Scheduling

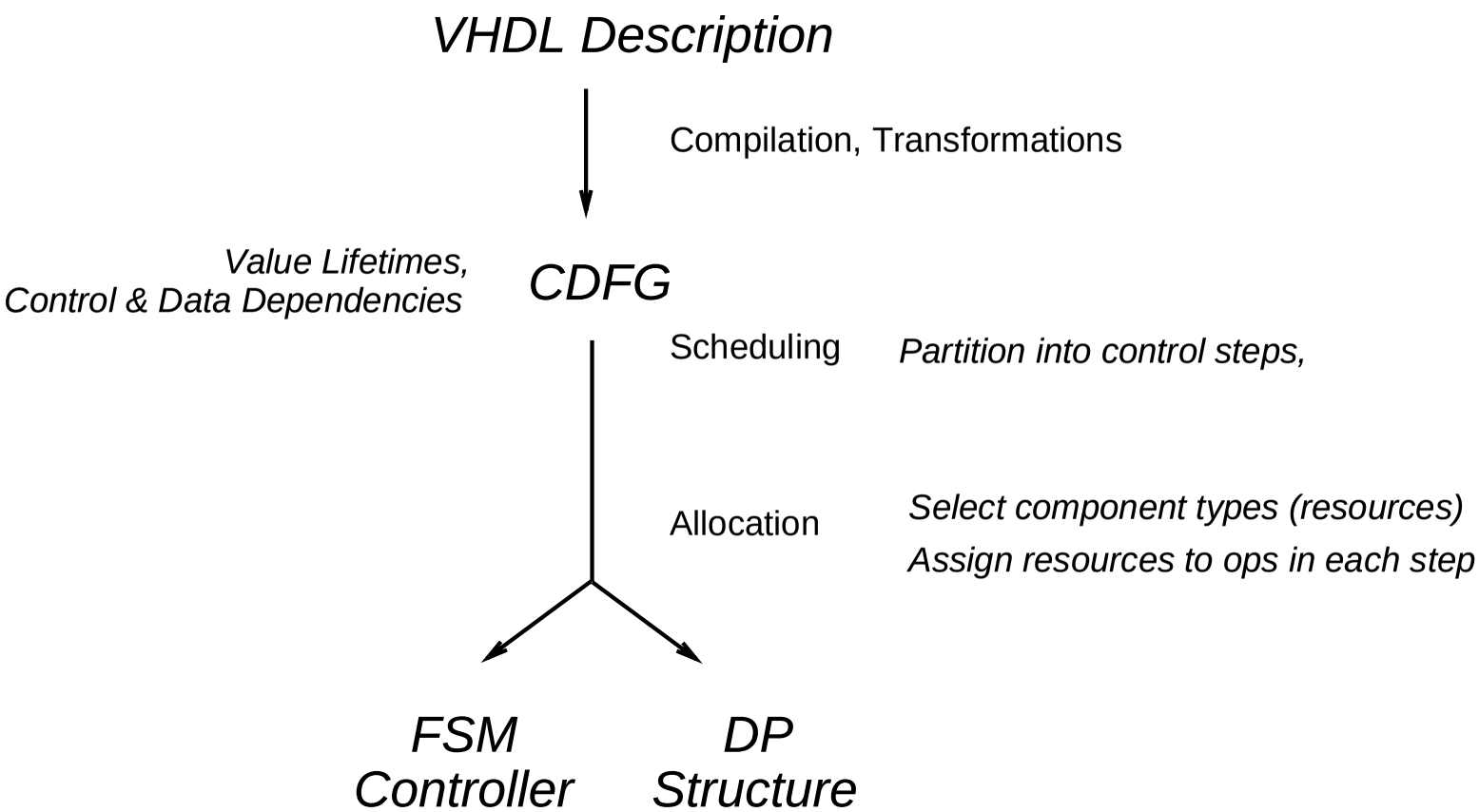
Chapter 7

Source: Gajski, Dutt, Wu, Lin

"High−Level Synthesis"

