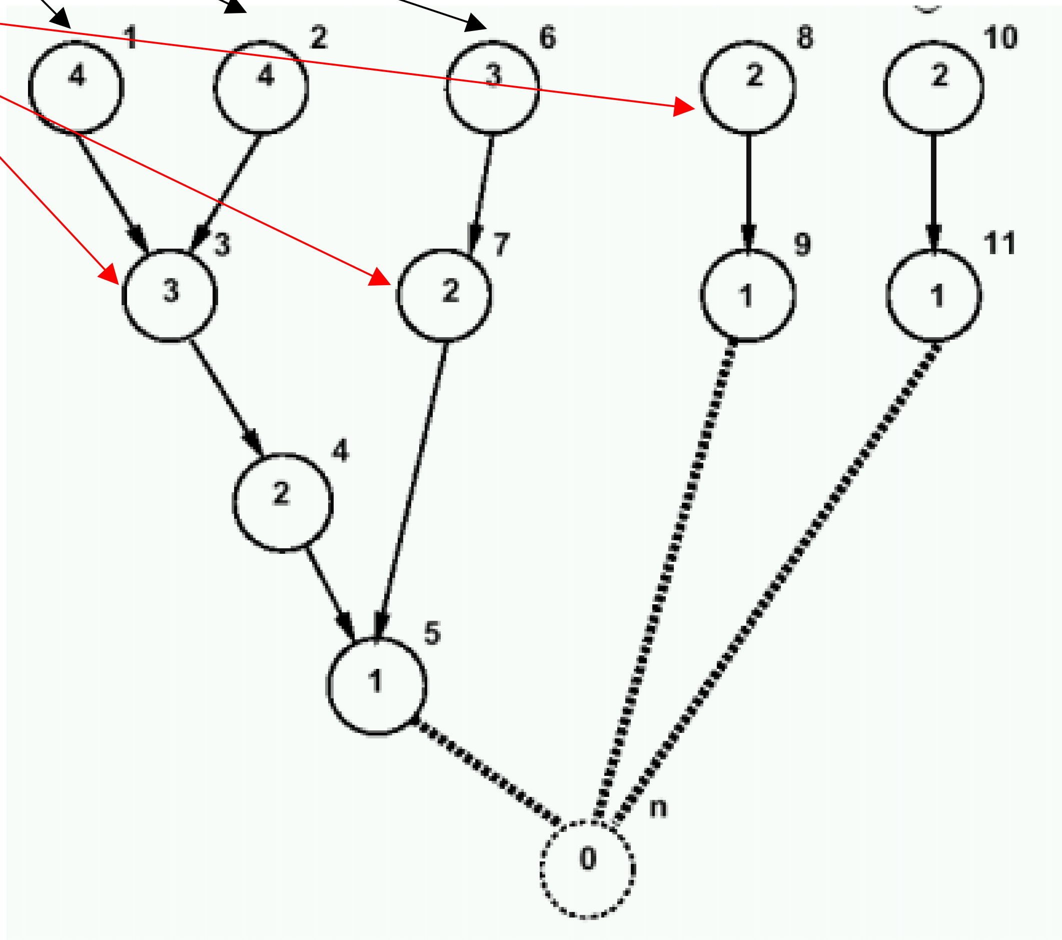
Algorithm
Hu's schedule with a resources

- Set step, l = 1.
- Repeat until all operations are scheduled: –
  - – Select $\mathbf{s} \leq \mathbf{a}$ resources with:
    - All predecessors scheduled.
    - Maximal labels.
  - – Schedule the **s** operations at step l.
  - – Increment step $l = l + 1$.

•All operations have unit delay.
•All operations have the same type.

Algorithm
Hu's schedule with a resources

- Minimum latency with a = 3 resources.
  - Step 1:
    - Select $\{v_1, v_2, v_6\}$.
  - Step 2:
    - Select $\{v_3, v_7, v_8\}$.
  - Step 3:
    - Select $\{v_4, v_9, v_{10}\}$.
  - Step 4:
    - Select $\{v_5, v_{11}\}$.



We always select 3 resources

Exactness of Hu's algorithm

- **Theorem1:**
  - Given a DAG with operations of the same type.

$$\overline{a} = \max_{\gamma} \left\lceil \frac{\sum_{j=1}^{\gamma} p(\alpha + 1 - j)}{\gamma + \lambda - \alpha} \right\rceil$$

- $\overline{a}$ 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 unit-cycle resources achieves latency λ with $\overline{a}$ resources.
- Corollary:
  - Since $\overline{a}$ is a lower bound on the number of resources for achieving λ, then λ is minimum.

List scheduling algorithms

- Heuristic method for:
  - **1.** Minimum ***latency*** subject to resource bound.
  - **2.** Minimum ***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.

