

ESE535: Electronic Design Automation

Day 4: January 28, 2009
Scheduling Introduction



Penn ESE535 Spring 2009 -- DeHon

Today

- Scheduling
 - Basic problem
 - Variants
 - List scheduling approximation

2

Penn ESE535 Spring 2009 -- DeHon

General Problem

- Resources are not free
 - wires, io ports
 - functional units
 - LUTs, ALUs, Multipliers,
 - memory locations
 - memory access ports

3

Penn ESE535 Spring 2009 -- DeHon

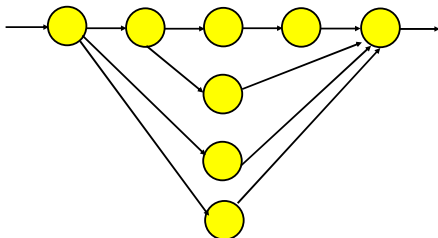
Trick/Technique

- Resources can be shared (reused) in time
- Sharing resources can reduce
 - instantaneous resource requirements
 - total costs (area)
- **Pattern:** scheduled operator sharing

4

Penn ESE535 Spring 2009 -- DeHon

Example



5

Penn ESE535 Spring 2009 -- DeHon

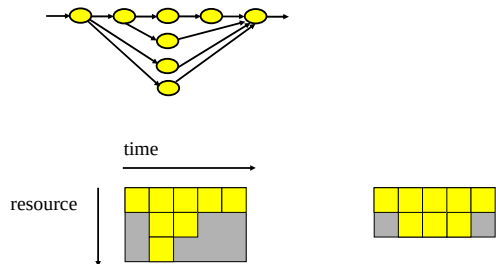
Sharing

- Does not have to increase delay
 - w/ careful time assignment
 - can often reduce peak resource requirements
 - while obtaining original (unshared) delay
- **Alternately:** Minimize delay given fixed resources

6

Penn ESE535 Spring 2009 -- DeHon

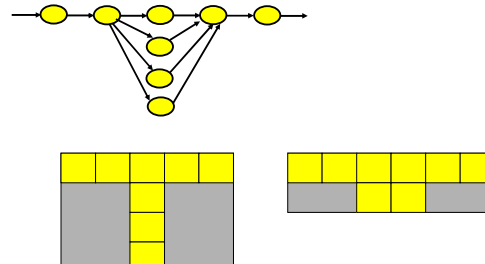
Schedule Examples



Penn ESE535 Spring 2009 -- DeHon

7

More Schedule Examples



Penn ESE535 Spring 2009 -- DeHon

8

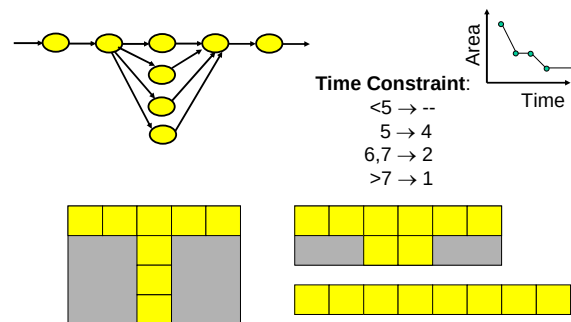
Scheduling

- **Task:** assign time slots (and resources) to operations
 - **time-constrained:** minimizing peak resource requirements
 - *n.b.* time-constrained, not always constrained to minimum execution time
 - **resource-constrained:** minimizing execution time

Penn ESE535 Spring 2009 -- DeHon

9

Resource-Time Example



Penn ESE535 Spring 2009 -- DeHon

10

Scheduling Use

- Very general problem formulation
 - HDL/Behavioral → RTL
 - Register/Memory allocation/scheduling
 - Instruction/Functional Unit scheduling
 - Processor tasks
 - TDMA, bus scheduling, static routing
 - Routing (share channel)

Penn ESE535 Spring 2009 -- DeHon

11

Two Types (1)

- **Data independent**
 - graph static
 - resource requirements and execution time
 - independent of data
 - schedule statically
 - maybe bounded-time guarantees
 - typical ECAD problem

Penn ESE535 Spring 2009 -- DeHon

12

Two Types (2)

