



EE6094 CAD for VLSI Design



Chapter 3 Scheduling

Spring 2021
Andy, Yu-Guang Chen
Assistant Professor, Department of EE
National Central University
andygchen@ee.ncu.edu.tw



2021/3/16

Andy Yu-Guang Chen

1



Outline



- ◆ High Level Synthesis (HLS)
- ◆ Scheduling and Binding
- ◆ Operation Scheduling
 - ASAP Scheduling
 - ALAP Scheduling
- ◆ Constrained Scheduling
 - Hu's Algorithm
 - List Scheduling
 - Force-Directed Scheduling
 - Linear Programming



2021/3/16

Andy Yu-Guang Chen

2



High Level Synthesis (HLS)



- ◆ A process of converting high-level description of a design to a netlist
 - Input:
 - High-level languages (ex: C/C++/System C)
 - Behavioral hardware description languages (ex: Verilog, VHDL)
 - Structural HDLs (ex: Verilog, VHDL)
 - State diagrams & logic networks
 - Tools:
 - Parser
 - Library of modules
 - Constraints:
 - Area constraints (ex: # modules of a certain type)
 - Delay constraints (ex: set of operations should finish in λ clock cycles)
 - Output:
 - Operation scheduling (time) and binding (resource)
 - Control generation and detailed interconnections



2021/3/16

Andy Yu-Guang Chen

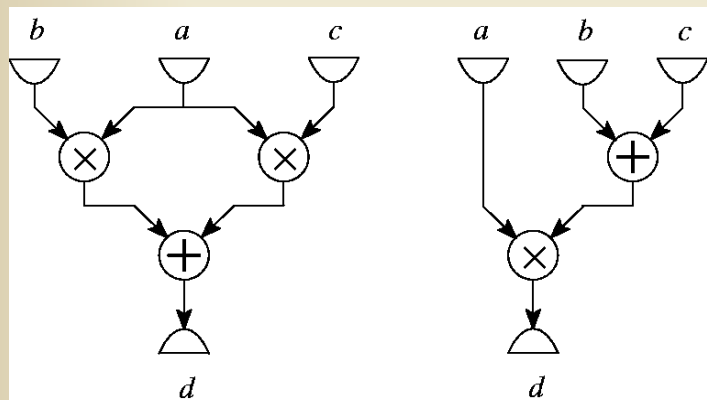
3



Example of High-Level Optimization



- ◆ Applying the distributivity law to reduce resource requirement.



Unit 2

4



Internal Representation



- ◆ Most systems use some form of a **data-flow graph (DFG)**.

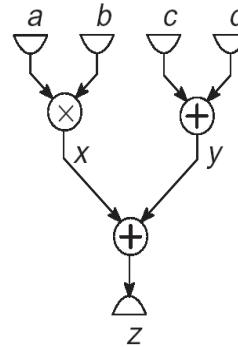
➤ A DFG may or may not contain information on control flow.

- ◆ A data-flow graph is built from

➤ Vertices (nodes): representing computation, and

➤ Edges: representing **precedence** relations.

```
x := a * b; y := c + d;
z := x + y;
```



scheduling Nop

Synchronous Graph



Unit 2

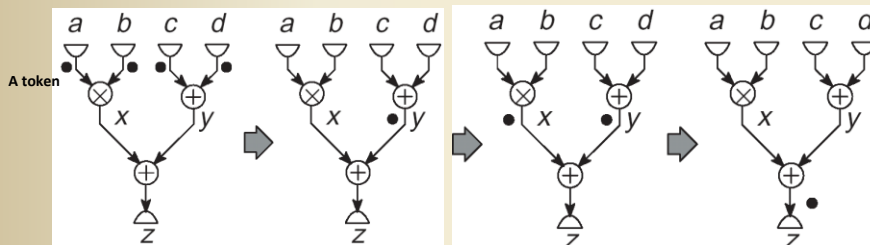
5



Token Flow in a DFG

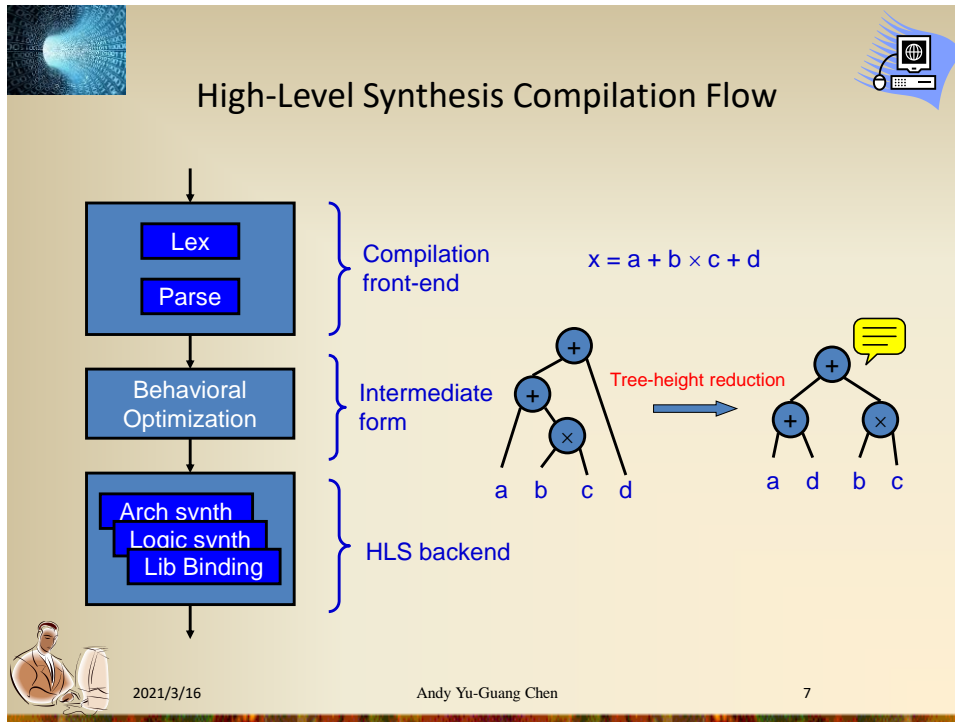


- ◆ A node in a DFG **fires** when all **tokens** are present at its inputs.
- ◆ The input tokens are consumed and an output token is produced.



Unit 2

6



Behavioral Optimization

The diagram illustrates Behavioral Optimization techniques, categorized into two main groups:

- Techniques used in software compilation:**
 - Expression tree-height reduction (indicated by a red arrow pointing to a handwritten diagram of an expression tree for $x = a + b \times c + d$ with a yellow box around the root node).
 - Constant and variable propagation
 - Common sub-expression elimination (indicated by a yellow speech bubble).
 - Dead-code elimination (indicated by a yellow speech bubble).
 - Operator strength reduction (ex: $*4 \rightarrow << 2$) (indicated by a yellow speech bubble).
- Typical Hardware transformations:**
 - Conditional expansion
 - If (c) then $x=A$ else $x=B$
 \rightarrow compute A and B in parallel, $x=(C)?A:B$
 - Loop expansion
 - Instead of three iterations of a loop, replicate the loop body three times (unrolling)

A logic diagram of a multiplexer is shown, with inputs A and B, a select input c, and output x.