1. List scheduling algorithm for minimum latency for resource bound

LIST_L (G(V, E), a) {

$l = 1$;

repeat {

for each resource type $k = 1, 2, \ldots, n_{res}$ {

Determine candidate operations $U_{l,k}$;

Determine unnished operations $T_{l,k}$;

Select $S_k \subseteq U_{l,k}$ vertices, such that $|S_k| + |T_{l,k}| \leq a_k$;

Schedule the S_k operations at step l;

}

$l = l + 1$;

}

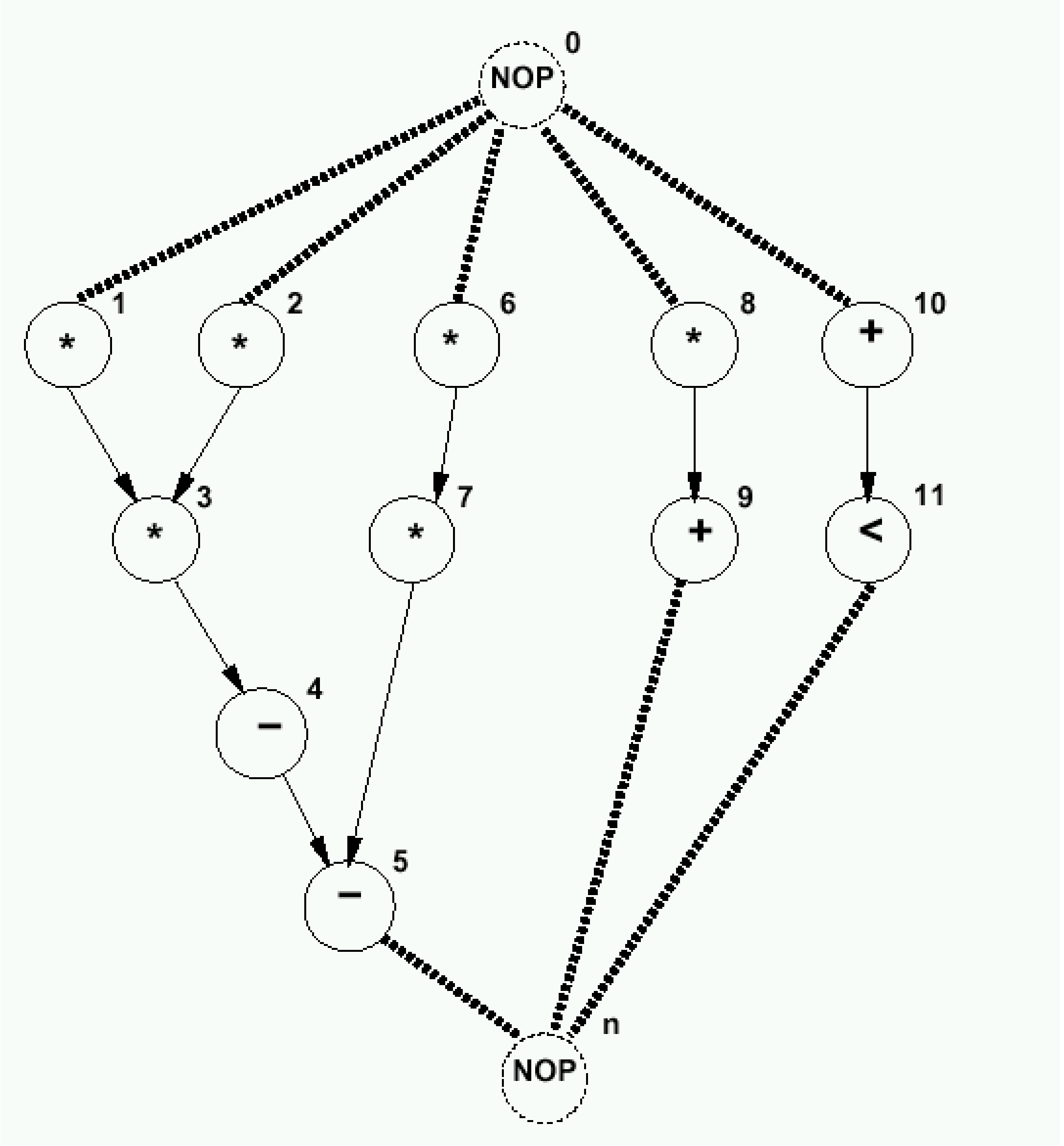
until (v_n is scheduled) ;

return (t);