Kluwer Academic Publishers, 1992

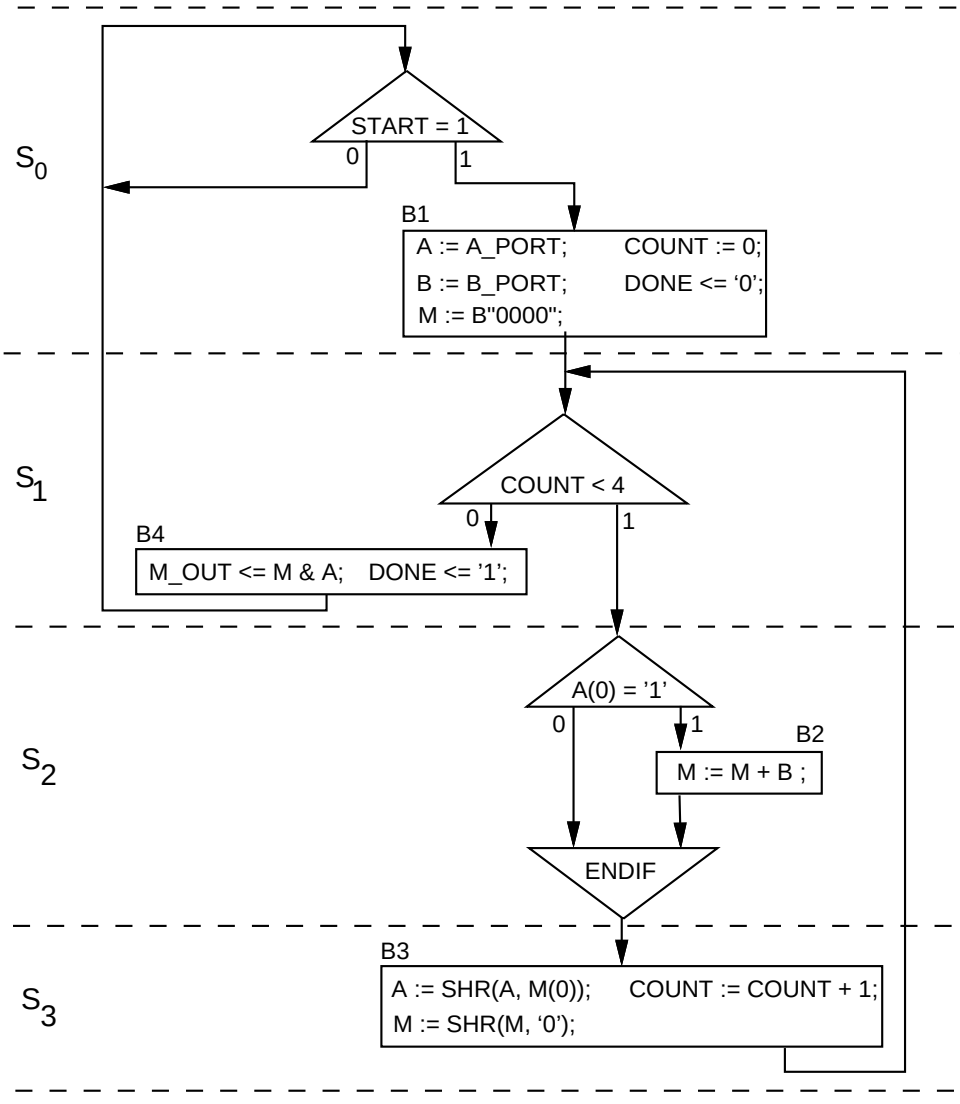
High−Level Synthesis Trajectory



Scheduling

- *Definition:* Task of assigning behavioral operators to control steps
- *Input:* CDFG (data lifetimes with control and data dependencies)
- *Output:* Temporal ordering of individual operations (FSM states)
- *Constraints:* Hardware Resources, Timing, Testability, Power, ...
- *Goal:* Exploit parallelism to achieve fastest design within constraints

SCHEDULED CDFG



Scheduling: Assumptions

- *Target Architecture:*

  Pipelining, Clocking, Busing, Component Sets, ...

- *Language Constructs Permitted:*

  Conditionals, Loops, Arrays, ADTs,...

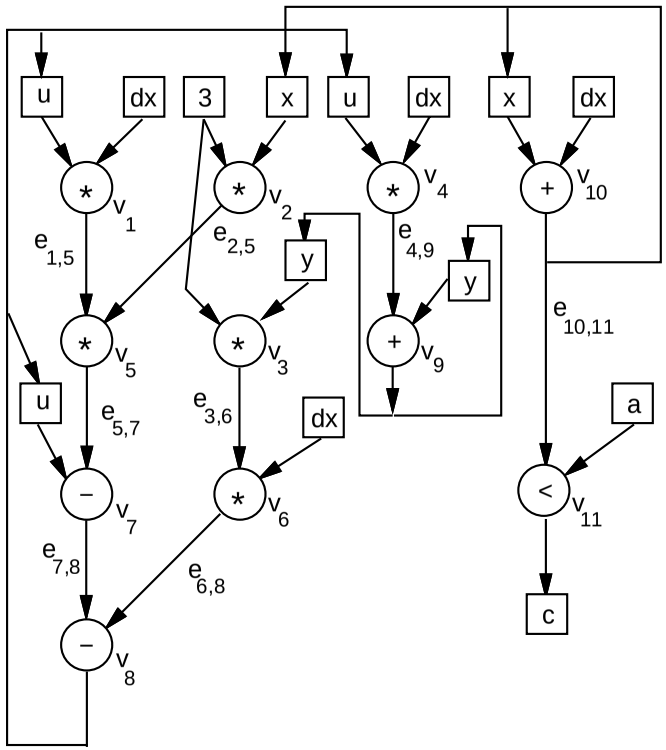
- *Temporary Assumptions for Illustration:*

  1. No pipelining, single−phase clock
  2. Straight−line code, simple data−types
  3. Each operation executes in 1 control step
  4. Each operation performed by 1 component type

Scheduling Example: HAL

```
while (x < a) loop
    x1 := x + dx;
    u1 := u − (3 * x * u * dx) − (3 * y * dx);
    y1 := y + (u*dx);
    x  := x1; u := u1; y := y1;
end loop;
```

VHDL Behavior



DFG Representation

ASAP ALGORITHM

Algorithm 7.1

page 217

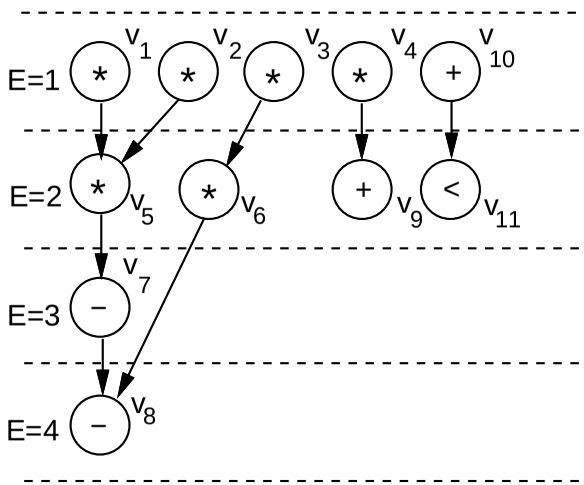
ALAP ALGORITHM

Algorithm 7.2

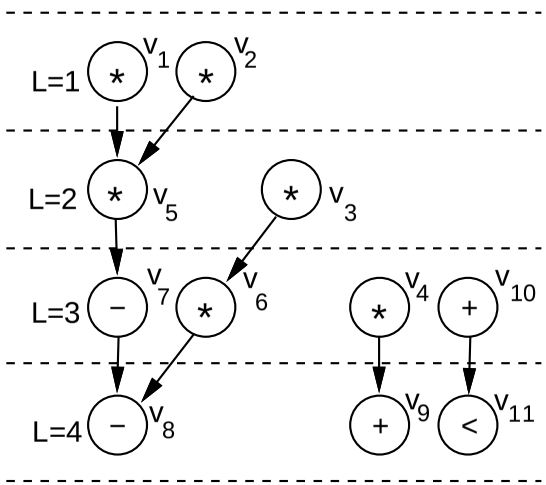
page 219

(Check errata)

HAL: ASAP and ALAP Schedules



ASAP Schedule



ALAP Schedule

Scheduling Formulations

- **Resource Constrained Scheduling:**
  Minimize control steps for given resources

  List−Based Scheduling

  Static−List Scheduling [JMSW91]

