Architecture Synthesis Part 1

SS 2012

Jun.-Prof. Dr. Christian Plessl

Custom Computing
University of Paderborn

Version 1.1.3 – 2012-07-23

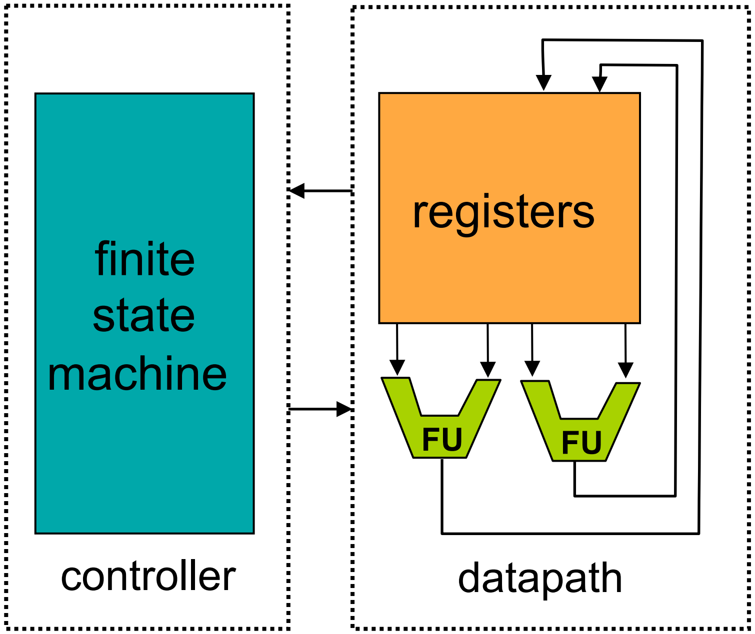
Motivation

- translate program/algorithm into dedicated hardware
- units of translation
  - single basic block (combinational)
  - complete program (sequential)

```
int diffeq(x,y,u,dx,a){

 do {
   x1 = x + dx;
   u1 = u – (3*x*u*dx) – (3*y*dx);
   y1 = y + u*dx;
   c = x1 < a;
   x = x1; y = y1; u = u1;
 } while (!c)

 return y;
}
```



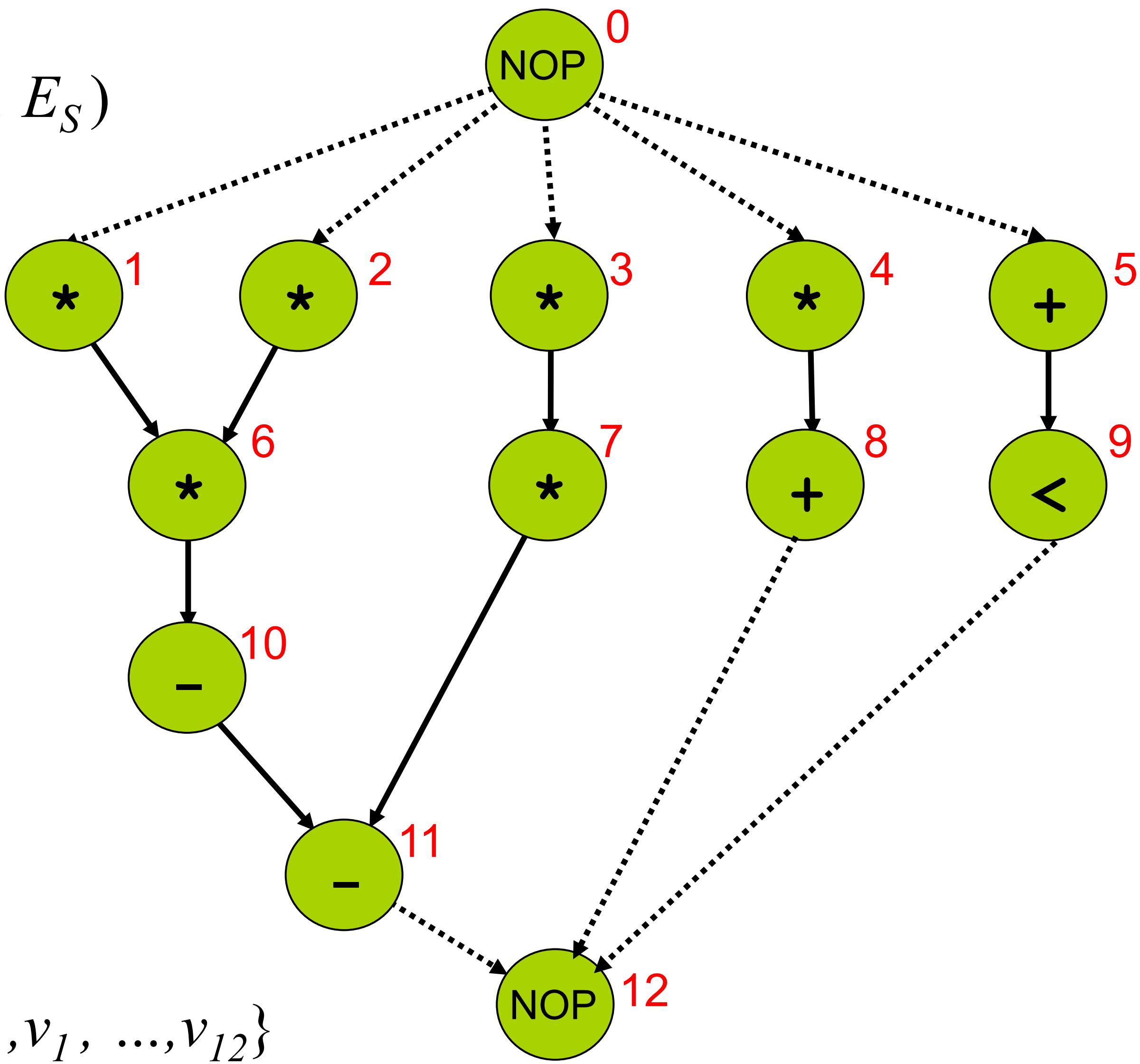
- FSM controls of datapath (FSM actions)
- datapath sends feedback to FSM (FSM conditions)

Overview

- translation of basic blocks
  - formal modeling
    - sequence graph, resource graph
    - cost function, execution times
    - synthesis problems: allocation, binding, scheduling
  - scheduling algorithms
    - ASAP, ALAP
    - extended ASAP, ALAP
    - list scheduling
    - integer linear programming
- translation of complete programs
  - finite state machine and data path
  - micro-coded controllers