}

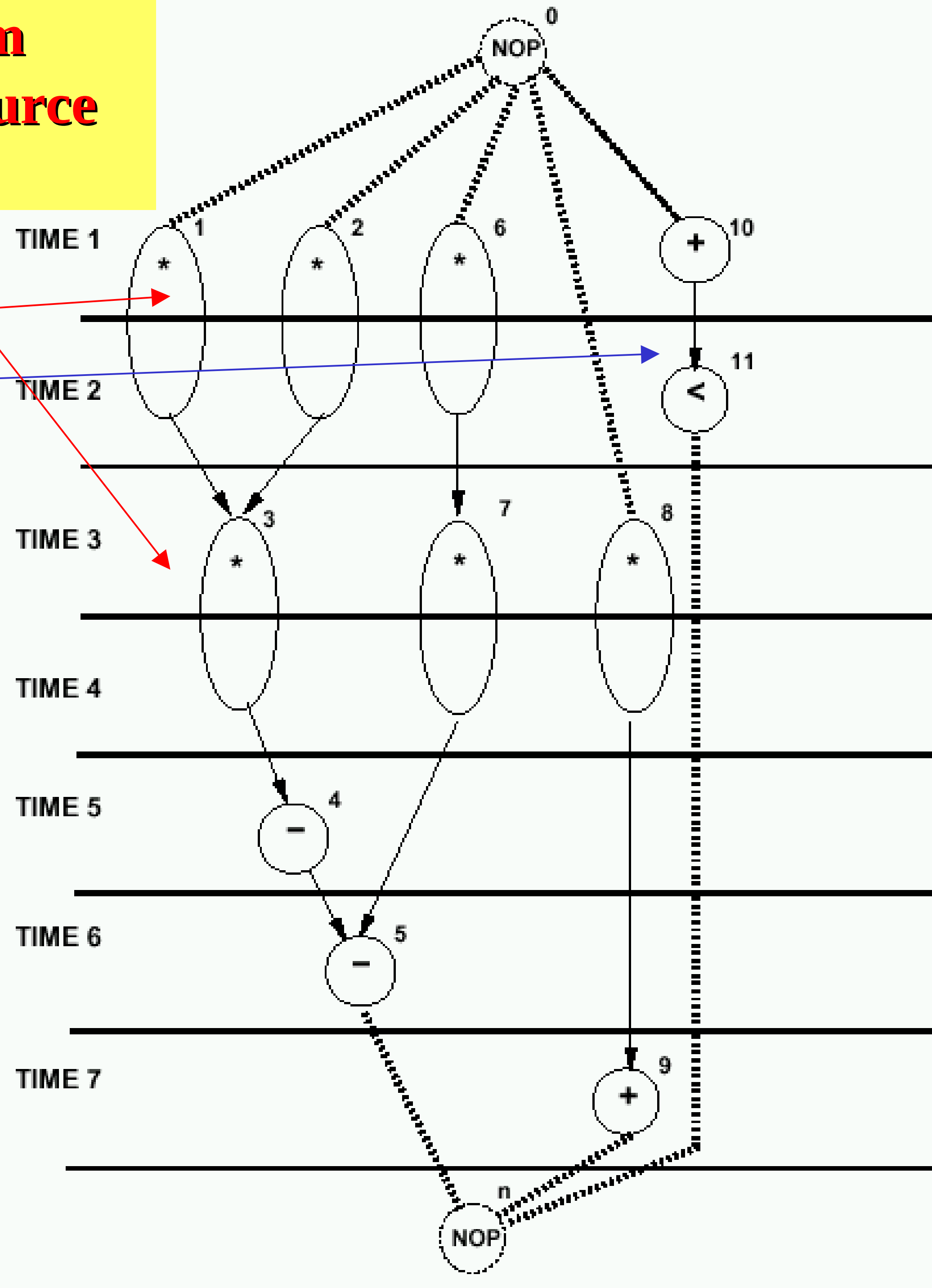
1. List scheduling algorithm for minimum latency for resource bound

- Assumptions:
  - $a_1 = 3$ multipliers with delay 2.
  - $a_2 = 1$ ALUs with delay 1.



1. List scheduling algorithm for minimum latency for resource bound

- **Assumptions:**
  - **a_1 = 3 multipliers with delay 2.**
  - **a_2 = 1 ALUs with delay 1.**



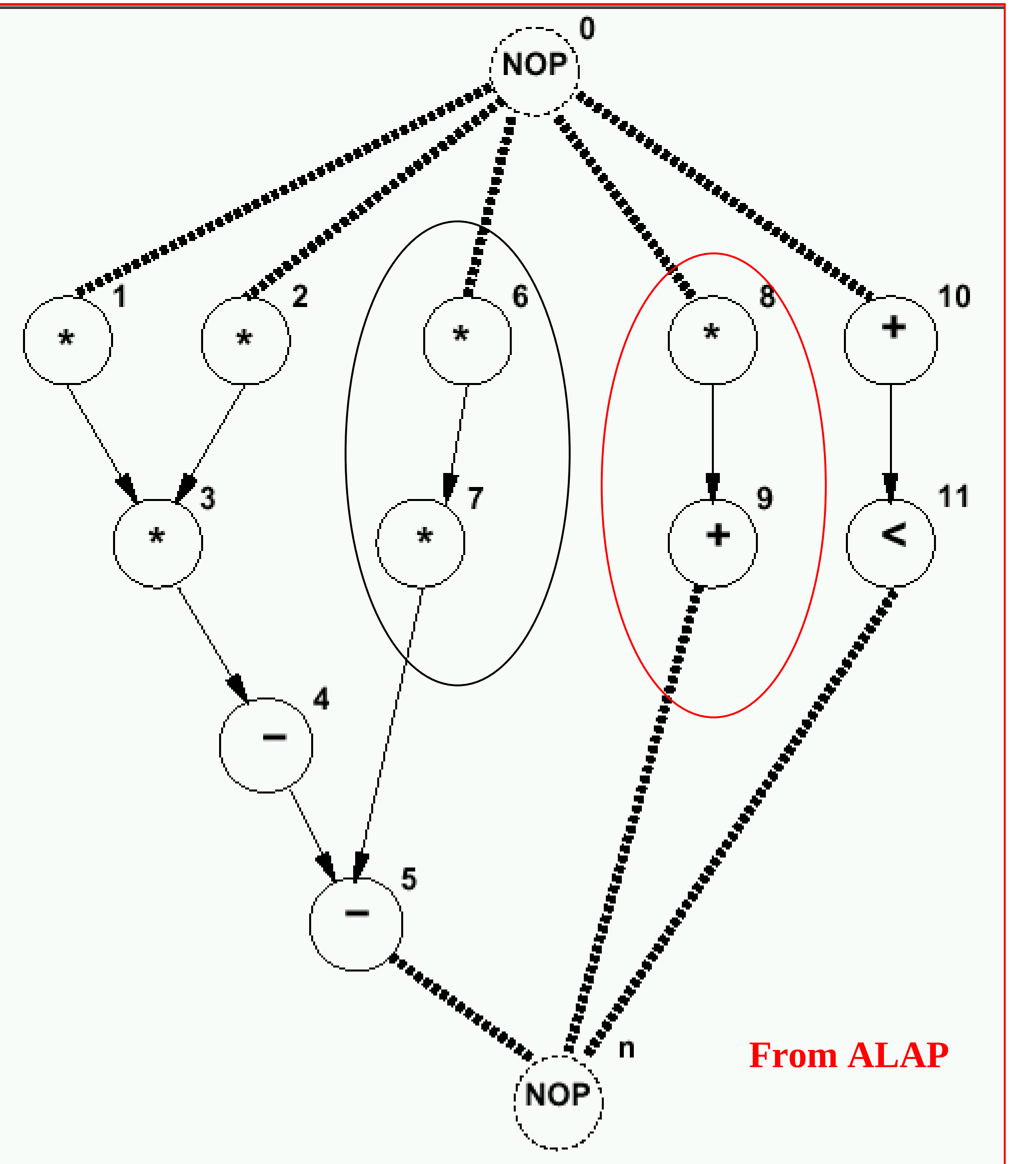
- Solution
- 3 multipliers as assumed
- 1 ALU as assumed
- LATENCY 7

