

Homework

- #5
 - Due Now
- #6
 - Problem 5-2
 - Problem 5-3
 - Problem 5-8
 - Problem 5-9
 - Problem 6-1
 - Problem 6-2
 - Problem 6-3

Final Examination

- Final Examination
 - Surge 284
 - 6/4 Tuesday 12:40PM-2PM
 - Chapter 8 (Pages 380-386, 396-397, 418-423)
 - Chapter 9 (Pages 441-446, 449-452, 458-465)
 - Chapter 10 (Pages 504-517, 522-529, 544-546)
 - Chapter 4 (Pages 141-174, 178-181)
 - Chapter 5 (Pages 185-215)
 - Chapter 6 (Pages 229-243)
 - Close notes, close book, close calculator, close everything
 - You will be provided with algorithms and formula

Final Project Presentation

- 6/6 Thursday, 2PM-5:20PM
 - 2-2:20 EquiYAPI (Nikhil)
 - 2:20-2:40 LOC2SPIN (Artur)
 - 2:40-3 MOCinMMM (Betul & Lingling)
 - 3-3:20 MOCinMMM (Betul & Lingling)
 - 3:20-3:40 Ptolemy2MMM (Dinesh)
 - 3:40-4 ConforminMMM (Yun)
 - 4-4:20 MMM2Promela (Xi & Fang)
 - 4:20-4:40 MMM2Promela (Xi & Fang)
 - 4:40-5 TCAM (Philip & Suvu)
 - 5-5:20 TCAM (Philip & Suvu)
- Be prepare to run late
 - Cold pizza...

Final project report

- Due 6/6 midnight
 - All relevant software, case study, tared and e-mailed
 - A hardcopy of the report turned in (office)
 - 8 page report, 11pt, 2 cols, single space (conference format)
 - 1 page introduction and motivation
 - 1 page related and background work
 - 5 pages of substantive material
 - 1 page conclusion, future work and reference
 - Leave long codes, large data sets,...etc in appendix, if any
 - Be concise, be formal,
 - use “definition”, “theorem”, “lemma”, “proof”, “example”
 - Organize with sections, subsections, subsubsections, paragraphs
 - Use tables and figures
 - Summarize your result

Final project presentation

- Use PowerPoint
- Approximately 30 slides for 2 person project
 - 2 slides introduction and motivation
 - 2 slides related and background work
 - 24 slides of substantive material
 - 2 slides of conclusion and future work
- Approximately 15 slides for 1 person project
 - 1 slides introduction and motivation
 - 1 slides related and background work
 - 12 slides of substantive material
 - 1 slides of conclusion and future work

Hints on presentation

- Some “rules of thumb” for good presentation
 - Include at least one relevant figure on every slide
 - A picture is worth a thousand words
 - Use single line item whenever possible
 - Use at most three levels of bullets
 - Use font no smaller than 18 pt
 - Use animation iff it is helpful to the understanding
 - Write bullets, use shorthands, no complete sentences

Force-directed scheduling

- Heuristic scheduling methods [Paulin]
 - Min latency subject to resource bound
 - Variation of list scheduling: FDLS
 - Min resource subject to latency bound
 - Schedule one operation at a time
- Rationale
 - Reward uniform distribution of operations across schedule steps
 - Reminiscent of force-directed placement and routing of VLSI circuit

7

CS220: Synthesis of Digital System, Spring 02

Force-directed scheduling definitions

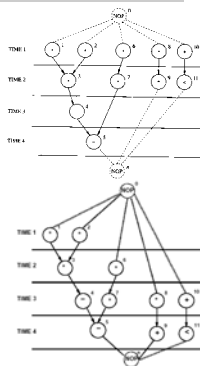
- Operation interval: mobility plus one (u_i+1)
 - Computed by ASAP and ALAP scheduling [t_i^s, t_i^l]
- Operation probability $p_i(l)$
 - Probability of executing in a given step
 - $1/(u_i+1)$ inside interval; 0 elsewhere
- Operation-type distribution $q_k(l)$
 - Sum of the operation probability for each type

