

VLSI System Design

Part V : High-Level Synthesis(2)

Lecturer : Tsuyoshi Isshiki

Dept. Information and Communications Engineering,
Tokyo Institute of Technology
issiki@ict.e.titech.ac.jp

High-Level Synthesis Flow

- A) Design capture (HDLs, C/C++, signal-flow graph, etc)
- B) Compilation to internal representation
 - Data-flow graph (DFG)
 - Control-flow graph (CFG)
 - Control-data-flow graph (CDFG)
- C) Resource allocation
 - Specify available functional units
- D) Operation scheduling
 - Assign each operation to control steps
- E) Resource binding
 - Assign each data to registers
 - Assign each operation to functional units

Synthesis Constraints and Cost Functions

- Constraints : must be satisfied
- Cost function : want to minimize
- ✓ Time-constrained → minimize area
- ✓ Area-constrained → minimize latency (maximize throughput)
- How to evaluate area before logic synthesis?
 - simple approximation : only count the number of functional units (ignore control units, registers and memories)
- Other parameters : power consumption, testability

Module Library

□ Specify the types of functional units

$$M = \{ m \mid m : \text{functional units} \}$$

- ✓ single function units : add, subtract, multiply, compare, shift
- ✓ multi-function units : add/subtract, add/subtract/compare (ALU)
- Speed/area choices : slow & small \leftrightarrow fast & large
- Clocking choices : single-cycle, multi-cycle, pipelined

□ Characterization of functional units

- ✓ $p(m)$: # of pipeline stages
- ✓ $d_p(m)$: Maximum combinational logic delay per pipeline stage
- ✓ $d(m) = p(m) \times d_p(m)$: computation latency
- ✓ $a(m)$: area

module	delay per stage	# pipe stages	area
m0 : ADD-I	20ns	1	200
m1 : ADD-II	10ns	1	300
m2 : MULT-I	80ns	1	2600
m3 : MULT-II	40ns	2	3000

Resource Assignment and Allocation

A) *Resource assignment* : for each operation $v \in V$ in the target data-flow graph $G(V, E)$, allocate a compatible functional unit $m \in M$:

$$\rho : V \rightarrow M \text{ or } \rho(v) = m$$

→ Mapping $\rho : V \rightarrow M$ determines the latency of each operation
 $v \in V : d(v) = d(\rho(v))$

B) *Resource allocation* : specify the number of units $r(m)$ for each type $m \in M$ to be used in the hardware implementation

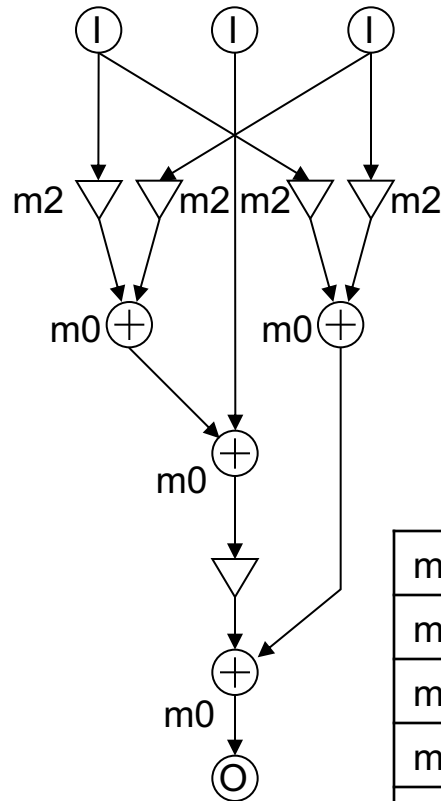
$$R = \{ r(m) \mid m \in M \}$$

→ Typically specified by the designer as a part of the synthesis parameters

→ Determines the circuit area occupied by the functional units :

$$Area(R) = \sum_{m \in M} r(m) a(m)$$

Resource Assignment Example



- Additions can be mapped either to m0 or m1
- Constant multipliers can be mapped either to m2 or m3
- What is the best mapping $\rho : V \rightarrow M$ when there are multiple module candidates? (usually not trivial problem)
- Popular approach is to allow only 1 type of functional units for all operations with the same type

module	delay per stage	# pipe stages	area
m0 : ADD-I	20ns	1	200
m1 : ADD-II	10ns	1	300
m2 : MULT-I	80ns	1	2600
m3 : MULT-II	40ns	2	3000

Operation Scheduling (1)

□ Problem inputs

- ✓ Data-flow graph $G(V, E)$
- ✓ Module library M
- ✓ Resource assignment $\rho: V \rightarrow M$
- ✓ Resource allocation $R = \{ r(m) \mid m \in M \}$
- ✓ Clock cycle period P
 - Computation latency is quantized to # of clock cycles :

$$\delta_p(m) = \lceil d_p(m) / P \rceil : \text{sampling interval}$$

$$\delta(m) = \delta_p(m) \times p(m) : \text{latency}$$

$(d_p(m) : \text{delay per stage, } p(m) : \# \text{ of pipeline stages})$

$$\delta_p(v) = \delta_p(\rho(v))$$

$$\delta(v) = \delta(\rho(v))$$

Operation Scheduling (2)

- ✓ Scheduling time set $T = \{0, 1, \dots, T_{\max} - 1\}$
 - Each scheduling time (control step) represents a duration of P
 - Scheduling of each operation is specified by the clock cycle index (between 0 and $T_{\max} - 1$)

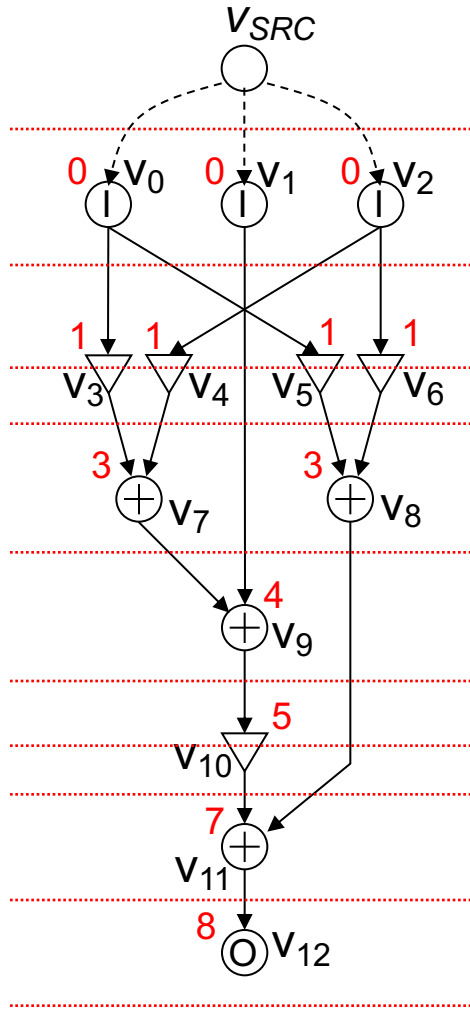
□ Scheduling σ is a mapping of operations $v \in V$ to scheduling time set T

$$\sigma: V \rightarrow T$$

□ while satisfying the data dependencies :