2. List scheduling algorithm for minimum resource usage

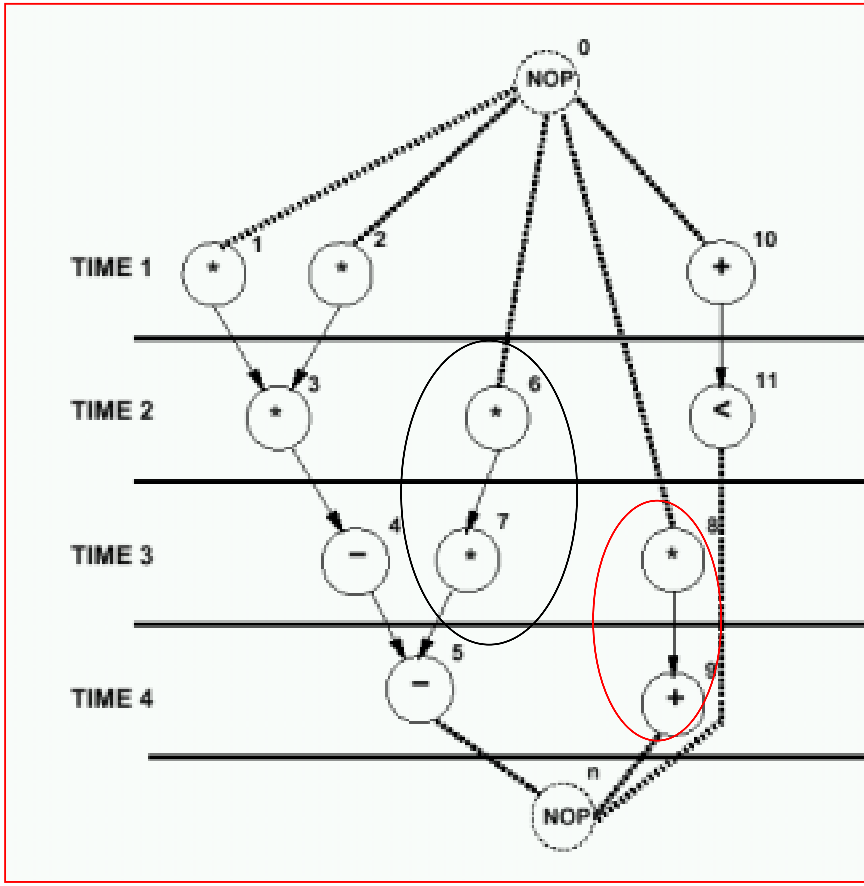
```
LIST_R ( G(V,E), λ̄ ) {
      a = 1 ;
      Compute the latest possible start times t^L by ALAP ( G(V,E), λ̄);
        if ( t^L_0 < 0 )
          return (φ);
      l = 1;
            repeat {
               for each resource type k = 1,2,…, n_res {
                     Determine candidate operations 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 );
}
```

$$\{s_i = t_i^L - l \ \forall v_i \in U_{lk}\}$$

2. Example: List scheduling algorithm for minimum resource usage



From ALAP



Force-directed scheduling

- Heuristic scheduling methods [Paulin]:
  - 1. Minimum ***latency*** subject to ***resource bound***.
    - Variation of **l**ist **s**cheduling: **FDLS**.
  - 2. Minimum ***resource*** subject to ***latency bound***.
    - Schedule one operation at a time.
- Rationale:
  - Reward ***uniform distribution*** of operations across schedule steps.

