RESOURCE SHARING

©*Giovanni De Micheli*

Stanford University

Outline

© GDM

- 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

© GDM

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

Binding

© GDM

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

Optimum sharing problem

© GDM

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

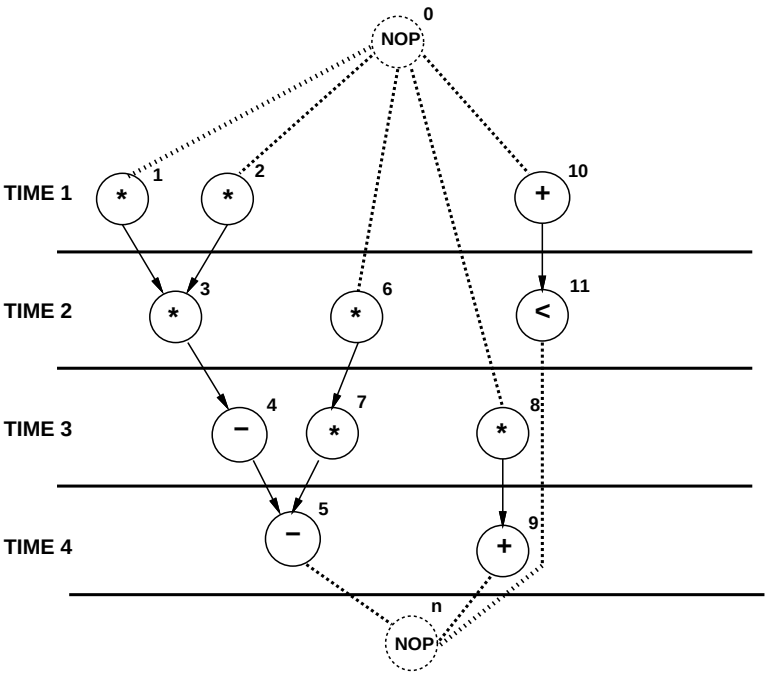
Compatibility and conflicts

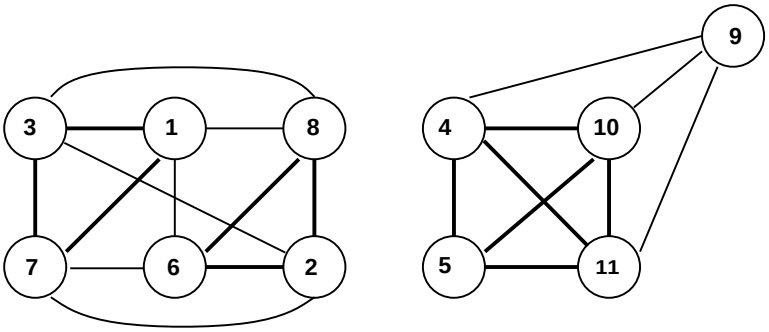
© GDM

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

Example

© GDM





- Multiplier ALU

Algorithmic solution to the optimum binding problem

© GDM

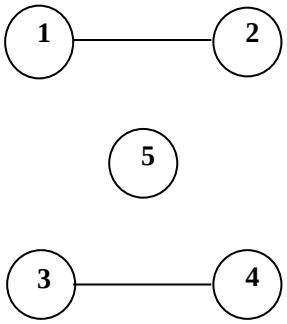
- *Compatibility* graph.
  - Partition the graph into a minimum number of cliques.
  - Find *clique cover number* $\kappa(G_+)$.
- *Conflict* graph.
  - Color the vertices by a minimum number of colors.
  - Find *chromatic number* $\chi(G_-)$.
- NP-complete problems – Heuristic algorithms.

Example

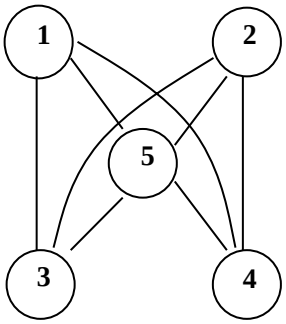
© GDM

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

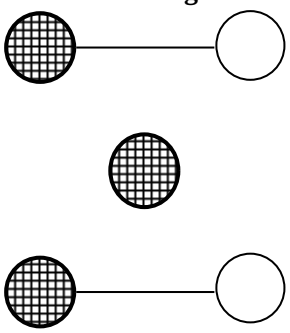
Conflict



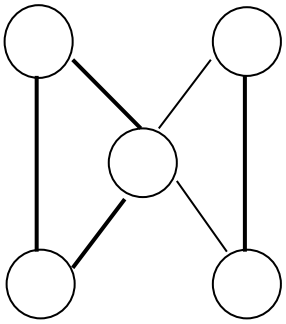
Compatibility



Coloring



Covering



ALU1: 1,3,5

ALU2: 2,4

Left-edge algorithm

© GDM

- 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.