- $\sigma(v_j) \geq \sigma(v_i) + \delta(v_i)$ for $\forall e_{ij} = (v_i, v_j) \in E$
- $\sigma(v_j)$: execution starting cycle of node v_j
- $\sigma(v_i) + \delta(v_i)$: execution terminating cycle of node v_i

ASAP (As-Soon-As-Possible) Scheduling



- A) Add “source node” v_{SRC} to $G(V, E)$
 - $\delta(v_{SRC}) = 0$
 - $\sigma(v_{SRC}) = 0$
- B) Add arcs (v_{SRC}, v_{IN}) to $G(V, E)$ for each input node v_{IN}
- C) Let $\delta(v_{IN}) = 1$ (actually, delay of input nodes depends on the type of device connecting to v_{IN})
- D) Solve the longest path problem on $G(V, E)$ from v_{SRC}

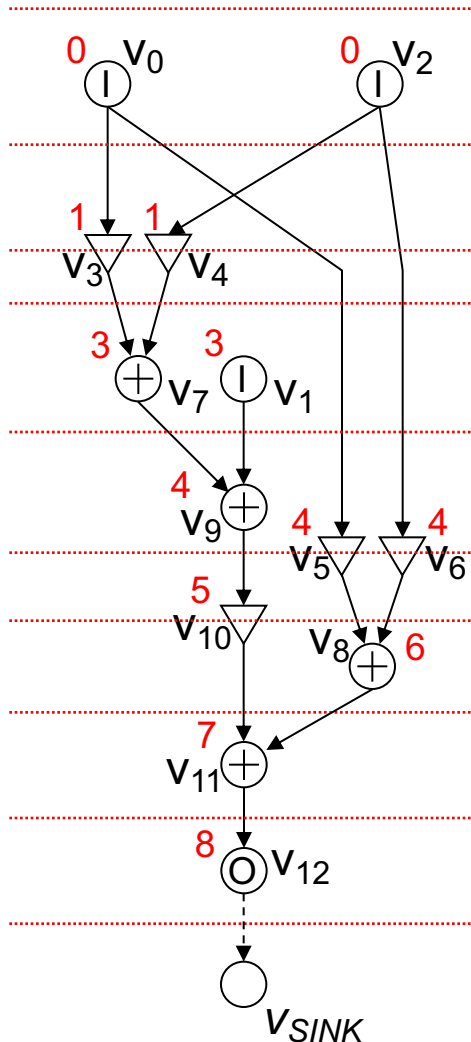
$$\sigma(v_j) = \max \{ \sigma(v_i) + \delta(v_i) \mid (v_i, v_j) \in E \}$$

- $\sigma(v_j) - \sigma(v_{SRC})$ is the longest path length from v_{SRC} (basically the same as computing the **arrival time** as in *delay-optimal technology mapping*)

◆ Example (clock cycle periods $P = 40\text{ns}$)

- Resource assignment :
 - ✓ map all additions to ADD-I (delay = 20ns, $\delta = 1$)
 - ✓ map all multiplications to MULT-I (delay = 80ns, $\delta = 2$)

ALAP (As-Late-As-Possible) Scheduling



A) Add “sink node” v_{SINK} to $G(V, E)$

- $\delta(v_{SINK}) = 0$
- $\sigma(v_{SINK}) = T_{max}$

B) Add arcs (v_{OUT}, v_{SINK}) to $G(V, E)$ for each output node v_{OUT}

C) Let $\delta(v_{OUT}) = 1$ (actually, delay of output nodes depends on the type of device connecting to v_{OUT})

D) Solve the longest path problem on $G(V, E)$ to v_{SINK}

$$\sigma(v_j) = \min \{ \sigma(v_i) - \delta(v_j) \mid (v_j, v_i) \in E \}$$

- $\sigma(v_{SINK}) - \sigma(v_j)$ is the longest path length to v_{SINK} (basically the same as computing the **required time** as in *delay-optimal technology mapping*)

◆ Example (same resource assignment and clock period as ASAP example)

- $T_{max} = 9$

(T_{max} needs to be set so that $\sigma(v) \geq 0$ for all $v \in V$)

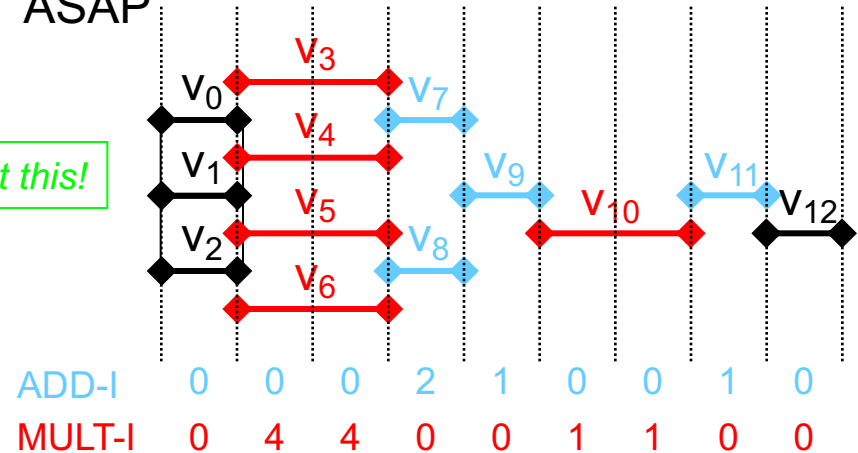
Resource Occupancy

- $r(\sigma, m, t)$: number of functional units m being used simultaneously at cycle t with scheduling σ
- $r(\sigma, m) = \max\{r(\sigma, m, t) \mid t \in T\}$: number of functional units m required to implement scheduling σ
- If resource allocation $R = \{r(m) \mid m \in M\}$ is specified, resource occupancy needs to satisfy

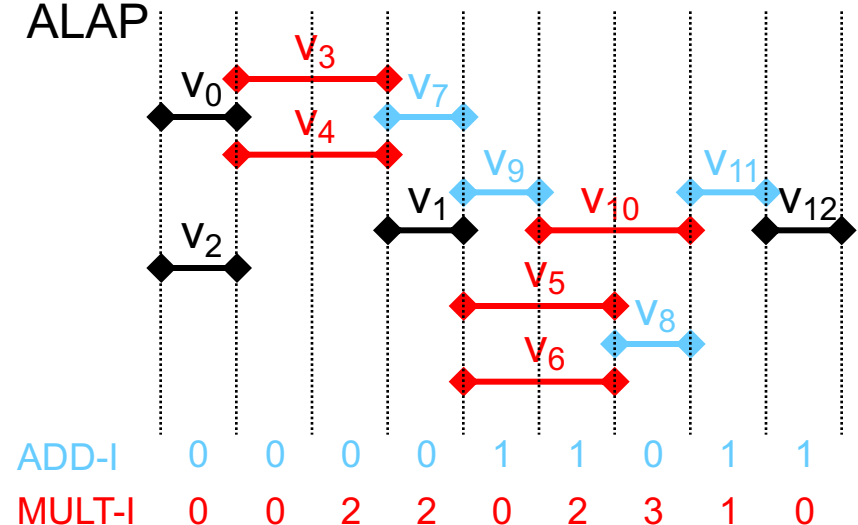
$$r(\sigma, m) \leq r(m) \text{ for all } m \in M$$
- ASAP and ALAP schedulings do not have the ability to optimize the resource occupancy
- ASAP and ALAP scheduling minimize the scheduling latency

Correct this!