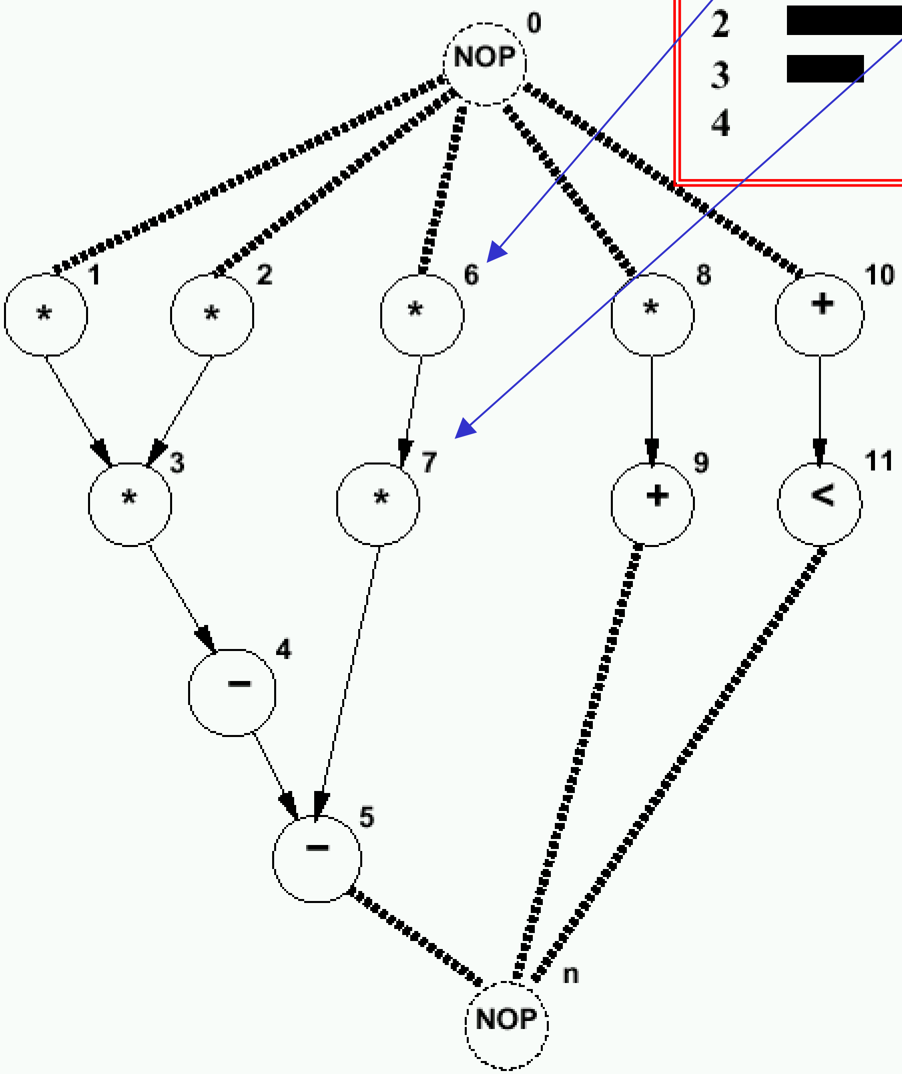
Force-directed scheduling definitions

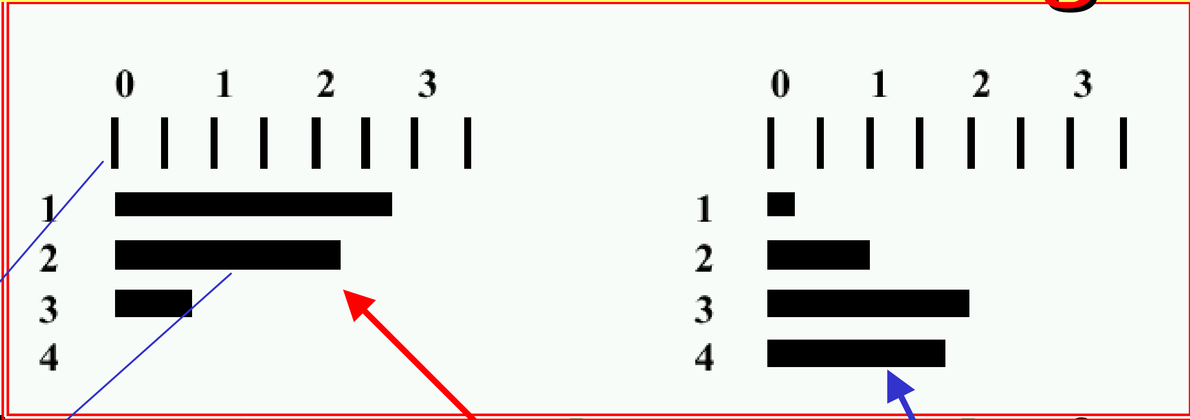
- Operation interval: mobility plus one (μ_i+1).
  - Computed by ASAP and ALAP scheduling

$$[t^S_i , t^L_i].$$

- Operation probability p_i **(l)** :
  - Probability of executing in a given step.
  - **1/ (μ_i +1)** inside interval; **0** elsewhere.
- Operation-type distribution q_k **(l)** :
  - Sum of the operation probabilities for each type.