2021/3/16 Andy Yu-Guang Chen 8



Architectural Synthesis



- ◆ Deals with "computational" behavioral descriptions
 - Behavior as sequencing graph (aka dependency graph, or data flow graph DFG)
 - Hardware resources as library elements
 - Pipelined or non-pipelined
 - Resource performance in terms of execution delay
 - Constraints on operation timing
 - Constraints on hardware resource availability
 - Storage as registers, data transfer using wires
- ◆ Objective
 - Generate a synchronous, single-phase clock circuit
 - Might have multiple feasible solutions (explore tradeoff)
 - Satisfy constraints, minimize objective:
 - Maximize performance subject to area constraint
 - Minimize area subject to performance constraints



2021/3/16

Andy Yu-Guang Chen

9



Pipelining Is Natural!



- ◆ Laundry example:

Ann, Brian, Cathy, Dave
each have one load of
clothes to wash, dry,
and fold



Washer takes 30 minutes



Dryer takes 40 minutes



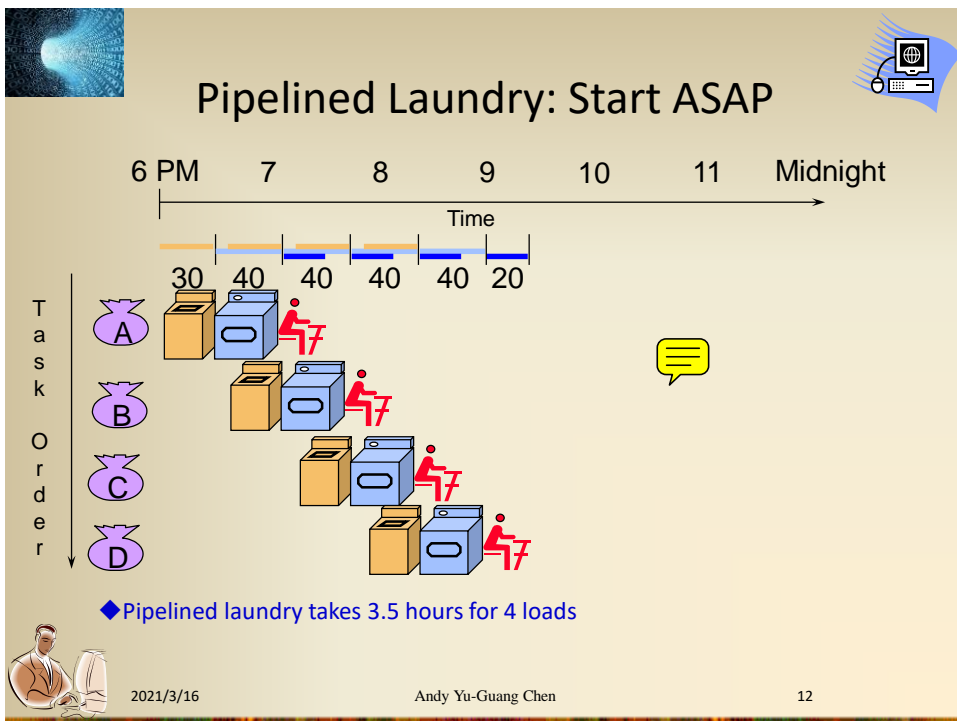
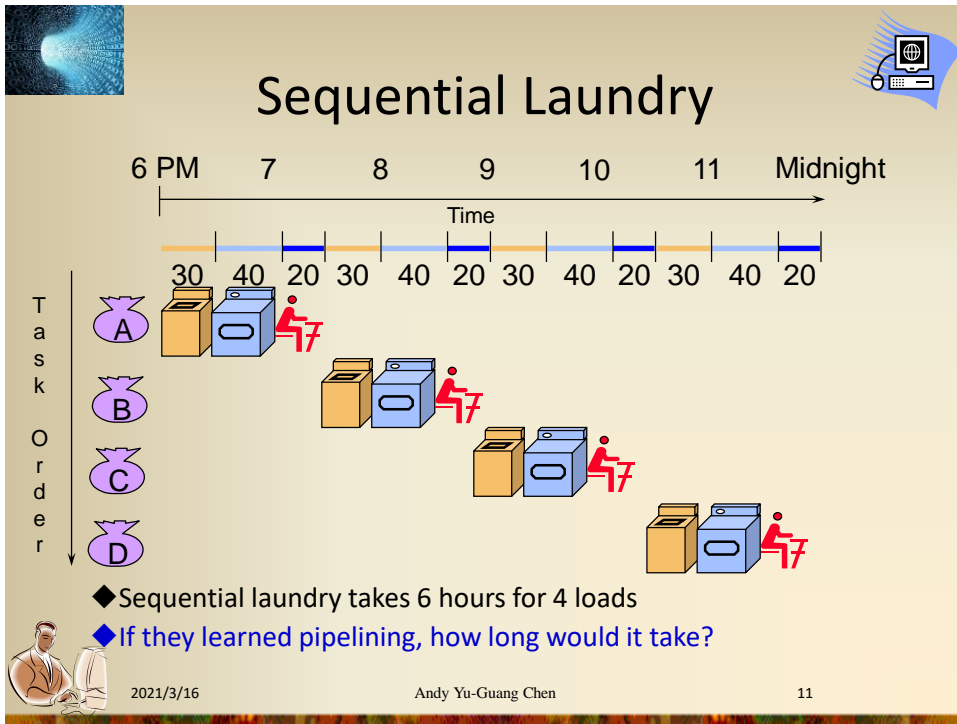
"Folder" takes 20 minutes



2021/3/16

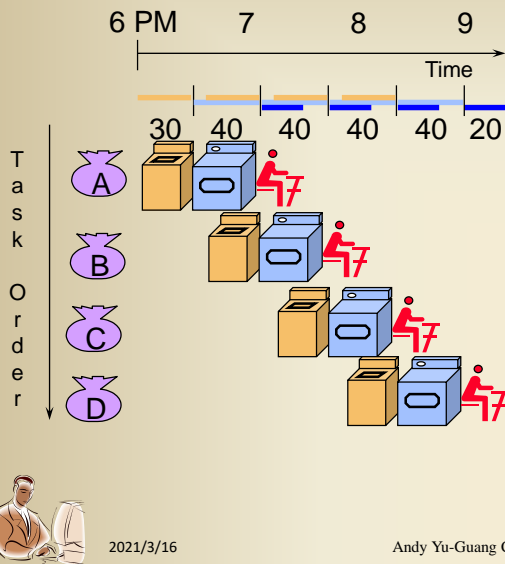
Andy Yu-Guang Chen

10





Pipelining Lessons



- ◆ Doesn't help **latency** of single task, but **throughput** of entire
- ◆ Pipeline rate limited by **slowest** stage
- ◆ **Multiple** tasks working at same time using different resources
- ◆ Potential speedup = **Number pipe stages**
- ◆ Unbalanced stage length; time to "fill" & "drain" the pipeline **reduce speedup**
- ◆ **Stall** for dependences