ASAP



ALAP



Operation Scheduling (3)

□ Time-constrained scheduling

- T_{max} (scheduling time set) specified
- Minimize resource occupancy $r(\sigma, m)$ for each $m \in M$
- ✓ Force-directed scheduling

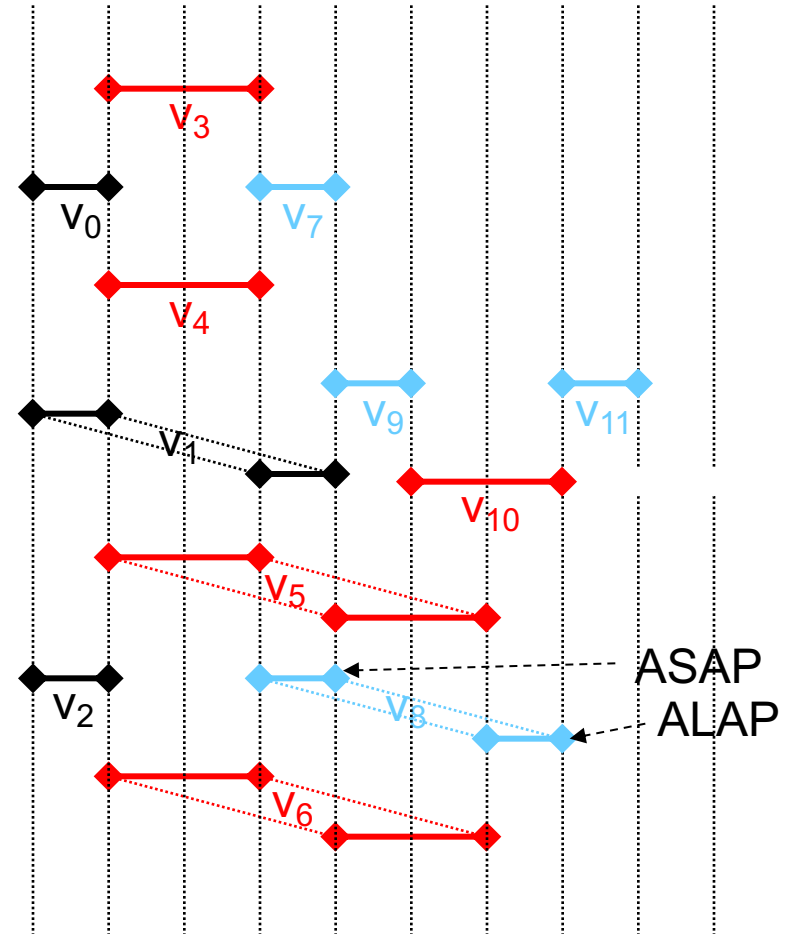
□ Resource-constrained scheduling

- $R = \{ r(m) \mid m \in M \}$ (resource allocation) specified
- Minimize T_{max}
- ✓ List scheduling

Mobility and Partial Scheduling

- Partial scheduling σ' is a mapping of operations $v \in V$ to a scheduling range $\sigma'(v) = [\sigma'_{min}(v), \sigma'_{max}(v)]$

$$\sigma' : V \rightarrow [T, T]$$
- σ'_{min} : earliest possible scheduling (ex. ASAP)
- σ'_{max} : latest possible scheduling (ex. ALAP)
- Mobility : $\mu(v) = \sigma'_{min}(v) - \sigma'_{max}(v)$
- When the mobilities of all operations $v \in V$ are 0, then the partial scheduling is complete.



Force-Directed Scheduling (1)

A) Operation scheduling distribution :

- ✓ assume that each operation v has the equal probability of being scheduled within the scheduling range $\sigma'(v) \rightarrow$

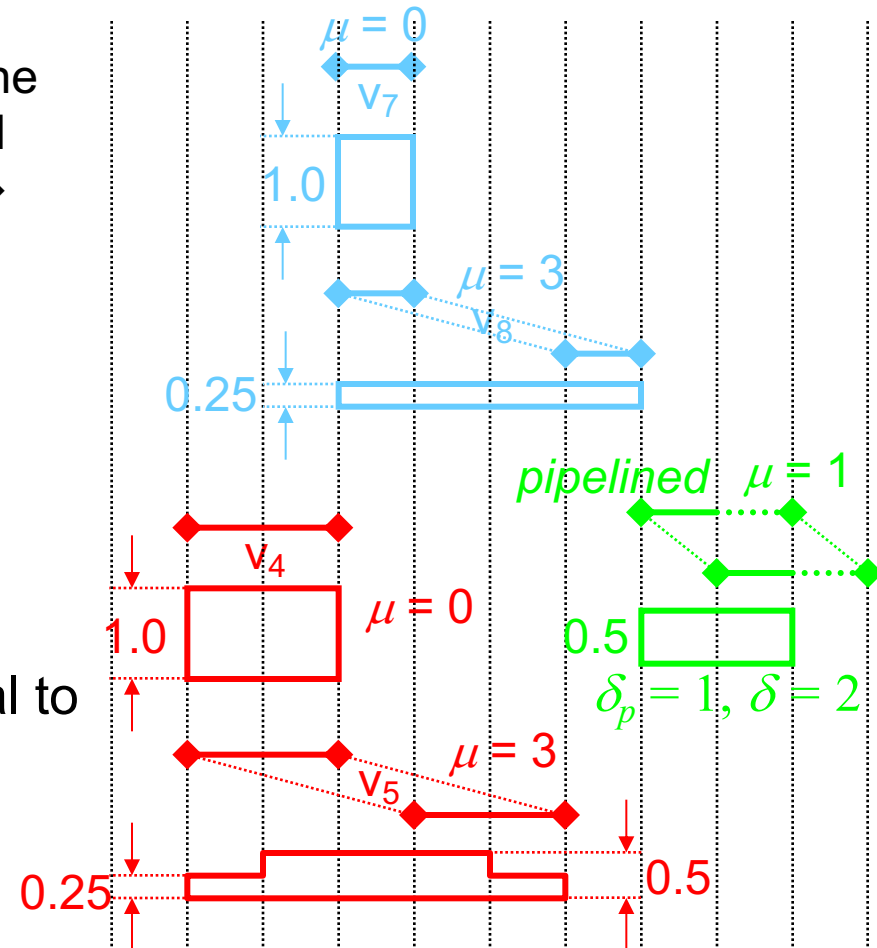
$$\theta(\sigma', v, t) = \sum_{k=0}^{\delta_p(v)-1} \phi(\sigma', v, t-k) / (\mu(v) + 1)$$

where

$$\begin{aligned} \phi(\sigma', v, t-k) &= 1 \quad (t-k \in \sigma'(v)) \\ &= 0 \quad (\text{otherwise}) \end{aligned}$$