Sequencing Graph

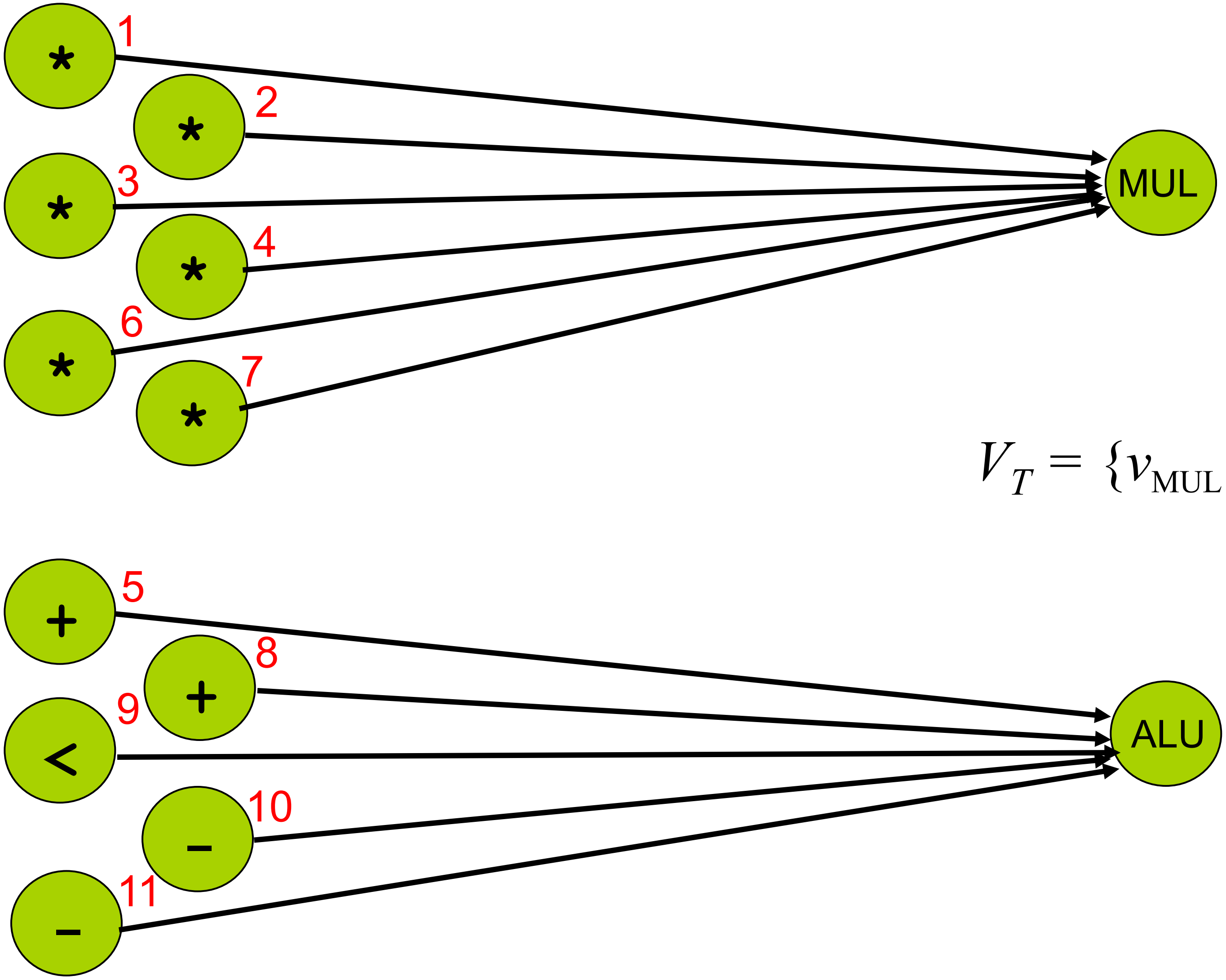
$G_S(V_S, E_S)$



$V_S = \{v_0, v_1, \ldots, v_{12}\}$

- $G_R(V_R, E_R)$
  - set of nodes $V_R = V_S \cup V_T$
    - V_S are the nodes of the sequencing graph (without NOPs)
    - V_T represent resource types (adder, multiplier, ALU, …)
  - set of edges $(v_S, v_T) \in E_R$ with $v_S \in V_S$, $v_T \in V_T$
    - an instance of resource type v_T can be used to implement operation v_S

Resource Graph – Example



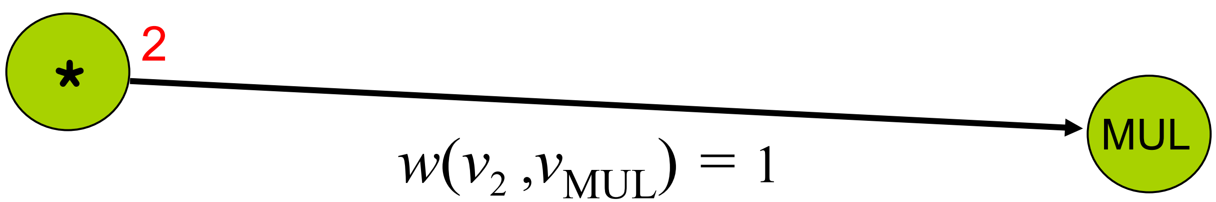
$$V_T = \{v_{\mathrm{MUL}}, v_{\mathrm{ALU}}\}$$

Cost, Execution Time

- cost function $c\colon V_T \rightarrow \boldsymbol{Z}$
  - assigns a cost value to each resource type
  - example:

MUL $c(v_{\mathrm{MUL}}) = 8$ ALU $c(v_{\mathrm{ALU}}) = 4$

- execution times $w\colon E_R \rightarrow \boldsymbol{Z}^+$
  - assigns the execution time of operation $v_S \in V_S$ on resource type $v_T \in V_T$ to the edge $(v_S, v_T) \in E_R$
  - example:



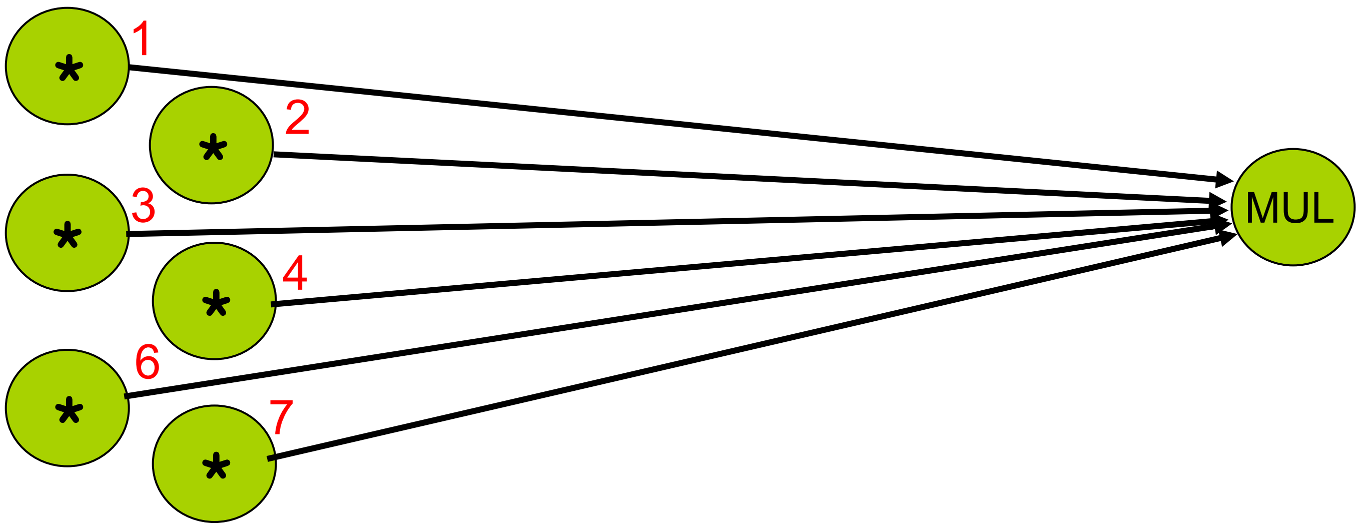
$$w(v_2, v_{\mathrm{MUL}}) = 1$$

Allocation, Binding

- allocation α: $V_T \rightarrow \boldsymbol{Z}^+$
  - assigns a number $\alpha(v_T)$ of available instances to each resource type v_T

- binding is given by the two functions
  β: $V_S \rightarrow V_T$ and γ: $V_S \rightarrow \boldsymbol{Z}^+$
  - $\beta(v_S) = v_T$ means that operation v_S is implemented by resource type v_T (possible β's shown in the resource graph)
  - $\gamma(v_S) = r$ denotes that v_S is implemented by the r–th instance of v_T ; $r \leq \alpha(v_T)$

Allocation, Binding – Example (1)



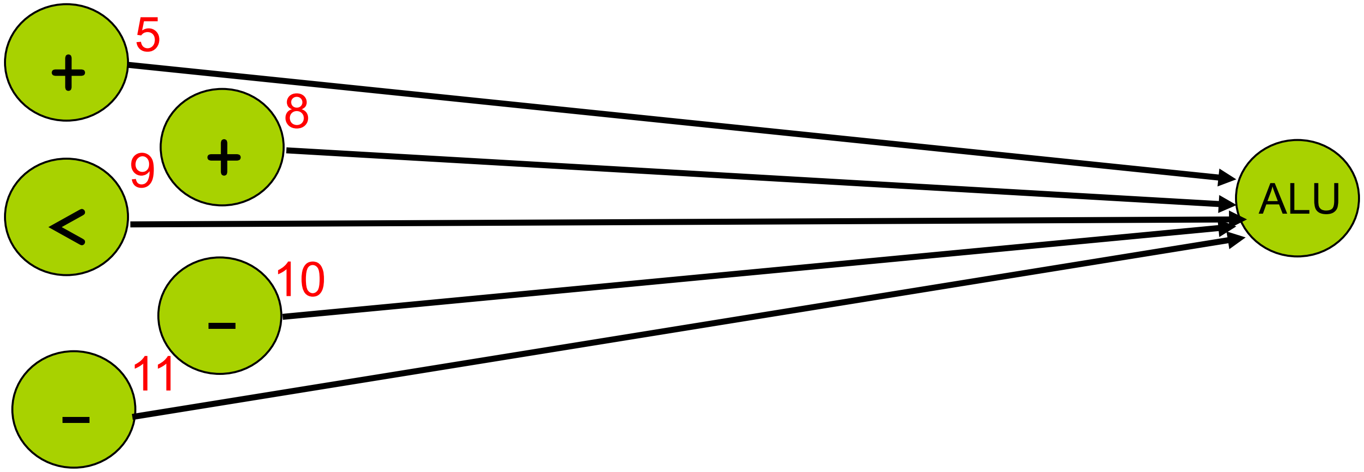
$c(v_{\text{MUL}}) = 8$

$\alpha(v_{\text{MUL}}) = 3$

$$
\begin{aligned}
w(v_1, v_{\text{MUL}}) &= 1\\
w(v_2, v_{\text{MUL}}) &= 1\\
w(v_3, v_{\text{MUL}}) &= 1\\
w(v_4, v_{\text{MUL}}) &= 1\\
w(v_6, v_{\text{MUL}}) &= 1\\
w(v_7, v_{\text{MUL}}) &= 1
\end{aligned}
$$

$$
\begin{aligned}
\beta(v_1) &= v_{\text{MUL}} & \gamma(v_1) &= 1\\
\beta(v_2) &= v_{\text{MUL}} & \gamma(v_2) &= 2\\
\beta(v_3) &= v_{\text{MUL}} & \gamma(v_3) &= 3\\
\beta(v_4) &= v_{\text{MUL}} & \gamma(v_4) &= 1\\
\beta(v_6) &= v_{\text{MUL}} & \gamma(v_6) &= 2\\
\beta(v_7) &= v_{\text{MUL}} & \gamma(v_7) &= 3
\end{aligned}
$$

Allocation, Binding – Example (2)



