ESE535: Electronic Design Automation

Day 16: March 18, 2013
Architecture Synthesis (Provisioning, Allocation)

Today
• Problem
• Brute-Force/Exhaustive
• Greedy
• Estimators
• Analytical Provisioning
• ILP Schedule and Provision

Previously
• General formulation for scheduled operator sharing
  – VLIW
• Fast algorithms for scheduling onto fixed resource set
  – List Scheduling

Today
• How do we determine the set of resources?

Today: Provisioning
• Given
  – An area budget
  – A graph to schedule
  – A library of operators
• Determine:
  – Delay minimizing set of operators
  – Or delay-achieving set of operators
  – i.e. select the operator set
Exhaustive

1. Identify all area-feasible operator sets
   - E.g. preclass exercise
2. Schedule for each
3. Select best
   - → optimal
   - Drawbacks?

Exhaustive

• How large is space of feasible operator sets?
  – As function of
    • operator types – \( O \)
    – Types: add, multiply, divide, ....
    • Maximum number of operators of type \( m \)

Implication

• Feasible operator space can be too large to explore exhaustively

Greedy Incremental

• Start with one of each operator
• While (there is area to hold an operator)
  – Which single operator
    • Can be added without exceeding area limit?
    • And provides largest benefit/operator-area?
  – Add one operator of that type
• How long does this run?
  – \( T_{\text{schedule}}(E)^* O(\text{operator-types} \times A) \)
• Weakness?

Example

Find best 5 operator solution.

Example

Find best 5 operator solution.
Example

Find best 5 operator solution.

One of each.

<table>
<thead>
<tr>
<th>Sq</th>
<th>Dia</th>
<th>Circ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>F</td>
<td>I</td>
</tr>
<tr>
<td>D</td>
<td>G</td>
<td>J</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K</td>
</tr>
</tbody>
</table>

Example

Find best 5 operator solution.

Two Squares

<table>
<thead>
<tr>
<th>Sq</th>
<th>Dia</th>
<th>Circ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,D</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>I</td>
</tr>
<tr>
<td>G</td>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>H</td>
<td>J</td>
<td>K</td>
</tr>
</tbody>
</table>

Example

Find best 5 operator solution.

Two Diamonds

<table>
<thead>
<tr>
<th>Sq</th>
<th>Dia</th>
<th>Circ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>F</td>
<td>I</td>
</tr>
<tr>
<td>D</td>
<td>G</td>
<td>J</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K</td>
</tr>
</tbody>
</table>

Example

Find best 5 operator solution.

Two Circles

<table>
<thead>
<tr>
<th>Sq</th>
<th>Dia</th>
<th>Circ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>F</td>
<td>I</td>
</tr>
<tr>
<td>D</td>
<td>G</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

Find best 5 operator solution.

Which should greedy add?

Incremental addition does not accelerate.

Two sqs + Two diamonds

<table>
<thead>
<tr>
<th>Sq</th>
<th>Dia</th>
<th>Circ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,D</td>
<td>E,F</td>
<td></td>
</tr>
<tr>
<td>G,H</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>K</td>
<td></td>
</tr>
</tbody>
</table>

Example

Find best 5 operator solution.

Two Squares

Max effect: Incremental may not suggest next single addition.

(maybe better with cost function that accounts for total slack?)
Analytic Formulation

Challenge
- Scheduling expensive
  - $O(|E|) \text{ or } O(|E|^* \log(|V|))$ using list-schedule
- Results not analytic
  - Cannot write an equation around them
- Bounds are sometimes useful
  - No precedence $\rightarrow$ is resource bound
  - Often one bound dominates
    - Latency bound unaffected by operator count

Estimations
- Step 1: estimate with resource bound
  - $O(|E|)$ vs. $O(|V|)$ evaluation
- Step 2: use estimate in equations
  - $T = \max(N_1/M_1, N_2/M_2, \ldots)$
- Most useful when RB$>>$CP

Constraints
- Let $A_i$ be area of operator type $i$
- Let $M_i$ be number of operators of type $i$
  \[
  \sum A_i \times M_i \leq \text{Area}
  \]

Achieve Time Target
- Want to achieve a schedule in $T$ cycles
- Each resource bound must be less than $T$ cycles:
  - $N_i/M_i \leq T$

Algebraic Solve
- Set of equations
  - $N_i/M_i \leq T$
  - $\sum A_i M_i \leq \text{Area}$
- Assume equality for time bound
  - $N_i/M_i = T \Rightarrow M_i = N_i/T$
  \[
  \sum \frac{A_i \times N_i}{T} \leq \text{Area}
  \]
Rearranging

$$\sum \frac{A_i \times N_i}{T} \leq Area$$

$$\sum \frac{A_i \times N_i}{Area} \leq T$$

Bounding T

• Gives Lower Bound on T

$$\sum \frac{A_i \times N_i}{Area} \leq T$$

Intuition: N of each is right balance given unbounded area; Scale to area available.

Preclass

• What is $T_{\text{lower}}$ for preclass?

$$\sum \frac{A_i \times N_i}{Area} \leq T$$

$$T \geq \frac{1 \times 8 + 2 \times 4}{7} = \frac{16}{7} = 2.3 \quad T \geq 3$$

Back Substitute from T to x

• $M_i = \frac{N_i}{T}$

$$\sum \frac{A_i \times N_i}{Area} \leq T$$

• $M_i$ won’t necessarily be integer
  – Round down definitely feasible solution
  – May have room to move a few up by 1
• Reduces range may need to search
  – (just over the residual area once rounded down)

Preclass

• $M_i = \frac{N_i}{T}$
• $T \geq 3$
• $M_{\text{add}}, M_{\text{mpy}}$?
  • $M_{\text{add}} = \frac{8}{3} \Rightarrow 2$ or $3$
  • $M_{\text{mpy}} = \frac{4}{3} \Rightarrow 1$ or $2$

Counter Example

• 1 Unit each
• Area = 4 Units
• What would analytic predict?
• What is best?
• How does CP compare to RB?