2021/3/16

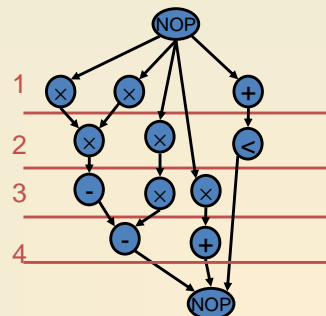
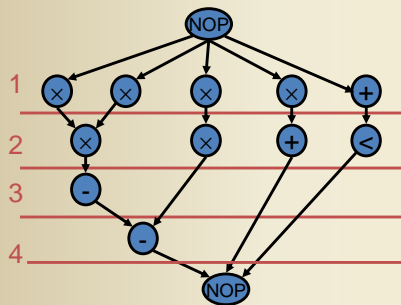
Andy Yu-Guang Chen

13



Synthesis in Temporal Domain

- ◆ Scheduling and binding can be done in different orders or together
- ◆ Scheduling:
 - Mapping of operations to time slots (cycles)
 - A scheduled sequencing graph is a labeled graph



2021/3/16

Andy Yu-Guang Chen

14



Operation Types



- ◆ For each operation, define its type
- ◆ For each resource, define a resource type and its delay (in terms of # cycles)
- ◆ T is a relation that maps an operation to a resource type that can implement it
 - $T: V \rightarrow \{1, 2, \dots, n_{res}\}$.
- ◆ More general case:
 - A resource type may implement more than one operation type (ex: ALU)
- ◆ Resource binding:
 - Map each operation to a resource with the same type
 - Might have multiple options



2021/3/16

Andy Yu-Guang Chen

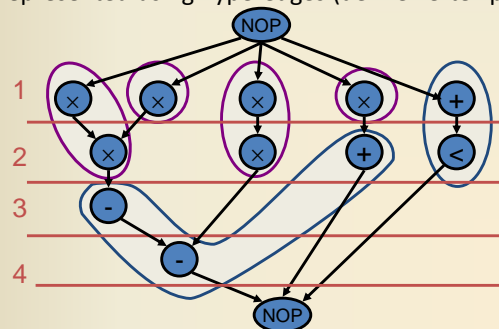
15



Schedule in Spatial Domain



- ◆ Resource sharing
 - More than one operation is bound to same resource
 - Operations have to be serialized
 - Can be represented using hyperedges (define vertex partition)



2021/3/16

Andy Yu-Guang Chen

16



HLS Subtasks: Allocation, Scheduling, Assignment

◆ Subtasks in high-level synthesis

- **Allocation (Module selection):** specify the hardware resources that will be necessary.
- **Scheduling:** determine for each operation the time at which it should be performed such that no precedence constraint is violated.
- **Assignment (Binding):** map each operation to a specific functional unit and each variable to a register.

◆ Remarks:

- Though the subproblems are strongly interrelated, they are often solved separately. However, to attain a better solution, an iterative process executing these three subtasks must be performed.
- Most scheduling problems are NP-complete \Rightarrow heuristics are used.



Unit 2

17



Scheduling and Binding

◆ Resource constraints:

- Number of resource instances of each type
 $\{a_k : k=1, 2, \dots, n_{res}\}$

◆ Scheduling:

- Labeled vertices $\phi(v_3)=1$

◆ Binding:

- Hyperedges (or vertex partitions) $\beta(v_2)=adder_1$

◆ Cost:

- Number of resources \approx area **Resource dominated**
- Registers, steering logic (mux, bus), wiring, control unit **Control dominated**

◆ Delay:

- Start time of the "sink" node
- Might be affected by steering logic and scheduling (control logic) – resource-dominated vs. control-dominated



2021/3/16

Andy Yu-Guang Chen

18



Architectural Optimization



- ◆ Optimization in view of design space flexibility
- ◆ A multi-criteria optimization problem:
 - Determine schedule ϕ and binding β .
 - Under area A , latency λ , and cycle time τ objectives
- ◆ Find non-dominated points in solution space
- ◆ Solution space tradeoff curves:
 - Non-linear, discontinuous
 - Area, latency, cycle time (more?)
- ◆ Evaluate (estimate) cost functions
- ◆ Unconstrained optimization problems for resource dominated circuits:
 - Min area: solve for minimal binding
 - Min latency: solve for minimum λ scheduling



2021/3/16

Andy Yu-Guang Chen

19



Scheduling and Binding



- ◆ Cost λ and A determined by both ϕ and β
 - Also affected by floorplan and detailed routing
- ◆ β affected by ϕ :
 - Resources cannot be shared among concurrent operations
- ◆ ϕ affected by β :
 - Resources cannot be shared among concurrent operations
 - When register and steering logic delays added to execution delays, might violate cycle time
- ◆ Order?
 - Apply either one (scheduling, binding) first



2021/3/16

Andy Yu-Guang Chen

20

Hardware Concepts: Data Path + Control

- ◆ Hardware is normally partitioned into two parts:
 - **Data path:** a network of functional units, registers, multiplexers and buses.
 - The actual “computation” takes place in the data path.
 - **Control:** the part of the hardware that takes care of having the data present at the right place at a specific time, of presenting the right instructions to a programmable unit, etc.
- ◆ Often high-level synthesis concentrates on data-path synthesis.
 - The control part is then realized as a finite state machine or in microcode.

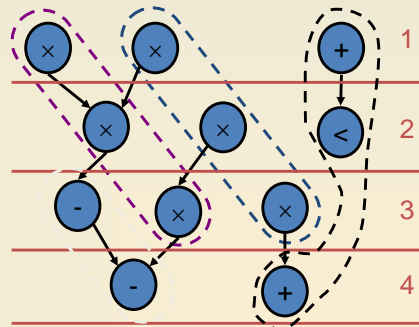


Unit 2

21

How Is the Datapath Implemented?

- ◆ Assuming the following scheduling and binding
 - Wires between modules?
 - Input selection?
 - How do binding and scheduling affect congestion?
 - How do binding and scheduling affect steering logic?



2021/3/16

Andy Yu-Guang Chen

22



Operation Scheduling



- ◆ Input:
 - Sequencing graph $G(V, E)$, with n vertices
 - Cycle time τ
 - Operation delays $D = \{d_i; i=0..n\}$
- ◆ Output:
 - Schedule ϕ determines start time t_i of operation v_i
 - Latency $\lambda = t_n - t_0$
- ◆ Goal: determine area & latency tradeoff
- ◆ Issues:
 - Non-hierarchical and unconstrained
 - Latency constrained
 - Resource constrained
 - Hierarchical



2021/3/16

Andy Yu-Guang Chen

23



Min Latency Unconstrained Scheduling



- ◆ Simplest case: no constraints, find minimum latency
- ◆ Given set of vertices V , delays D , and partial order \succ on operations E , find an integer labeling of operations $\phi: V \rightarrow \mathbb{Z}^+$, such that:
 - $t_i = \phi(v_i)$
 - $t_i \geq t_j + d_j \quad \forall (v_j, v_i) \in E$
 - $\lambda = t_n - t_0$ is minimum
- ◆ Solvable in polynomial time
- ◆ Bounds on latency for resource constrained problems
- ◆ ASAP algorithm used: topological order
- ◆ Applying the DFG algorithm to finding the longest path between the start and end nodes leads to the latency of the result