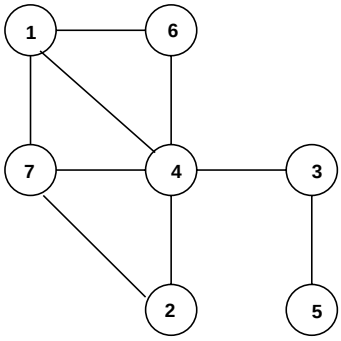
Left-edge algorithm

© GDM

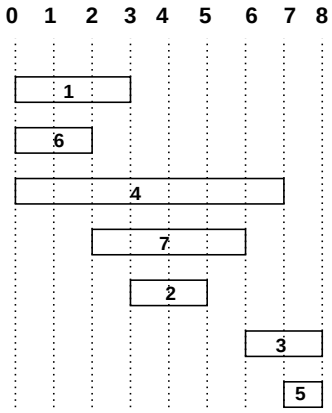
```
LEFT_EDGE(I) {
    Sort elements of I in a list L in ascending order of l_i;
    c = 0;
    while (some interval has not been colored ) do {
        S = ∅;
        r = 0;
        while ( ∃s ∈ L such that l_s > r) do{
            s = First element in the list L with l_s > r;
            S = S ∪ {s};
            r = r_s;
            Delete s from L;
        }
        c = c + 1;
        Label elements of S with color c;
    }
}
```

Example

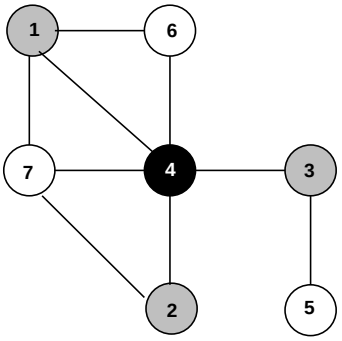
© GDM



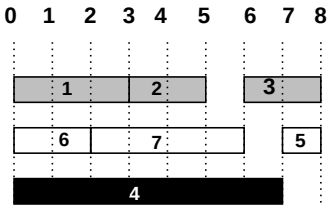
(a)



(b)



(c)



(d)

Register binding problem

© GDM

- 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.

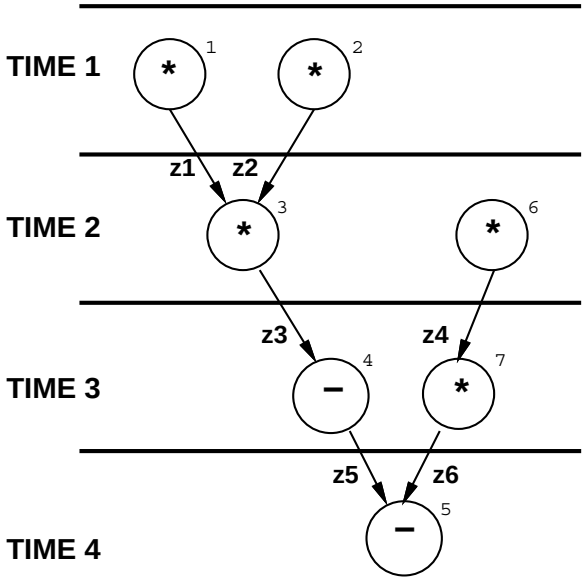
Register sharing data-flow graphs

© GDM

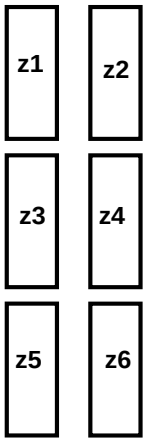
- Given:
  - Variable lifetime conflict graph.
- Find:
  - Minimum number of registers storing all the variables.
- Key point:
  - Interval graph:
    - Left-edge algorithm. (Polynomial-time).

Example

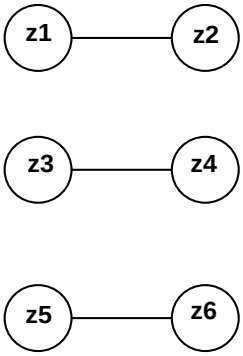
© GDM



(a)



(b)



(c)