- **Data Dependent**
 - execution time of operators variable
 - depend on data
 - flow/requirement of operators data dependent
 - if cannot bound range of variation
 - must schedule online/dynamically
 - cannot guarantee bounded-time
 - general case (i.e. halting problem)
 - typical “General-Purpose” (non-real-time) OS problem

Penn ESE535 Spring 2009 -- DeHon

13

Unbounded Problem

- **Easy:**
 - compute ASAP schedule
 - i.e. schedule everything as soon as predecessors allow
 - will achieve minimum time
 - won't achieve minimum area
 - (meet resource bounds)

Penn ESE535 Spring 2009 -- DeHon

14

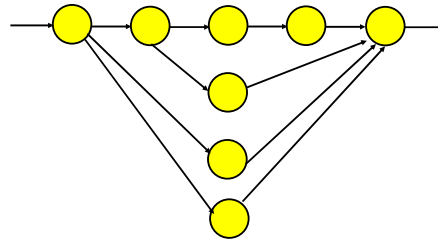
ASAP Schedule

- For each input
 - mark input on successor
 - if successor has all inputs marked, put in visit queue
- While visit queue not empty
 - pick node
 - update time-slot based on latest input
 - mark inputs of all successors, adding to visit queue when all inputs marked

Penn ESE535 Spring 2009 -- DeHon

15

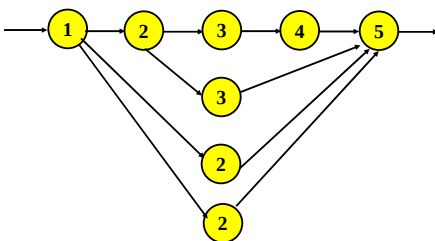
ASAP Example



Penn ESE535 Spring 2009 -- DeHon

16

ASAP Example



Penn ESE535 Spring 2009 -- DeHon

17

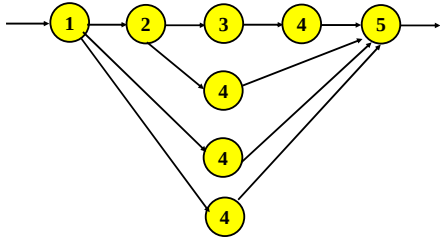
Also Useful to Define ALAP

- As Late As Possible
- Work backward from outputs of DAG
- Also achieve minimum time w/ unbounded resources

Penn ESE535 Spring 2009 -- DeHon

18

ALAP Example

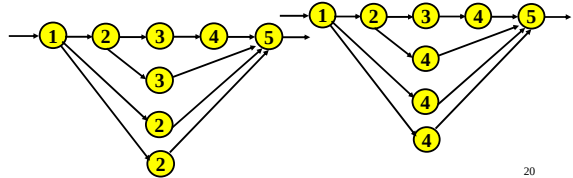


Penn ESE535 Spring 2009 -- DeHon

19

ALAP and ASAP

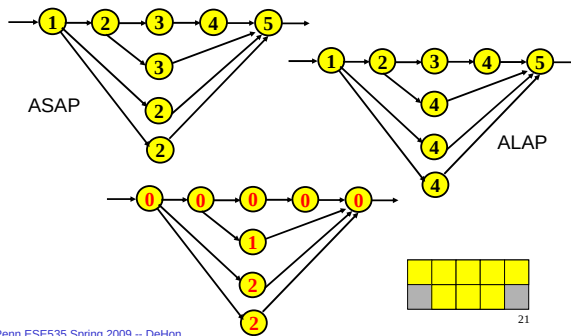
- Difference in labeling between ASAP and ALAP is slack of node
 - Freedom to select timeslot
 - **Class theme:** exploit freedom to reduce costs
- If ASAP=ALAP, no freedom to schedule



Penn ESE535 Spring 2009 -- DeHon

20

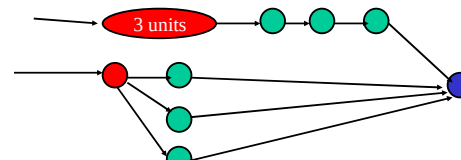
ASAP, ALAP, Difference



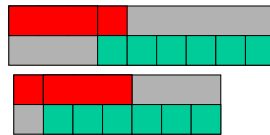
Penn ESE535 Spring 2009 -- DeHon

21

Why hard?



