COE 561
Digital System Design & Synthesis
Resource Sharing and Binding

Dr. Aiman H. El-Maleh

Computer Engineering Department

King Fahd University of Petroleum & Minerals

[Adapted from slides of Prof. G. De Micheli: Synthesis & Optimization of Digital Circuits]

Outline

- **Sharing and Binding**
- **Resource-dominated circuits.**
  - Flat and hierarchical graphs.
- **Register sharing**
- **Multi-port memory binding**
- **Bus sharing and binding**
- **Non resource-dominated circuits.**
- **Module selection.**

Allocation and Binding

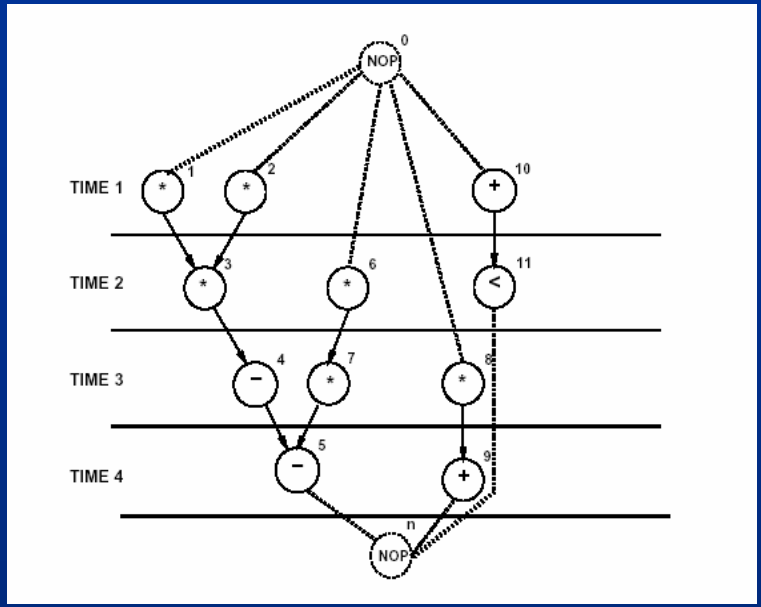
- **Allocation**
  - Number of resources available.
- **Binding**
  - Mapping between operations and resources.
- **Sharing**
  - Assignment of a resource to more than one operation.
- **Optimum binding/sharing**
  - Minimize the resource usage.

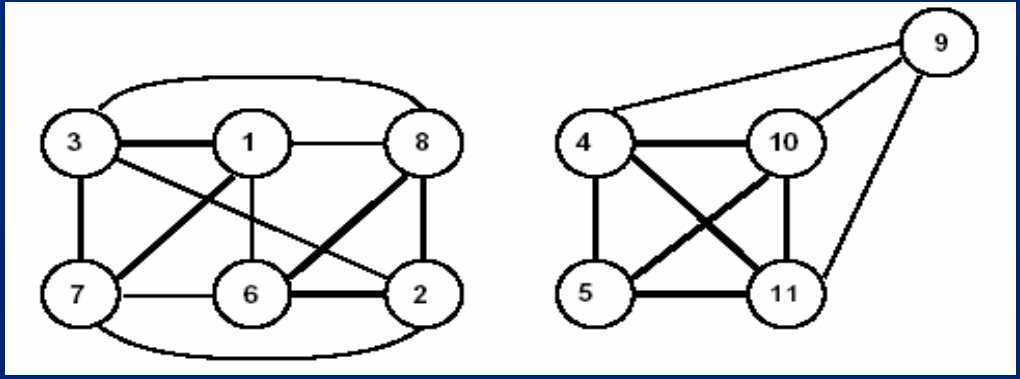
Optimum Sharing Problem

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

Compatibility and Conflicts

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





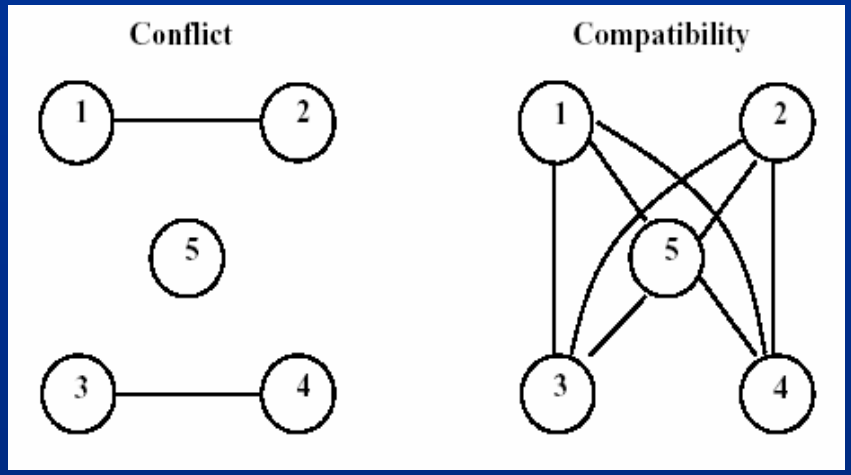
Multiplier ***ALU***

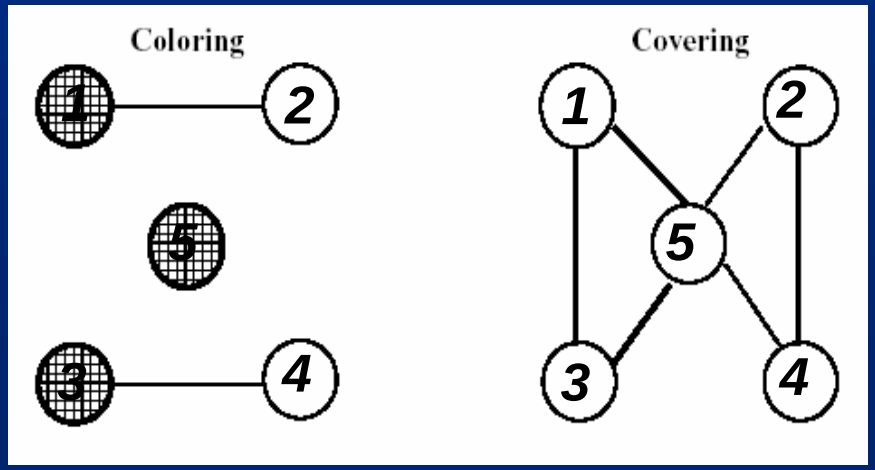
Algorithmic Solution to the Optimum Binding Problem

- **Compatibility graph.**
  - Partition the graph into a minimum number of cliques.
  - Find clique cover number.
- **Conflict graph.**
  - Color the vertices by a minimum number of colors.
  - Find chromatic number.
- **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	