$c(v_{\mathrm{ALU}}) = 4$

$\alpha(v_{\mathrm{ALU}}) = 2$

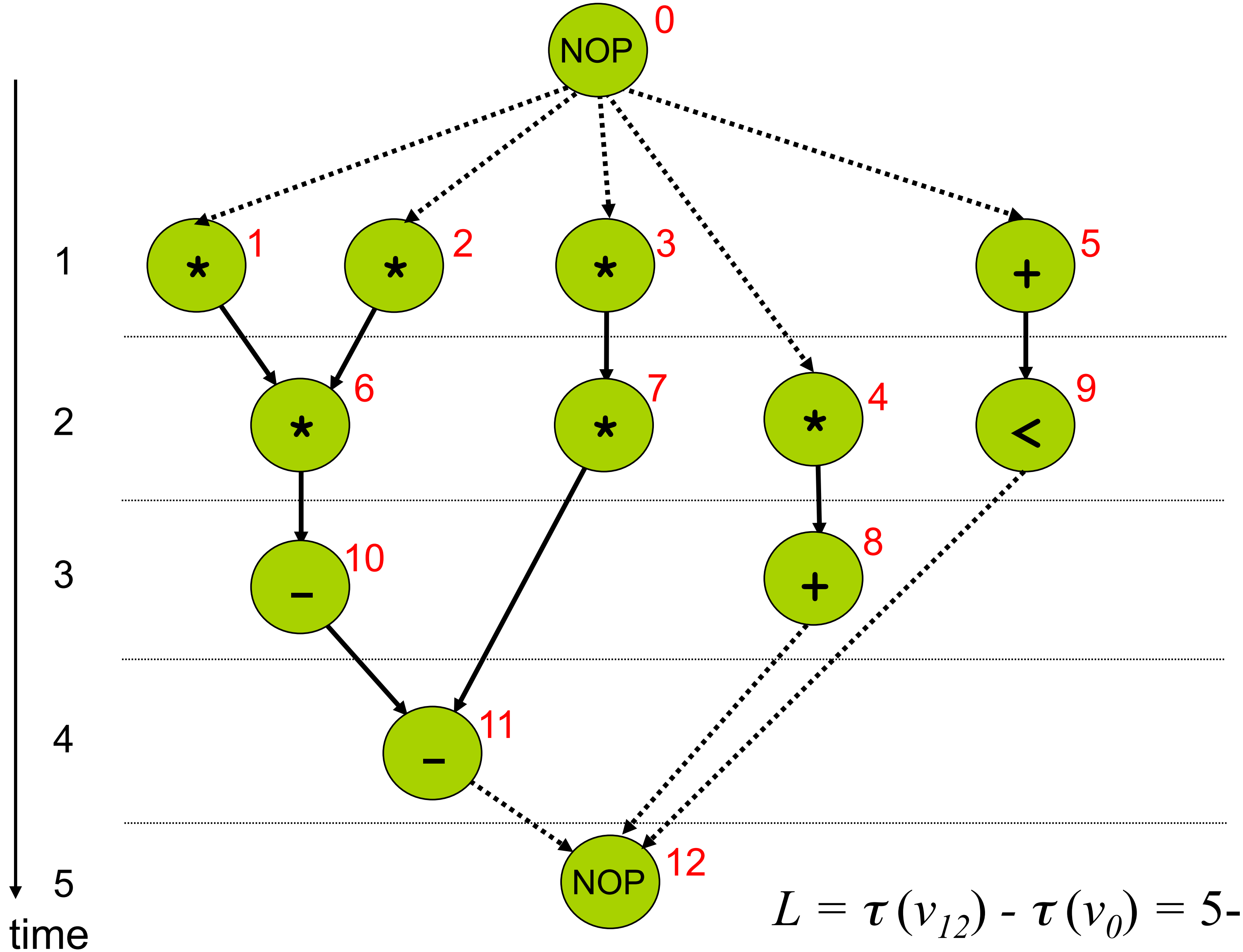
$w(v_5, v_{\mathrm{ALU}}) = 1$
$w(v_8, v_{\mathrm{ALU}}) = 1$
$w(v_9, v_{\mathrm{ALU}}) = 1$
$w(v_{10}, v_{\mathrm{ALU}}) = 1$
$w(v_{11}, v_{\mathrm{ALU}}) = 1$

$\beta(v_5) = v_{\mathrm{ALU}} \quad \gamma(v_5) = 1$
$\beta(v_8) = v_{\mathrm{ALU}} \quad \gamma(v_8) = 1$
$\beta(v_9) = v_{\mathrm{ALU}} \quad \gamma(v_9) = 1$
$\beta(v_{10}) = v_{\mathrm{ALU}} \quad \gamma(v_{10}) = 2$
$\beta(v_{11}) = v_{\mathrm{ALU}} \quad \gamma(v_{11}) = 1$

- schedule τ: $V_S \longrightarrow \boldsymbol{Z}^+$
  - assigns a start time to each operation under the constraint
    $\tau(v_j) - \tau(v_i) \geq w(v_i, \beta(v_i)) \quad \forall\ (v_i, v_j) \in E_S$

- latency L of a scheduled sequencing graph
  - difference in start times between end node and start node
    $L = \tau(v_n) - \tau(v_0)$

Schedule - Example



$$L = \tau(v_{12}) - \tau(v_0) = 5\text{-}1 = 4$$

- allocation, binding, scheduling
  - finding ($\alpha, \beta, \gamma, \tau$) that optimize latency and cost under resource and timing constraints
  - algorithms for architecture synthesis discussed in, e.g.
    - J.Teich, C. Haubelt, *Digitale Hardware/Software-Systeme*, Springer 2007
    - G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGrawHill 1994
  - synthesis problem variants
    - multicycle operations, operator chaining
    - several possible resource types for an operation
    - iterative schedules, pipelining

- in the following
  - scheduling without resource constraints
    - ASAP, ALAP
  - scheduling under resource constraints
    - extended ASAP, ALAP
    - list scheduling

Scheduling without Resource Constraints

- ASAP (as soon as possible)
  - determines the earliest possible start times for the operations
  - minimal latency

- ALAP (as late as possible)
  - determines the latest possible start times for the operations under a given latency bound

