Scheduling: Exact Methods
Announcements

▶ Sign up for the first student-led discussions today
  – One slot remaining
  – Presenters for the 1st session will meet with instructor on Tuesday 9/19
  – Form groups on CMS by tomorrow (9/15) 11:59pm

▶ HW 1 due tomorrow at 11:59pm

▶ Lab 2 to be released tomorrow
Example: Basic Blocks and CFG

- Can you partition the program into basic blocks and draw the corresponding CFG?

for (i = 0; i < N; i++)
    if (i > 0) A[i-1] = 0;
return;

Leader statement is:
(1) the first in the program
(2) any that is the target of a branch
(3) any that immediately follows a branch
(4) A[i-1] = 0
(5) i++
(6) goto (2)
(7) return
Review: Dominator Tree and Dominance Frontier

Algorithm to compute DF set
For each convergence point $X$ in the CFG
For each predecessor, $Y$, of $X$ in the CFG
Run up to $Z=\text{IDOM}(X)$ in the dominator tree,
adding $X$ to $\text{DF}(N)$ for each $N$ between $[Y, Z)$
Example: PHI Node Placement

- $X$ are defined in $B_0$ and $B_4$ in non-SSA form
- Can you identify all the basic blocks where $\phi$-nodes need to be inserted for $X$ in the SSA form?
Agenda

- Unconstrained scheduling
  - ASAP and ALAP

- Constrained scheduling
  - Resource constrained scheduling (RCS)
  - Exact formulations with integer linear programming (ILP)
Review: A Typical HLS Flow

High-level Programming Languages
(C/C++, OpenCL, SystemC, ...)

Parsing

Transformations

Intermediate Representation (IR)

Allocation

Scheduling

Binding

RTL generation

if (condition) {
  ...
} else {
  t_1 = a + b;
  t_2 = c * d;
  t_3 = e + f;
  t_4 = t_1 * t_2;
  z = t_4 - t_3;
}

Control data flow graph (CDFG)

Finite state machines with datapath

3 cycles

Intermediate Representation (IR)
Scheduling in High-Level Synthesis

- Scheduling: a central problem in HLS
  - Introduce clock boundaries to untimed or partially timed input specification
  - Significant impact on QoR
    - Frequency
    - Latency
    - Throughput
    - Area
    - Power
    ...

Scheduling: Untimed to Timed

Control-Data Flow Graph

Latency

Area

Throughput

\[ out_1 = f(in1, in2, in3, in4) \]

\[
\begin{align*}
  t_{clk} & = 3 \, d_{add} \\
  T_1 &= 1 / t_{clk} \\
  A_1 &= 3 \, A_{add} \\
  t_{clk} & \approx d_{add} + d_{setup} \\
  T_2 &= 1 / (3 \, t_{clk}) \\
  A_2 &= A_{add} + 2 \, A_{reg} \\
  t_{clk} & = d_{add} + d_{setup} \\
  T_3 &= 1 / t_{clk} \\
  A_3 &= 3 \, A_{add} + 6 \, A_{reg}
\end{align*}
\]
Scheduling Input

- Control data flow graph (CDFG)
  - Generated by a compiler front end from high-level description
  - Nodes
    - Operations (and pseudo operations)
  - Directed edges
    - Data edges, control edges, precedence edges

- Without control flow, the basic structure is a data flow graph (DFG)

\[ xl = x+dx; \]
\[ ul = u-3*x*u*dx-3*y*dx \]
\[ yl = y+u*dx \]
\[ c = xl<a; \]
\[ x = xl; u = ul; y = yl; \]
Scheduling Output

- Scheduling: map operations to states
- Each clock cycle corresponds to a state in the FSM
  - Commonly referred to as control step (c-step)

DFG

FSM or State Transition Diagram (STG)
Unconstrained Scheduling

- Only consideration: dependence

- As soon as possible (ASAP)
  - Schedule an operation to the earliest possible step

- As late as possible (ALAP)
  - Schedule an operation to the earliest possible step, without increasing the total latency
ASAP Schedule

\[ Y = ((a*b)+c)+(d*e)-(f+g) \]