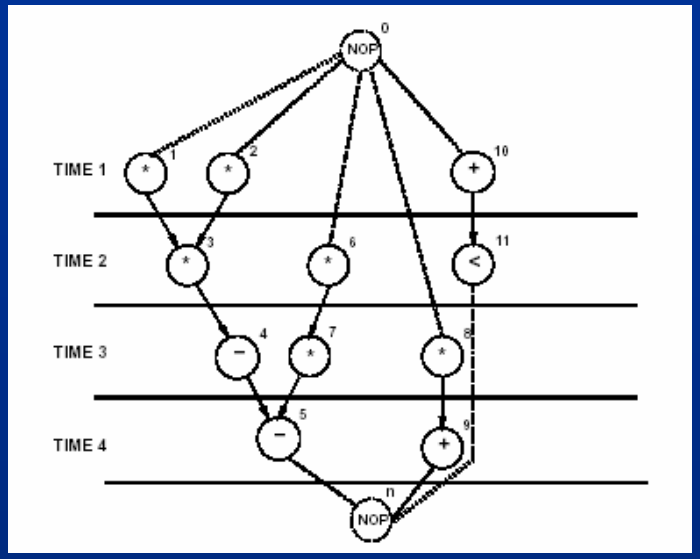
ALU1: 1, 3, 5
ALU2: 2, 4

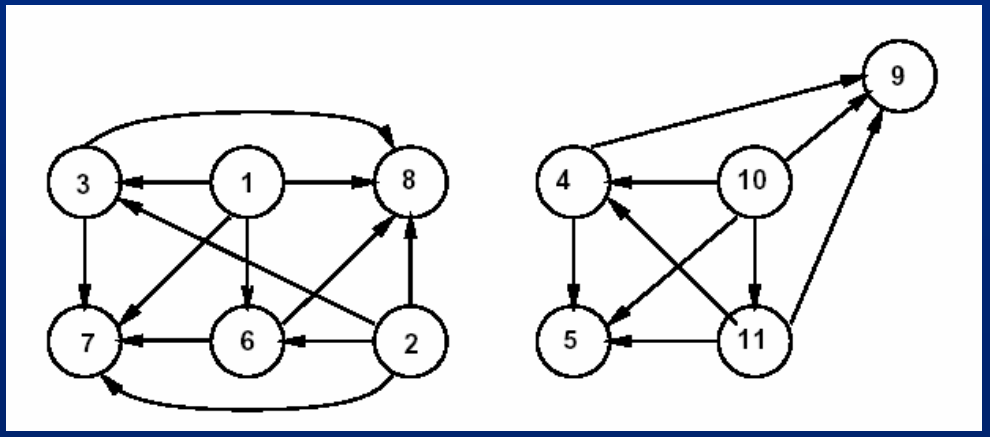
Perfect Graphs

- **Comparability graph**
  - Graph *G(V, E)* has an orientation (i.e. directed edges) *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 (i.e. an edge joining two non-consecutive vertices in the cycle).
- **Efficient algorithms exist for coloring and clique partitioning of interval, chordal, and comparability graphs.**

Non-Hierarchical Sequencing Graphs

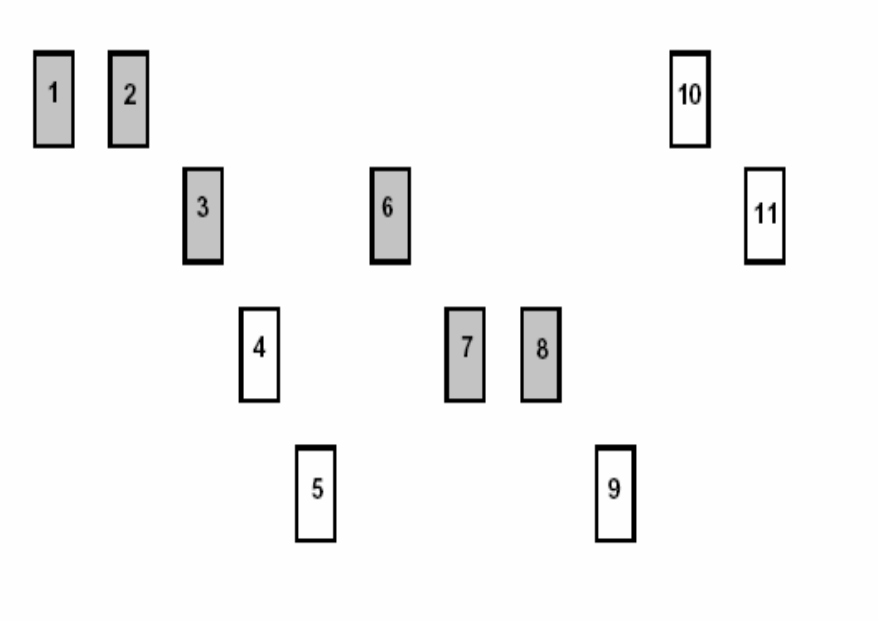
- **The compatibility/conflict graphs have special properties**
  - Compatibility: Comparability graph.
  - Conflict: Interval graph.
- **Polynomial time solutions**
  - Golumbic's algorithm.
  - Left-edge algorithm.

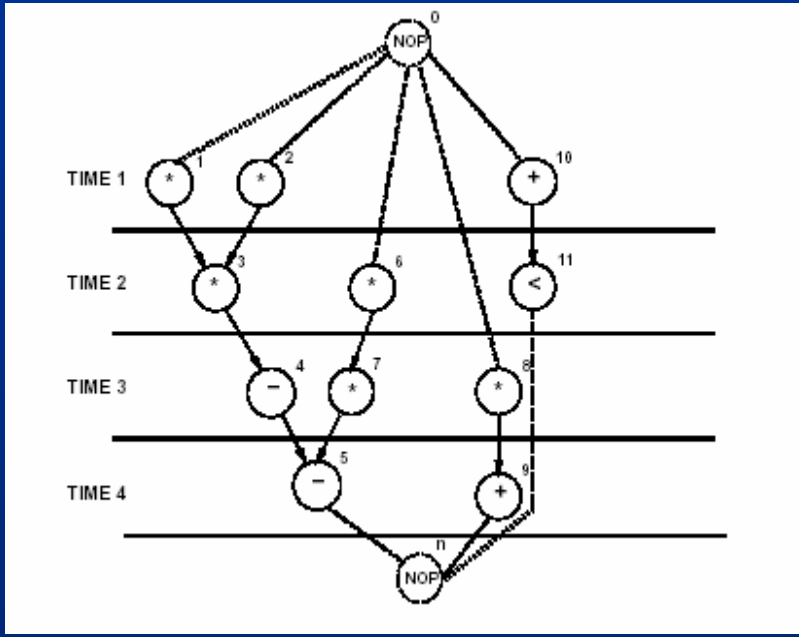




Comparability Graph

Example





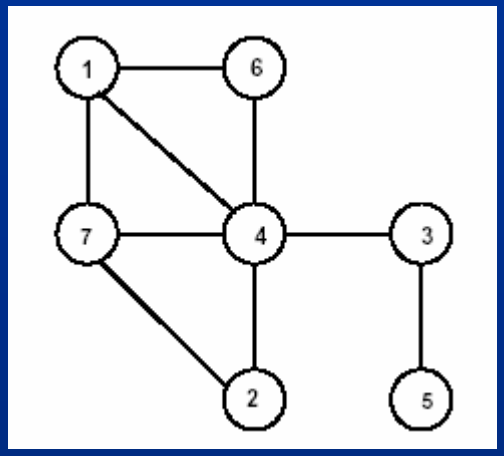
Intervals Corresponding to Conflict Graph