- **Time Constrained Scheduling:**
  Minimize resources for given number of time steps

  Integer Linear Programming(ILP) [LeHL89]

  Force−Directed Scheduling (FDS) [PaKn89]

  Iterative Refinement [PaKy91]

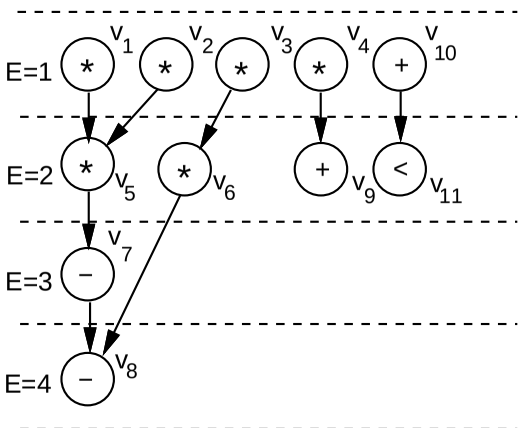
LIST−BASED SCHEDULING

Algorithm 7.5

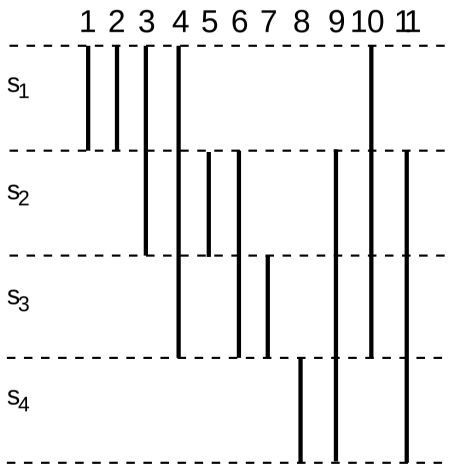
page 234

HAL: List Scheduling

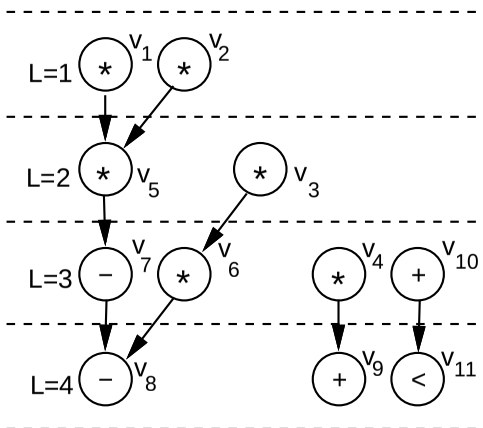
ASAP Schedule



Operator Mobility



ALAP Schedule



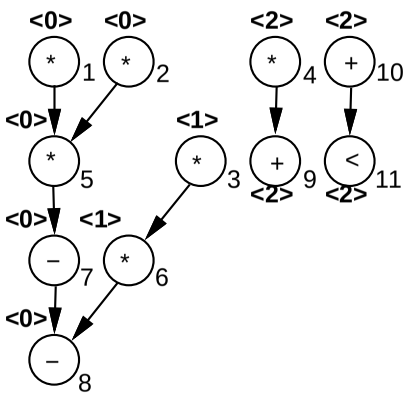
Mobility(op) = ASAP − ALAP

Node:	*v1*	*v2*	*v3*	*v4*	*v5*	*v6*	*v7*	*v8*	*v9*	*v10*	*v11*
Operation:	*	*	*	*	*	*	−	−	+	+	<
Mobility:	*0*	*0*	*1*	*2*	*0*	*1*	*0*	*0*	*2*	*2*	*2*

- *Maintain priority list for each component type*
- *Schedule critical nodes first*

List Scheduling (Cont'd)

- *ASAP + Priority function (mobility) for each resource*
- *Resource conflicts resolved by priority function*
- *Constructive Method: schedule, reevaluate priority, sort,...*



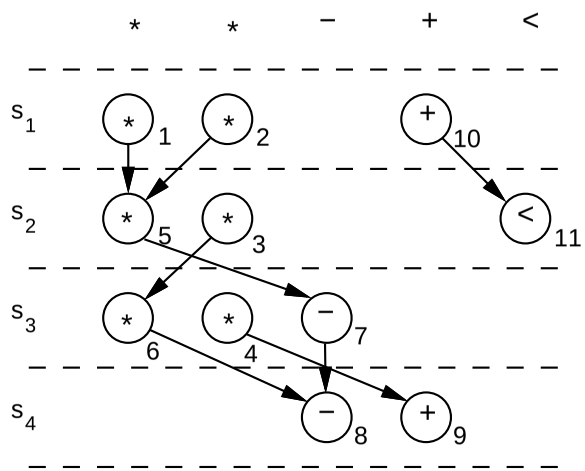
DFG with mobilities

$PList_{*}$: 1<0>, 2<0>, 3<1>, 4<2>
$PList_{+}$: 10<2>
$PList_{-}$: NIL
$PList_{<}$: NIL

Ready List

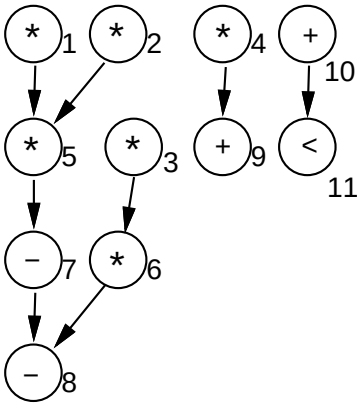
$Resources_{*}$: 2
$Resources_{+}$: 1
$Resources_{-}$: 1
$Resources_{<}$: 1

