# Architectural Synthesis through Sequencing Graphs

Dr. Shubhajit Roy Chowdhury,
Centre for VLSI and Embedded Systems Technology,
IIIT Hyderabad, India
Email: src.vlsi@iiit.ac.in

# Architectural Synthesis Problem

- Specification
  - A sequencing graph
  - A set of functional resources
  - characterized by area and execution delay
- Constraints
- Tasks
  - Place operations in time and space
  - Determine detailed interconnection and control
- This is what we need to do in behavioral synthesis! :)
- Constraints include: area, cycle time, latency, and throughput.
- Area: number of modules/resources available or size of your silicon die.
- Cycle time: how fast your clock runs
- Latency: number of cycles for input data to result in a solution or result.
- Throughput: Amount of data that can be processed in a given amount of time (usually involves pipelining)

## Resource Binding

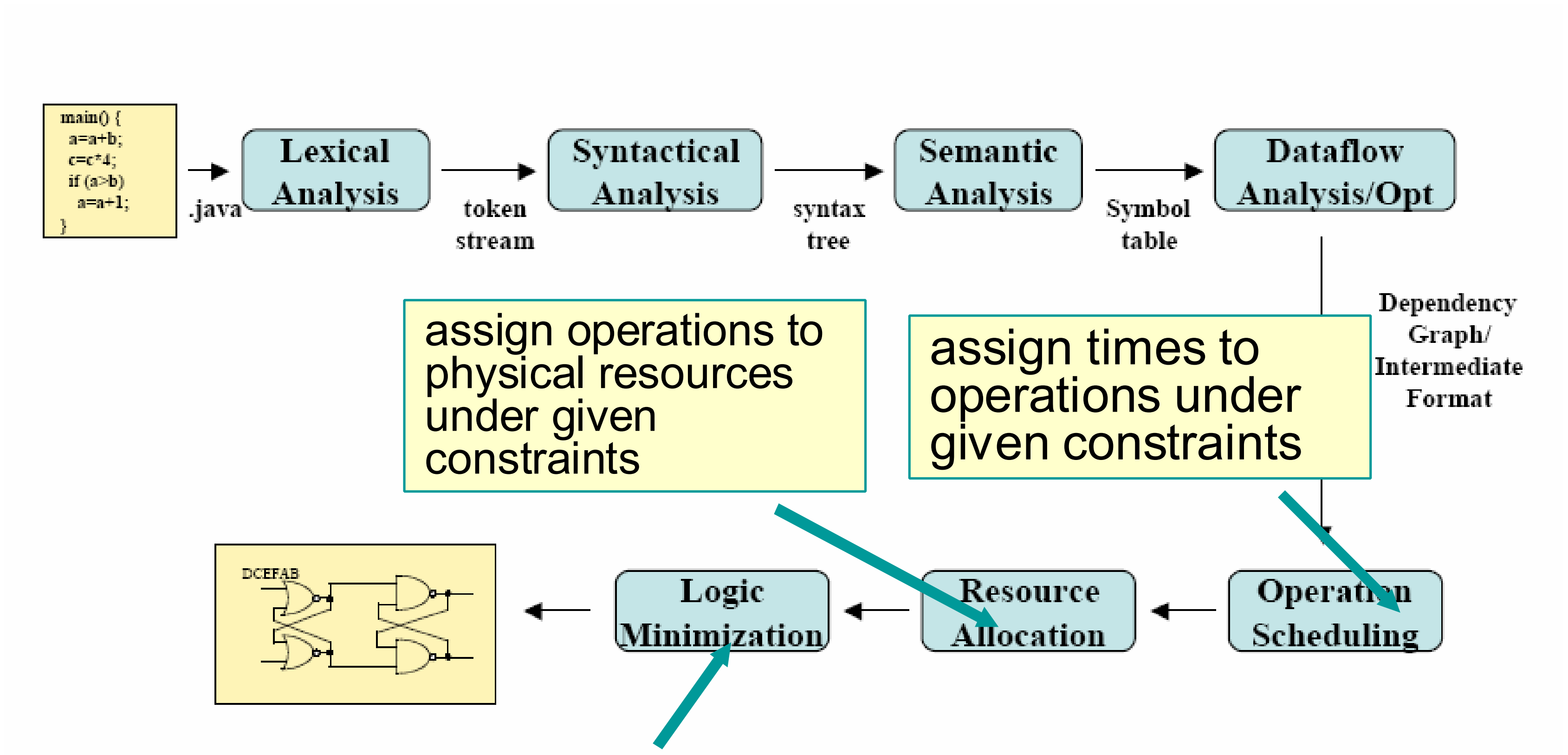
- May have a pool of resources larger than required for problem
- Map a constrained set of resources to given operations
- Dedicated resources: each operation is bound to a single resource.