- Total area of the distribution is equal to the sampling interval:

$$\sum_{t \in T} \theta(\sigma', v, t) = \delta_p(v)$$

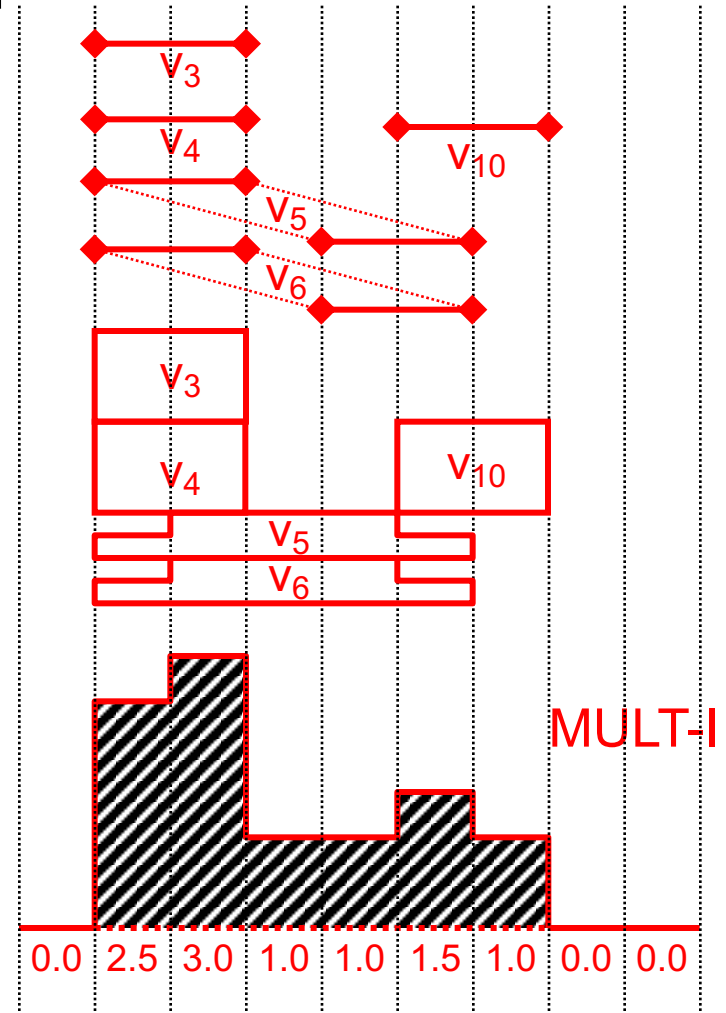
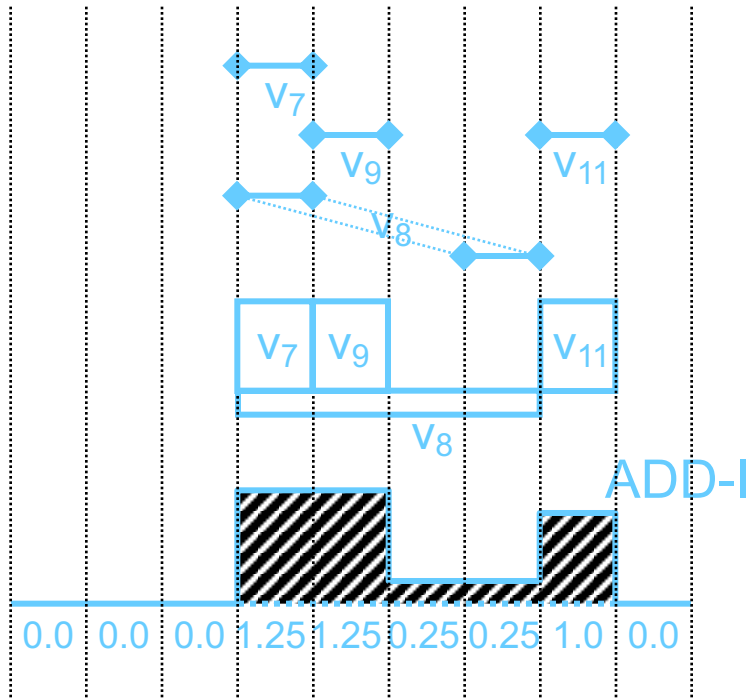


Force-Directed Scheduling (2)

B) Resource occupation distribution

$$r(\sigma', m, t) = \sum_{\rho(v)=m} \theta(\sigma', v, t)$$

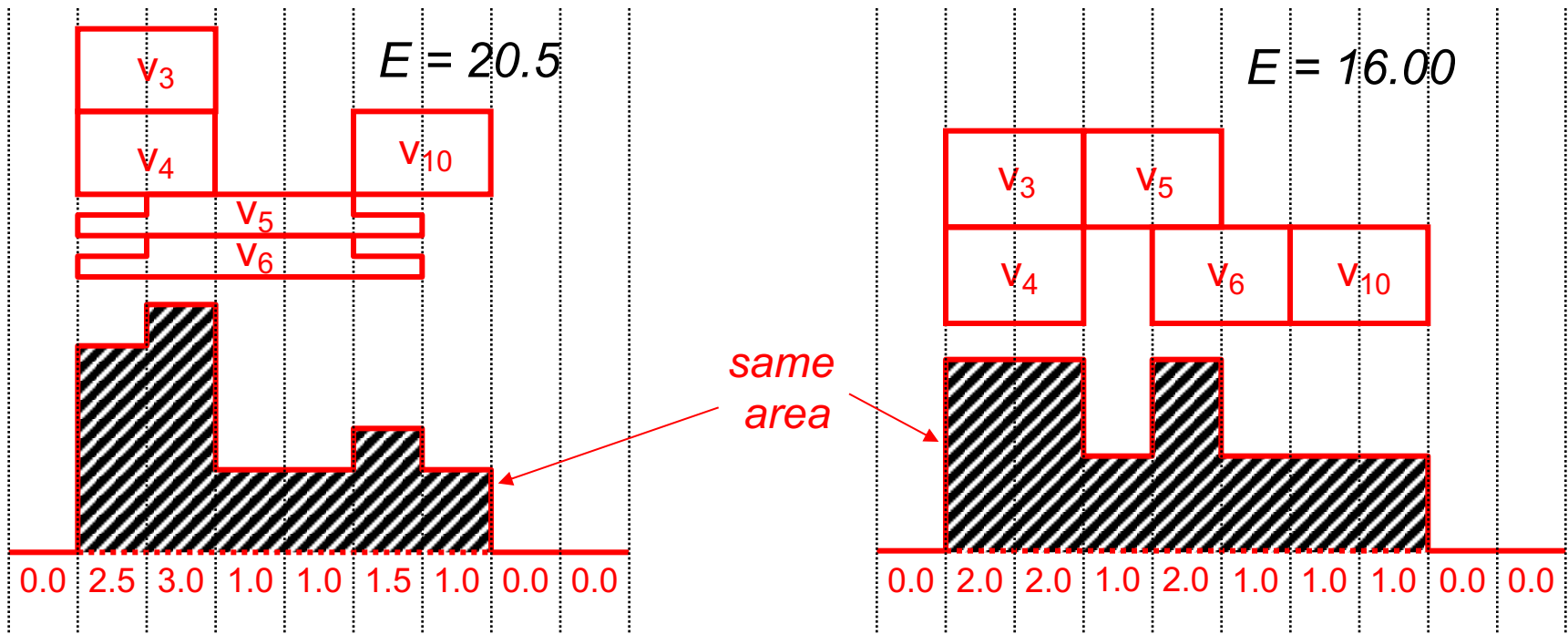
$$\triangleright \sum_{t \in T} r(\sigma', m, t) = \sum_{v \in V^p} \delta_p(v)$$



Force-Directed Scheduling (3)

- Basic idea :
 - Minimize the maximum resource occupancy:

$$\text{minimize } \max\{r(\sigma', m, t) \mid t \in T\}$$
 - Make the distribution as flat as possible (most balanced)
 - Minimize “energy” : $E(\sigma', m) = \sum_{t \in T} r(\sigma', m, t)^2$



Force-Directed Scheduling (4)

C) Operation distribution energy (force):

$$F(\sigma', v) = \sum_{t \in T} \theta(\sigma', v, t) \times r(\sigma', m, t)$$

➤ $E(\sigma', m) = \sum_{v \in V} F(\sigma', v)$

Correct this!