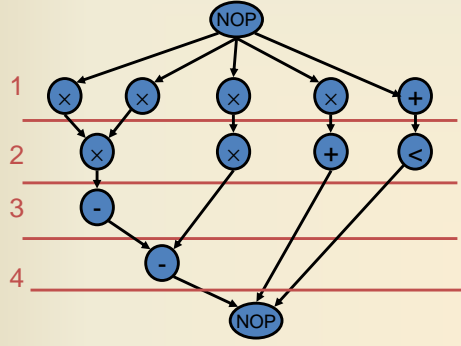
2021/3/16

Andy Yu-Guang Chen

24

ASAP Scheduling

Schedule v_0 at $t_0=0$
 While (v_n not scheduled)
 Select v_i with all scheduled predecessors
 Schedule v_i at $t_i = \max \{t_j + d_j\}$, v_j being a predecessor of v_i
 Return t_n

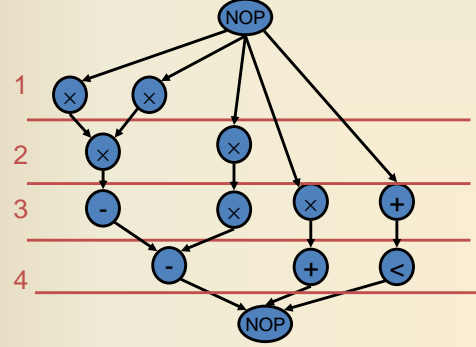


The diagram illustrates the ASAP scheduling process on a control flow graph. The graph starts with a NOP node at the top. It branches into five nodes at level 1: two multiplication nodes (x), one multiplication node (x), one addition node (+), and one less-than node (<). Level 2 contains three nodes: a multiplication node (x) from the first x, a multiplication node (x) from the second x, and an addition node (+) from the first +. Level 3 contains two nodes: a subtraction node (-) from the first x, and a subtraction node (-) from the second x. Level 4 contains a final NOP node. Red horizontal lines separate the levels, and the levels are numbered 1 through 4 on the left. The graph shows the earliest possible execution time for each node based on its predecessors.

2021/3/16 Andy Yu-Guang Chen 25

ALAP Scheduling (Latency Constrained)

Schedule v_n at $t_n = \bar{\lambda} + 1$
 While (v_0 not scheduled)
 Select v_i with all scheduled successors
 Schedule v_i at $t_i = \min \{t_j - d_i\}$, v_j being a successor of v_i
 Return t_n



The diagram illustrates the ALAP scheduling process on the same control flow graph as the ASAP diagram. The graph structure is identical, but the nodes are scheduled as late as possible while meeting the latency constraint. The levels are numbered 1 through 4 on the left, and red horizontal lines separate the levels. The graph shows the latest possible execution time for each node based on its successors.

2021/3/16 Andy Yu-Guang Chen 26



Related Terminologies

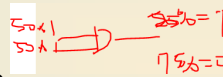


◆ ASAP, ALAP

- Slack, mobility
- Critical node, critical path
- Criticality

◆ Static timing analysis

- Arrival time
- Required time



◆ Statistical static timing analysis

- Advanced process technology: 45nm, 32nm, ...



2021/3/16

Andy Yu-Guang Chen

27



Constrained Scheduling



◆ Constrained scheduling

- General case NP-complete
- Minimize latency, given constraints on area or the resources (ML-RCS)
- Minimize resources subject to bound on latency (MR-LCS)

◆ Exact solution methods

- ILP: Integer Linear Programming
- Hu's heuristic algorithm for identical processors (operations)

◆ Heuristics

- List scheduling
- Force-directed scheduling

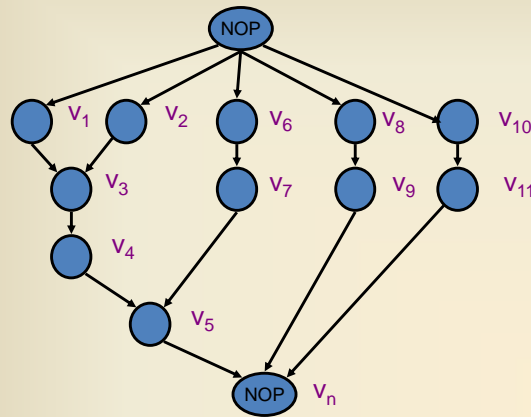


2021/3/16

Andy Yu-Guang Chen

28

Precedence-constrained Multiprocessor Scheduling



2021/3/16

Andy Yu-Guang Chen

29

Hu's Algorithm

- ◆ Simple case of the scheduling problem
 - Operations (and resources) of the same type
 - Operations of unit delay
- ◆ Hu's algorithm
 - Greedy
 - Polynomial & optimal
 - Compute lower bound on number of resources for a given latency (MR-LCS)
 - OR-
 - Compute lower bound on latency subject to resource constraints (ML-RCS)
- ◆ Basic idea:
 - Label operations based on their distances from the sink
 - Try to schedule nodes with higher labels first (i.e., most "critical" operations have highest priority)



2021/3/16

Andy Yu-Guang Chen

30

Hu's Algorithm

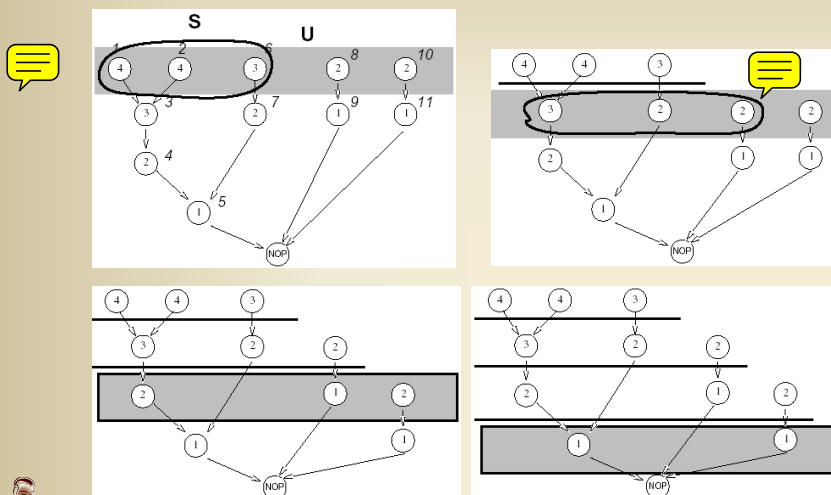
$HU(G(V,E), a)$
 Label the vertices // label = length of longest path
 passing through the vertex
 $l = 1$
 repeat {
 U = unscheduled vertices in V whose
 predecessors have been scheduled
 (or have no predecessors) \rightarrow "ready state"
 Select $S \subseteq U$ such that $|S| \leq a$ and labels in S
 are maximal
 Schedule the S operations at step l by setting
 $t_i = l, i: v_i \in S$
 $l = l + 1$
 } until v_n is scheduled

2021/3/16

Andy Yu-Guang Chen

31

Hu's Algorithm: Example



2021/3/16

Andy Yu-Guang Chen

32



Constrained Scheduling



- ◆ Constrained scheduling
 - General case NP-complete
 - Minimize latency, given constraints on area or the resources (ML-RCS)
 - Minimize resources subject to bound on latency (MR-LCS)
- ◆ Exact solution methods
 - ILP: Integer Linear Programming
 - Hu's heuristic algorithm for identical processors (operations)
- ◆ Heuristics
 - List scheduling
 - Force-directed scheduling