1. Resource pool: may include various kinds of multipliers (booth, array, etc) adders (tree, carry-lookahead, etc.) multipurpose units (ALUs, multiplier/divider, etc.)

2. Mapping a given set of resources to a set of known operations is one type of problem to solve.

3. Dedicated resource allocation is a one-to-one mapping.

# Overview of Hardware Synthesis



assign operations to physical resources under given constraints

assign times to operations under given constraints

reduce the amount of hardware, optimize the design in general. May be done with the consideration of additional constraints.

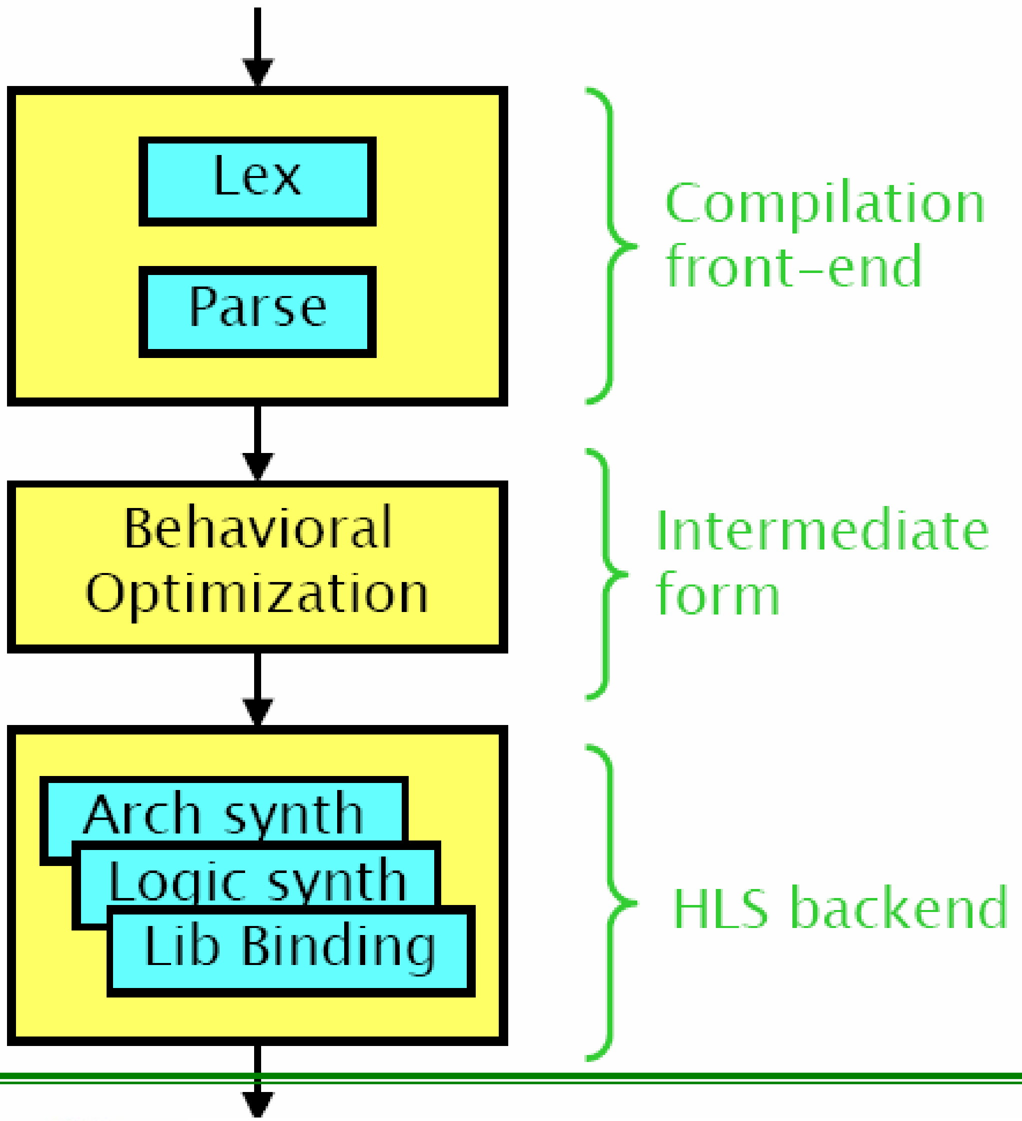
## Architectural versus Logic Synthesis

- Transform behavioral into structural view.