Example of Force-directed scheduling

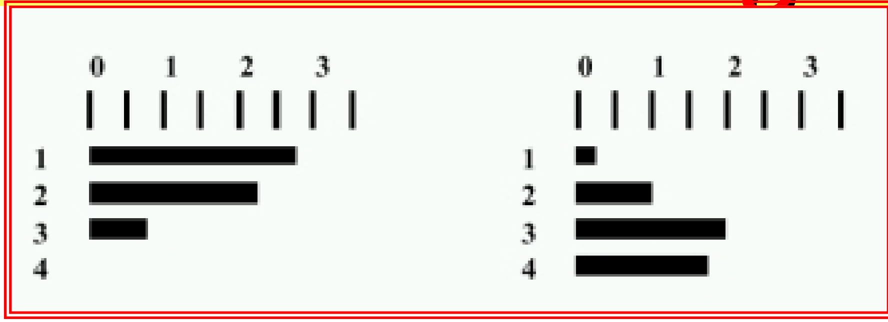




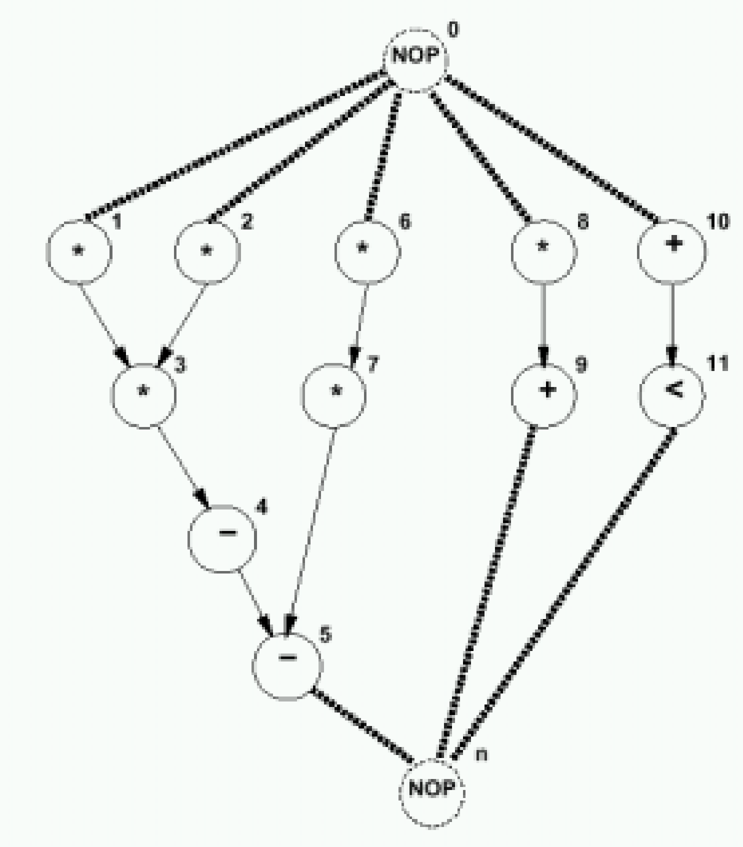
- Distribution graphs for multiplier and ALU.

Example of Force-directed scheduling

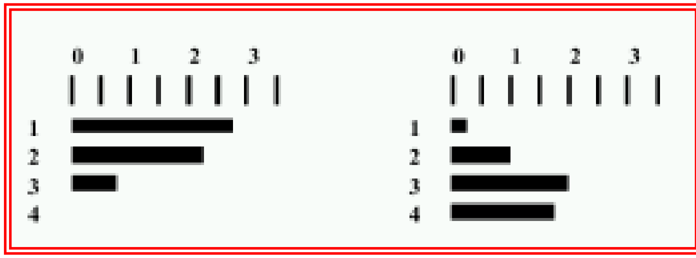
Force



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

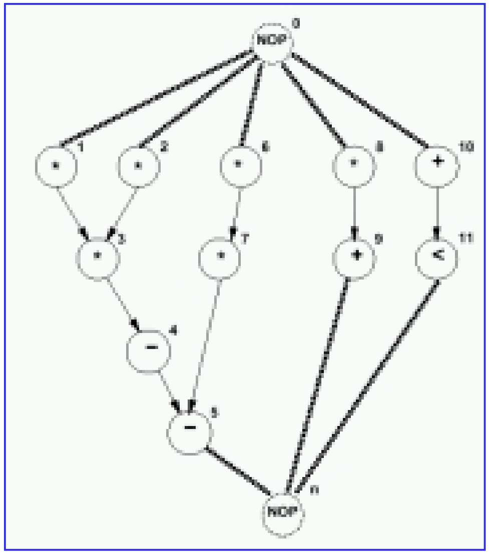


Forces related to the assignment of an operation to a control step



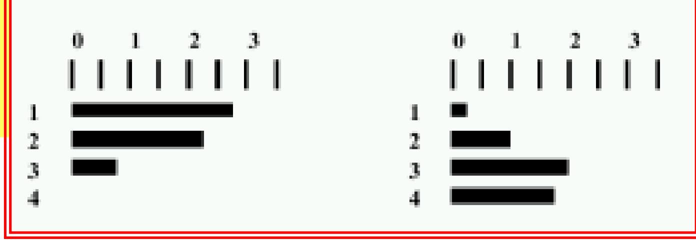
- Self-force:
  - Sum of forces to other steps.
  - Self-force for operation $\mathbf{v}_{\mathbf{i}}$ in step $\boldsymbol{l}$

$$\sum_{m=t_i^S}^{t_i^L} q_k(m)(\delta_{lm} - p_i(m))$$

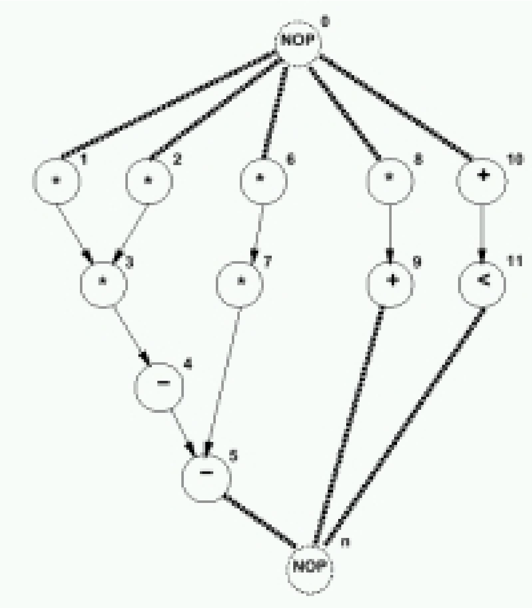


- Successor-force:
  - Related to the successors.
  - Delaying an operation implies delaying its successors.