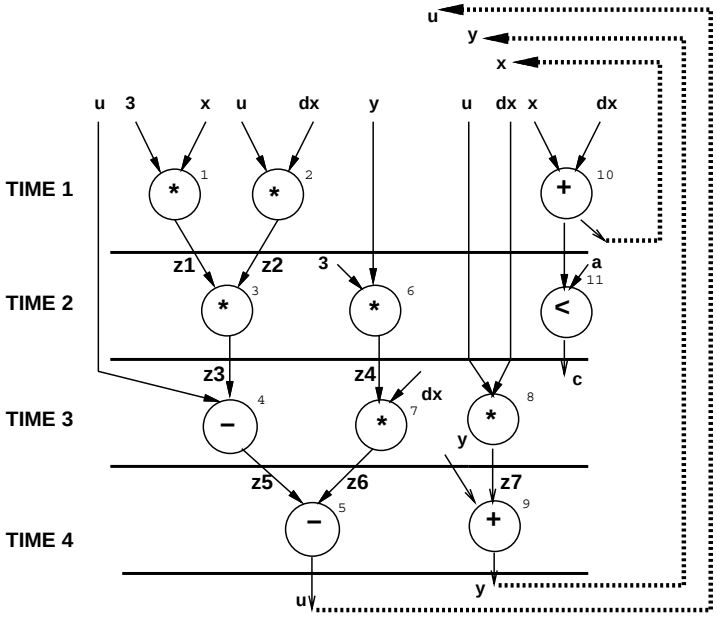
Register sharing
general case

© GDM

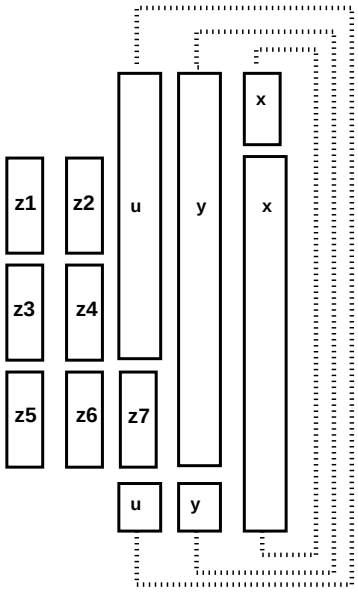
- Iterative constructs:
  - Preserve values across iterations.
  - *Circular-arc* conflict graph:
    - Coloring is intractable.
- Hierarchical graphs:
  - General conflict graphs:
    - Coloring is intractable.
- Heuristic algorithms.

Example

© GDM



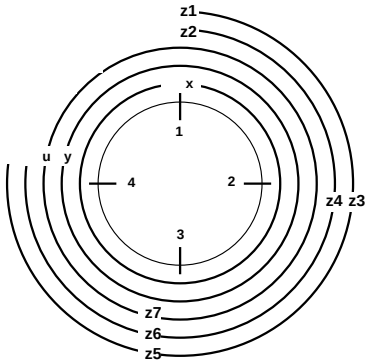
(a)

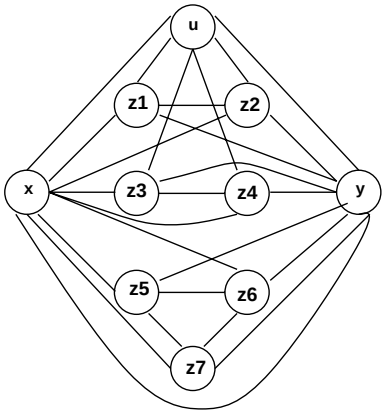


(b)

Example
Variable-lifetimes and circular-arc conflict graph

© GDM





Module selection problem

© GDM

- 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.

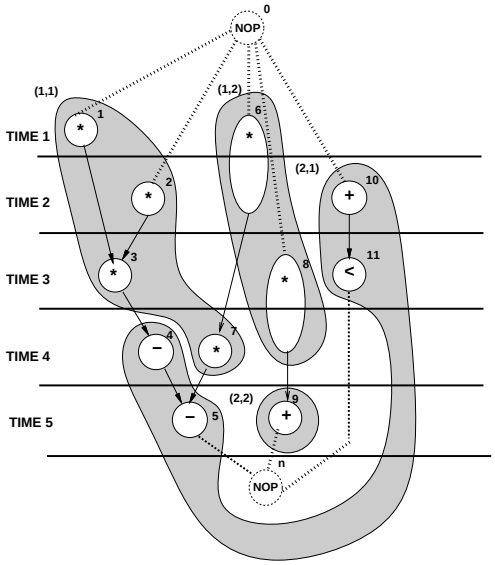
Module selection solution

© GDM

- ILP formulation:
  - Decision variables:
    - Select resource sub-type.
    - Determine *(area, delay)*.
- Heuristic algorithms:
  - Determine *minimum latency* with fastest resource subtypes.
  - Recover area by using slower resources on non-critical paths.

Example

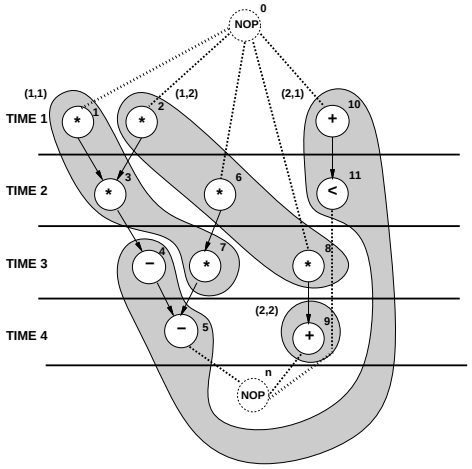
© GDM



- Multipliers with:
  - (Area, delay) = (5,1) and (2,2)
- Latency bound of 5.

Example (2)

© GDM



- 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

© GDM

- 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.