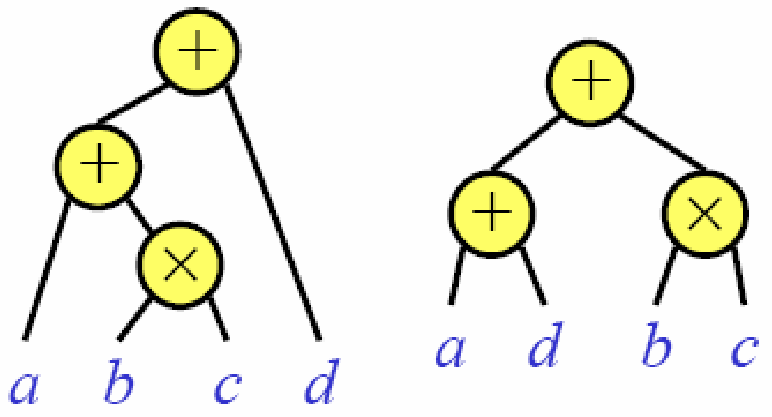
- Architectural-level synthesis:
  - Architectural abstraction level.
  - Determine ***macroscopic*** structure.
  - Example of synthesis: major building blocks.

- Logic-level synthesis:
  - Logic abstraction level.
  - Determine ***microscopic*** structure.
  - Example of synthesis: logic gate interconnection.

# High-Level Synthesis Compilation Flow



$$x = a + b \times c + d$$



## Compilation

- Front-end:

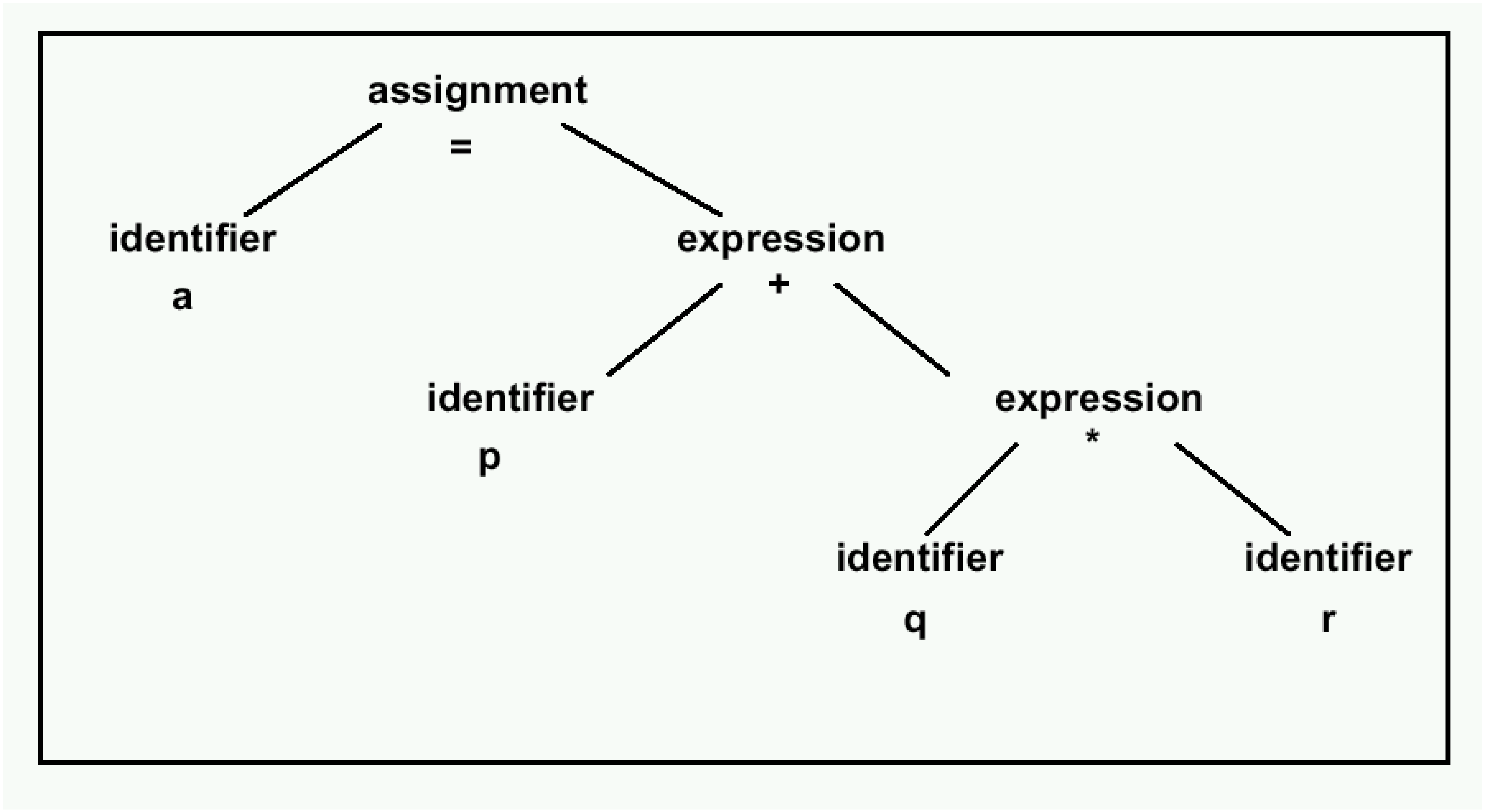
  1. Lexical and syntax analysis.
  2. Parse-tree generation.
  3. Macro-expansion.