D) Operation scheduling energy (force):

(fix the scheduling $\sigma(v) \rightarrow t$)

$$F(\sigma', v, t) = \sum_{t' \in T} \phi(v, t' - t) \times r(\sigma', m, t')$$

where

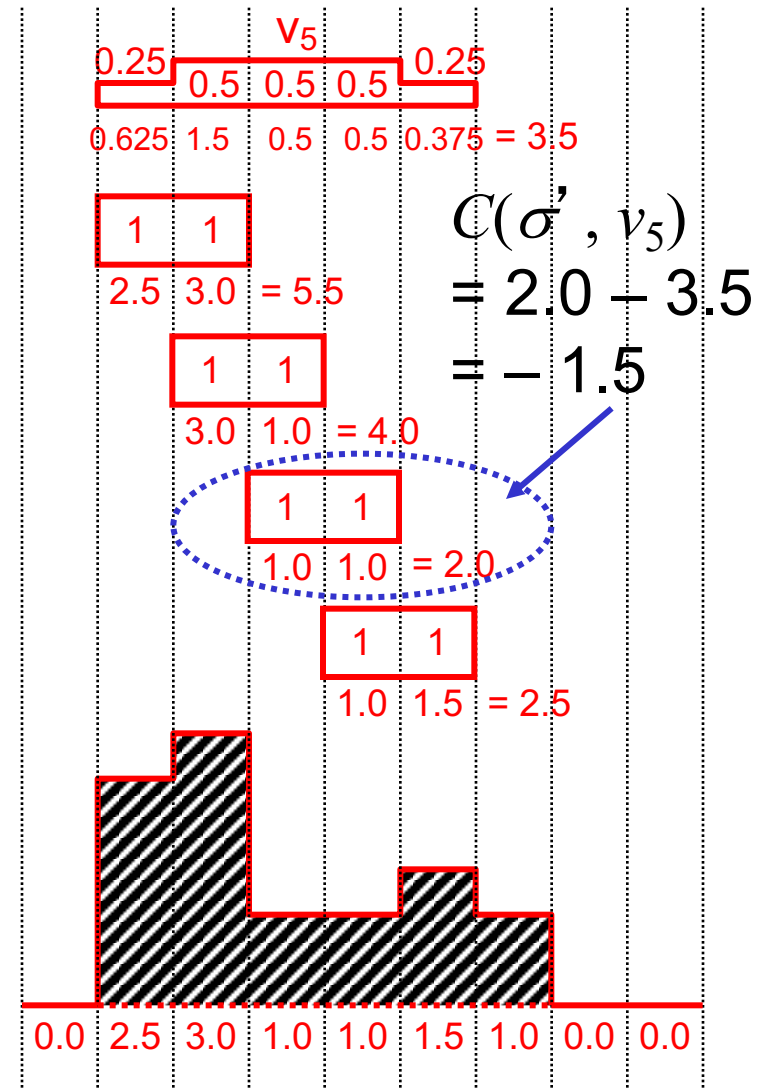
$$\begin{aligned}\phi(v, k) &= 1 \quad (0 \leq k < \delta(v)) \\ &= 0 \quad (\text{otherwise})\end{aligned}$$

E) Operation scheduling cost :

$$C(\sigma', v, t) = F(\sigma', v, t) - F(\sigma', v)$$

F) Minimum operation scheduling cost :

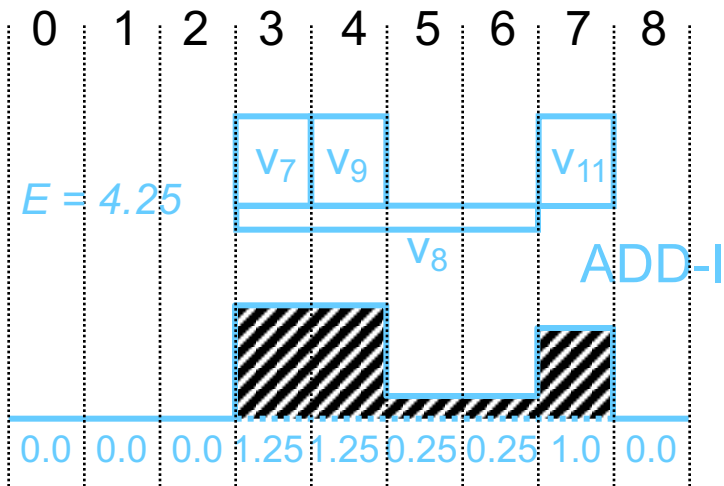
$$C(\sigma', v) = \min\{C(\sigma', v, t) \mid t \in T\}$$



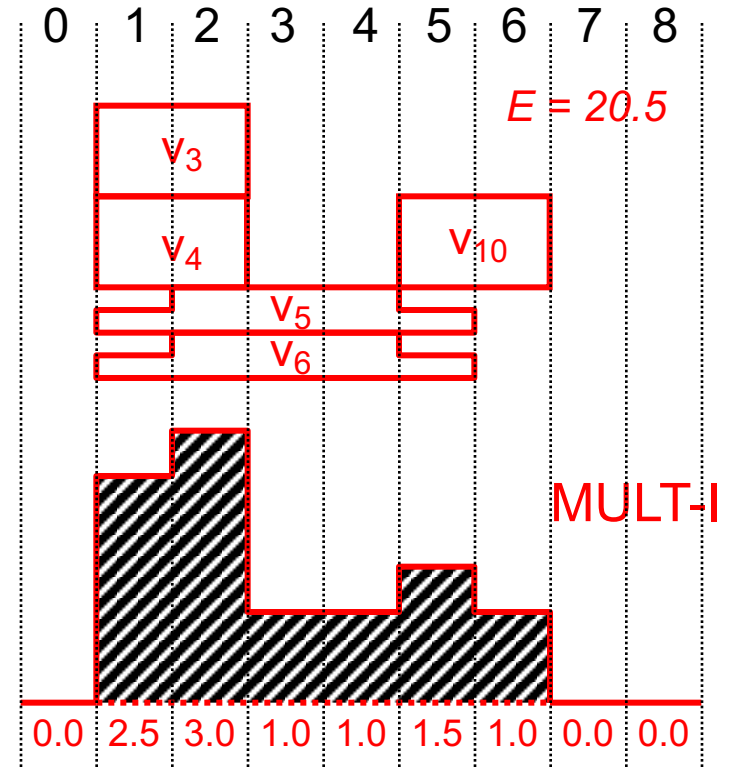
Force-Directed Scheduling (5)