Resource Constraints



Scheduled DFG

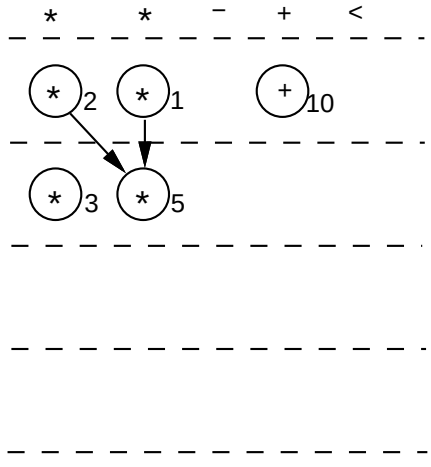
STATIC–LIST SCHEDULING



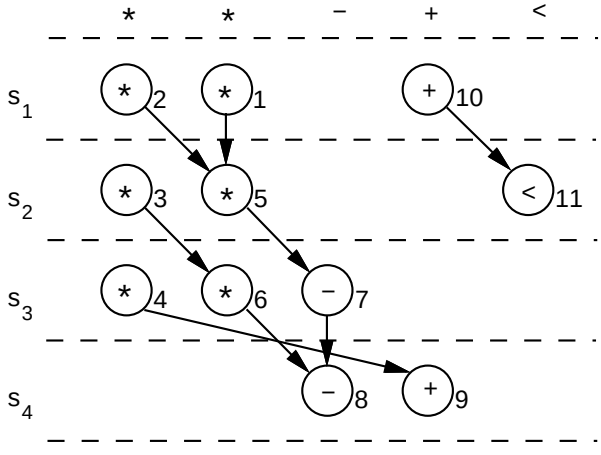
DFG

node	8	9	11	7	6	4	10	5	3	1	2
ALAP	1	1	1	2	2	2	2	3	3	4	4
ASAP	4	2	2	3	2	1	1	2	1	1	1
priority	1	2	3	4	5	6	7	8	9	10	11

Priority List



Partial Schedule



Final Schedule

Scheduling Formulations

- **Resource Constrained Scheduling:**
  Minimize control steps for given resources

  List−Based Scheduling

  Static−List Scheduling [JMSW91]

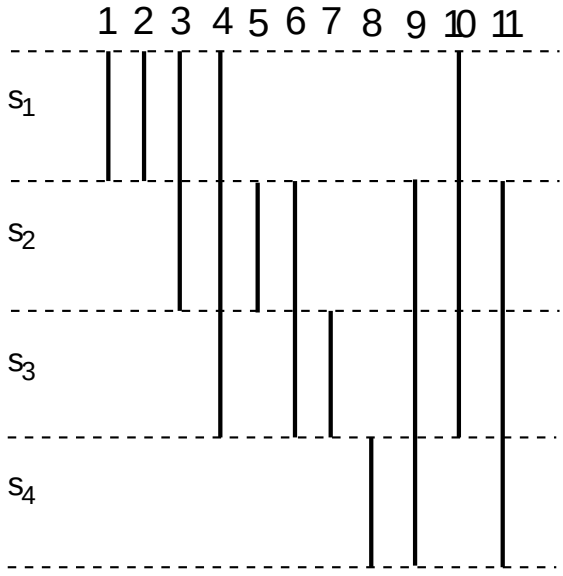
- **Time Constrained Scheduling:**
  Minimize resources for given number of time steps

  Integer Linear Programming(ILP) [LeHL89]

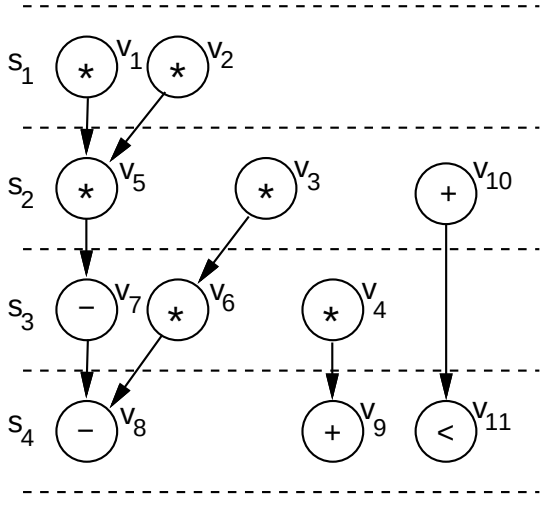
  Force−Directed Scheduling (FDS) [PaKn89]

  Iterative Refinement [PaKy91]

HAL: ILP FORMULATION



Operation ranges



Final Schedule

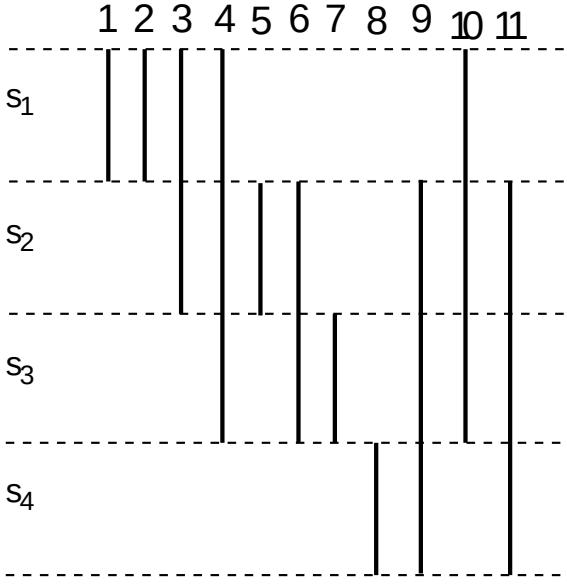
ILP FORMULATION

Formulation on

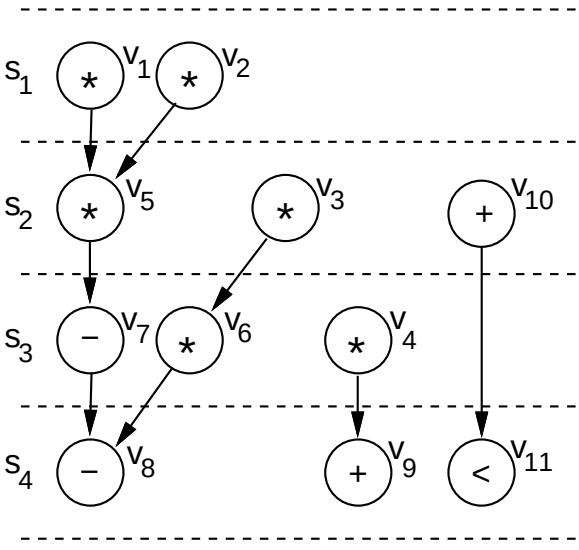
pages 221−222

(Check errata)

HAL: ILP FORMULATION



Operation ranges



Final Schedule

Eq. 7.5 with constraints (pages 222−223)