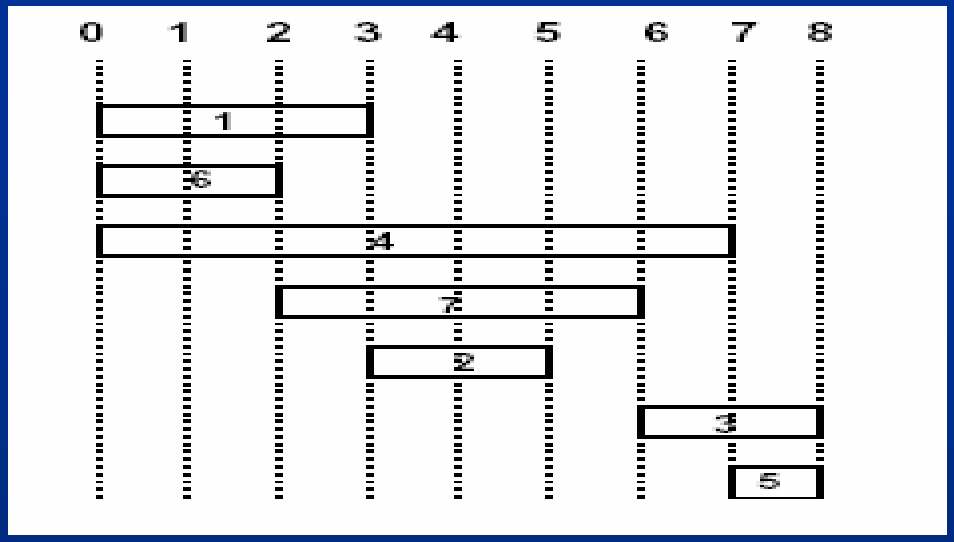
Left-Edge Algorithm

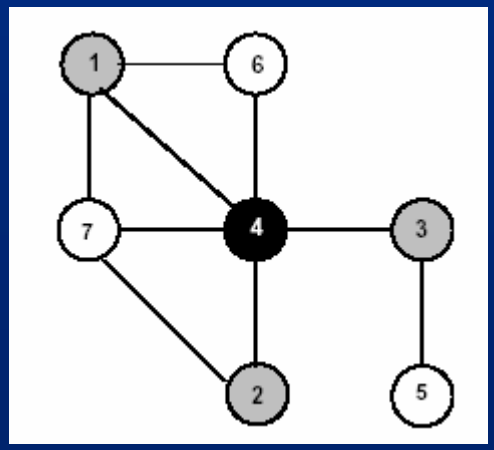
- **Input**
  - Set of intervals with *left* and *right* edge.
- **Rationale**
  - Sort intervals by *left* edge.
  - Assign non-overlapping intervals to first color using the sorted list.
  - When possible intervals are exhausted increase color counter and repeat.