Example: operation v_6

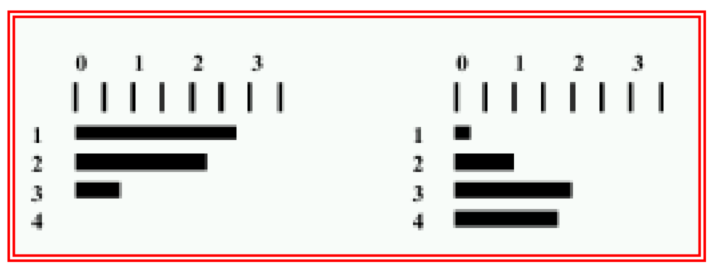


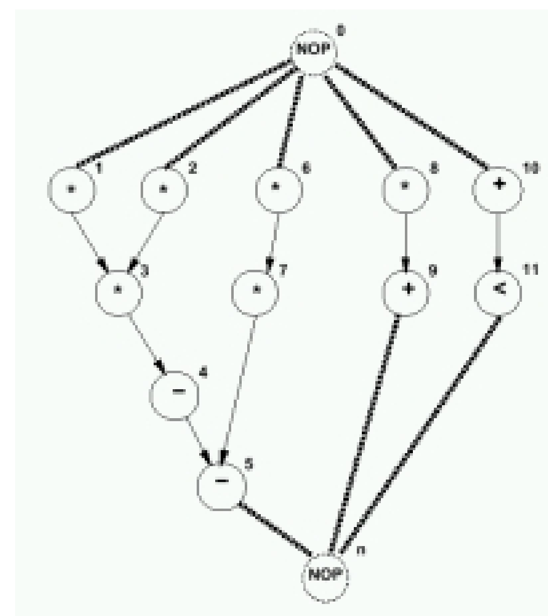
- **It can be scheduled in the first two steps.**
  - $p(1) = 0.5$; $p(2) = 0.5$; $p(3) = 0$; $p(4) = 0$.
- <u>Distribution:</u> $q(1) = 2.8$; $q(2) = 2.3$.
- Assign v_6 to step 1:
  - variation in probability $1 - 0.5 = 0.5$ for step 1
  - variation in probability $0 - 0.5 = -0.5$ for step 2
- **Self-force:** 2.8 * 0.5 - 2.3 * 0:5 = +0.25



