RESOURCE SHARING

©Giovanni De Micheli

Stanford University
Outline

- Resource-dominated circuits.
  - Flat and hierarchical graphs.
  - Functional and memory resources.

- Extensions.
  - Non resource-dominated circuits.
  - Concurrent scheduling and binding.
  - Module selection.
Allocation 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.
• Limiting cases:
  – Dedicated resources:
    * One resource per operation.
    * No sharing.
  – One multi-task resource:
    * ALU.
  – One resource per type.
Optimum sharing problem

- Scheduled sequencing graphs.
  - Operation concurrency well defined.

- Consider operation types independently.
  - Problem decomposition.
  - Perform analysis for each resource type.
Compatibility and conflicts

- Operation compatibility:
  - Same type.
  - Non concurrent.

- Compatibility graph:
  - Vertices: operations.
  - Edges: compatibility relation.

- Conflict graph:
  - Complement of compatibility graph.
Example

- Multiplier
- ALU
Algorithmic solution to the optimum binding problem

- **Compatibility** graph.
  - Partition the graph into a minimum number of cliques.
  - Find *clique cover number* $\kappa(G_+)$. 

- **Conflict** graph.
  - Color the vertices by a minimum number of colors.
  - Find *chromatic number* $\chi(G_-)$. 

- NP-complete problems – Heuristic algorithms.
Example

| t1 | x=a+b   | y=c+d   | 1 | 2 |
| t2 | s=x+y   | t=x−y   | 3 | 4 |
| t3 | z=a+t   |         | 5 |

Conflict

Compatibility

Coloring

Covering

ALU1: 1,3,5
ALU2: 2,4
Perfect graphs

- **Comparability graph:**
  
  - Graph $G(V, E)$ has an orientation $G(V, F)$ with the transitive property.
  
  - $(v_i, v_j) \in F \cup (v_j, v_k) \in F \Rightarrow (v_i, v_k) \in F$.

- **Interval graph:**
  
  - Vertices correspond to *intervals*.
  
  - Edges correspond to interval intersection.
  
  - Subset of *chordal* graphs:
    
    * Every loop with more than three edges has a chord.
Data-flow graphs
(flattening graphs)

- The compatibility/conflict graphs have special properties.
  - Compatibility:
    * Comparability graph.
  - Conflict:
    * Interval graph.

- Polynomial time solutions:
  - Golumbic’s algorithm.
  - Left-edge algorithm.
Example
Example
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.
Left-edge algorithm

LEFT_EDGE(I) { 
    Sort elements of I in a list L in ascending order of $l_i$; 
    $c = 0$; 
    while (some interval has not been colored) do { 
        $S = \emptyset$; 
        $r = 0$; 
        while ($\exists s \in L$ such that $l_s > r$) do{ 
            $s =$ First element in the list L with $l_s > r$; 
            $S = S \cup \{s\}$; 
            $r = r_s$; 
            Delete s from L; 
        } 
        $c = c + 1$; 
        Label elements of $S$ with color $c$; 
    } 
}
Example
ILP formulation of binding

- Boolean variables $b_{ir}$
  - Operation $i$ bound to resource $r$.

- Boolean variables $x_{il}$
  - Operation $i$ scheduled to start at step $l$.

\[ \sum_{r=1}^{a} b_{ir} = 1 \quad \forall i \]
\[ \sum_{i=1}^{n_{ops}} b_{ir} \sum_{m=l-d_i+1}^{l} x_{im} \leq 1 \quad \forall l \quad \forall r \]
Hierarchical sequencing graphs

• Hierarchical conflict/compatibility graphs.
  – Easy to compute.
  – Prevent sharing across hierarchy.

• Flatten hierarchy.
  – Bigger graphs.
  – Destroy nice properties.
Example

(a) TIME 1
(a) TIME 2
(a) TIME 3
(a) TIME 4
(a) TIME 5
(a) TIME 6
(a) TIME 7

(b) a + a *

(c) a 2 3 4

© GDM
Example

(a) (b) (c)
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.
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).
Example

TIME 1

TIME 2

TIME 3

TIME 4

(a)  (b)  (c)
Register sharing
general case

• Iterative constructs:
  – Preserve values across iterations.
  – *Circular-arc* conflict graph:
    * Coloring is intractable.

• Hierarchical graphs:
  – General conflict graphs:
    * Coloring is intractable.

• Heuristic algorithms.
Example
Example
Variable-lifetimes and circular-arc conflict graph
Multiport-memory binding

- Find *minimum number of ports* to access the required number of variables.

- Variables use the same port:
  
  - Port compatibility/conflict.
  
  - Similar to resource binding.

- Variables can use any port:
  
  - Decision variable $x_{il}$ is TRUE when variable $i$ is accessed at step $l$.