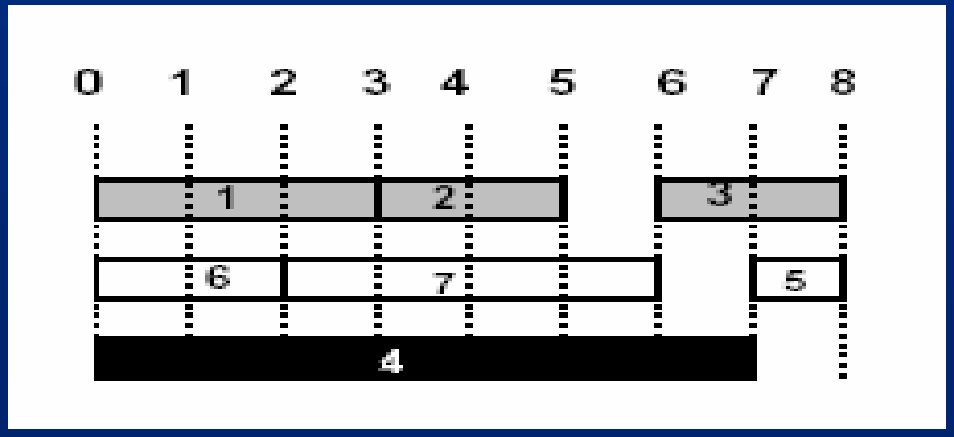
```
LEFT_EDGE(I) {
    Sort elements of I in a list L in ascending order of l_i;
    c = 0;
    while (some interval has not been colored ) do {
        S = ∅;
        r = 0;
        while ( ∃s ∈ L such that l_s > r) do{
            s = First element in the list L with l_s > r;
            S = S ∪ {s};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

Example









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*.**
- Each operation v_i should be assigned to one resource

$$\sum_{r=1}^{a} b_{ir} = 1 \quad \forall i$$

- At most, one operation can be executing, among those assigned to resource r, at any time step

$$\sum_{i=1}^{n_{ops}} b_{ir} \sum_{m=l-d_i+1}^{l} x_{im} \leq 1 \quad \forall l \quad \forall r$$

Example…

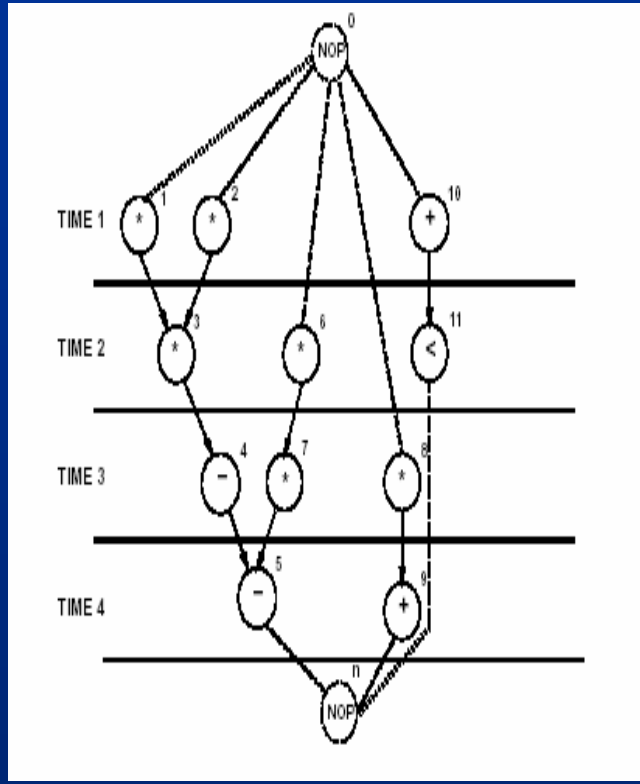
- Operation types: Multiplier, ALU
- Unit execution delay
- A feasible binding satisfies constraints

$$\sum_{r=1}^{a_1} b_{ir} = 1, \quad \forall i : Type(v_i) = 1$$

$$\sum_{i:Type(v_i)=1} b_{ir}\ x_{il} \le 1, \quad l = 1,2,\ldots,\lambda+1, \quad r = 1,2,\ldots,a_1$$

$$\sum_{r=1}^{a2} b_{ir} = 1, \quad \forall i : Type(v_i) = 2$$

$$\sum_{i:Type(v_i)=2} b_{ir}\ x_{il} \le 1, \quad l = 1,2,\ldots,\lambda+1, \quad r = 1,2,\ldots,a_2$$