2021/3/16

Andy Yu-Guang Chen

33



List Scheduling



- ◆ Greedy algorithm for ML-RCS and MR-LCS
 - Does NOT guarantee optimum solution
- ◆ Similar to Hu's algorithm
 - Operation selection decided by criticality
 - $O(n)$ time complexity
- ◆ More general input
 - Resource constraints on different resource types



2021/3/16

Andy Yu-Guang Chen

34

List Scheduling Algorithm: ML-RCS

```

LIST_L (G(V,E), a) {
  l = 1
  repeat {
    for each resource type k {
       $U_{l,k}$  = available vertices in V  $\rightarrow$  "ready state"
       $T_{l,k}$  = operations in progress  $\rightarrow$  "ongoing state"
      Select  $S_k \subseteq U_{l,k}$  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
}

```

必須知道parent狀態

要知道每個type正在進行的數量



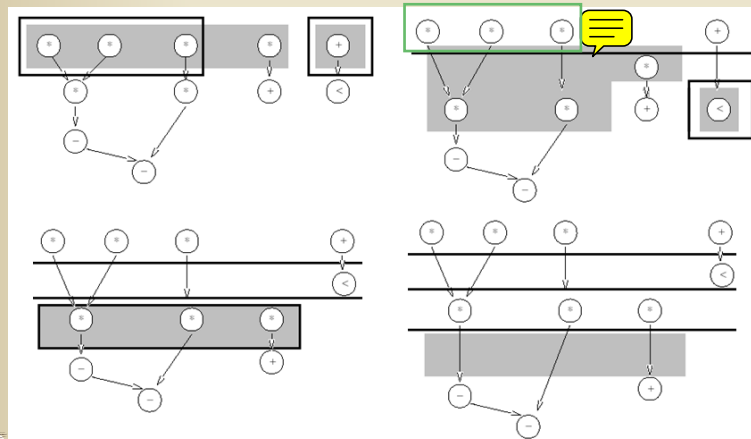
2021/3/16

Andy Yu-Guang Chen

35

List Scheduling Example

◆ OP *: delay = 2, number: 3 OP +: delay = 1, number 1



2021/3/16

Andy Yu-Guang Chen

36



List Scheduling Algorithm: MR-LCS



```

LIST_R (G(V,E),  $\lambda'$ ) {
   $\alpha = 1$ ,  $l = 1$ 
  Compute the ALAP times  $t^L$ 
  if  $t_0^L < 0$ 
    return (not feasible)
  repeat {
    for each resource type k {
       $U_{l,k}$  = available vertices in V  $\rightarrow$  "ready state"
      Compute the slacks  $\{s_i = t_i^L - l, \forall v_i \in U_{l,k}\}$ 
      Schedule operations with zero slack, update  $\alpha$ 
      Schedule additional  $S_k \subseteq U_{l,k}$  under  $\alpha$  constraints
    }
     $l = l + 1$ 
  } until  $v_n$  is scheduled
}

```



2021/3/16

Andy Yu-Guang Chen

37



Constrained Scheduling



- ◆ Constrained scheduling
 - General case NP-complete
 - Minimize latency, given constraints on area or the resources (ML-RCS)
 - Minimize resources subject to bound on latency (MR-LCS)
- ◆ Exact solution methods
 - ILP: Integer Linear Programming
 - Hu's heuristic algorithm for identical processors (operations)
- ◆ Heuristics
 - List scheduling
 - Force-directed scheduling



2021/3/16

Andy Yu-Guang Chen

38



Force-Directed Scheduling



- ◆ Similar to list scheduling
 - Can handle ML-RCS and MR-LCS
 - For ML-RCS, schedules step-by-step
 - BUT, selection of the operations tries to find the *globally* best set of operations
- ◆ Idea – time frame:
 - Find the **mobility** $\mu_i = t_i^L - t_i^S$ of operations
 - Look at the operation type probability distributions
 - Try to flatten the operation type distributions
- ◆ Definition: operation probability density
 - $p_i(l) = \Pr \{ v_i \text{ starts at step } l \}$
 - Assume uniform distribution:

$$p_i(l) = \frac{1}{\mu_i + 1} \quad \text{for } l \in [t_i^S, t_i^L]$$



2021/3/16

Andy Yu-Guang Chen

39



Force-Directed Scheduling: Definitions



- ◆ Operation-type distribution (NOT normalized to 1)
 - $q_k(l) = \sum_{i: T(v_i)=k} p_i(l)$
- ◆ Operation probabilities over control steps:
 - $p_i = \{ p_i(0), p_i(1), \dots, p_i(n) \}$
- ◆ Distribution graph of type k over all steps:
 - $\{ q_k(0), q_k(1), \dots, q_k(n) \}$
 - $q_k(l)$ can be thought of as *expected* operator cost for implementing operations of type k at step l .



2021/3/16

Andy Yu-Guang Chen

40



Example



$$q_{mult}(1) = 1 + 1 + \frac{1}{2} + \frac{1}{3} = 2.83$$

$$q_{mult}(2) = 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{3} = 2.33$$

$$q_{mult}(3) = \frac{1}{2} + \frac{1}{3} = 0.83$$

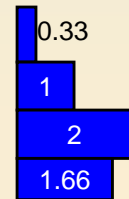
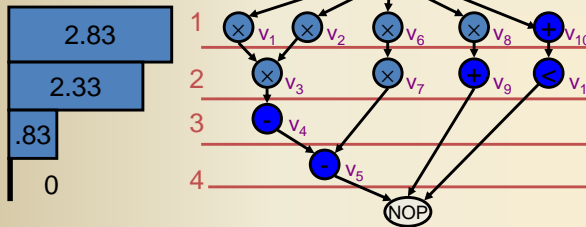
$$q_{mult}(4) = 0$$

$$q_{add}(1) = \frac{1}{3} = 0.33$$

$$q_{add}(2) = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$$

$$q_{add}(3) = 1 + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 2$$

$$q_{add}(4) = 1 + \frac{1}{3} + \frac{1}{3} = 1.66$$



2021/3/16

Andy Yu-Guang Chen

41



Force



- ◆ Used as *priority* function
- ◆ Force is related to concurrency:
 - Sort operations for least force
- ◆ Mechanical analogy:
- ◆ Force = constant \times displacement
 - Constant = operation-type distribution, $q_k(l)$
 - Displacement = change in probability