• Analytic Resource Estimate
  – Most useful when RB $>>$ CP
Analytic Counter Example

• How would greedy incremental work on this one?

ILP

Maybe we can do exhaustive, if we formulate properly.

ILP

• Integer Linear Programming
  • Formulate set of linear equation constraints (inequalities)
    ▪ \( A x_0 + B x_1 + C x_2 \leq D \)
    ▪ \( x_i + x_f = 1 \)
    ▪ \( A,B,C,D \) – constants
    ▪ \( x \) – variables to satisfy
    ▪ No products on variables, just linear weighted sums
  • Can constrain variables to integers
  • No polynomial time guarantee
    – But often practical
    – Solvers exist (significant piece next lecture)

ILP Provision and Schedule

Now to make it look like an ILP nail…

• Formulate operator selection and scheduling as ILP problem

Formulation

• Integer variables \( M_i \)
  – number of operators of type \( i \)
• 0-1 (binary) variables \( x_{i,j} \)
  – 1 if node \( i \) is scheduled into timestep \( j \)
  – 0 otherwise
• Variable assignment completely specifies operator selection and schedule
• This formulation for achieving a target time \( T \)
  – \( j \) ranges 0 to \( T-1 \)

Target \( T \rightarrow \text{Min } T \)

• Formulation targets \( T \)
  • What if we don’t know \( T \)?
    – Want to minimize \( T \)
  • Do binary search for minimum \( T \)
    – How does that impact solution time?
Constraints

What properties must hold true for a solution to be valid?
1. Total area constraints
2. Not assign too many things to a timestep
3. Assign every node to some timestep
4. Maintain precedence

(1) Total Area

- Same as before

\[ \sum A_i \times M_i \leq Area \]

(2) Not overload timestep

- For each timestep \( j \)
  - For each operator type \( k \)

\[ \sum_{o_i \in F \cup U_k} x_{i,j} \leq M_k \]

(3) Node is scheduled

- For each node in graph

\[ \sum_j x_{i,j} = 1 \]

Can narrow to sum over slack window.

(4) Precedence Holds

- For each edge from node \( src \) to node \( snk \)

\[ \sum_j j \times x_{src,j} - \sum_j j \times x_{snk,j} \leq -1 \]

Can narrow to sum over slack windows.

Constraints

1. Total area constraints
2. Not assign too many things to a timestep
3. Assign every node to some timestep
4. Maintain precedence
ILP Solver

- ILP Solver can take these constraints and find a solution (satisfying assignment)
- On Wednesday, will see how to start to make this practical

Two Constraint Challenge

- Processing elements have limited memory
  - Instruction memory (data memory)
- Tasks have different requirements for compute and instruction memory
  - i.e. Run length not correlated to code length
- No provisioning, scheduling

Task

- **Task**: schedule tasks onto PEs obeying both memory and compute capacity limits

Example and ILP solution From Plischker et al. NSCD2004

SAT/ILP

Scheduling Variant

(Demonstration)

<if time permits>

Plishker Task Example

Example: 4 Port DiffServ

Task

- **Task**: schedule tasks onto PEs obeying both memory and compute capacities
  - two capacity assignment problem
  - two capacity bin packing problem
  - Task: $i < C_{i,1}$
SAT Packing

Variables:
- $A_{i,j}$ – task $i$ assigned to resource $j$

Constraints
- Coverage constraints
- Uniqueness constraints
- Cardinality constraints
  - PE compute: $\sum_j (A_{i,j} \times C_i) \leq PE.cap(j)$

Allow Code Sharing

- Two tasks of same type can share code
- Instead of memory capacity
  - Vector of memory usage
- Compute PE Imem vector
  - As OR of task vectors assigned to it
- Compute mem space as sum of non-zero vector entry weights (dot product)

Allow Code Sharing

- Two tasks of same type can share code
- Task has vector of memory usage
  - Task $i$ needs set of instructions $k$: $T_{i,k}$
- Compute PE Imem vector
  - OR (all $i$): PE.Imem$_{j,k} += A_{i,j} \times T_{i,k}$
- PE Mem space
  - PE.Total_Imem$_j = \sum$ (PE.Imem$_{j,k} \times$Instrs($k$))

Symmetries

- Many symmetries
- Speedup with symmetry breaking
  - Tasks in same class are equivalent
  - PEs indistinguishable
  - Total ordering on tasks and PEs
  - Add constraints to force tasks to be assigned to PEs by ordering
  - Plishker claims "significant runtime speedup"
  - Using GALENA [DAC 2003] pseudo-Boolean SAT solver

Plishker Task Example

Example: 4 Port DiffServ

Results

Greedy (first-fit) binpack
SAT/ILP Solve

Solutions in < 1 second
Why can they do this?

- Ignore precedence?
- Ignore Interconnect?

Interconnect Buffers

- Allow “Software Pipelining”
  
  Each data item

  Spatial we would pipeline, running all three at once

  Think of each schedule instance as one timestep in spatial pipeline.

Round up Algorithms and Runtimes

- Exhaustive Schedule
- Greedy Schedule
- Analytic Estimates
- ILP formulation

Big Ideas:

- Estimators
- Dominating Effects
- Reformulate as a problem we already have a solution for
  - ILP
- Technique: Greedy
- Technique: ILP
Admin

• Reading for Wednesday on web
• My grading priority now will be 5a