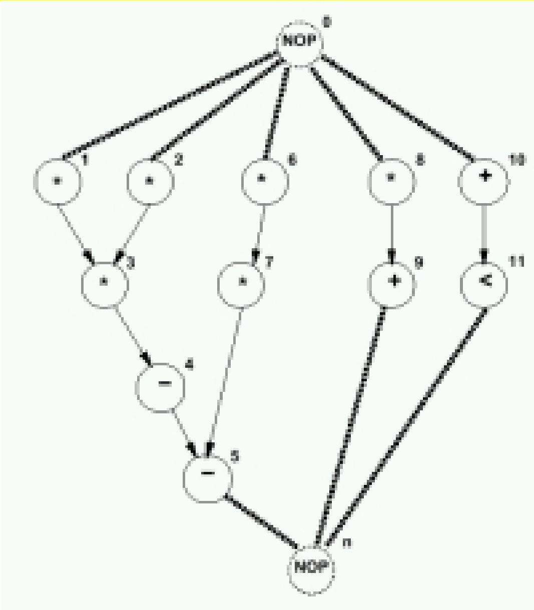
Example: operation v_6

- 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

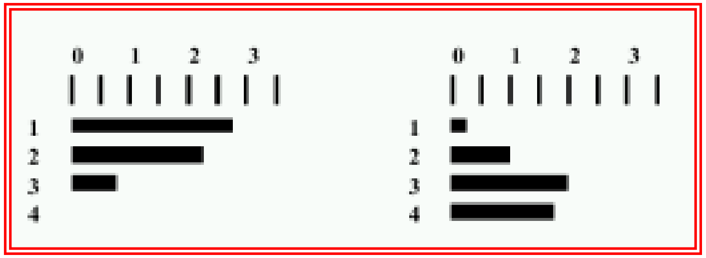




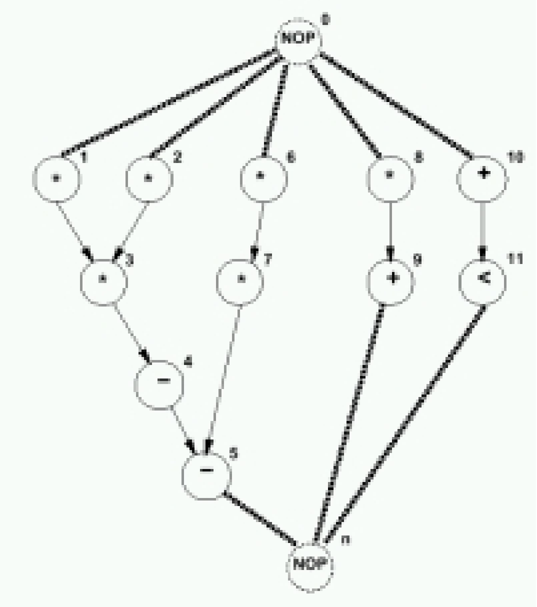
Example: operation v_6



- **Successor-force:**
  - Operation v_7 assigned to step 3.
  - $2.3\ (0-0.5) + 0.8\ (1 - 0.5) = -0.75$
- **Total-force** = -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), λ) {

repeat {

Compute the time-frames;

Compute the operation and type probabilities;

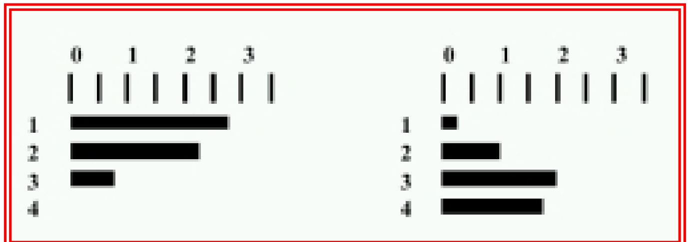
Compute the self-forces, p/s-forces and total forces;

Schedule the operation with least force, update time-frame;

} until (all operations are scheduled)

return (t);

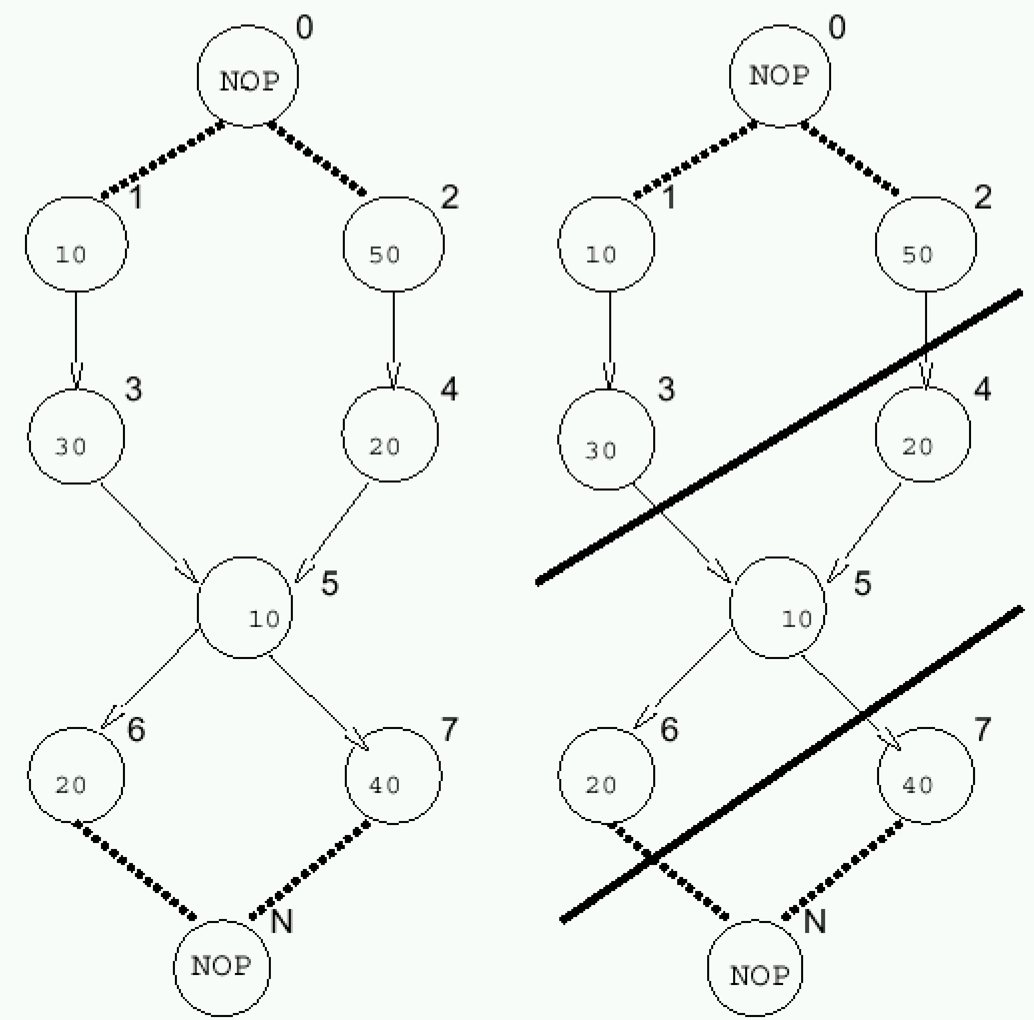
}



Scheduling with 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 of Scheduling with chaining



(a) (b)

- Cycle-time: 60.

Summary

- Scheduling determines area/latency trade-off.
- Intractable problem in general:
  - Heuristic algorithms.
  - ILP formulation (small-case problems).
- Chaining:
  - Incorporate *cycle-time* considerations.