Force−Directed Scheduling [PaKn89]

- *Time−constrained scheduling, minimize resources*
- *Goal:* Achieve high unit utilization by uniformly distributing operations of a particular type over all states
- *Iterative Approach:*

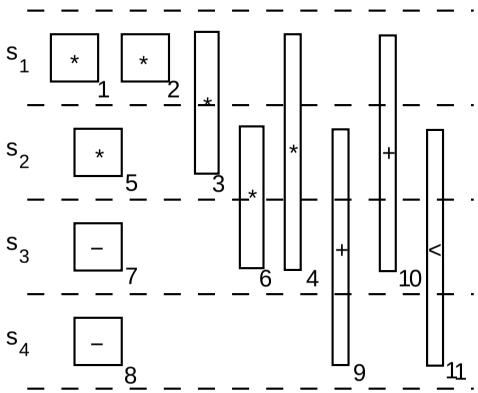
  Determine global effect of scheduling an operation into a state

  - *Compute probability of scheduling op into a state*
  - *Compute expected operator cost for each op type*

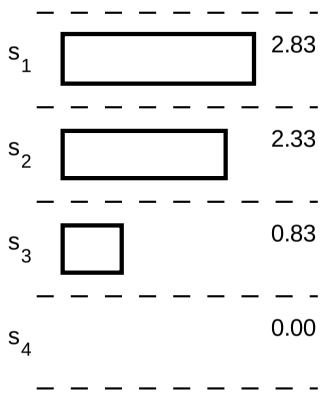
  Schedule each operation to balance hardware utilization

  - *Choose assignment of op to state with min cost*
  - *Recompute probability distr. and expected op costs*

HAL: Force−Directed Scheduling



Probabilities of op scheduling
(uniform distr from mobility)

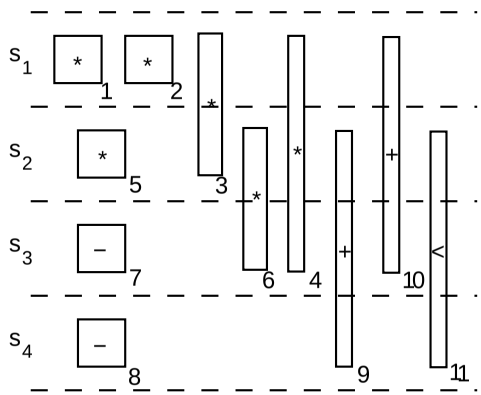


Operator cost for Mult (*)
(sum of op prob's per state)

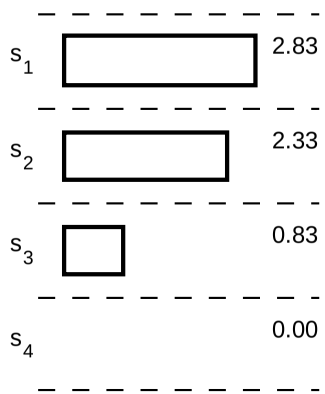
For each iteration of FDS do:

- Compute Expected Operator Costs (EOC) for scheduling each operator type in every state
- Assign one operation to control step based on minimal EOC

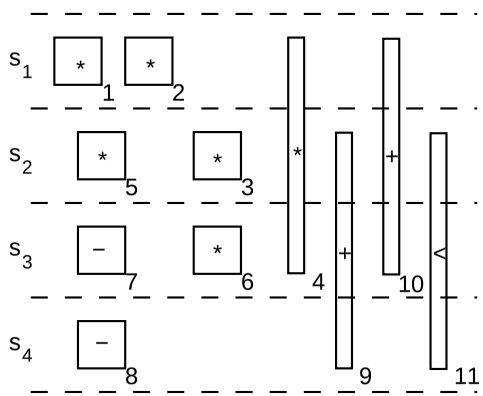
HAL: Force−Directed Scheduling(Cont'd)



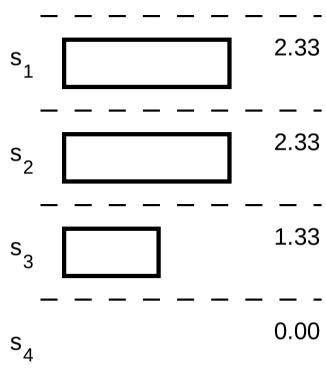
Probabilities of op scheduling



Operator cost for Mult (*)



Probabilities after o3 is scheduled in s2



Revised operator cost for Mult (*)

FORCE–DIRECTED SCHEDULING

Algorithm 7.3

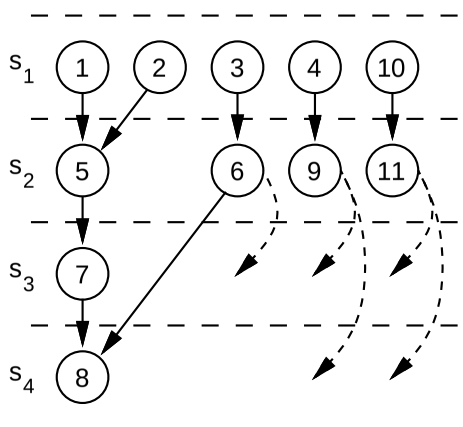
page 227

Iterative Refinement Scheduling [PaKy91]

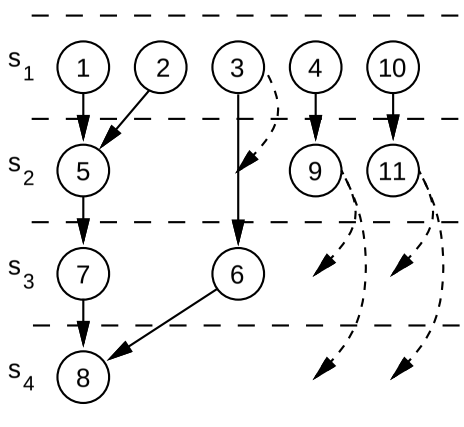
FDS: Constructive, no backtracking (rescheduling)

Iterative Refinement Scheduling: allows rescheduling

- *Based on KL Graph Bisection [KeLi70]*
- *Start with any initial schedule*
- *Reschedule one operation at a time, and lock operation*
- *Maximize cumulative gain*