  - Optimum: $\max_{1 \leq l \leq \lambda + 1} \sum_{i=1}^{n_{\text{var}}} x_{il}$.
Multiport-memory binding

- Find maximum number of variables to be stored through a fixed number of ports \( a \).

  - Boolean variables \( \{b_i, i = 1, 2, \ldots, n_{\text{var}}\} \):
    * Variable \( i \) is stored in array.

  - \( \max \sum_{i=1}^{n_{\text{var}}} b_i \) such that

  - \( \sum_{i=1}^{n_{\text{var}}} b_i \ x_{il} \leq a \quad l = 1, 2, \ldots, \lambda + 1 \)
Example
formulation

Time – step 1  :  \( r_3 = r_1 + r_2 \); \( r_{12} = r_1 \)
Time – step 2  :  \( r_5 = r_3 + r_4 \); \( r_7 = r_3 \times r_6 \); \( r_{13} = r_3 \)
Time – step 3  :  \( r_8 = r_3 + r_5 \); \( r_9 = r_1 + r_7 \); \( r_{11} = r_{10}/r_5 \)
Time – step 4  :  \( r_{14} = r_{11} \land r_8 \); \( r_{15} = r_{12} \lor r_9 \)
Time – step 5  :  \( r_1 = r_{14} \); \( r_2 = r_{15} \)

\[
\max \sum_{i=1}^{15} b_i \text{ such that }
\]

\[
\begin{align*}
  b_1 + b_2 + b_3 + b_{12} & \leq a \\
  b_3 + b_4 + b_5 + b_6 + b_7 + b_{13} & \leq a \\
  b_1 + b_3 + b_5 + b_7 + b_8 + b_9 + b_{10} + b_{11} & \leq a \\
  b_8 + b_9 + b_{11} + b_{12} + b_{14} + b_{15} & \leq a \\
  b_1 + b_2 + b_{14} + b_{15} & \leq a
\end{align*}
\]
Example solution

- One port $a = 1$:
  - $\{b_2, b_4, b_8\}$ non-zero.
  - 3 variables stored: $v_2, v_4, v_8$.

- Two ports $a = 2$:
  - 6 variables stored: $v_2, v_4, v_5, v_{10}, v_{12}, v_{14}$

- Three ports $a = 3$:
  - 9 variables stored: $v_1, v_2, v_4, v_6, v_8, v_{10}, v_{12}, v_{13}$
Bus sharing and binding

- Find the *minimum number of busses* to accommodate *all* data transfer.

- Find the *maximum number of data transfers* for a fixed number of busses.

- Similar to memory binding problem.

- ILP formulation or heuristic algorithms.
Example

- One bus:
  - 3 variables can be transferred.

- Two busses:
  - All variables can be transferred.
Scheduling and binding
Resource dominated circuits

- Area and delay of resources dominate.

- Strategy:
  - Scheduling under area constraints:
    * Minimize latency.
  - Binding.
    * Share resource within bounds.

- Decoupling between scheduling and binding.
Scheduling and binding
General circuits

• Area and delay influenced by:
  
  – Sparse logic, wiring, registers and control circuit.

• Binding affects the cycle-time:
  
  – It may invalidate a schedule.

• Scheduling after binding:
  
  – Binding under restrictive assumptions.
  
  – Time-frame of operations not yet known.
Scheduling and binding approaches

- *Concurrent* scheduling and binding.
  - ILP model- exact.
  - Some heuristic algorithms.

- *Scheduling before binding:*
  - Good for DSP application.

- *Binding before scheduling:*

- *Iterative* techniques.
Module selection problem

• Library of resources:
  – More than one resource per type.

• Example:
  – Ripple-carry adder.
  – Carry look-ahead adder.

• Resource modeling:
  – Resource subtypes with:
    * (area, delay) parameters.
Module selection solution

- **ILP formulation:**
  - Decision variables:
    * Select resource sub-type.
    * Determine \((\text{area, delay}).\)

- **Heuristic algorithms:**
  - Determine *minimum latency* with fastest resource subtypes.
  - Recover area by using slower resources on non-critical paths.
• Multipliers with:

  - (Area, delay) = (5,1) and (2,2)

• Latency bound of 5.
Example (2)

- Latency bound of 4.
  - Fast multipliers for \( \{v_1, v_2, v_3\} \).
  - Slower multipliers can be used elsewhere.
    * Less sharing.

- Minimum-area design uses fast multipliers only.
Summary

- Resource sharing is reducible to coloring/clique-covering.

- Simple for flat graphs.

- Intractable, but still easy in practice, for other graphs.

- More complicated for non resource-dominated circuits.

- Extension: module selection.