- Semantic analysis:

  1. Data-flow and control-flow analysis.
  2. Type checking.
  3. Resolve arithmetic and relational operators.

## Parse tree example

a = p + q * r



## Behavioral-level optimization

- **Semantic-preserving** transformations aiming at simplifying the model.

- Applied to parse-trees or during their generation.

- ***Taxonomy:***
  1. ***Data-flow*** based transformations.
  2. ***Control-flow*** based transformations.

## Data-Flow Based Transformations

1. Tree-height reduction.
2. Constant and variable propagation.
3. Common sub-expression elimination.
4. Dead-code elimination.
5. Code motion.

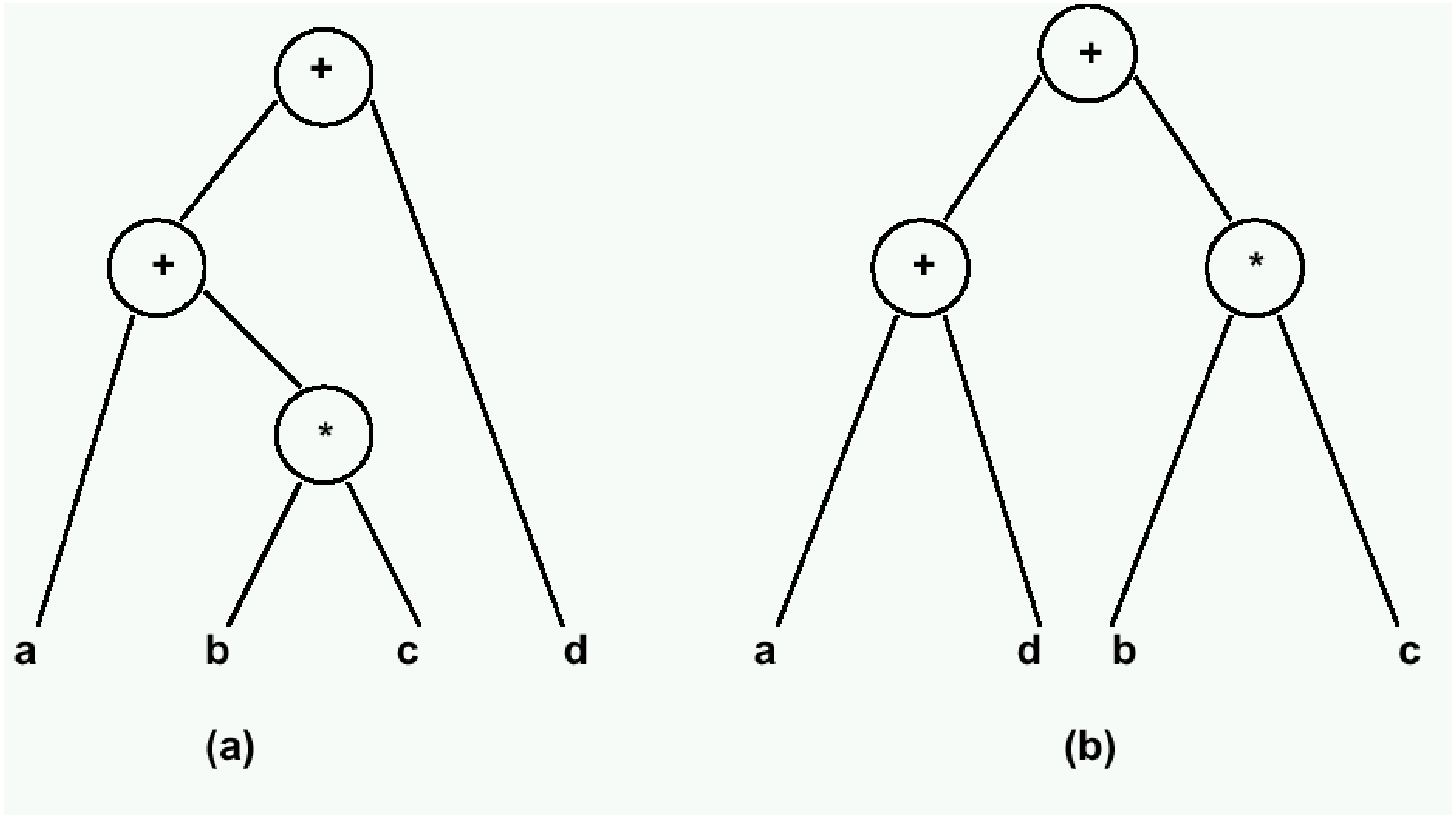
## Tree-height reduction

- Applied to arithmetic expressions.

- **Goal**:
  - Split into two-operand expressions to exploit hardware parallelism at best.

- **Techniques:**
  - Balance the expression tree.
  - Exploit ***commutativity, associativity and distributivity.***

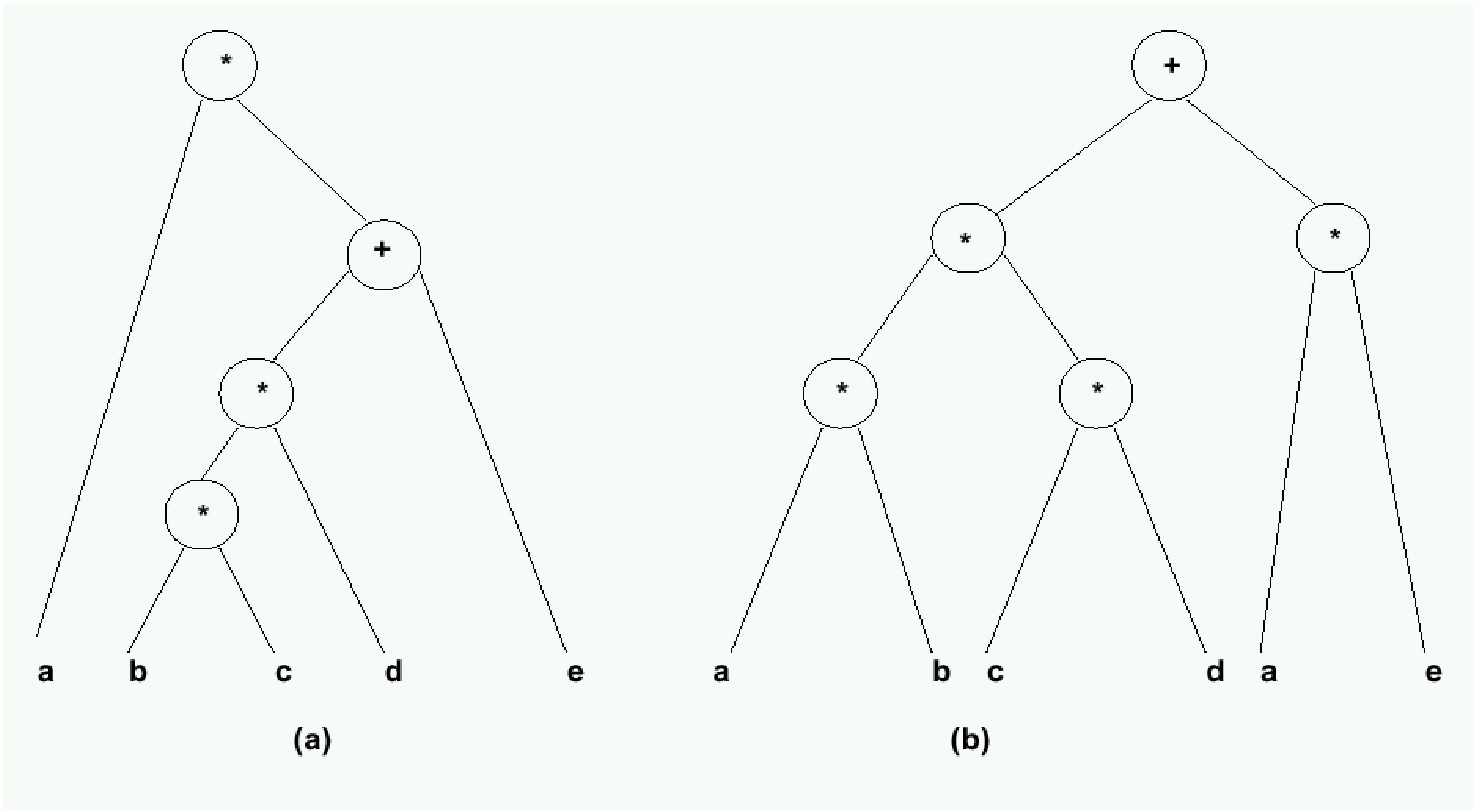
# Example of tree-height reduction using commutativity and associativity