8

CS220: Synthesis of Digital System, Spring 02

Example

- $p_1(1)=1, p_1(2)=p_1(3)=p_1(4)=0$
- $p_6(1)=p_6(2)=0.5, p_6(3)=p_6(4)=0$
- $p_8(1)=p_8(2)=p_8(3)=0.3, p_8(4)=0$
- $q_1(1)=1+1+0.5+0.3=2.8$



CS220: Synthesis of Digital System, Spring 02

Force

- Used as priority function
- Force is related to concurrency
 - Sort operations for least force
- Mechanical analogy
 - Force = constant * displacement
 - Constant = operation-type distribution
 - Displacement = change in probability

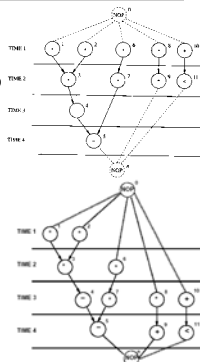
10

CS220: Synthesis of Digital System, Spring 02

Forces related to the assignment of an operation to a control step

- Self-force
 - Sum of forces to other steps
 - Self-force for operation v_i in step

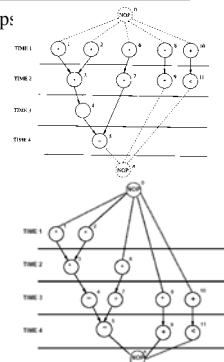
$$\text{self-force}(i, l) = q_i(l) - \frac{i}{\mu_i + 1} \sum_{m=1}^{l-1} q_i(m)$$



CS220: Synthesis of Digital System, Spring 02

E.g. Operation v_6

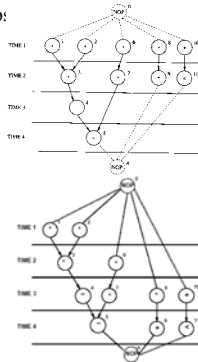
- It can be scheduled in the first 2 steps:
 - $p(1)=p(2)=0.5; p(3)=p(4)=0$
- Distribution: $q(1)=2.8; q(2)=2.3$
- Assign v_6 to step 1
 - Variation in probability $1-0.5=0.5$
 - for step 1
 - Variation in probability $0-0.5=-0.5$
 - for step 2
- Self force
 - $2.8*0.5-2.3*0.5=0.25$
 - To assign v_6 to step 1
 - $2.8-0.5(2.3+2.8)=0.25$



CS220: Synthesis of Digital System, Spring 02

E.g. Operation v_6

- It can be scheduled in the first 2 step:
 - $p(1)=p(2)=0.5$; $p(3)=p(4)=0$
- Distribution: $q(1)=2.8$; $q(2)=2.3$
- 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
 - $-2.8*0.5+2.3*0.5=-0.25$
 - To assign v_6 to step 2
 - $2.3-0.5(2.3+2.8)=-0.25$



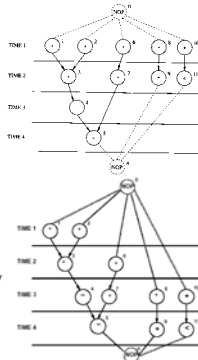
Forces related to the assignment of an operation to a control step

- Successor-force
 - Related to the successors
 - Delaying an operation implies delaying its successors
- Predecessor-force
 - Related to the predecessor
 - “make early” an operation implies “make early” its predecessors

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

Example: operatino v_6

- Successor-force
 - Assign v_6 to step 1
 - v_7 's mobility is not affected
 - Assign v_6 to step 2
 - v_7 must be assigned to step 3
 - $2.3(0-0.5)+0.8(1-0.5)=-0.75$
 - $1*0.8-0.5(0.8+2.3)=-0.75$
- Total-force
 - V_6 at step 1 = 0.25
 - V_6 at step 2 = -1
- Conclusion
 - Least force is for step 2
 - Assign v_6 to step 2 reduces concurrency



Force-directed scheduling algorithm for minimum resources

```

FDS( G(V, E),  $\vec{\lambda}$  ) {
  repeat {
    Compute the time-frames;
    Compute the operation and type probabilities;
    Compute the self-forces, predecessor/successor forces and total forces;
    Schedule the operation with least force and update its time-frame;
  }
  until (all operations are scheduled);
  return (0);
}
    
```