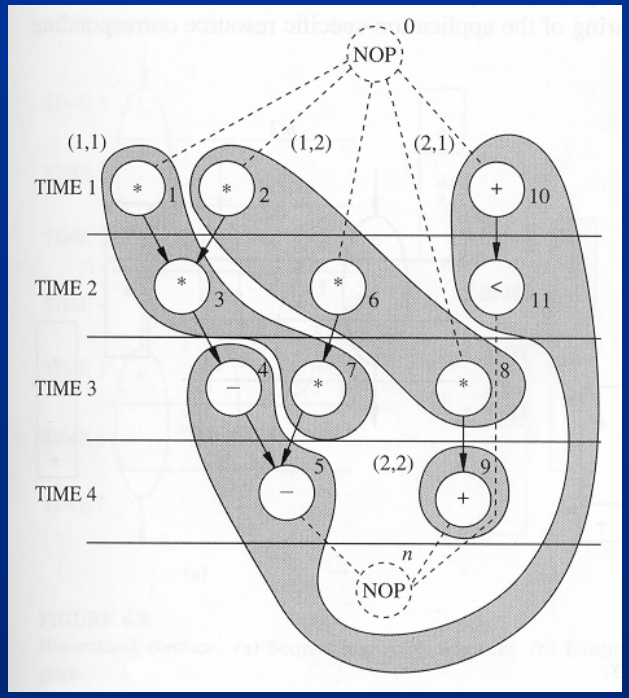
… Example

- Constants in X are 0 except
  $x_{1,1}$, $x_{2,1}$, $x_{3,2}$, $x_{4,3}$, $x_{5,4}$, $x_{6,2}$, $x_{7,3}$, $x_{8,3}$, $x_{9,4}$, $x_{10,1}$, $x_{11,2}$.
- An implementation with a_1=2 multipliers:

$$b_{i1} + b_{i2} = 1, \quad \forall i \in \{1, 2, 3, 6, 7, 8\}$$

$$\sum_{i \in \{1,2,3,6,7,8\}} b_{i1} x_{il} \le 1, \quad l = 1, 2, \ldots, 5$$

$$\sum_{i \in \{1,2,3,6,7,8\}} b_{i2} x_{il} \le 1, \quad l = 1, 2, \ldots, 5$$



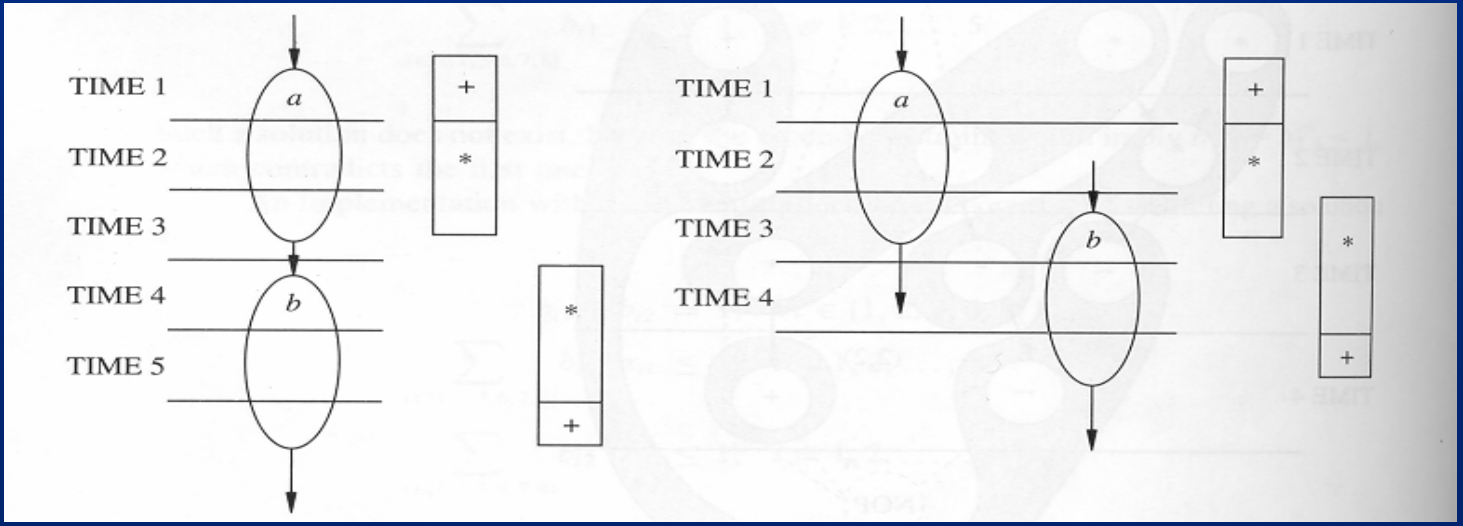
- Solutions
  - $b_{1,1}$=1, $b_{2,2}$=1, $b_{3,1}$=1, $b_{6,2}$=1, $b_{7,1}$=1, $b_{8,2}$=1.

Hierarchical Sequencing Graphs …

- **Hierarchical conflict/compatibility graphs.**
  - Easy to compute.
  - Prevent sharing across hierarchy.
- **Flatten hierarchy.**
  - Bigger graphs.
  - Destroy nice properties.
    - Graphs may no longer have special properties i.e., comparability graph, interval graph.
    - Clique partitioning and vertex coloring intractable problems.

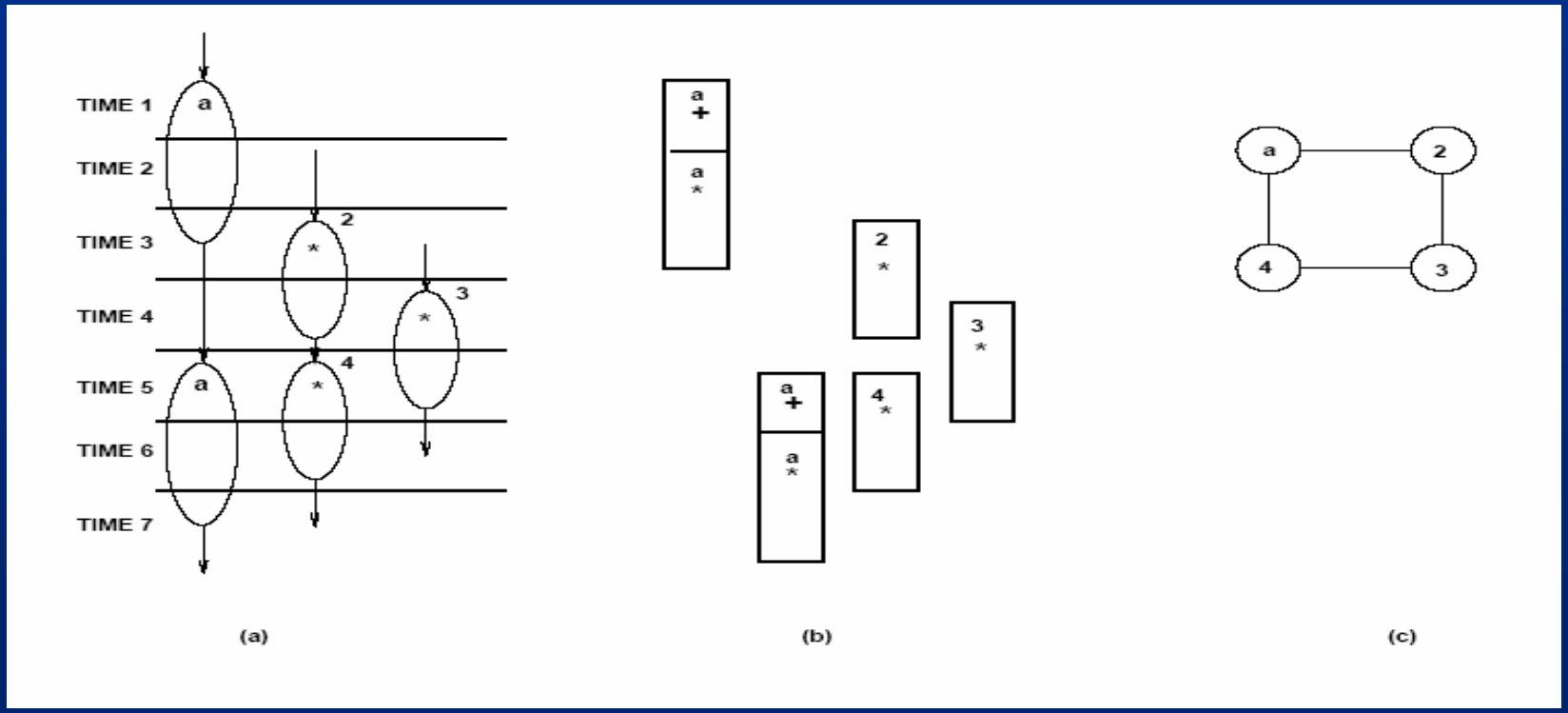
… Hierarchical Sequencing Graphs

- **Model calls**
  - When two link vertices corresponding to different called models are not concurrent
    - Any operation pair of same resource type in the different called models is compatible.