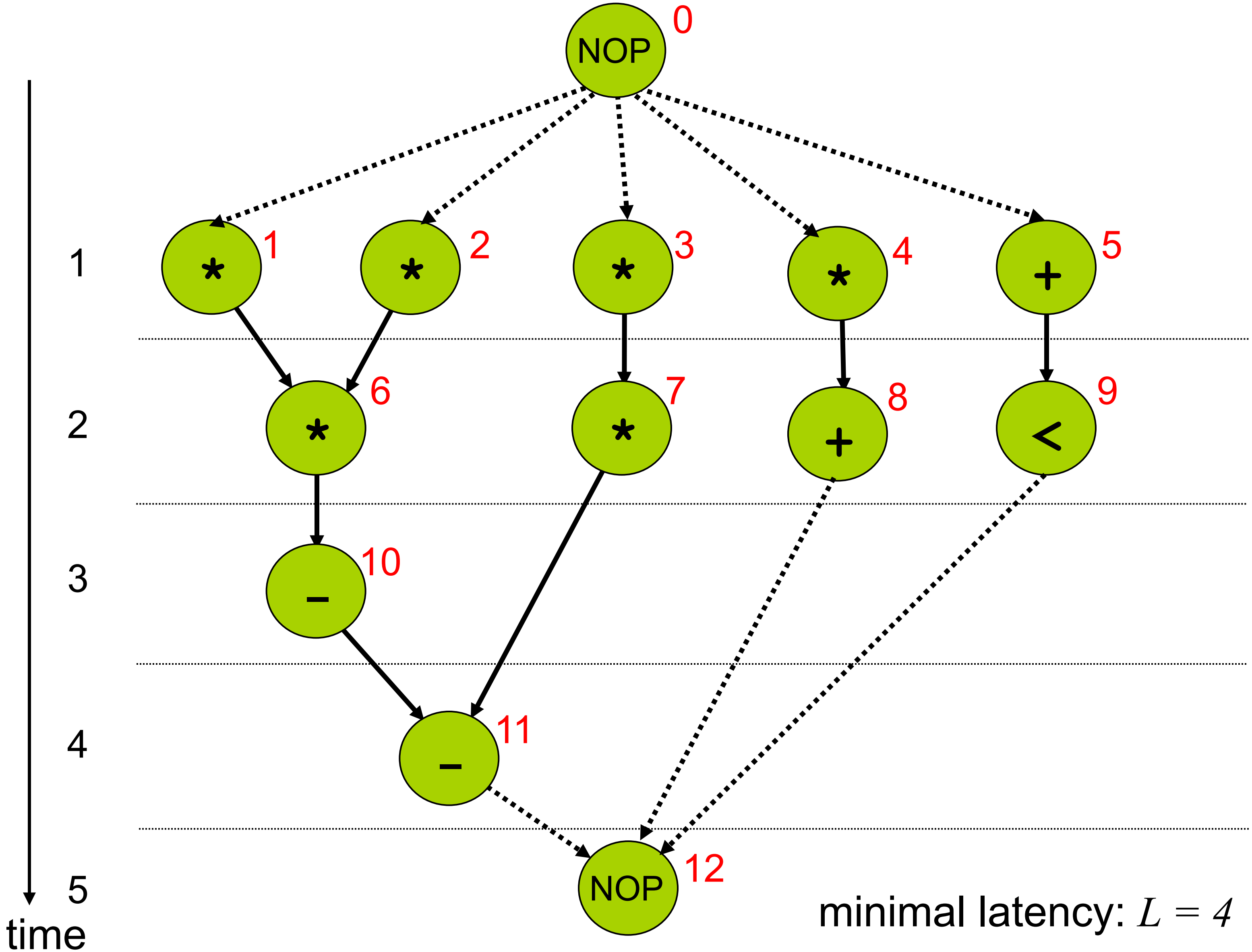
- slack (mobility) of operations
  - difference of start times: (ALAP with ASAP latency bound) - ASAP
  - if slack = 0 → operation is on the critical path

ASAP Scheduling

- ASAP: as soon as possible scheduling
- algorithm

ASAP($G_S(V,E)$) {
  schedule v_0 by setting $\tau(v_0) = 1$
  repeat {
    select a vertex v_j whose predecessors are all scheduled
    schedule v_j by setting $\tau(v_j) = \max_{i:(v_i,v_j)\in E} \tau(v_i) + w(v_i, \beta(v_i))$
  }
  until (v_n is scheduled)
  return τ
}