Start with Critical Path?



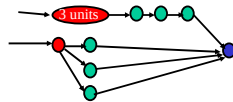
Schedule on:
1 Red Resource
1 Green Resource

Penn ESE535 Spring 2009 -- DeHon

22

General

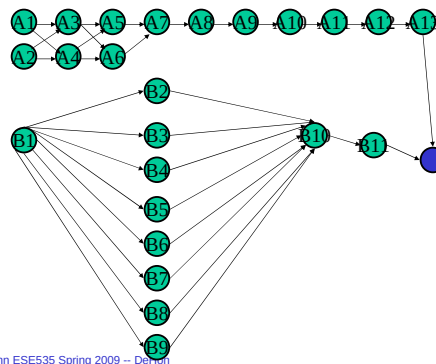
- When selecting, don't know
 - need to tackle **critical path**
 - need to run task to **enable work** (parallelism)
- Can generalize example to single resource case



Penn ESE535 Spring 2009 -- DeHon

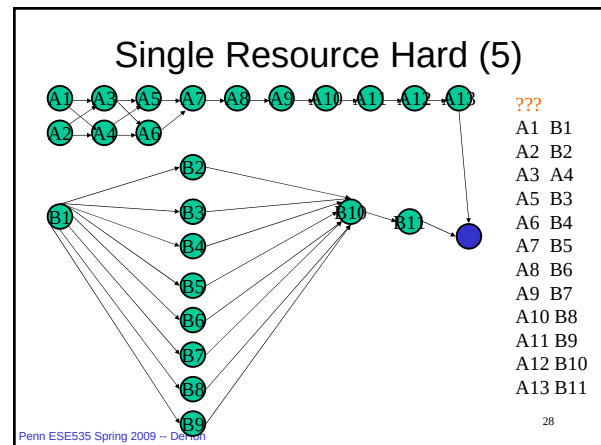
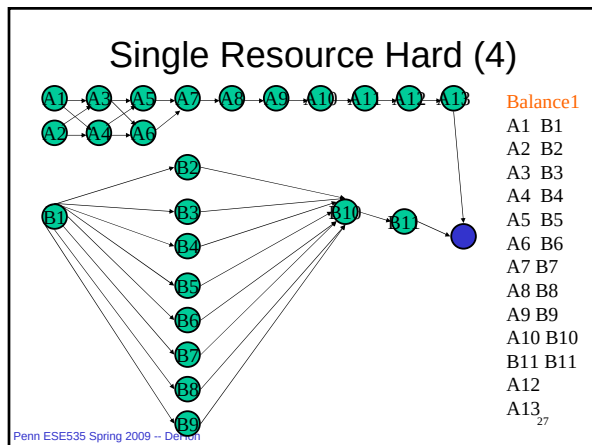
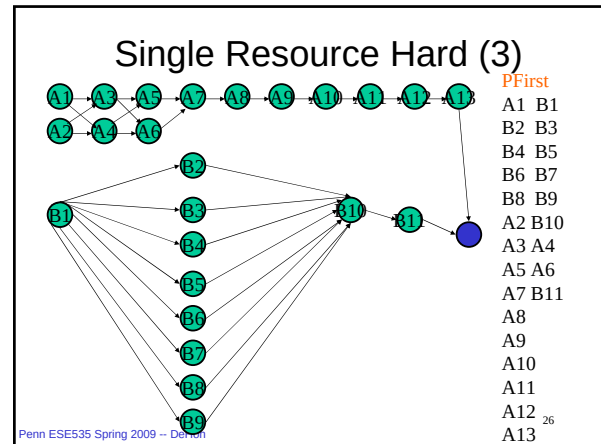
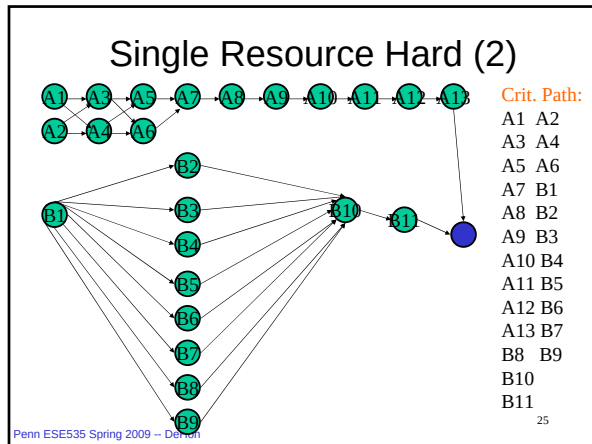
23