G) Optimal scheduling refinement

($\sigma(v_5) \rightarrow 3$ or $\sigma(v_6) \rightarrow 3$)



ADD-I	v_7	v_8	v_9	v_{11}
$F(\sigma', v)$	1.25	0.75	1.25	1.0
$C(\sigma', v)$	-	-0.5	-	-
t_{opt}	-	5, 6	-	-

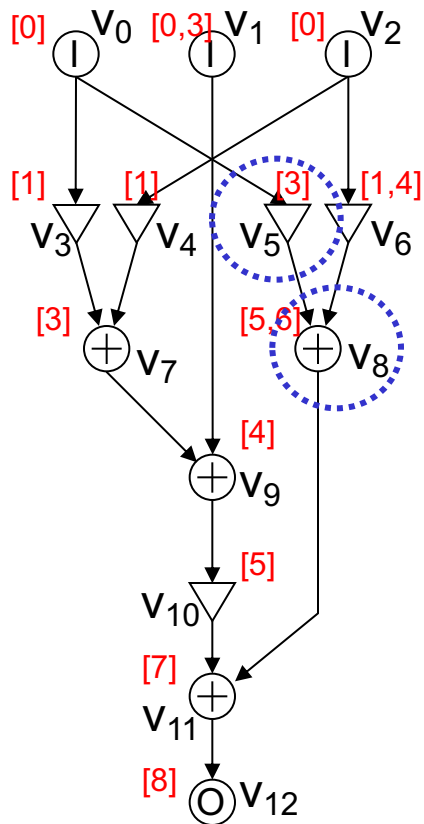


MULT-I	v_3	v_4	v_5	v_6	v_{10}
$F(\sigma', v)$	5.5	5.5	3.5	3.5	2.5
$C(\sigma', v)$	-	-	-1.5	-1.5	-
t_{opt}	-	-	3	3	-

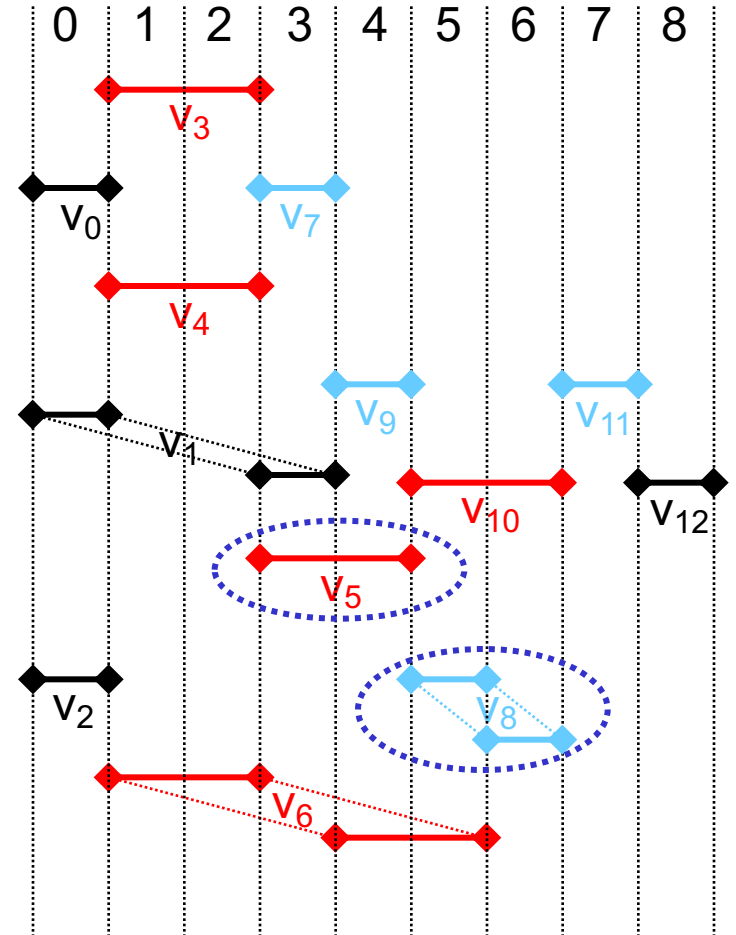
Force-Directed Scheduling (6)

H) Update operation mobilities

$$(\sigma(v_5) \rightarrow 3)$$

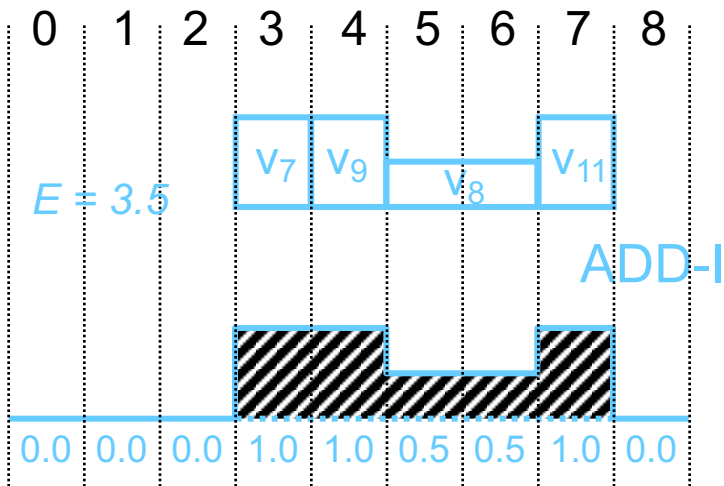


refining the scheduling to an operation affects mobilities of other operations

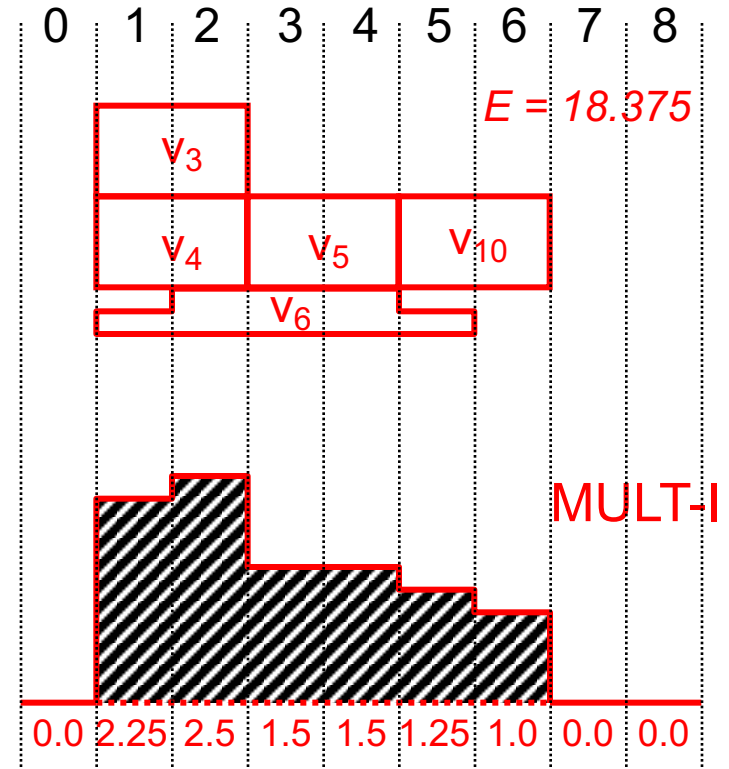


Force-Directed Scheduling (7)