  - Concurrency of called models does not necessarily imply conflicts of operation pairs in the models.



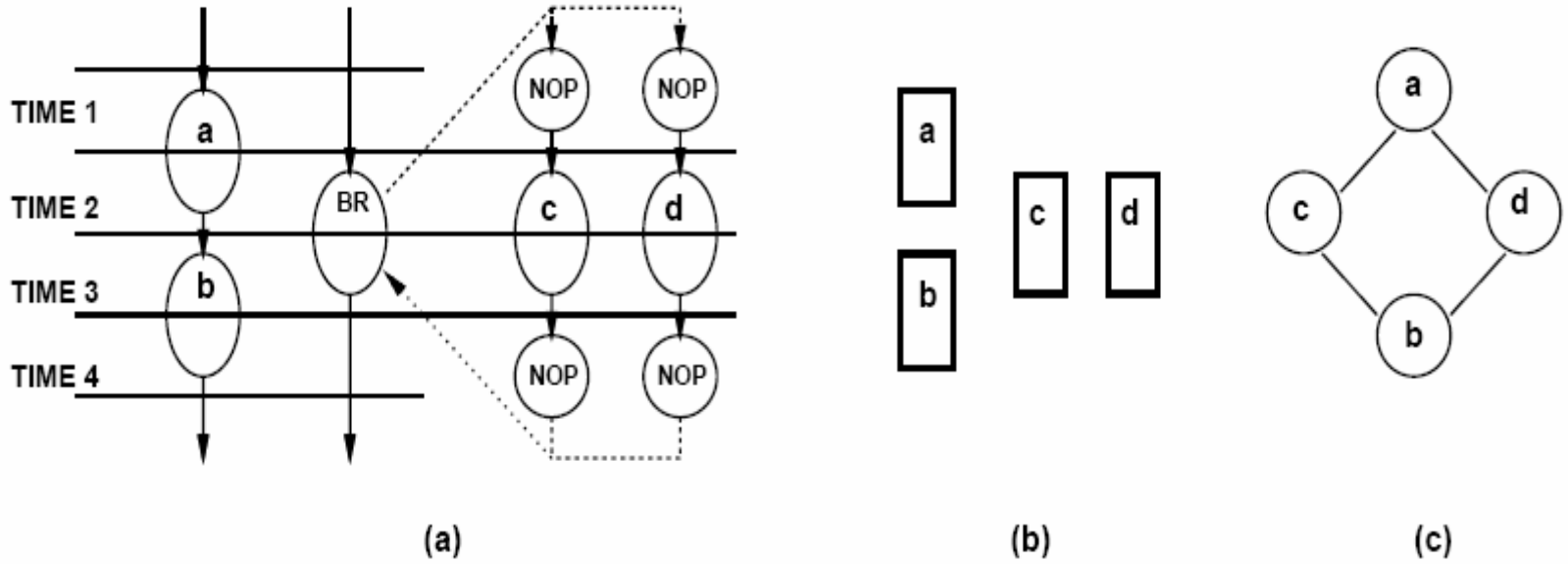
Example: Model Calls

- Model a consists of two operations: addition, followed by multiplication
- Addition delay is 1, multiplication delay is 2



Example: Branching Constructs

- All operations take 2 time units
- Start times: $t_a=1$, $t_b=3$, $t_c=t_d=2$

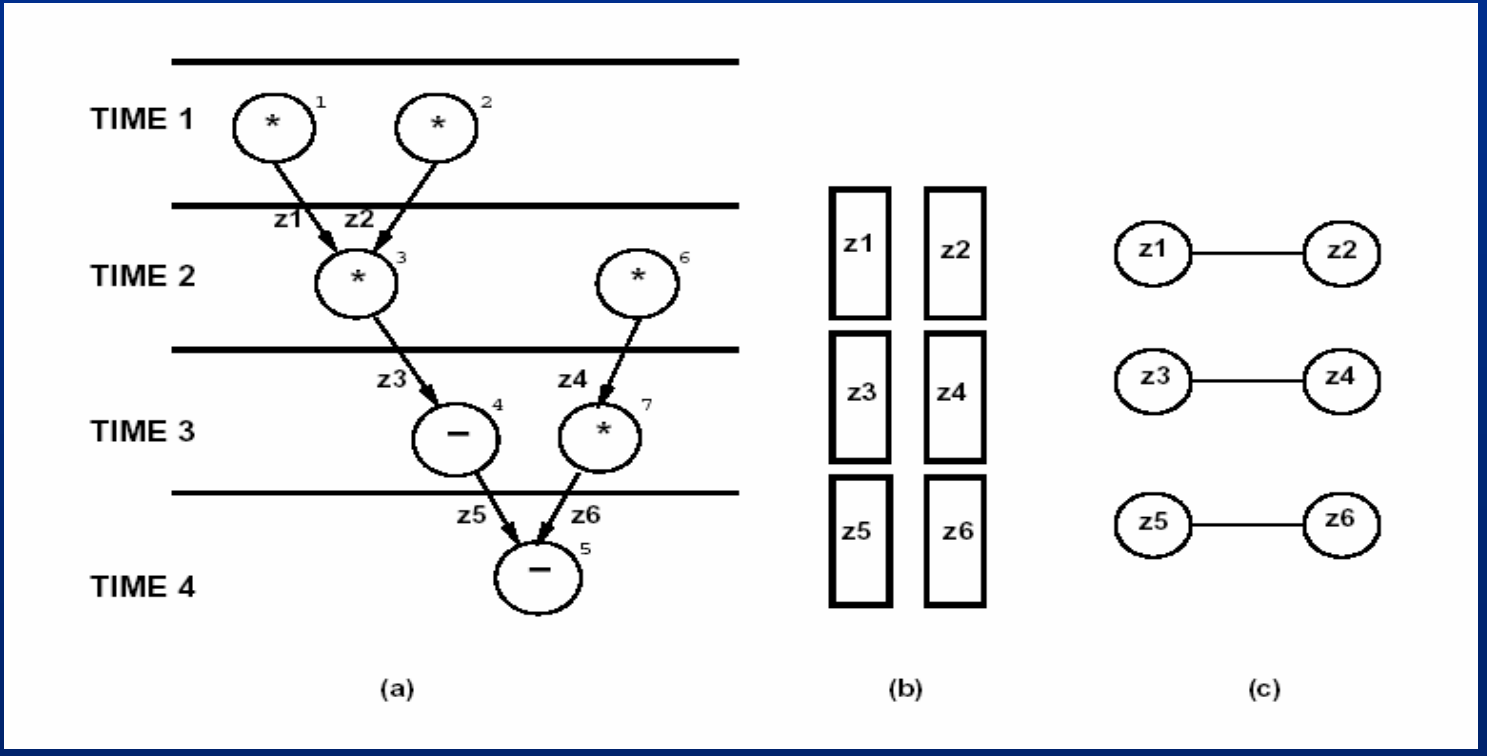


Register Binding Problem

- **Given a schedule**
  - Lifetime intervals for variables.
  - Lifetime overlaps.
- **Conflict graph (interval graph).**
  - Vertices $\leftrightarrow$ variables.
  - Edges $\leftrightarrow$ overlaps.
  - Interval graph.
  - Left-edge algorithm. (Polynomial-time).
- **Find minimum number of registers storing all the variables.**
- **Compatibility graph (comparability graph).**

Example

- Six intermediate variables that need to be stored in registers {z1, z2, z3, z4, z5, z6}
- Six variables can be stored in two registers



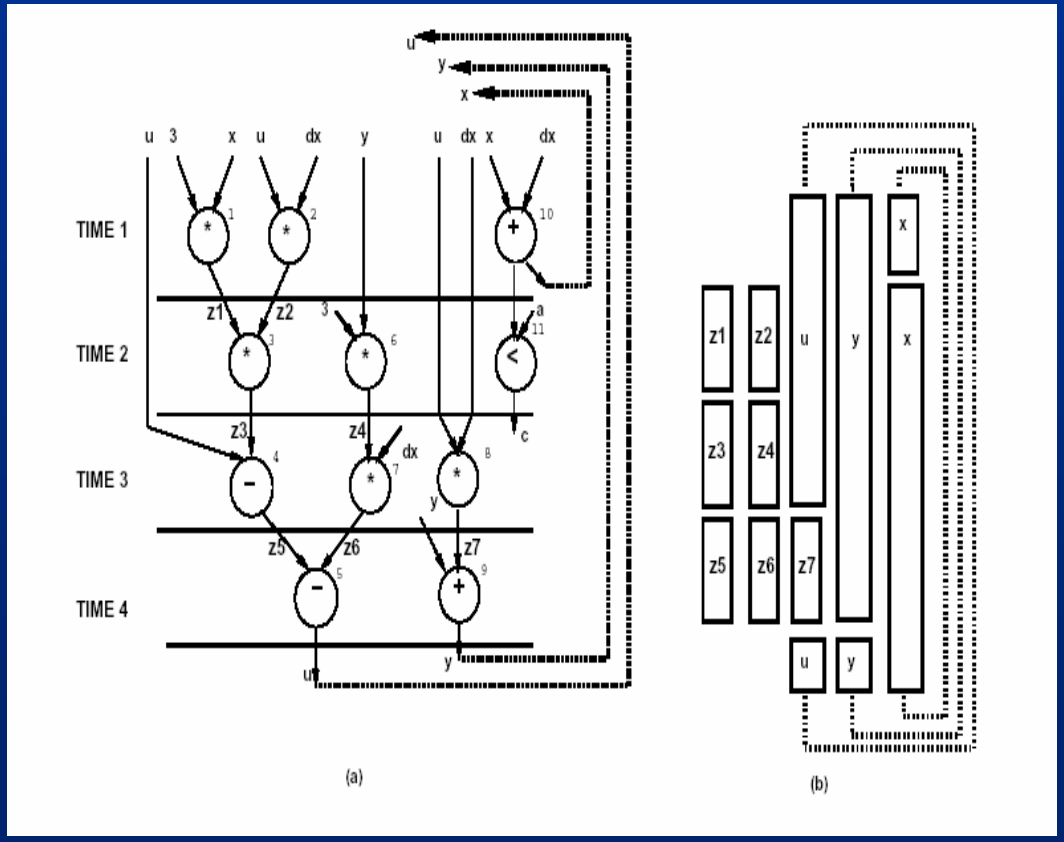
Register Sharing: General Case

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

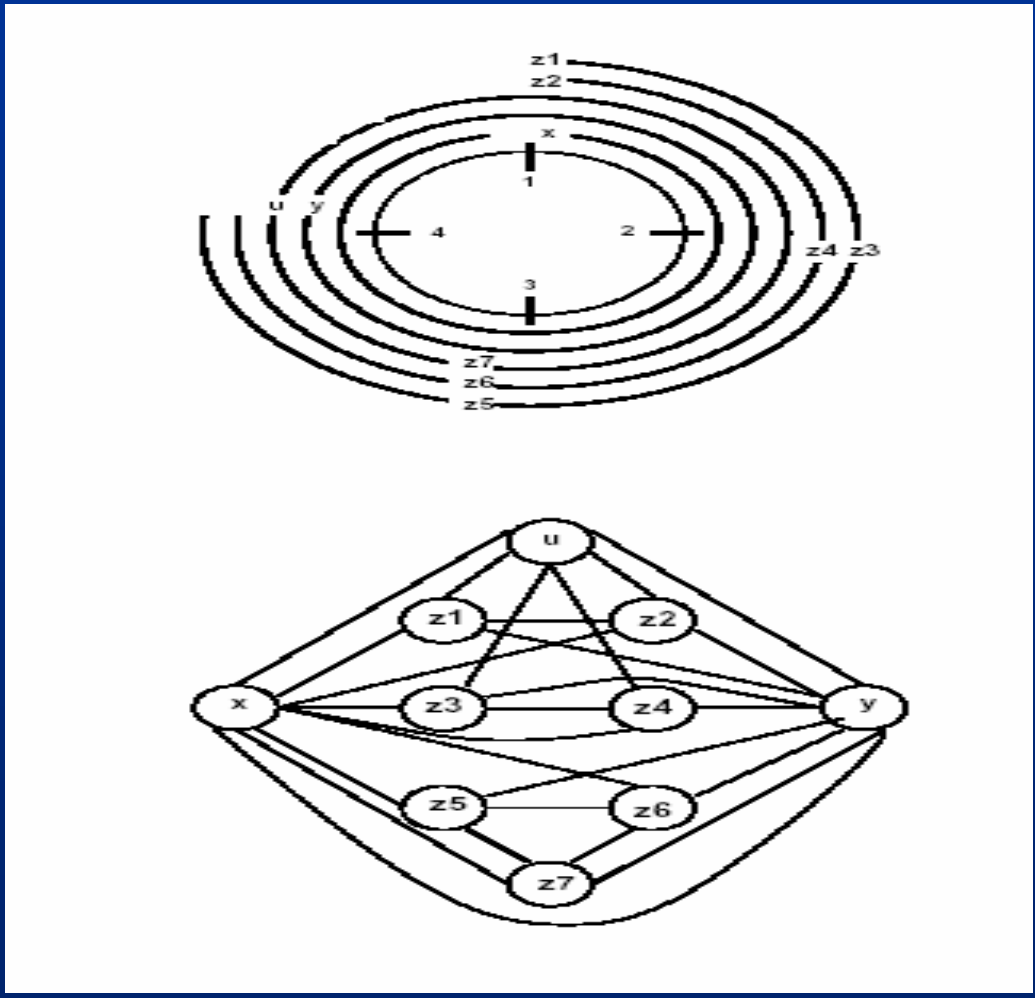
Example

- 7 intermediate variables, 3 loop variables, 3 loop invariants
- 5 registers suffice to store 10 intermediate loop variables