ASAP Example



minimal latency: $L = 4$

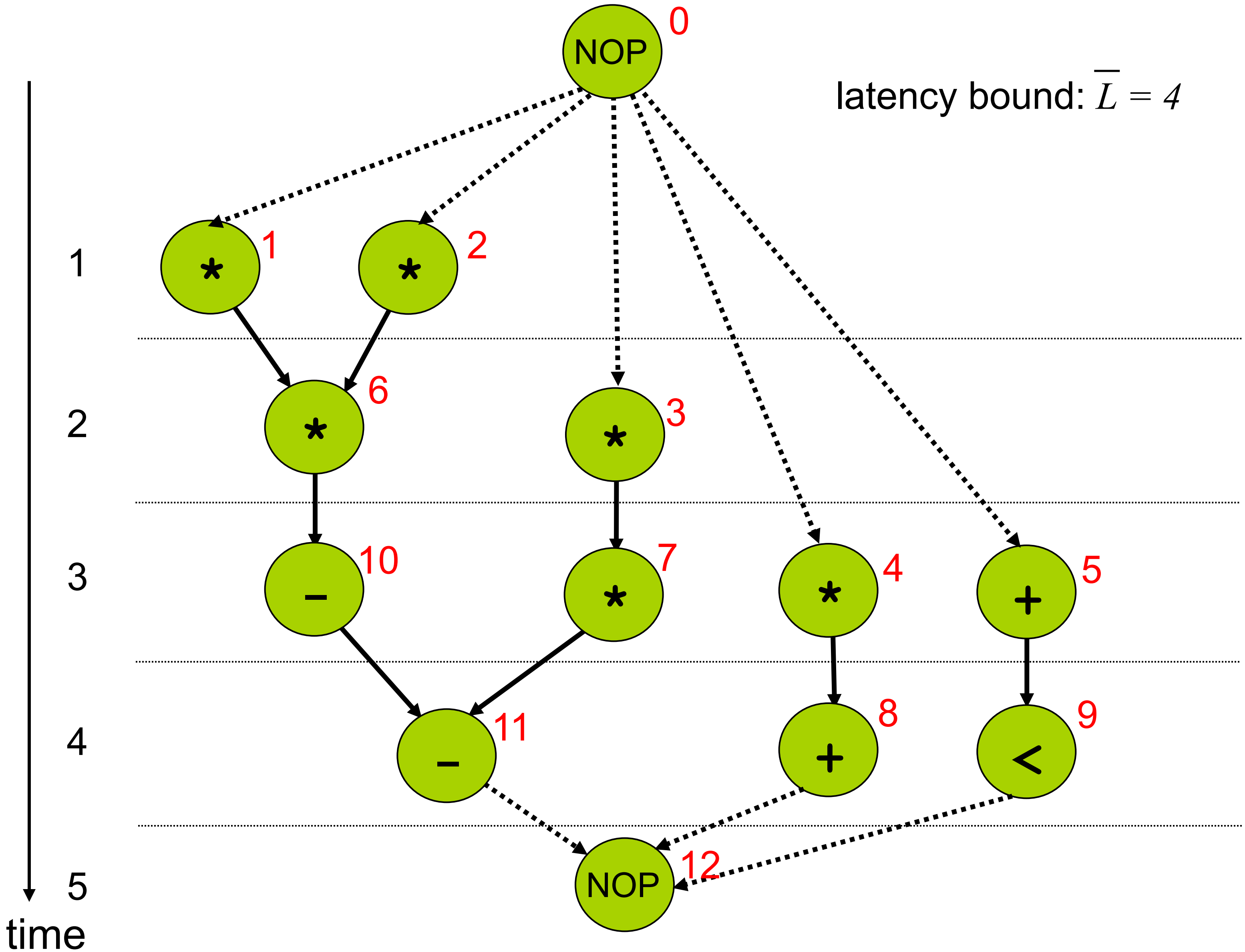
ALAP Scheduling

- ALAP: as late as possible scheduling
- requires a latency bound $\overline{L}$
  - otherwise nodes could be arbitrarily delayed
  - typically the schedule length of ASAP schedule is used as latency bound

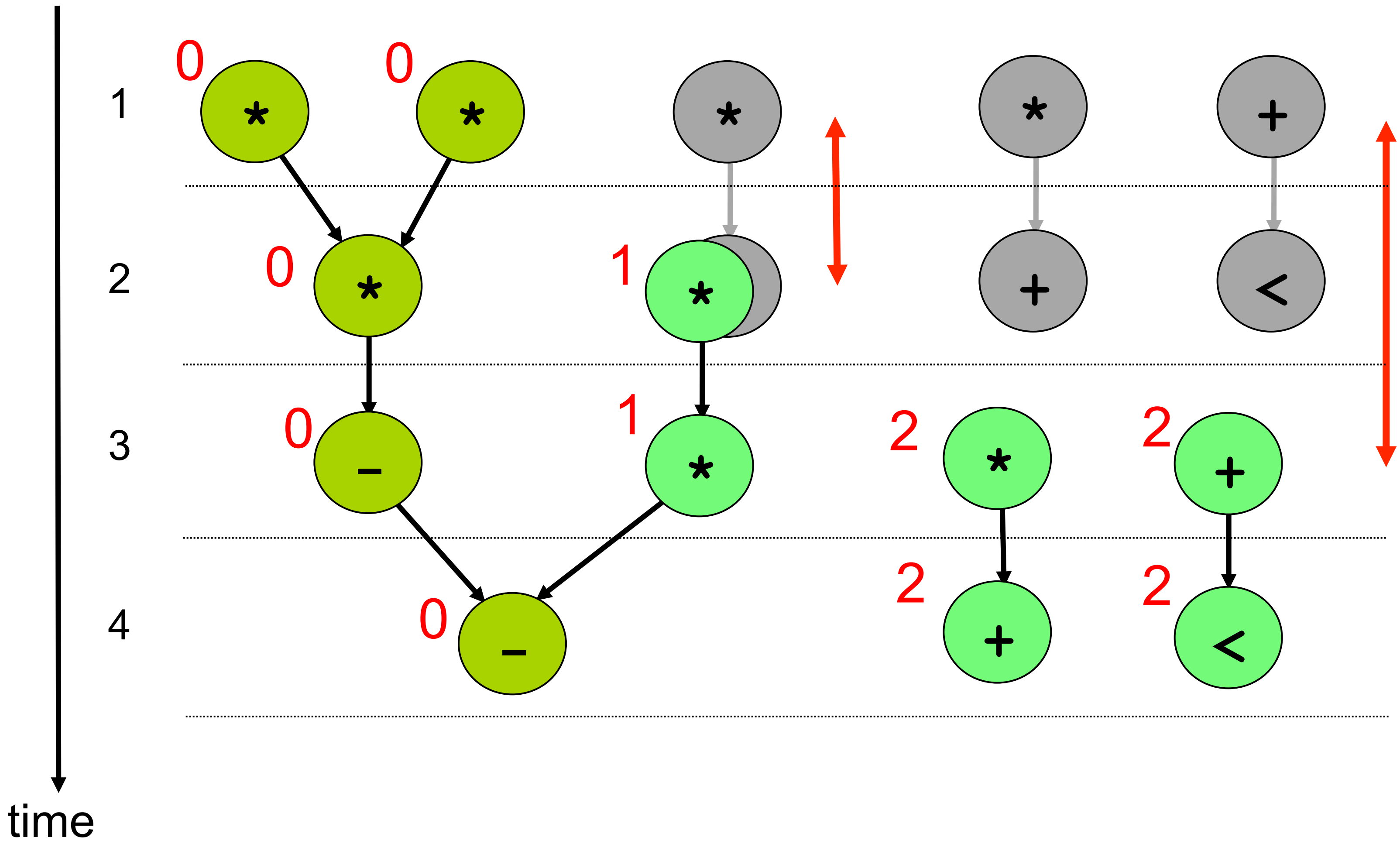
- algorithm

ALAP($G_S(V,E), \overline{L}$) {
  schedule v_n by setting $\tau(v_n) = \overline{L} + 1$
  repeat {
    select a vertex v_i whose successors are all scheduled
    schedule v_i by setting $\tau(v_i) = \min\limits_{j:(v_i,v_j)\in E} \tau(v_j) - w(v_i, \beta(v_i))$
  }
  until (v_n is scheduled)
  return τ
}

ALAP Example



Slack (mobility)



Scheduling under Resource Constraints (1)