2021/3/16

Andy Yu-Guang Chen

42



Self Force



- ◆ Sum of forces to feasible schedule steps
- ◆ Self-force for operation v_i in step l

$$\begin{aligned}\text{self-force}(i, l) &= \sum_{m=t_i^S}^{t_i^L} q_k(m)(\delta_{lm} - p_i(m)) \\ &= q_k(l) - \frac{1}{\mu_i + 1} \sum_{m=t_i^S}^{t_i^L} q_k(m)\end{aligned}$$

$$\delta_{lm} = \begin{cases} 1, & \text{if } l = m \\ 0, & \text{if } l \neq m \end{cases}$$

$$p_i(m) = \frac{1}{\mu_i + 1}$$



2021/3/16

Andy Yu-Guang Chen

43



Self Force Example

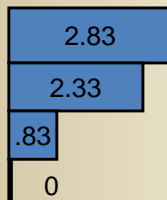


$$q_{mult}(1) = 1 + 1 + \frac{1}{2} + \frac{1}{3} = 2.83$$

$$q_{mult}(2) = 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{3} = 2.33$$

$$q_{mult}(3) = \frac{1}{2} + \frac{1}{3} = 0.83$$

$$q_{mult}(4) = 0$$

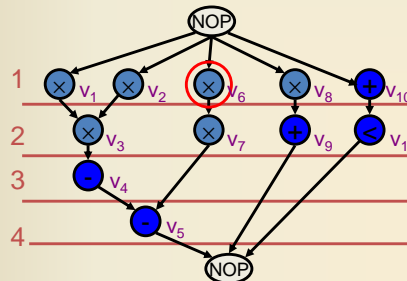
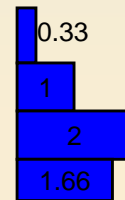


$$q_{add}(1) = \frac{1}{3} = 0.33$$

$$q_{add}(2) = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$$

$$q_{add}(3) = 1 + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 2$$

$$q_{add}(4) = 1 + \frac{1}{3} + \frac{1}{3} = 1.66$$



2021/3/16

Andy Yu-Guang Chen

44



Self Force Example: v_6



- ◆ Op v_6 can be scheduled in the first two steps
 - $p(1) = 0.5$; $p(2) = 0.5$; $p(3) = 0$; $p(4) = 0$
 - Distribution: $q(1) = 2.8$; $q(2) = 2.3$
- ◆ 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: $2.8 * 0.5 - 2.3 * 0.5 = +0.25$
 - No successor force
- ◆ 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: $-2.8 * 0.5 + 2.3 * 0.5 = -0.25$
 - Successor-force:
 - Successor (v_7) force is $2.3 * (0 - 0.5) + 0.8 * (1 - 0.5) = -0.75$
 - Total force = $-1 \quad 1 * 0.8 - 0.5 * (2.3 + 0.8) = -0.75$
- ◆ Hence, assign v_6 to step 2



2021/3/16

Andy Yu-Guang Chen

45



Predecessor/successor Force



- ◆ Related to the predecessors/successors
 - Fixing an operation timeframe restricts timeframe of predecessors/successors
 - Ex: Delaying an operation implies delaying its successors

$$\text{ps-force}(i, l) = \frac{1}{\tilde{\mu}_i + 1} \sum_{m=\tilde{t}_i^s}^{\tilde{t}_i^L} q_k(m) - \frac{1}{\mu_i + 1} \sum_{m=t_i^s}^{t_i^L} q_k(m)$$



2021/3/16

Andy Yu-Guang Chen

46



P/S Force Example



$$q_{mult}(1) = 1 + 1 + \frac{1}{2} + \frac{1}{3} = 2.83$$

$$q_{mult}(2) = 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{3} = 2.33$$

$$q_{mult}(3) = \frac{1}{2} + \frac{1}{3} = 0.83$$

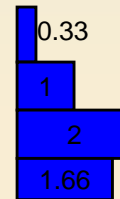
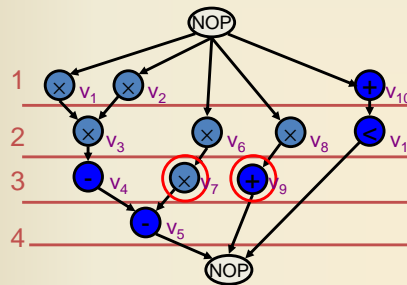
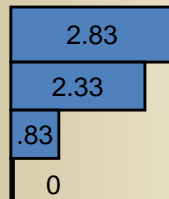
$$q_{mult}(4) = 0$$

$$q_{add}(1) = \frac{1}{3} = 0.33$$

$$q_{add}(2) = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$$

$$q_{add}(3) = 1 + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 2$$

$$q_{add}(4) = 1 + \frac{1}{3} + \frac{1}{3} = 1.66$$



2021/3/16

Andy Yu-Guang Chen

47



P/S Force Example: v_7 & v_9



- ◆ Type 1 (v_7) distribution:
 - $q(1) = 2.8$; $q(2) = 2.3$; $q(3) = 0.8$; $q(4) = 0$
- ◆ Assign v_6 to step 2:
 - Time frame of v_7 is reduced
 - $1 * (0.8) - 0.5 * (2.3 + 0.8) = -0.75$
- ◆ Type 2 (v_9) distribution:
 - $q(1) = 0.3$; $q(2) = 1$; $q(3) = 2$; $q(4) = 1.6$
- ◆ Assign v_8 to step 2:
 - Time frame of v_9 is reduced
 - $0.5 * (2 + 1.6) - 0.3 * (1 + 2 + 1.6) = 0.3$



2021/3/16

Andy Yu-Guang Chen

48



Force-Directed Scheduling: Algorithm



```

FDS ( $G(V, E), \bar{\lambda}$ ) {
  repeat {
    Compute/update the time-frames
    Compute the operation and type probabilities
    Compute the self-force, ps-force and total force
    Schedule the op. with least force
  }
  until (all operations are scheduled)
  return (t)
}

```



2021/3/16

Andy Yu-Guang Chen

49



Force-Directed Scheduling: Algorithm



- ◆ Very similar to LIST_L($G(V, E), a$)
 - Compute mobility of operations using ASAP and ALAP
 - Select and schedule operations
 - Go to next control step
- ◆ Difference with list scheduling in selecting operations
 - Compute operation probabilities and type distributions
 - Select operations with least force
 - Update operation probabilities and type distributions
 - Consider the effect on the type distribution
 - Consider the effect on p/s nodes and their type distributions
 - Complexity: $O(n^3)$



2021/3/16

Andy Yu-Guang Chen

50



Resource Constraint Scheduling



- ◆ Constrained scheduling
 - General case NP-complete
 - Minimize latency given constraints on area or the resources (ML-RCS)
 - Minimize resources subject to bound on latency (MR-LCS)
- ◆ Exact solution methods
 - ILP: Integer Linear Programming
 - Hu's heuristic algorithm for identical processors (operations)
- ◆ Heuristics
 - List scheduling
 - Force-directed scheduling



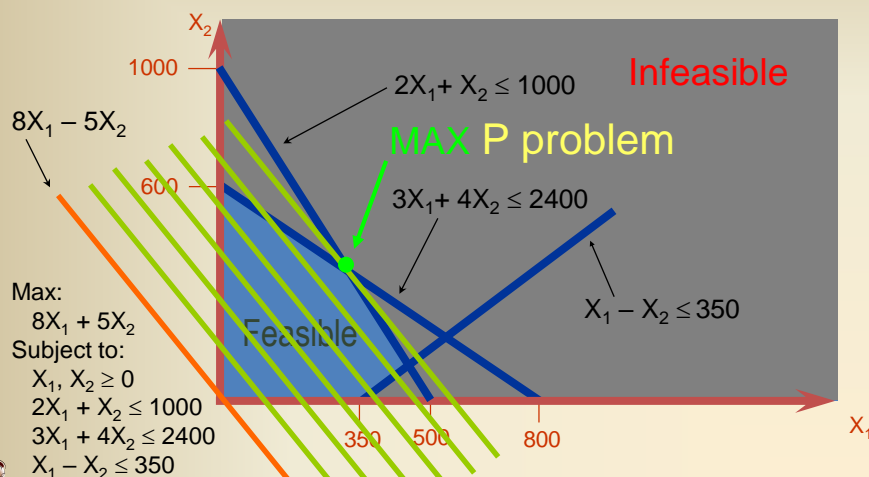
2021/3/16

Andy Yu-Guang Chen

51



Linear Programming Example



2021/3/16

Andy Yu-Guang Chen

52



Mixed Integer Linear Programming



- ◆ A mathematical programming such that:
 - The objective is a linear function
 - All constraints are linear functions
 - Some variables are real numbers and some are integers, i.e., "mixed integer"
- ◆ It is almost like a linear programming, except that some variables are integers