The start time for each operation is the least one allowed by the dependencies

**ASAP(G(V, E)):**

\[ V' = \text{Topological\_Sort}(G) \]

**foreach** \( v_i \) in \( V' \):

// Primary inputs (PIs) to first cycle

if \( v_i \in \text{PIs} \): \( t_i = 1 \)

// Assume no chaining & single-cycle operations

else: \( t_i = \max(t_j + 1) \); // \((v_j, v_i) \in E\)
ALAP Schedule

ALAP(G(V, E), L): // L is the latency bound
V’ = Reverse_Topological_Sort(G)
foreach vᵢ in V’:
  // Primary outputs (Pos) to last cycle
  if vᵢ ∈ POs: tᵢ = L
  // Assume no chaining & single-cycle operations
else: tᵢ = min(tⱼ) - 1; // (vᵢ, vⱼ) ∈ E

The end time of each operation is the latest one allowed by the dependencies and the latency constraint

Y = ((a*b)+c)+(d*e)-(f+g)
Operation Mobility (Slack)

Mobility (or slack) is the difference of the start times computed by the ALAP and ASAP

\[ Y = ((a \times b) + c) + (d \times e) - (f + g) \]
Constrained Scheduling

- Constrained scheduling
  - General case NP-hard
  - Resource-constrained scheduling (RCS)
    - Minimize latency given constraints on area or resources
  - Time-constrained scheduling (TCS)
    - Minimize resources subject to bound on latency

- Exact methods
  - Integer linear programming (ILP)
  - Hu’s algorithm for a very restricted problem

- Heuristics
  - List scheduling
  - Force-directed scheduling
  - SDC-based scheduling
  …
Linear Programming

- **Linear programming (LP)** solves the problem of maximizing or minimizing a linear objective function subject to linear constraints
  - Efficiently solvable both in theory and in practice

- **Integer linear programming (ILP)**: in addition to linear constraints and objective, the values for the variables have to be integer
  - NP-Hard in general (A special case, 0-1 ILP)
  - Modern ILP solvers can handle problems with nontrivial size

- Enormous number of problems can be expressed in LP or ILP
Canonical Form of ILP

maximize $c_1x_1 + c_2x_2 + \ldots + c_nx_n$ // objective function
subject to // linear constraints
    $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1$
    $a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2$
    ....
    $a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m$
    $x_i \geq 0$
    $x_i \in \mathbb{Z}$

Vector form

maximize $c^Tx$ // $c = (c_1, c_2, \ldots, c_n)$
subject to // $A$ is a $m \times n$ matrix; $b = (b_1, b_2, \ldots, b_n)$
    $Ax \leq b$
    $x \geq 0$ and
    $x_i \in \mathbb{Z}$
Example: Course Selection Problem

A student is about to finalize her course selection for the coming semester, given the following information:

- Minimum credits / semester: 8

<table>
<thead>
<tr>
<th>Course</th>
<th>Schedule</th>
<th>Credits</th>
<th>Est. workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Big data analytics</td>
<td>MW 2:00-3:30pm</td>
<td>3</td>
<td>8 hrs</td>
</tr>
<tr>
<td>2. How to build a start-up</td>
<td>TT 2:00-3:00pm</td>
<td>2</td>
<td>4 hrs</td>
</tr>
<tr>
<td>3. Linear programming</td>
<td>MW 9:00-11:00am</td>
<td>4</td>
<td>10 hrs</td>
</tr>
<tr>
<td>4. Analog circuits</td>
<td>TT 1:00-3:00pm</td>
<td>4</td>
<td>12 hrs</td>
</tr>
</tbody>
</table>

Question: Which courses to take to minimize the amount of work?
ILP Formulation for Course Selection

- Define decision variables \((i = 1, 2, 3, 4)\):
  \[ x_i = \begin{cases} 
  1 & \text{if course } i \text{ is taken} \\
  0 & \text{if not} 
  \end{cases} \]

- The total expected work hours: \(8x_1 + 4x_2 + 10x_3 + 12x_4\)
- The total credits taken: \(3x_1 + 2x_2 + 4x_3 + 4x_4\)
- Account for the schedule conflict: \(x_2 + x_4 \leq 1\)