Initial Schedule



After Op 6 is moved and locked

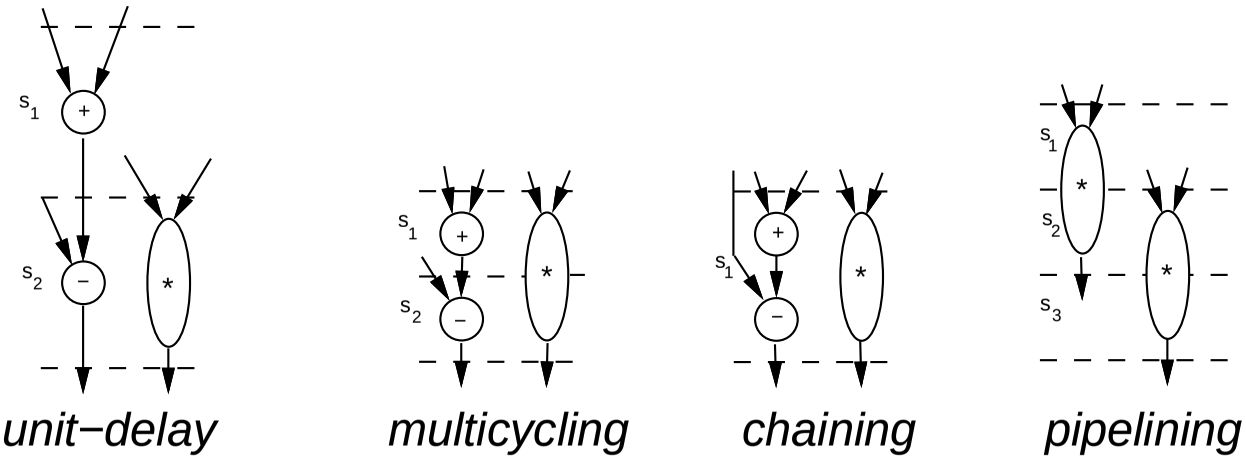
ITERATIVE RESCHEDULING

Algorithm 7.4

page 231

Scheduling with Realistic Assumptions

- Functional Units with Varying Delays



unit−delay *multicycling* *chaining* *pipelining*

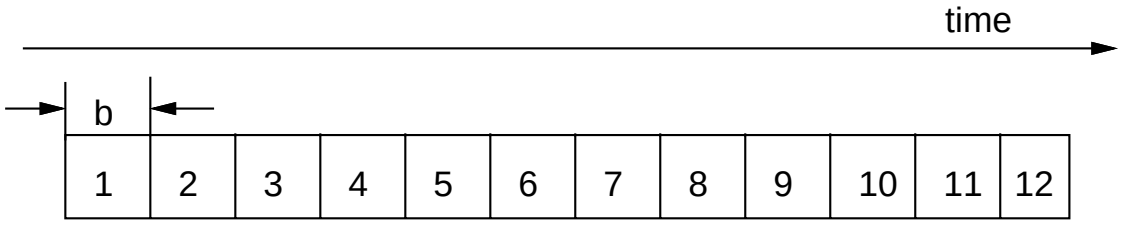
- Multi−Functional Units

  ALUs, Shift−registers, etc.

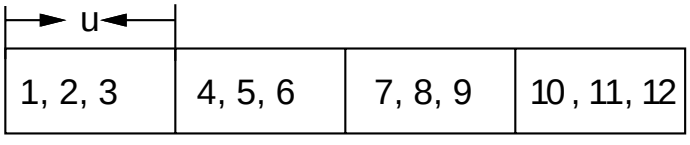
- Realistic Design Descriptions

  Conditionals, loops, nested loops

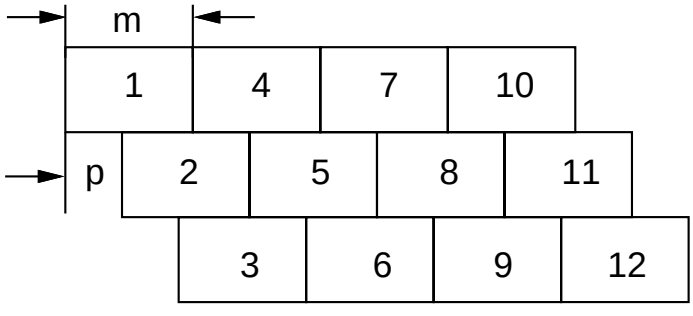
LOOP SCHEDULING



Sequential Execution

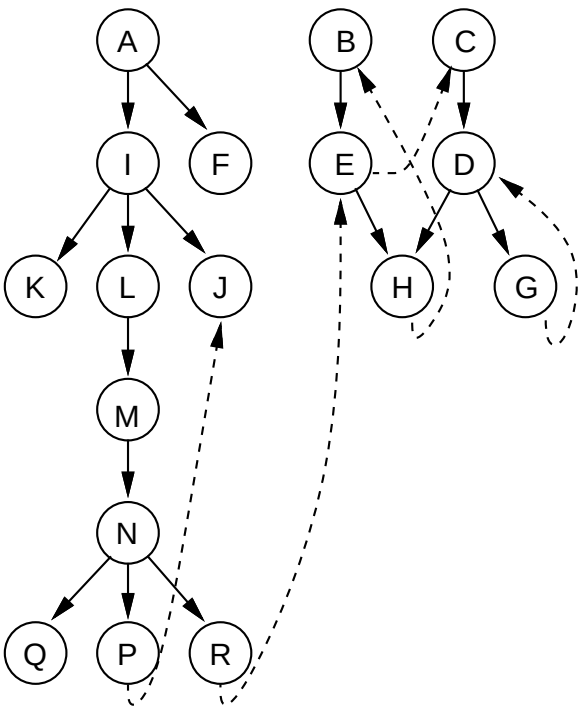


Partial Loop Unrolling

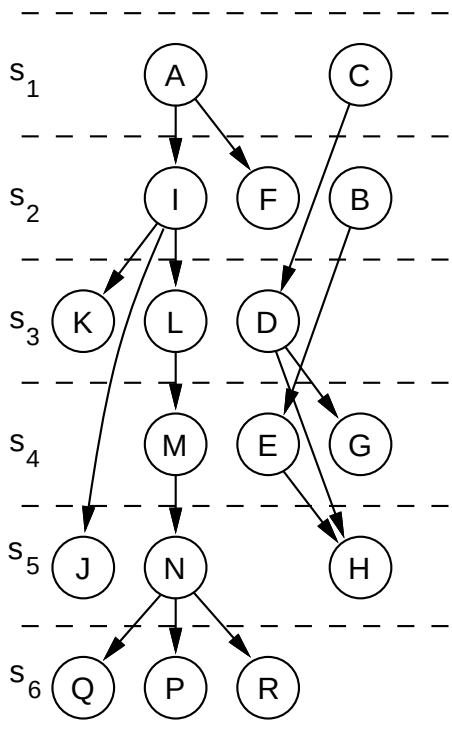


Loop Folding

LOOP SCHEDULING (Cont'd)