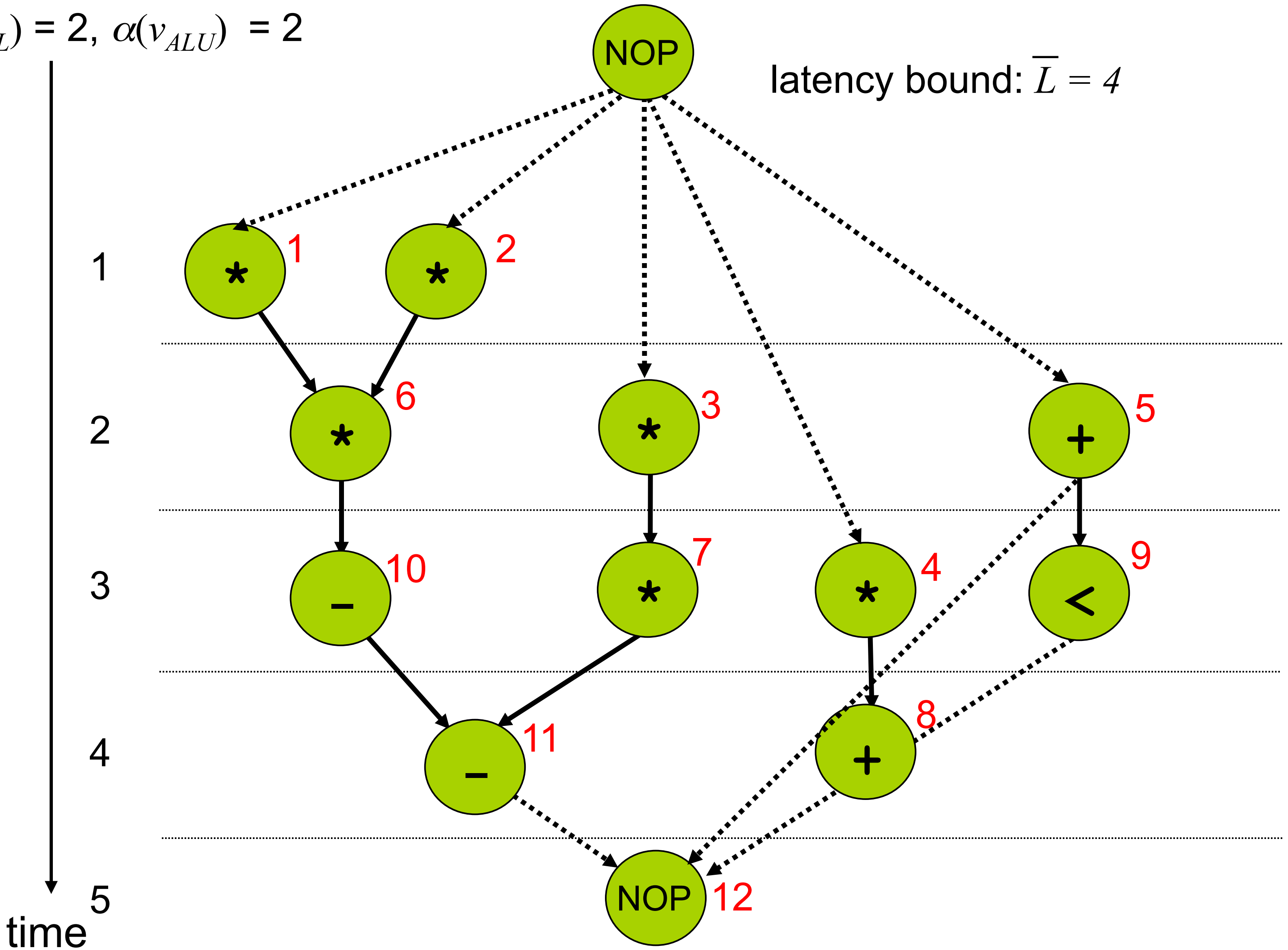
- Extended ASAP, ALAP
  - first ASAP or ALAP
  - then move operations down (ASAP) or up (ALAP) until resource constraints are satisfied

Extended ALAP

resource constraints:

$\alpha(v_{MUL}) = 2, \ \alpha(v_{ALU}) = 2$

latency bound: $\overline{L} = 4$



Scheduling under Resource Constraints (2)

- list scheduling
  - operations are prioritized according to some criterion, e.g. number of successor nodes, slack, …

time = 1

repeat

for each resource $(v_T, \alpha(v_T))$

determine all ready operations v_S with $\beta(v_s) = v_T$ and

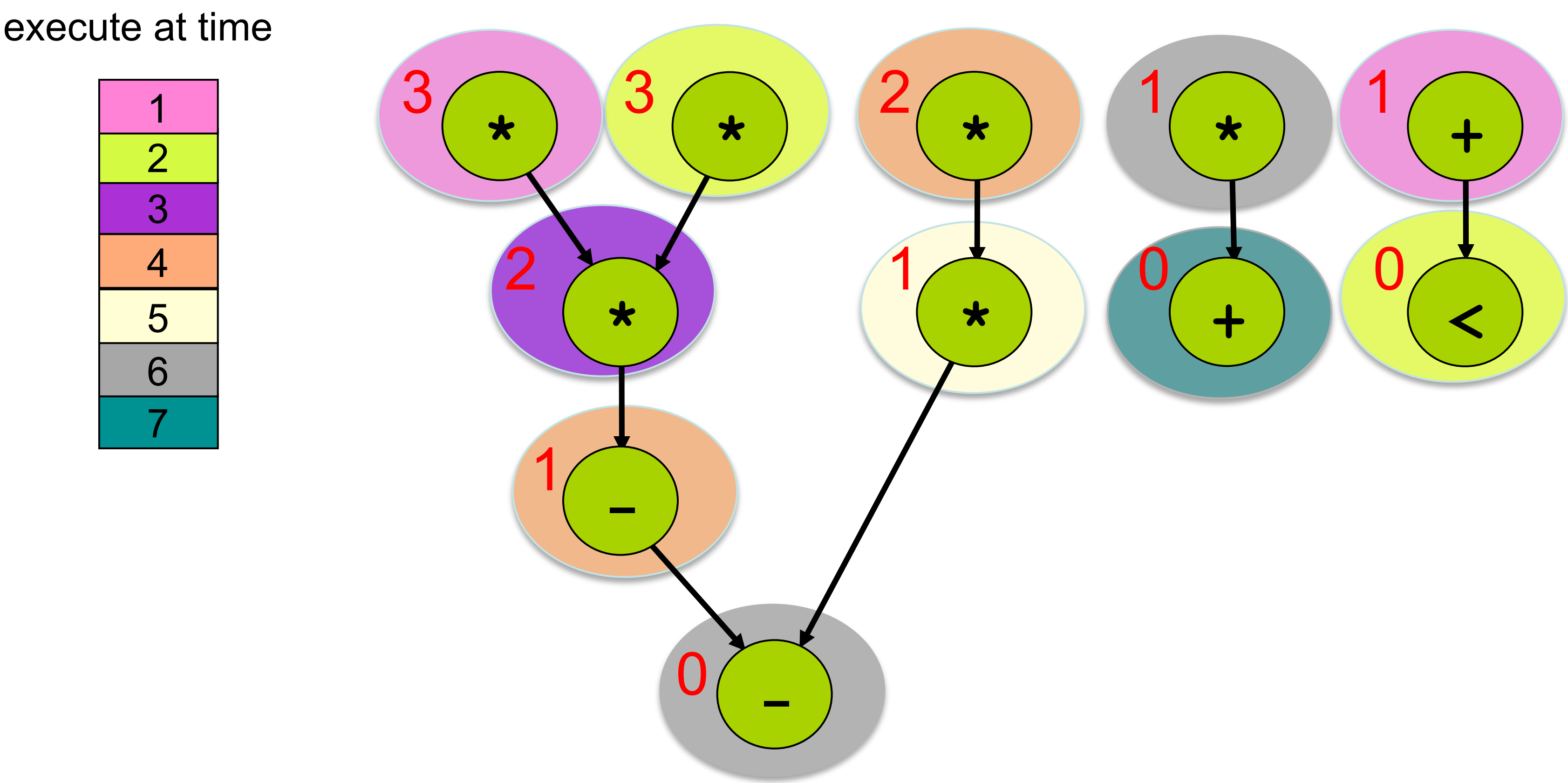
schedule the one with the highest priority