Resource sharing and binding

- Allocation
 - Number of resources available
- Binding
 - Relation between operations and resources
- Sharing
 - Many to one relation
- Optimum binding/sharing
 - Minimize the resource usage



Binding

- Limiting cases
 - Dedicated resources
 - One resource per operation
 - No sharing
 - One multi-task resource
 - ALU
 - One resource per type

19

CS220: Synthesis of Digital System, Spring 02

Optimum sharing problem

- Scheduled sequencing graphs
 - Operation concurrency well defined
- Consider operation types independently
 - Problem decomposition
 - Perform analysis for each resource type

20

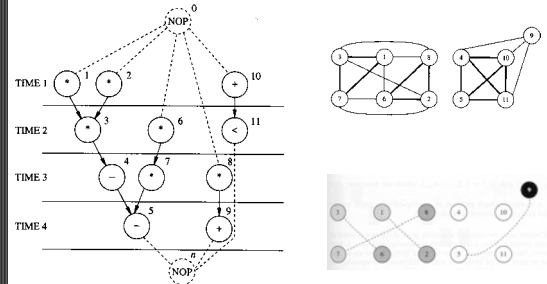
CS220: Synthesis of Digital System, Spring 02

Compatibility and conflicts

- Operation compatibility
 - Same type
 - Non concurrent
- Compatibility graph
 - Vertices: operations
 - Edges: compatibility relation
- Conflict graph
 - Complement of compatibility graph

21

CS220: Synthesis of Digital System, Spring 02



22

CS220: Synthesis of Digital System, Spring 02

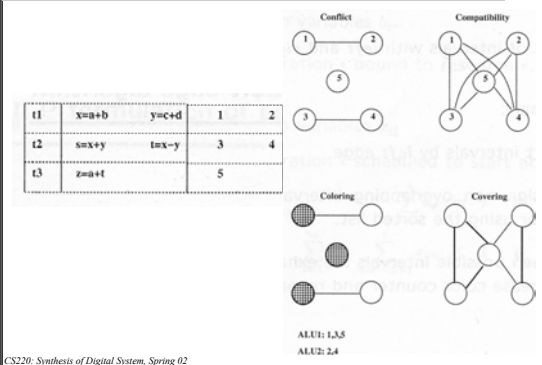
Algorithmic solution to the optimum binding problem

- Compatibility graph
 - Partition the graph into a minimum number of cliques
 - Find clique covering number
- Conflict graph
 - Color the vertices by a minimum number of colors
 - Find chromatic number
- NP-complete problems
 - Heuristic algorithms

23

CS220: Synthesis of Digital System, Spring 02

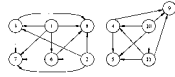
Example



CS220: Synthesis of Digital System, Spring 02

Perfect graphs

- **Comparability graph**
 - Graph $G(V,E)$ has an orientation $G(V,F)$ with the transitive property
 - $(v_i, v_j) \in F \cup (v_j, v_k) \in F \Rightarrow (v_i, v_k) \in F$
 - Resource sharing has an ordering relationship
 - Ordering is transitive
- **Interval graph**
 - Vertices correspond to intervals
 - Edges correspond to interval intersection

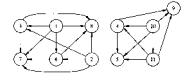


25

CS220: Synthesis of Digital System, Spring 02

Data-flow graphs (flat sequencing graphs)

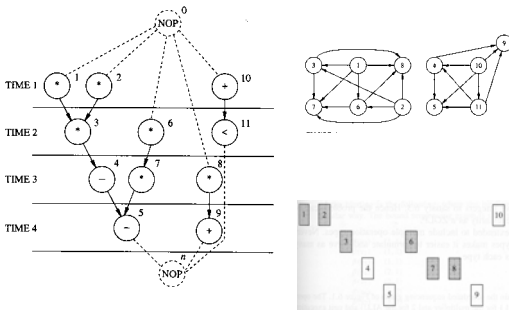
- The compatibility/conflict graphs
 - Compatibility
 - Comparability graph
 - Conflict
 - Interval graph
- Polynomial time solutions
 - Golumbic's algorithm
 - Left-edge algorithm