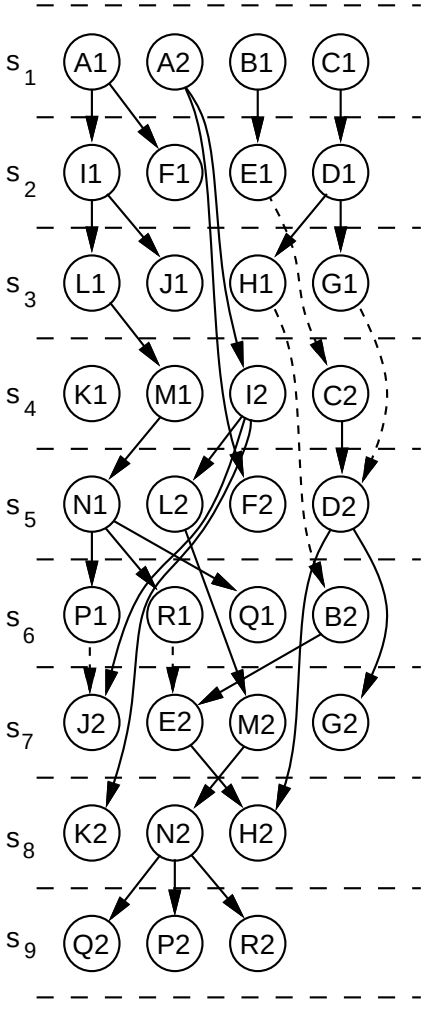


DFG with Depend. Across Iterations

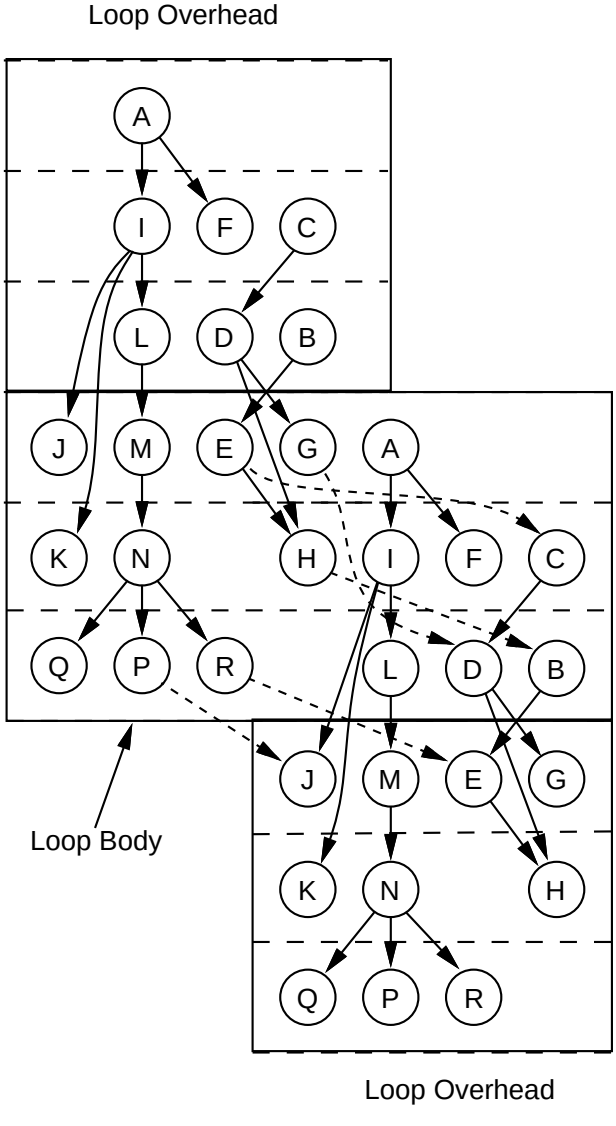


Seq. Schedule w/ 3 FU's

LOOP UNROLLING AND LOOP UNFOLDING



Loop Unrolling



Loop Folding

SIMULATED ANNEALING FORMULATION

	ALU1	ALU2	ALU3
s1	v1 = v2 + v3	v4 = v2 * v3	
s2	v5 = v1 + v4		v6 = v4 / v1
s3		v2 = v4 * v5	

Initial Schedule

	ALU1	ALU2	ALU3
s1	v4 = v2 * v3	v1 = v2 + v3	
s2	v5 = v1 + v4		v6 = v4 / v1
s3		v2 = v4 * v5	

After Swapping Two Operations

	ALU1	ALU2	ALU3
s1	v4 = v2 * v3	v1 = v2 + v3	
s2	v5 = v1 + v4		v6 = v4 / v1
s3	v2 = v4 * v5		