time ++;

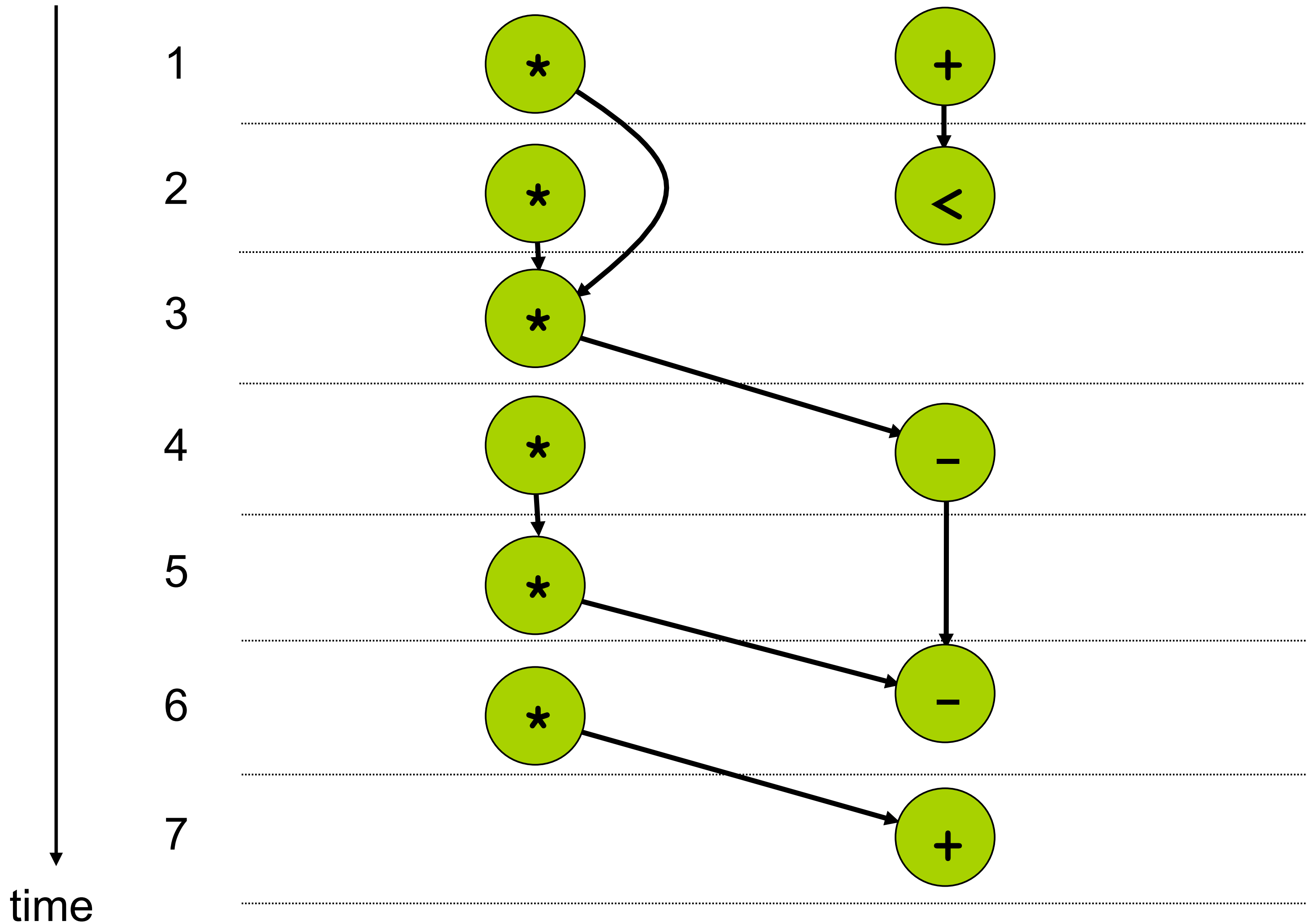
until (v_n is scheduled)

List Scheduling (1)

- example
  - criterion: number of successor nodes
  - resource constraints: $\alpha(v_{MUL}) = 1$, $\alpha(v_{ALU}) = 1$



List Scheduling (2)



$$L = \tau(v_{12}) - \tau(v_0) = 8\text{-}1 = 7$$

Changes

- 2012-07-23 (v1.1.3)
  - correct definition of ASAP algorithm on slide 15.
- 2012-06-11 (v1.1.2)
  - remove sentence "assume all operator delays are equal) from slide 13, materials have been extended such that we can handle arbitrary delays.
- 2012-05-07 (v1.1.1)
  - added algorithms for ASAP and ALAP
- 2012-05-07 (v1.1.0)
  - updated for SS2012
- 2011-05-04 (v1.0.1)
  - cosmetics: fix typo, add label to slide 21
- 2011-05-25 (v1.0.2)
  - minor corrections