Single Resource Hard (1)



Penn ESE535 Spring 2009 -- DeHon

24



General: Why Hard

- When selecting, don't know
 - need to tackle **critical path**
 - need to run task to **enable work** (parallelism)

Penn ESE535 Spring 2009 -- DeHon 29

Two Bounds

Penn ESE535 Spring 2009 -- DeHon 30

Bounds

- Useful to have bounds on solution
- Two:
 - CP: Critical Path
 - RB: Resource Bound

Penn ESE535 Spring 2009 -- DeHon

31

Critical Path Lower Bound

- ASAP schedule ignoring resource constraints
 - (look at length of remaining critical path)
- Certainly cannot finish any faster than that

Penn ESE535 Spring 2009 -- DeHon

32

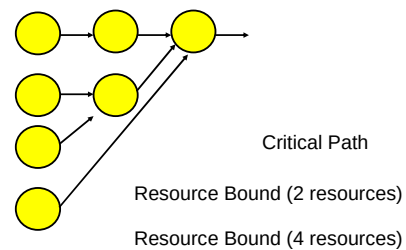
Resource Capacity Lower Bound

- Sum up all capacity required per resource
- Divide by total resource (for type)
- Lower bound on remaining schedule time
 - (best can do is pack all use densely)
 - Ignores schedule constraints

Penn ESE535 Spring 2009 -- DeHon

33

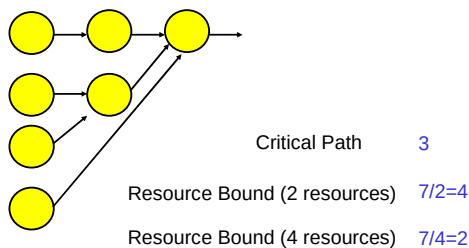
Example



Penn ESE535 Spring 2009 -- DeHon

34

Example



Penn ESE535 Spring 2009 -- DeHon

35

List Scheduling

Greedy Algorithm → Approximation

Penn ESE535 Spring 2009 -- DeHon

36

List Scheduling (basic algorithm flow)

- Keep a ready list of “available” nodes
 - (one whose predecessors have already been scheduled)
- Pick an unscheduled task and schedule on next available resource
- Put any tasks enabled by this one on ready list

Penn ESE535 Spring 2009 -- DeHon

37

List Scheduling

- Greedy heuristic
- **Key Question:** How prioritize ready list?
 - What is dominant constraint?
 - least slack (worst critical path) $\rightarrow$ LPT
 - enables work
 - utilize most precious (limited) resource
- So far:
 - seen that no single priority scheme would be optimal

Penn ESE535 Spring 2009 -- DeHon

38

List Scheduling

- Use for
 - resource constrained
 - time-constrained
 - give resource target and search for minimum resource set
- Fast: $O(N) \rightarrow O(N \log(N))$ depending on prioritization
- Simple, general
- How good?

Penn ESE535 Spring 2009 -- DeHon

39

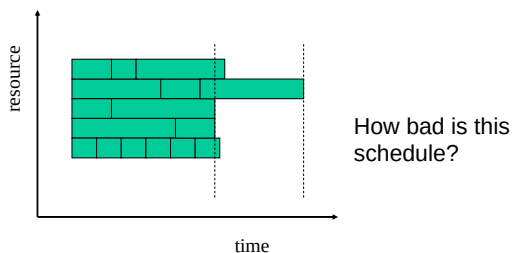
Approximation

- Can we say how close an algorithm comes to achieving the optimal result?
- Technically:
 - **If** can show
 - $\text{Heuristic}(\text{Prob}) / \text{Optimal}(\text{Prob}) \leq \alpha \quad \forall \text{ prob}$
 - **Then** the Heuristic is an α -approximation