```
diffeq {
    read (x, y, u, dx, a);
    repeat {
        xl = x + dx;
        ul = u – (3 · x · u · dx) – (3 · y · dx);
        yl = y + u · dx;
        c = x < a;
        x = xl; u = ul; y = yl;
        }
    until ( c ) ;
write (y);
}
```



Example: Variable-Lifetimes and Circular-Arc Conflict Graph



Multiport-Memory Binding …

- **Multi-port memory arrays used to store variables.**
- **Find minimum number of ports to access the required number of variables.**
- **Assuming variables access memory always through the same port**
  - Problem reduces to binding variables to ports.
  - Port compatibility/conflict.
  - Similar to resource binding.
- **Assuming variables can use any port**
  - Decision variable x_{il} is TRUE when variable *i* is accessed at step *l*.
  - Minimum number of ports

$$\max_{1 \le l \le \lambda+1} \sum_{i=1}^{n_{var}} x_{il}.$$

… Multiport-Memory Binding

- **Find maximum number of variables to be stored through a fixed number of ports a.**
  - Boolean variables $\{b_i, i = 1, 2, \ldots, n_{var}\}$:
  - Variable *i* is stored in array.
- **The maximum number of variables that can be stored in a multiport-memory with *a* ports is obtained by:**

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

$$- \sum_{i=1}^{n_{var}} b_i \, x_{il} \leq a \qquad l = 1, 2, \ldots, \lambda + 1$$

Example

- **One port a = 1**
  - {b2, b4, b8} non-zero.
  - 3 variables stored: {v2, v4, v8}.
- **Two ports a = 2**
  - 6 variables stored: {v2, v4, v5, v10, v12, v14}
- **Three ports a = 3**
  - 9 variables stored: {v1, v2, v4, v6, v8, v10, v12, v13}

$$
\begin{aligned}
&Time-step\ 1\ :\ r_3 = r_1 + r_2\ ;\ r_{12} = r_1\\
&Time-step\ 2\ :\ r_5 = r_3 + r_4\ ;\ r_7 = r_3 * r_6\ ;\ r_{13} = r_3\\
&Time-step\ 3\ :\ r_8 = r_3 + r_5\ ;\ r_9 = r_1 + r_7\ ;\ r_{11} = r_{10}/r_5\\
&Time-step\ 4\ :\ r_{14} = r_{11} \wedge r_8\ ;\ r_{15} = r_{12} \vee r_9\\
&Time-step\ 5\ :\ r_1 = r_{14}\ ;\ r_2 = r_{15}
\end{aligned}
$$

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

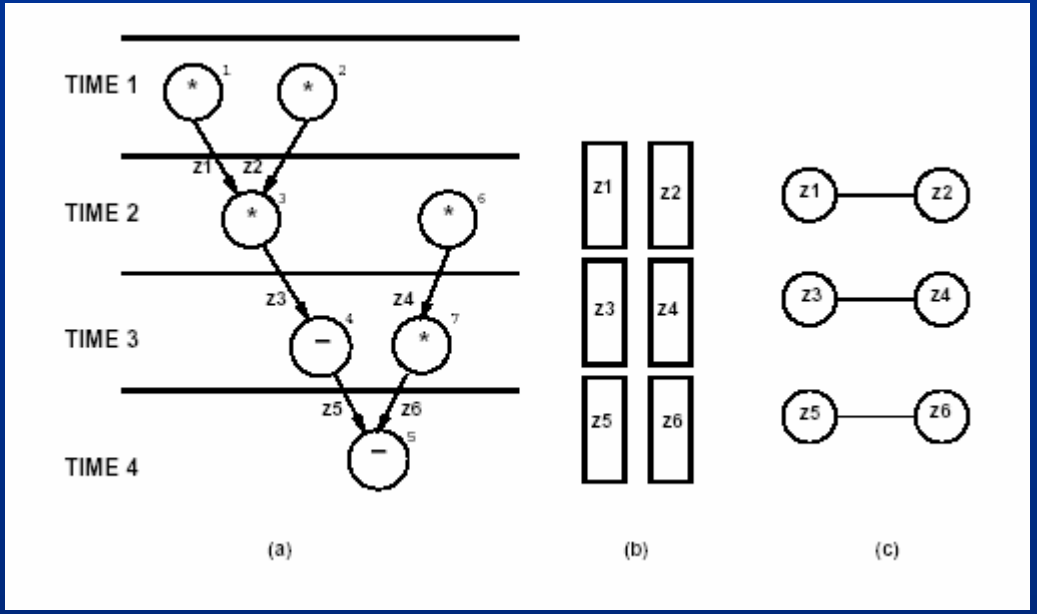
$$
\begin{aligned}
b_1 + b_2 + b_3 + b_{12} &\leq a\\
b_3 + b_4 + b_5 + b_6 + b_7 + b_{13} &\leq a\\
b_1 + b_3 + b_5 + b_7 + b_8 + b_9 + b_{10} + b_{11} &\leq a\\
b_8 + b_9 + b_{11} + b_{12} + b_{14} + b_{15} &\leq a\\
b_1 + b_2 + b_{14} + b_{15} &\leq a
\end{aligned}
$$

Bus Sharing and Binding

- Busses act as transfer resources that feed data to functional resources.
- Find the minimum number of busses to accommodate all data transfers.
- 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.



Sharing and Binding for General Circuits

- **Area and delay influenced by**
  - Steering logic, wiring, registers and control circuit.
  - E.g. multiplexers area and propagation delays depend on number of inputs.
  - Wire lengths can be derived from statistical models.
- **Binding affects the cycle-time**
  - It may invalidate a schedule.
- **Control unit is affected marginally by resource binding.**

Unconstrained Minimum Area Binding

- **Area cost function depends on several factors**
  - resource count, steering logic and wiring.
- **In limiting cases, resource sharing may affect adversely circuit area.**
- **Example**
  - Circuit with n 1-bit add operations
  - Area of 1-bit adder is $area_{add}$
  - Area of a MUX is a function of number of inputs
    $area_{mux} = area_{mux}^{\nabla} \cdot (i-1)$, where $area_{mux}^{\nabla}$ is a constant
  - Total area of a binding with a resources is $a\ (area_{add} + area_{mux}) \approx a\ (area_{add} - area_{mux}^{\nabla}) + n\ .\ area_{mux}^{\nabla}$
  - Area is increasing or decreasing function of a according to relation $area_{add} > area_{mux}^{\nabla}$.

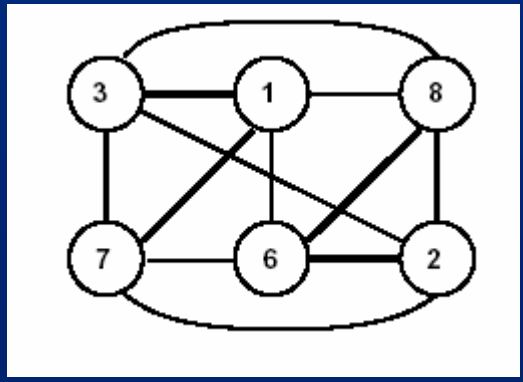
Unconstrained Minimum Area Binding

- **Edge-weighted compatibility graph**
  - Edge weights represent level of desirability of sharing
  - Clique covering