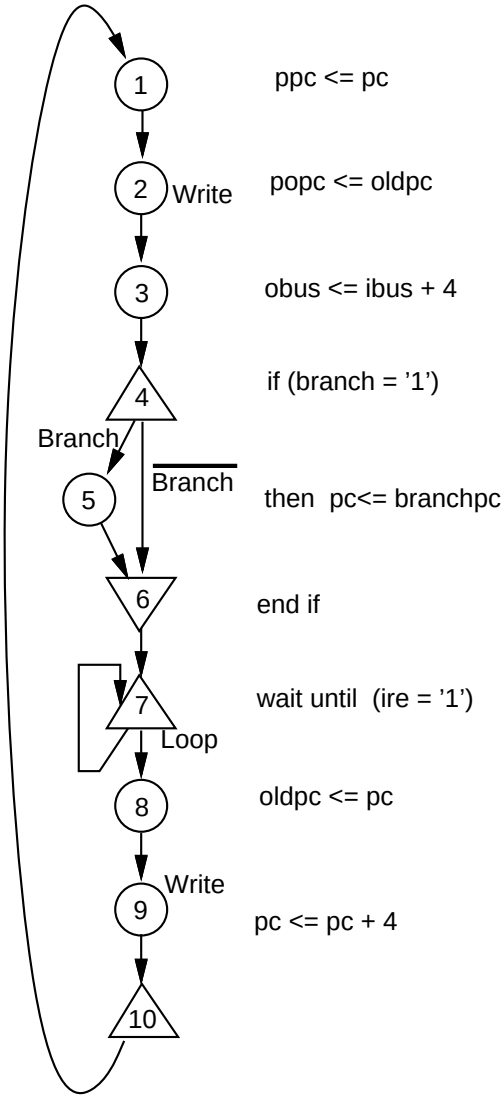
After Displacing an Operation

PATH−BASED SCHEDULING

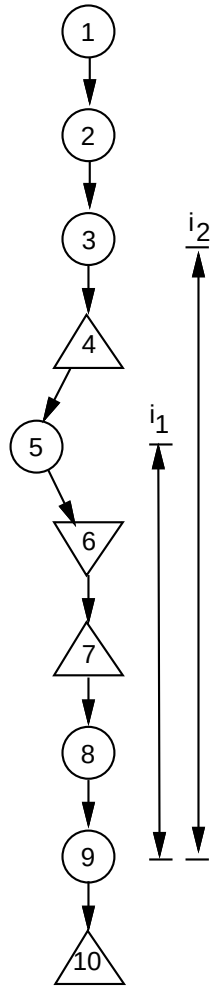
Input Ports: branchpc, ibus, branch, ire

Output Ports: ppc, popc, obus

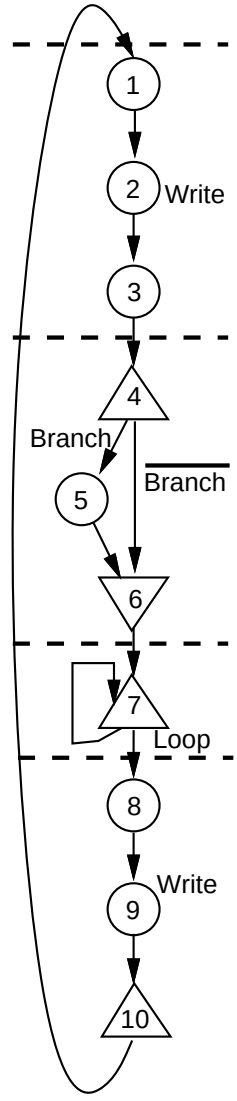
Variables: pc, oldpc



CDFG

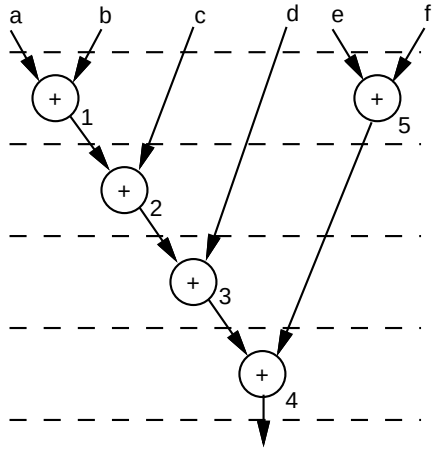


Constraint Intervals



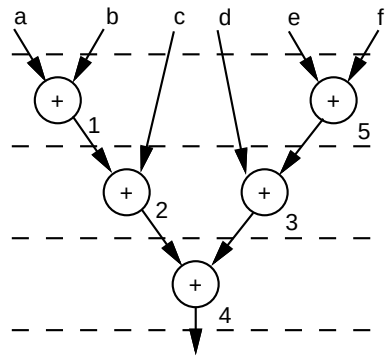
Scheduled CDFG

DFG RESTRUCTURING



(((a + b) + c) + d) + (e + f)

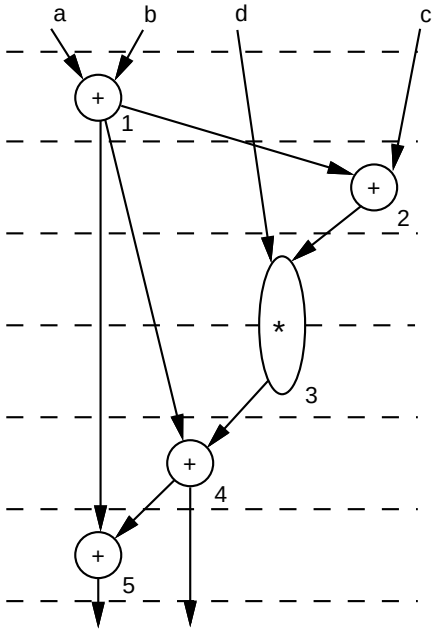
before



((a + b) + c) + (d + (e + f))

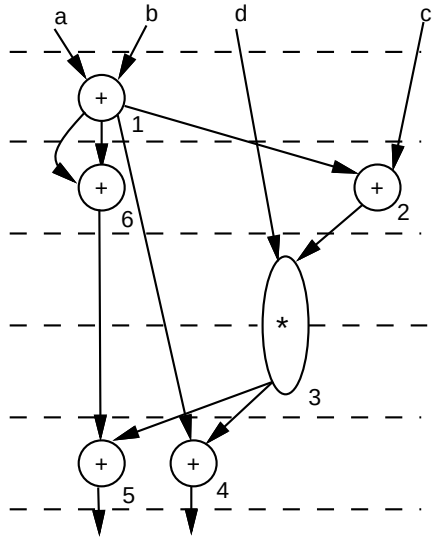
after

Tree−Height Reduction



d (a + b + c) + ab + ab

before



d (a + b + c) + 2ab

after

Redundant Operator Insertion

Scheduling: Future Directions

- More realistic libraries
- Realistic target architectures
- Better cost functions (layout−driven)
- Scheduling with allocation
- Arbitrary descriptions (nested loops, conditionals)
- Loop−pipelining, tree−height reduction
- Complex data structures (arrays, ADT)
- Application−specific scheduling (RISC, VLIW, DSP,..)