Penn ESE535 Spring 2009 -- DeHon

40

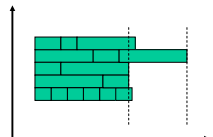
Scheduled Example Without Precedence



Penn ESE535 Spring 2009 -- DeHon

41

Observe



- $\exists$ optimal length L
- No idle time up to start of last job to finish
- start time of last job $\leq L$
- last job length $\leq L$
- Total LS length $\leq 2L$
- Algorithm is within factor of 2 of optimum

Penn ESE535 Spring 2009 -- DeHon

42

Results

- Scheduling of identical parallel machines has a 2-approximation
 - i.e. we have a polynomial time algorithm which is guaranteed to achieve a result within a factor of two of the optimal solution.
- In fact, for precedence unconstrained there is a 4/3-approximation
 - i.e. schedule Longest Processing Time first

Penn ESE535 Spring 2009 -- DeHon

43

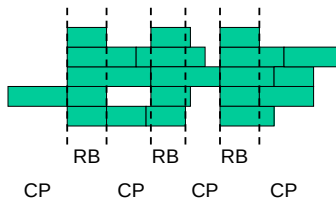
Recover Precedence

- With precedence we may have idle times, so need to generalize
- Work back from last completed job
 - two cases:
 - entire machine busy
 - some predecessor in critical path is running
- Divide into two sets
 - whole machine busy times
 - critical path chain for this operator

Penn ESE535 Spring 2009 -- DeHon

44

Precedence



Penn ESE535 Spring 2009 -- DeHon

45

Precedence Constrained

- Optimal Length > All busy times
 - Optimal Length $\geq$ Resource Bound
 - Resource Bound $\geq$ All busy
- Optimal Length > This Path
 - Optimal Length $\geq$ Critical Path
 - Critical Path $\geq$ This Path
- List Schedule = This path + All busy times
- List Schedule $\leq 2 * (\text{Optimal Length})$

Penn ESE535 Spring 2009 -- DeHon

46

Conclude

- Scheduling of identical parallel machines with precedence constraints has a 2-approximation.

Penn ESE535 Spring 2009 -- DeHon

47

Tighten

- LS schedule $\leq$ Critical Path + Resource Bound
- LS schedule $\leq \text{Min}(\text{CP}, \text{RB}) + \text{Max}(\text{CP}, \text{RB})$
- Optimal schedule $\geq \text{Max}(\text{CP}, \text{RB})$
- LS/Opt $\leq 1 + \text{Min}(\text{CP}, \text{RB}) / \text{Max}(\text{CP}, \text{RB})$
- The more one constraint dominates
 - the closer the approximate solution to optimal
 - ↳ (EEs think about 3dB point in frequency response)

Penn ESE535 Spring 2009 -- DeHon

48

Tightening

- Example of
 - More information about problem
 - More internal variables
 - ...allow us to state a tighter result
- 2-approx for any graph
 - Since CP may = RB
- Tighter approx as CP and RB diverge

Multiple Resource

- Previous result for homogeneous functional units
- For heterogeneous resources:
 - also a 2-approximation
 - Lenstra+Shmoys+Tardos, Math. Programming v46p259
 - (not online, no precedence constraints)

Bounds

- Precedence case, Identical machines
 - no polynomial approximation algorithm can achieve better than $4/3$ bound
 - (unless $P=NP$)
- Heterogeneous machines (no precedence)
 - no polynomial approximation algorithm can achieve better than $3/2$ bound

Summary

- Resource sharing saves area
 - allows us to fit in fixed area
- Requires that we schedule tasks onto resources
- General kind of problem arises
- We can, sometimes, bound the “badness” of a heuristic
 - get a tighter result based on gross properties of the problem
 - approximation algorithms often a viable alternative to finding optimum
 - play role in knowing “goodness” of solution

Admin

- Move office hours to W4:30pm
- Reading on web
- Assignment 1 Due Monday

Big Ideas:

- Exploit freedom in problem to reduce costs
 - (slack in schedules)
- Use dominating effects
 - (constrained resources)
 - the more an effect dominates, the “easier” the problem
- Technique: Approximation