26

CS220: Synthesis of Digital System, Spring 02

Example



27

CS220: Synthesis of Digital System, Spring 02

Left-edge algorithm

- **Input**
 - Set of intervals with left and right edge
- **Rationale**
 - Sort intervals by left edge
 - Assign non-overlapping intervals to first color using the sorted list
 - When possible intervals are exhausted, increase color counter and repeat

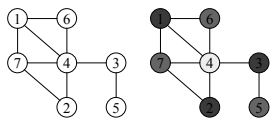
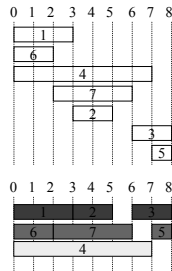
28

CS220: Synthesis of Digital System, Spring 02

Left edge algorithm

```

LEFT_EDGE(L)
{
  Sort elements of L in a list L with ascending order of l;
  c = 0;
  while (Some interval has not been colored) do {
    S = {};
    repeat {
      s = first element in the list L whose left edge
      L is higher than the rightmost edge in S;
      S = S ∪ {s};
    } until (an element s is found);
    c = c + 1;
    color elements of S with color c;
    delete elements of S from L;
  }
}
  
```

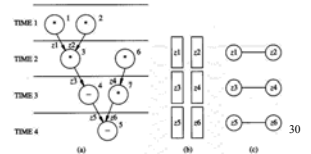


29

CS220: Synthesis of Digital System, Spring 02

Register binding problem

- Given a schedule
 - Lifetime intervals for variables
 - Lifetime overlaps
- Conflict graph (interval graph)
 - Vertices \Leftrightarrow variables
 - Edges \Leftrightarrow overlaps
 - Interval graph
- Compatibility graph (comparability graph)
 - Complement of conflict graph



30

CS220: Synthesis of Digital System, Spring 02

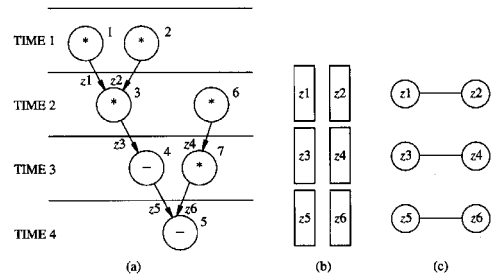
Register sharing data-flow graphs

- Given
 - Variable lifetime conflict graph
- Find
 - Minimum number of registers storing all the variables
- Key point
 - Interval graph
 - Left-edge algorithm (polynomial-time)

31

CS220: Synthesis of Digital System, Spring 02

Example

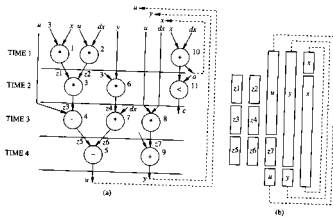


32

CS220: Synthesis of Digital System, Spring 02

Register sharing general case

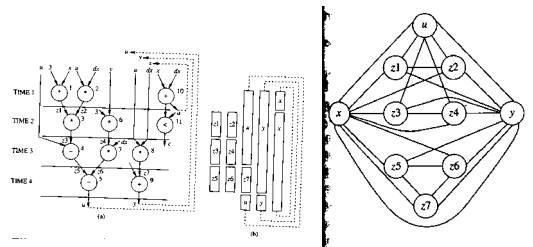
- Iterative constructs
 - Preserve values across iterations
 - Circular-arc conflict graph
 - Coloring is intractable



33

CS220: Synthesis of Digital System, Spring 02

Example



34

CS220: Synthesis of Digital System, Spring 02

What have we learned?

- Two-Level Combinational Logic Optimization
 - Logic optimization (exact, heuristic), multi-value manipulation, unate recursive paradigm, Espresso, Symbolic minimization, encoding
- Multiple-Level Combinational Logic Optimization
 - Logic networks, algebraic model, Boolean model, delay evaluation
- Sequential Logic Optimization
 - State minimization, state encoding, retiming
- Cell-Library Binding
 - Structural matching, boolean matching
- Architectural Synthesis
 - Performance estimation, cycle-time/latency/area tradeoff, control synthesis
- Scheduling Algorithm
 - ALAP, ASAP, Hu's algorithm, List scheduling
- Resource Sharing and Binding
 - Resource sharing, register sharing

35

CS220: Synthesis of Digital System, Spring 02