NP-C problem



2021/3/16

Andy Yu-Guang Chen

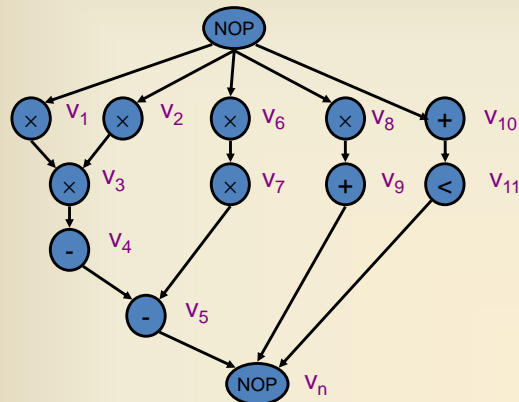
53



ILP Scheduling



- ◆ How to construct a mathematical model?



2021/3/16

Andy Yu-Guang Chen

54



ILP Formulation of ML-RCS



- ◆ Use binary decision variables
 - $i = 0, 1, \dots, n$
 - $l = 1, 2, \dots, \lambda' + 1$ λ' : given upper-bound on latency
 - $x_{i,l} = 1$ if operation i starts at step l , 0 otherwise.
- ◆ Set of linear inequalities (constraints), and an objective function (min latency)
- ◆ Observations
 - $x_{i,l} = 0$ for $l < t_i^S$ and $l > t_i^L$ feasibility
 $(t_i^S = ASAP(v_i), t_i^L = ALAP(v_i))$
 - $t_i = \sum_l l \cdot x_{i,l}$ t_i = start time of op i
 - $\sum_{m=l-d_i+1}^l x_{i,m} = 1$ If op v_i takes d_i steps,
 is op v_i (still) executing at step l ?



2021/3/16

Andy Yu-Guang Chen

55



Start Time vs. Execution Time



- ◆ For each operation v_i , only one start time
- ◆ If $d_i = 1$, then the following questions are the same:
 - Does operation v_i **start** at step l ?
 - Is operation v_i **running** at step l ?
- ◆ But if $d_i > 1$, then the two questions should be formulated as:
 - Does operation v_i **start** at step l ?
 - Does $x_{i,l} = 1$ hold?
 - Is operation v_i **running** at step l ?
 - Does $\sum_{m=l-d_i+1}^l x_{i,m} = 1$ hold?



2021/3/16

Andy Yu-Guang Chen

56

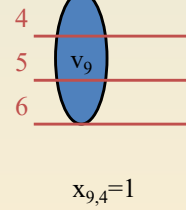
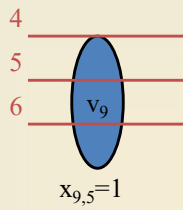
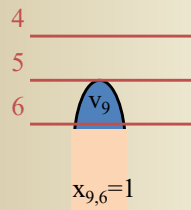


Operation v_i Still Running at Step l ?



◆ Assume that v_9 takes 3 steps, is v_9 running at step 6?

➤ Is $x_{9,6} + x_{9,5} + x_{9,4} = 1$?



◆ Note:

- Only one (if any) of the above three cases can happen
- To meet resource constraints, we have to ask the same question for ALL steps, and ALL operations of that type



2021/3/16

Andy Yu-Guang Chen

57

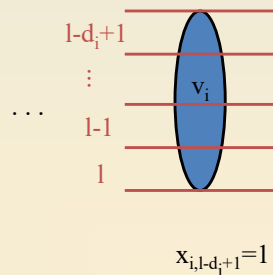
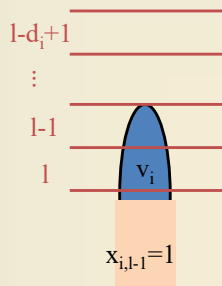
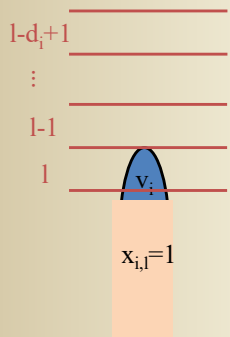


Operation v_i Still Running at Step l ?



◆ Is v_i running at step l ?

➤ Is $x_{i,l} + x_{i,l-1} + \dots + x_{i,l-d_i+1} = 1$?



2021/3/16

Andy Yu-Guang Chen

58

ILP Formulation of ML-RCS (Cont.)

◆ Constraints:

➤ Unique start times: $\sum_l x_{i,l} = 1, \quad i = 0, 1, \dots, n$

➤ Sequencing (dependency) relations must be satisfied

$$t_i \geq t_j + d_j \quad \forall (v_j, v_i) \in E \Rightarrow \sum_l l \cdot x_{i,l} \geq \sum_l l \cdot x_{j,l} + d_j$$

➤ Resource constraints

$$\sum_{i: T(v_i)=k} \sum_{m=l-d_i+1}^l x_{i,m} \leq a_k, \quad k = 1, \dots, n_{res}, \quad l = 1, \dots, \bar{\lambda} + 1$$

◆ Objective: $\min \mathbf{c}^T \mathbf{t}$.

➤ \mathbf{t} = start times vector, \mathbf{c} = cost weight (ex: $[0 \ 0 \ \dots \ 1]$)

➤ When $\mathbf{c} = [0 \ 0 \ \dots \ 1]$, $\mathbf{c}^T \mathbf{t} = \sum_l l \cdot x_{n,l}$



2021/3/16

Andy Yu-Guang Chen

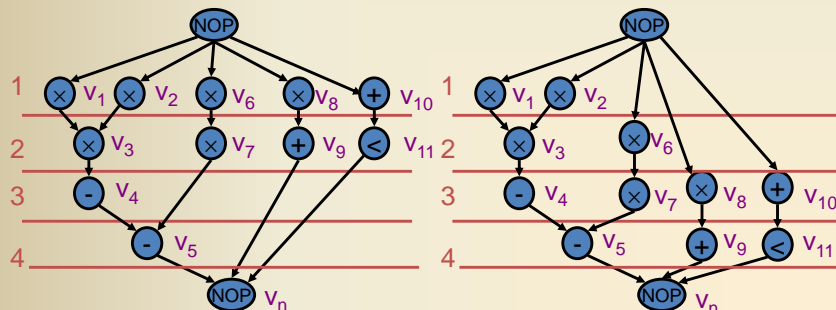
59

ILP Example

◆ Assume $\bar{\lambda} = 4$

◆ First, perform ASAP and ALAP

➤ (we can write the ILP without ASAP and ALAP, but using ASAP and ALAP will simplify the inequalities)



2021/3/16

Andy Yu-Guang Chen

60



ILP Example: Unique Start Times Constraint



◆ Without using ASAP and ALAP values:

$$x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} = 1$$

$$x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} = 1$$

...