G) Optimal scheduling refinement
($\sigma(v_6) \rightarrow 4$)



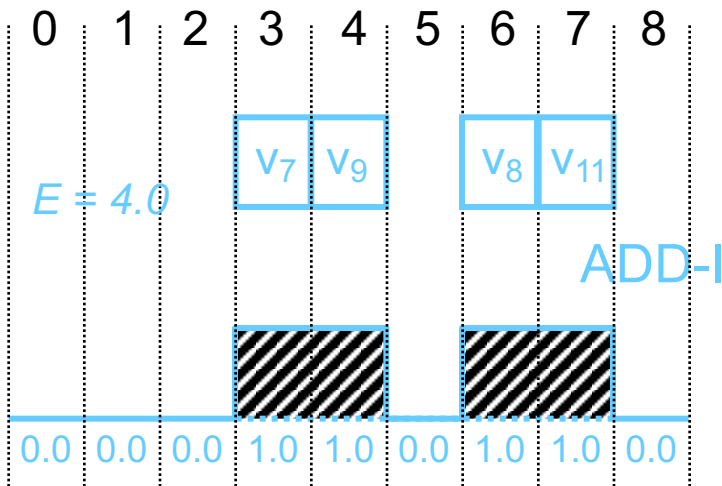
ADD-I	v_7	v_8	v_9	v_{11}
$F(\sigma', v)$	1.0	0.5	1.0	1.0
$C(\sigma', v)$	-	0	-	-
t_{opt}	-	5, 6	-	-



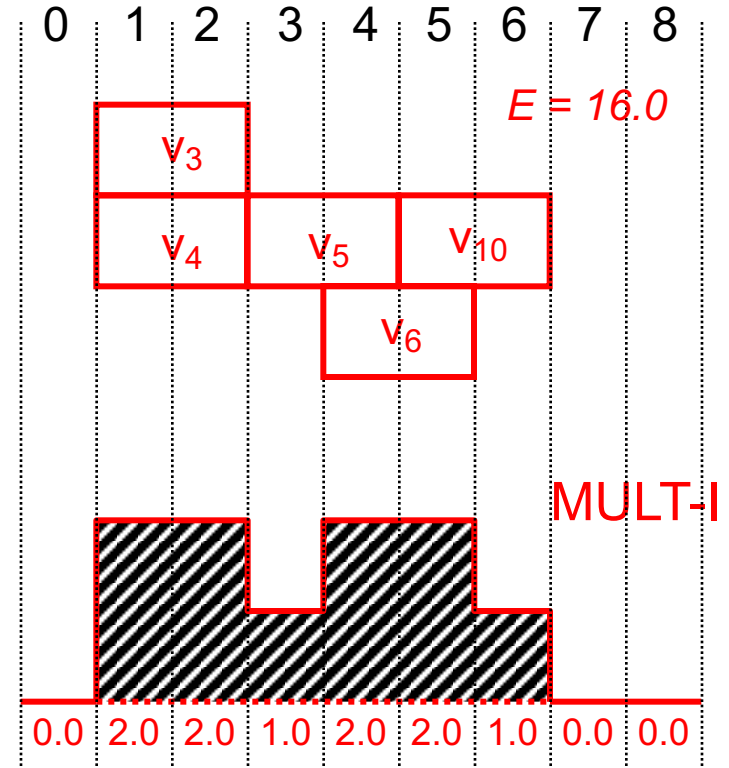
MULT-I	v_3	v_4	v_5	v_6	v_{10}
$F(\sigma', v)$	4.75	4.75	3.0	3.625	2.25
$C(\sigma', v)$	-	-	-	-0.875	-
t_{opt}	-	-	-	4	-

Force-Directed Scheduling (8)

G) Optimal scheduling refinement
($\sigma(v_6) \rightarrow 4$)



ADD-I	v_7	v_8	v_9	v_{11}
$F(\sigma', v)$	1.0	1.0	1.0	1.0
$C(\sigma', v)$	-	-	-	-
t_{opt}	-	-	-	-



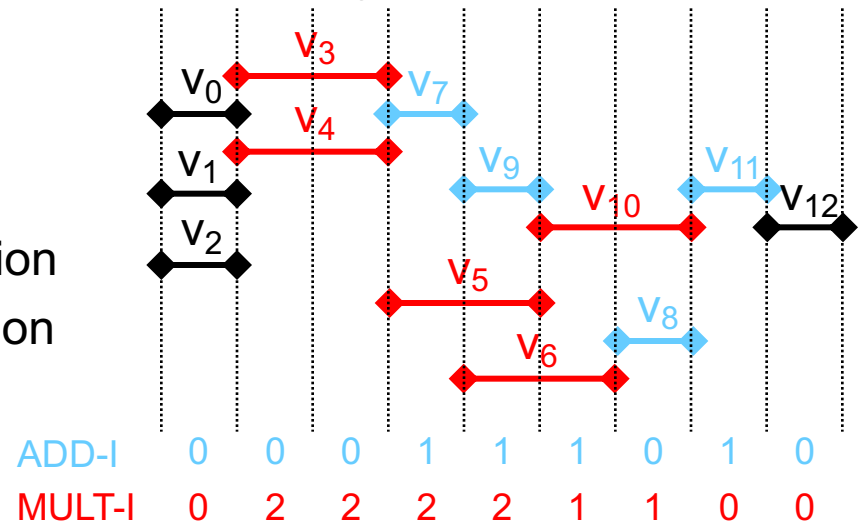
MULT-I	v_3	v_4	v_5	v_6	v_{10}
$F(\sigma', v)$	4.0	4.0	3.0	4.0	3.0
$C(\sigma', v)$	-	-	-	-	-
t_{opt}	-	-	-	-	-

Force-Directed Scheduling (9)

□ *Algorithm summary*

1. Compute ASAP and ALAP scheduling
2. Choose optimal scheduling refinement
 - Operation scheduling distribution
 - Resource occupation distribution
 - Operation scheduling cost
3. Update operation mobilities
4. If there are unscheduled operations, go to 2. Otherwise, END.

Final scheduling



Improvements in Force-Directed Scheduling

- Refining the scheduling for the target operation can affect the mobilities of other operations
 - Consider the *indirect forces* : forces of predecessors (connecting to input ports) and successors (connecting to output ports) of the target operation
 - *BUT actually, this is not enough (mobility changes can occur beyond predecessors or successors)*

- Operation scheduling energy equation

$$F(\sigma', v, t) = \sum_{t' \in T} \phi(v, t' - t) \times r(\sigma', m, t')$$

does not consider the changes of resource occupation distribution by the tentative scheduling refinement of $\sigma(v) \rightarrow t$

- Lookahead cost evaluation :

$$F(\sigma', v, t) = \sum_{t' \in T} \phi(v, t' - t) \times \underbrace{(r(\sigma', m, t') - \theta(\sigma', v, t))}_{\text{operation distribution energy}} + \underbrace{\phi(v, t' - t)}_{\text{operation occupancy}}$$

Correct this!