```
TSENG( G+(V, E, W) ) {
    while (E ≠ Ø) do {
        lw = max_{w∈W} w;                                   /* largest edge weight */
        E' = {{vi, vj} ∈ E such that wij = lw};
        G'+(V', E', W') = subgraph of G+(V, E, W) induced by E';
        while (E' ≠ Ø) do {
            Select {vi, vj} ∈ E' such that vi and vj have the most neighbors in common;
            C = {vi, vj};
            Delete edges {vl, vi} if {vl, vj} ∉ E' ∀vl ∈ V';
            Delete vertex vj from V';
            while (one vertex adjacent to vi in G'+(V', E', W')) do {
                Select vk such that {vi, vk} ∈ E' and vi and vk have the
                  most neighbors in common;
                C = C ∪ {vk};
                Delete edges {vl, vi} if {vl, vk} ∉ E' ∀vl ∈ V';
                Delete vertex vk from V';
            }
            Save clique C in the clique list;
        }
        Delete the vertices in the clique list from V;
    }
}
```

Unconstrained Minimum Area Binding

- **Tseng's algorithm considers repeatedly subgraphs induced by vertices with same weight edges.**
- **Graphs with decreasing values of weights considered.**
- **Unweighted clique partitioning of subgraphs.**
- **Example**
  - Assume following edges have weight of 2
    - {v1, v3}, {v1, v6}, {v1, v7}, {v3, v7}, {v6, v7}
  - Other edges have weight 1
  - Clique {v1, v3, v7} is first identified
  - Clique {v2, v6, v8} is then identified



Module Selection Problem …

- **Library of resources**
  - More than one resource per type.
- **Example**
  - Adder
    - Ripple-carry adder.
    - Carry look-ahead adder.
  - Multiplier
    - Fully parallel
    - Serial-Parallel
    - Fully serial
- **Resource modeling**
  - Resource subtypes with
    - (area, delay) parameters.

… Module Selection Problem

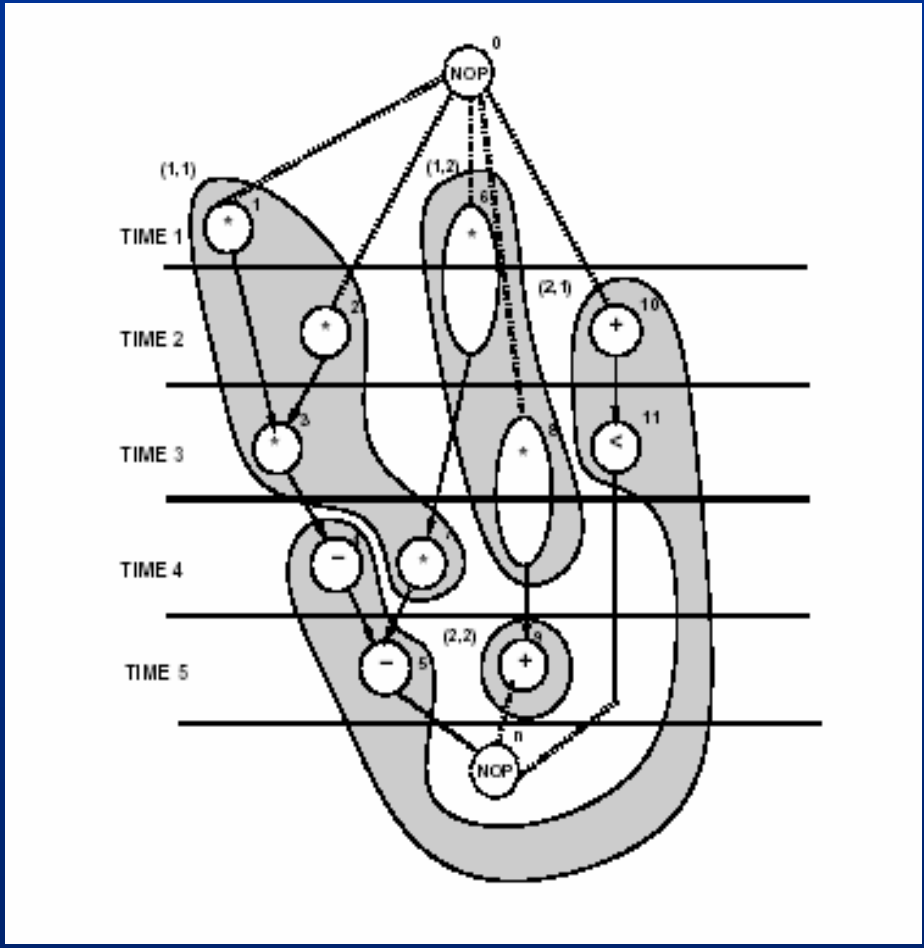
- **ILP formulation**
  - Decision variables b_{jr}
    - Select resource sub-type.
    - Determine (area, delay).

$$d_j = \sum_{r=1}^{\bar{a}} b_{jr}.delay_r; \quad j = 1,2,...,n_{ops};\ \bar{a} \text{ is a resource uppe bound}$$

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

Example

- Multipliers with
  - **(Area, delay) = (5,1) and (2,2)**
- ALU with
  - **(Area, delay) = (1,1)**
- Latency bound of 5.
- Area cost is 7+2=9



Example

- **Latency bound of 4.**
  - Fast multipliers for {v1, v2, v3}.
  - Slower multipliers can be used elsewhere.
    - Less sharing.
    - Assume v8 uses a slow multiplier: Area=12+2=14
- **Minimum-area design uses fast multipliers only.**
  - Area=10+2=12