...

...

$$x_{11,1} + x_{11,2} + x_{11,3} + x_{11,4} = 1$$

□ Using ASAP and ALAP:

$$x_{1,1} = 1$$

$$x_{2,1} = 1$$

$$x_{3,2} = 1$$

$$x_{4,3} = 1$$

$$x_{5,4} = 1$$

$$x_{6,1} + x_{6,2} = 1$$

$$x_{7,2} + x_{7,3} = 1$$

$$x_{8,1} + x_{8,2} + x_{8,3} = 1$$

$$x_{9,2} + x_{9,3} + x_{9,4} = 1$$

...



2021/3/16

Andy Yu-Guang Chen

61



ILP Example: Dependency Constraints



◆ Using ASAP and ALAP, the non-trivial inequalities are: (assuming unit delay for + and *)

$$2 \cdot x_{7,2} + 3 \cdot x_{7,3} - 1 \cdot x_{6,1} - 2 \cdot x_{6,2} - 1 \geq 0$$

$$2 \cdot x_{9,2} + 3 \cdot x_{9,3} + 4 \cdot x_{9,4} - 1 \cdot x_{8,1} - 2 \cdot x_{8,2} - 3 \cdot x_{8,3} - 1 \geq 0$$

$$2 \cdot x_{11,2} + 3 \cdot x_{11,3} + 4 \cdot x_{11,4} - 1 \cdot x_{10,1} - 2 \cdot x_{10,2} - 3 \cdot x_{10,3} - 1 \geq 0$$

$$4 \cdot x_{5,4} - 2 \cdot x_{7,2} - 3 \cdot x_{7,3} - 1 \geq 0$$

$$5 \cdot x_{n,5} - 2 \cdot x_{9,2} - 3 \cdot x_{9,3} - 4 \cdot x_{9,4} - 1 \geq 0$$

$$5 \cdot x_{n,5} - 2 \cdot x_{11,2} - 3 \cdot x_{11,3} - 4 \cdot x_{11,4} - 1 \geq 0$$



2021/3/16

Andy Yu-Guang Chen

62



ILP Example: Resource Constraints



- ◆ Resource constraints (assuming 2 adders and 2 multipliers)

$$x_{1,1} + x_{2,1} + x_{6,1} + x_{8,1} \leq 2$$

$$x_{3,2} + x_{6,2} + x_{7,2} + x_{8,2} \leq 2$$

$$x_{7,3} + x_{8,3} \leq 2$$

$$x_{10,1} \leq 2$$

$$x_{9,2} + x_{10,2} + x_{11,2} \leq 2$$

$$x_{4,3} + x_{9,3} + x_{10,3} + x_{11,3} \leq 2$$

$$x_{5,4} + x_{9,4} + x_{11,4} \leq 2$$

- ◆ Objective:

- Since $\lambda=4$ and sink has no mobility, any feasible solution is optimum, but we can use the following anyway:

$$\text{Min } 1 \cdot x_{n,1} + 2 \cdot x_{n,2} + 3 \cdot x_{n,3} + 4 \cdot x_{n,4}$$



2021/3/16

Andy Yu-Guang Chen

63



ILP Formulation of MR-LCS



- ◆ Dual problem to ML-RCS

- ◆ Objective:

- Goal is to optimize total resource usage, \mathbf{a} .
- Objective function is $\mathbf{c}^T \mathbf{a}$, where entries in \mathbf{c} are respective area costs of resources

- ◆ Constraints:

- Same as ML-RCS constraints, plus:
- Latency constraint added:

$$\sum_l l \cdot x_{n,l} \leq \bar{\lambda} + 1$$

- Note: unknown \mathbf{a}_k appears in constraints.



2021/3/16

Andy Yu-Guang Chen

64



Further Study



◆ Linear programming

➤ <http://www.cs.sunysb.edu/~algorith/files/linear-programming.shtml>

◆ Linear programming tools

➤ <http://lpsolve.sourceforge.net/5.5/>



2021/3/16

Andy Yu-Guang Chen

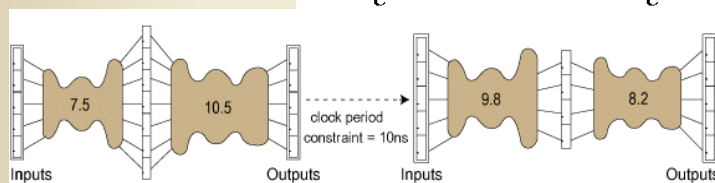
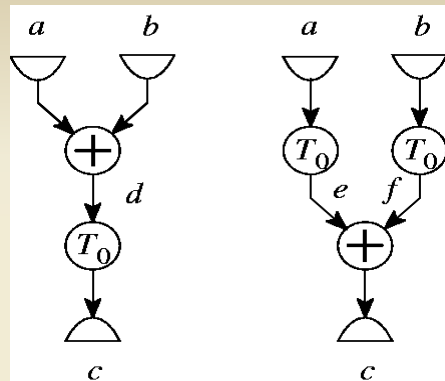
65



Behavior Retiming (BRT)



- By moving registers through logic and hierarchical boundaries, BRT reduces the clock period with minimum area impact.



Unit 2

66



Effectiveness of BRT



From Synopsys

DESIGN TYPE	RTL DESIGN		BEHAVIORAL RETIMING		SUMMARY
	SPEED (CLOCK PERIOD)	GATES	SPEED (CLOCK PERIOD)	GATES	
Control	44 ns	10,913-gates	30.6 ns	11,313 gates	30%-faster, 4%-more-area
Control	23.1 ns	3,598-gates	19.6 ns	4,575 gates	15%-faster, 27%-more-area
Control	28.6 ns	3,585-gates	28.6 ns	3,359 gates	same-speed, 6%-less-area
Dataflow&Control	17 ns	28,900-gates	12.5 ns	30,100 gates	26%-faster, 4%-more-area
Dataflow&Control	16 ns	7,620-gates	13 ns	8,019 gates	20%-faster, 5%-more-area
Dataflow	22 ns	4,990-gates	18.5 ns	5,109 gates	16%-faster, 2%-more-area
Dataflow	28 ns	31,226-gates	26 ns	32,032 gates	8%-faster, 2%-more-area
Dataflow	26.2 ns	14,351-gates	23.6 ns	13,847 gates	10%-faster, 4%-less-area
Dataflow	25.9 ns	16,798-gates	20.8 ns	15,550 gates	20%-faster, 7%-less-area
Dataflow	45 ns	28,705-gates	26 ns	30,987 gates	42%-faster, 8%-more-area

- RTL designs have a single clock net and were synthesized to gates using Synopsys Design Compiler.
- Design type: dataflow implies significant number of operators; control implies state machine dominated.



Unit 2

67



Summary



- ◆ High Level Synthesis (HLS)
- ◆ Scheduling and Binding
- ◆ Operation Scheduling
 - ASAP Scheduling
 - ALAP Scheduling
- ◆ Constrained Scheduling
 - Hu's Algorithm
 - List Scheduling
 - Force-Directed Scheduling
 - Linear Programming



2021/3/16

Andy Yu-Guang Chen

68