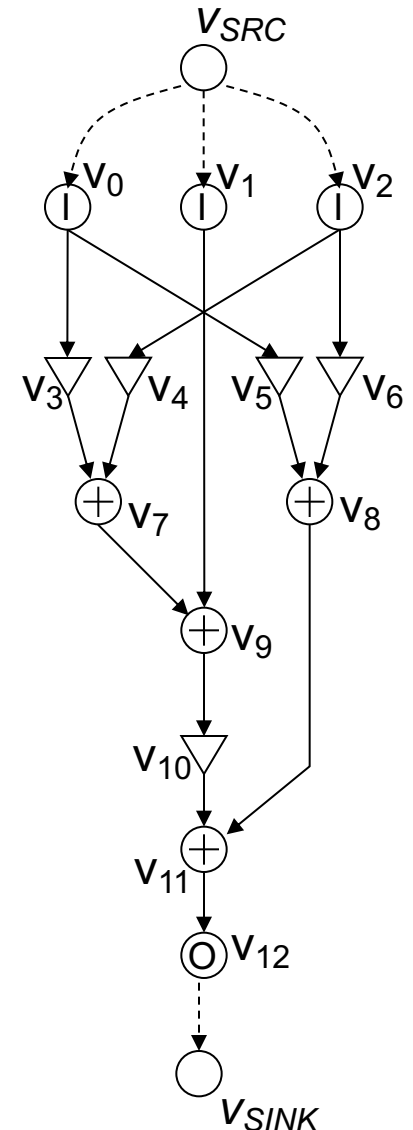
Correct this!

Force-Directed Scheduling Summary

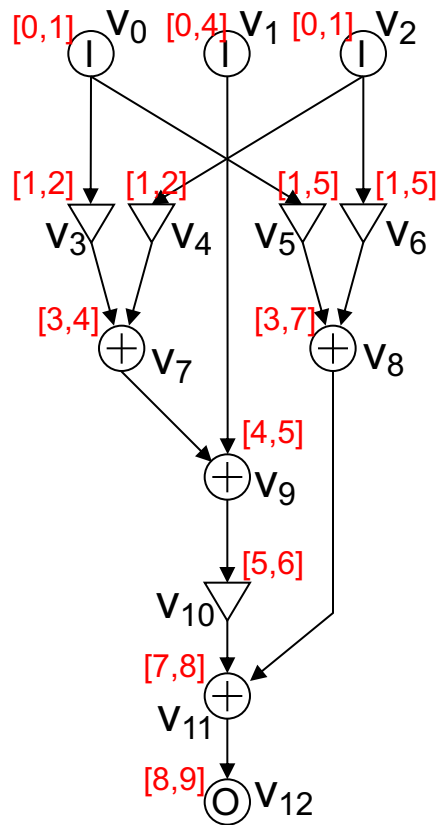
- Very popular time-constrained scheduling algorithm.
- Uses “forces” to balance the operation concurrency for high utilization of functional units.
- Cannot enforce resource constraints, (can only attempt to minimize them)

List Scheduling (1)

- Resource allocation : $R = \{ r(m) \mid m \in M \}$
- Start from $t = 0$, and increase t until all operations have been scheduled (let $\delta(v_{SRC}) = 0$, $\sigma(v_{SRC}) = 0$)
- Condition for operation v_j to be scheduled at t :
 - ✓ $\sigma(v_i) + \delta(v_i) \leq t$ for $\forall e_{ij} = (v_i, v_j) \in E$
(all predecessors of v_j must be scheduled)
 - ✓ $r(\sigma, \rho(v_j), t) \leq r(\rho(v_j))$
(resource occupancy must not exceed the constraint)
- If there are more operations to be scheduled than the resource constraint, choose the operations according to some priority function
 - ✓ Mobility $\mu(v)$: smaller mobility has higher priority
 - ✓ Longest path to v_{SINK} : longer path has higher priority



List Scheduling (2)

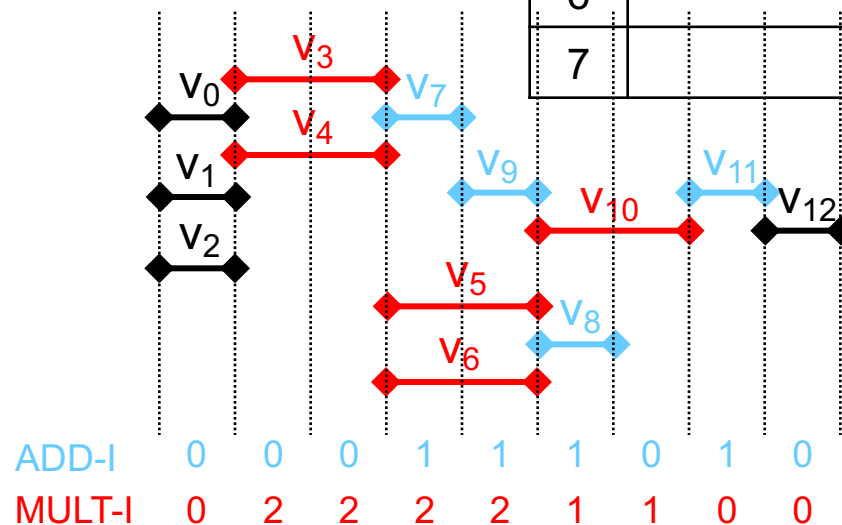


$$r(\text{MULT-I}) = 2$$

$$r(\text{ADD-I}) = 1$$

scheduled operations

t	Ready list (MULT-I)	Ready list (ADD-I)
0		
1	3[1,2], 4[1,2], 5[1,5], 6[1,5]	
2	5[2,5], 6[2,5]	
3	5[3,5], 6[3,5]	7[3,4]
4		9[4,5]
5	10[5,6]	8[5,7]
6		
7		11[7,8]



List Scheduling Summary

- Very simple, easy to implement
- Cannot enforce time constraints
- Scheduling quality depends on the definition of priority function used.
 - Scheduling quality depends on the definition of priority function used.

Other Topics on Scheduling Problems

- More realistic resource cost function
 - Not only # functional units, but also # registers, # buses, # IO ports
 - Formulate these costs in the force-directed scheduling
- Parallelism limited inside basic-blocks (data-flow graph)
 - Path-based scheduling : all control paths are extracted and scheduled independently (therefore, basic-block boundaries can be ignored), and later combined to obtain the overall scheduling.

High-Level Synthesis Flow

- A) Design capture (HDLs, C/C++, signal-flow graph, etc)
- B) Compilation to internal representation
 - Data-flow graph (DFG)
 - Control-flow graph (CFG)
 - Control-data-flow graph (CDFG)
- C) Resource allocation
 - Specify available functional units
- D) Operation scheduling
 - Assign each operation to control steps
- E) Resource binding
 - Assign each data to registers
 - Assign each operation to functional units