- Complete ILP formulation (in canonical form):
  \[
  \text{minimize } 8x_1 + 4x_2 + 10x_3 + 12x_4 \\
  \text{s.t. } 3x_1 + 2x_2 + 4x_3 + 4x_4 \geq 8 \\
  x_2 + x_4 \leq 1 \\
  x_i \in \{0, 1\} 
  \]

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>CRs</th>
<th>Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Big data</td>
<td>MW 2-3:30pm</td>
<td>3</td>
<td>8 hrs</td>
</tr>
<tr>
<td>2. Start-up</td>
<td>TT 2-3pm</td>
<td>2</td>
<td>4 hrs</td>
</tr>
<tr>
<td>3. Linear prog.</td>
<td>MW 9-11am</td>
<td>4</td>
<td>10 hrs</td>
</tr>
<tr>
<td>4. Analog</td>
<td>TT 1-3pm</td>
<td>4</td>
<td>12 hrs</td>
</tr>
</tbody>
</table>
When functional units are limited

- Each functional unit can only perform one operation at each clock cycle
  - e.g., if there are only K adders, no more than K additions can be executed in the same c-step

A resource-constrained scheduling problem for DFG

- Given the number of functional units of each type, minimize latency
- NP-hard problem
ILP Formulation of RCS

- Use binary decision variables
  - \( x_{ik} = 1 \) if operation \( i \) starts at step \( k \), otherwise \( = 0 \).
  - \( i = 1, 2, ..., N \) : \( N \) is the total number of operations
  - \( k = 1, ..., L \) : \( L \) is the given upper bound on latency

\[
t_i = \sum_{k=1}^{L} k x_{ik}
\]

\( t_i \) indicates the start time of operation \( i \)
ILP Formulation of RCS: Constraints (1)

- Linear constraints:
  - Unique start times: \[ \sum_{k} x_{ik} = 1, \quad i = 1, 2, ..., N \]
  - Dependence must be satisfied (no chaining)
    \[ t_j \geq t_i + d_i + 1 : \forall (v_i, v_j) \in E \Rightarrow \sum_{k} k \cdot x_{jk} \geq \sum_{k} k \cdot x_{ik} + d_i + 1 \]
    
    \( v_j \) must not start before \( v_i \) completes
    since \( v_j \) depends on \( v_i \)
Start Time vs. Time(s) of Execution

- $d_i$: latency of operation $i$
  - $d_i = 0$ indicates single-cycle combinational logic

- When $d_i = 0$, then the following questions are the same:
  - Does operation $i$ start at step $k$
  - Is operation $i$ running at step $k$

- But if $d_i > 0$, then the two questions should be formulated as:
  - Does operation $i$ start at step $k$
    - Check if $x_{ik} = 1$ hold
  - Is operation $i$ running at step $k$
    - Check if the following hold?
      \[
      \sum_{l=k-d_i}^{k} x_{il} \neq 1
      \]
Operation $v_i$ Still Running at Step $k$?

- Is $v_9$ ($d_9 = 2$) running at step 6?
  
  If and only if $x_{9,6} + x_{9,5} + x_{9,4}$ equals 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
ILP Formulation of RCS: Constraints (2)

- Linear constraints:
  
  - Unique start times: \( \sum_{k} x_{ik} = 1, \quad i = 1, 2, \ldots, N \)
  
  - Dependence must be satisfied (no chaining)
    \[
    t_j \geq t_i + d_i + 1: \forall (v_i, v_j) \in E \Rightarrow \sum_{k} k \cdot x_{jk} \geq \sum_{k} k \cdot x_{ik} + d_i + 1
    \]
  
  - Resource constraints
    \[
    \sum_{i: RT(v_i) = r} \sum_{l=k-d_i} x_{il} \leq a_r, \quad r = 1, \ldots, n_{res}, \quad k = 1, \ldots, L
    \]
    \(RT(v_i)\): resource type ID of operation \(v_i\) (between 1~\(n_{res}\))
    \(a_r\) is the number of available resources for resource of type \(r\)
Students presenting next Thursday are expected to set up an appointment with the instructor.

Next lecture: More scheduling algorithms
Acknowledgements

- These slides contain/adapt materials developed by
  - Ryan Kastner (UCSD)