x = ( a + (b * c ) ) + d

x = (a +d) + (b * c)

# Example of tree-height reduction using distributivity



x = a * (b c d +e)   x = (a b) (c d) + (a e);

## Examples of propagation

- First Transformation type: Constant propagation:
  - a = 0, b = a +1, c = 2 * b,
  - a = 0, b = 1, c = 2,

- Second Transformation type: Variable propagation:
  - a = x, b = a +1, c = 2 * a,
  - a = x, b = x +1, c = 2 * x,

## Sub-expression elimination

- Logic expressions:
  - Performed by logic optimization.

- Arithmetic expressions:
  - Search isomorphic patterns in the parse trees.
  - **Example:**
    - a = x +y,   b = a +1,   c = x +y,
    - a = x +y,   b = a +1,   c = a.

## Examples of other transformations

- Dead-code elimination:
  - a = x; b = x +1; c = 2 * x;
  - a = x; can be removed if not referenced.

- Code motion:
  - for (i = 1; i ≤ a * b; i++) { }
  - t = a * b;      for (i = 1; i ≤ t) { }
    - Multiplication only once.

## Control- flow based transformations

1. Model expansion.
2. Conditional expansion.
3. Loop expansion.
4. Block-level transformations.

## Model expansion

- Expand subroutine and flatten hierarchy as the result.
- Useful to expand scope of other optimization techniques.
- Problematic when routine is called more than once.
- **Example of model expansion:**
  - x = a +b; y = a * b; z = foo(x; y);
  - foo(p; q) {t = q - p; return(t); }
  - By expanding foo:
  - x = a +b; y = a * b; z = y - x

foo does subtraction

## Conditional expansion

- Transform conditional into parallel execution with test at the end.
- Useful when test depends on late signals.
- May preclude hardware sharing.
- Always useful for logic expressions.
- Example:
  - y = ab; **if** (a) {x = b + d; } **else** {x = bd;}
  - can be expanded to: $x = a(b + d) + a' bd$
  - and simplified as: $y = ab$; $x = y + d(a + b)$

Moves conditionals from control unit to data path

## Loop expansion

- Applicable to loops with data-independent exit conditions.
- Useful to expand scope of other optimization techniques.
- Problematic when loop has many iterations.
- Example of loop expansion:
  - $x = 0;$

    for (i = 1; i $\leq$ 3; i ++) {x = x +1; }
  - *Expanded to*:

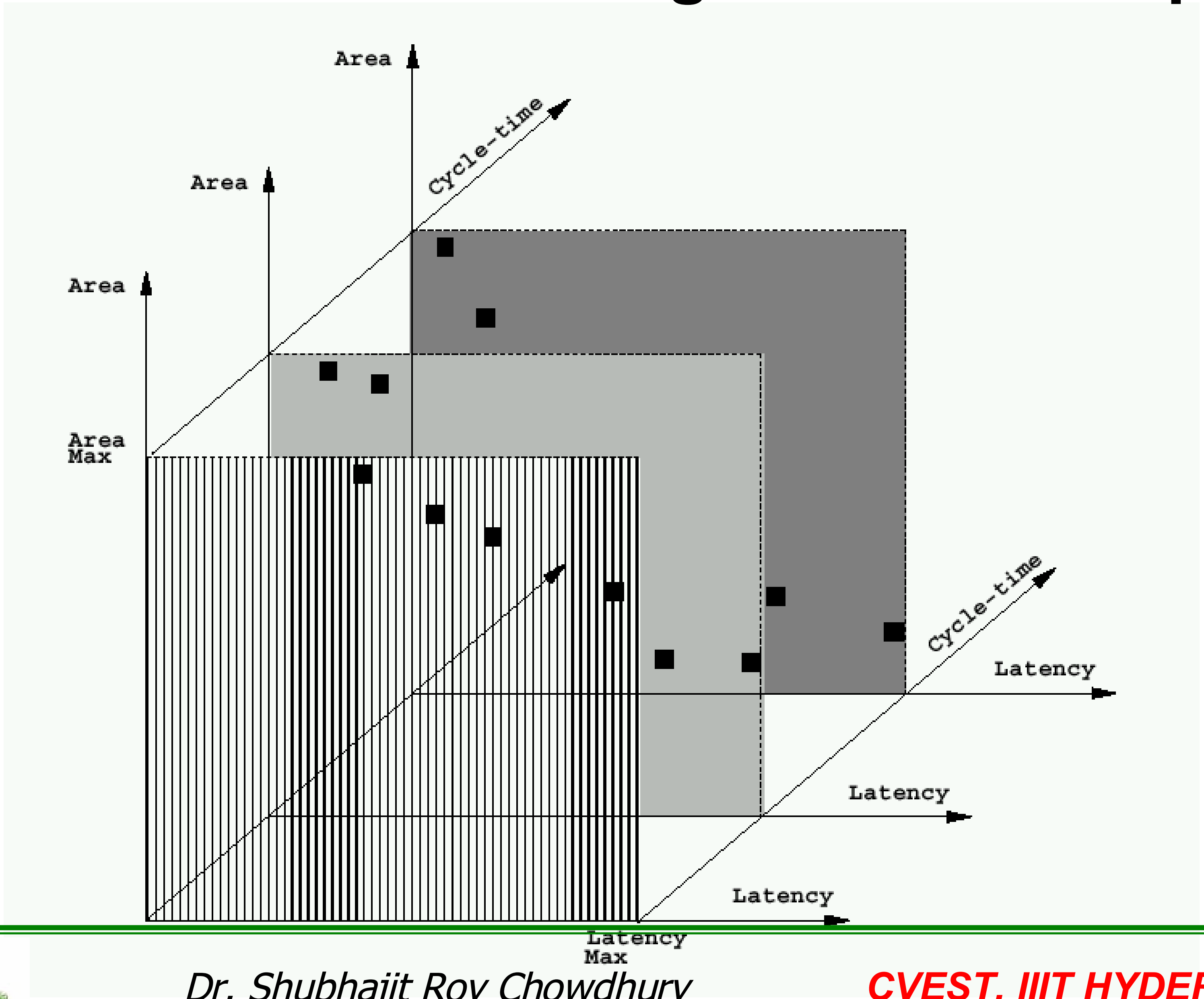
    $x = 0; \quad x = x + 1; \quad x = x + 2; \quad x = x + 3$

*Can use various variable semantics*

## Design space and objectives in architectural synthesis

- Design space:
  - Set of all feasible implementations.
- Implementation parameters:
  - Area.
  - *Performance:*
    - Cycle-time.
    - Latency.
    - Throughput (for pipelined implementations).
  - Power consumption

# Three dimensional Design evaluation space



## Hardware modeling

1. Circuit behavior:
   - Sequencing graphs.
2. Building blocks:
   - **Resources.**
3. Constraints:
   - Timing and resource usage.

*Our methods and data structures have to model them for architectural design*

## Synthesis in the temporal domain

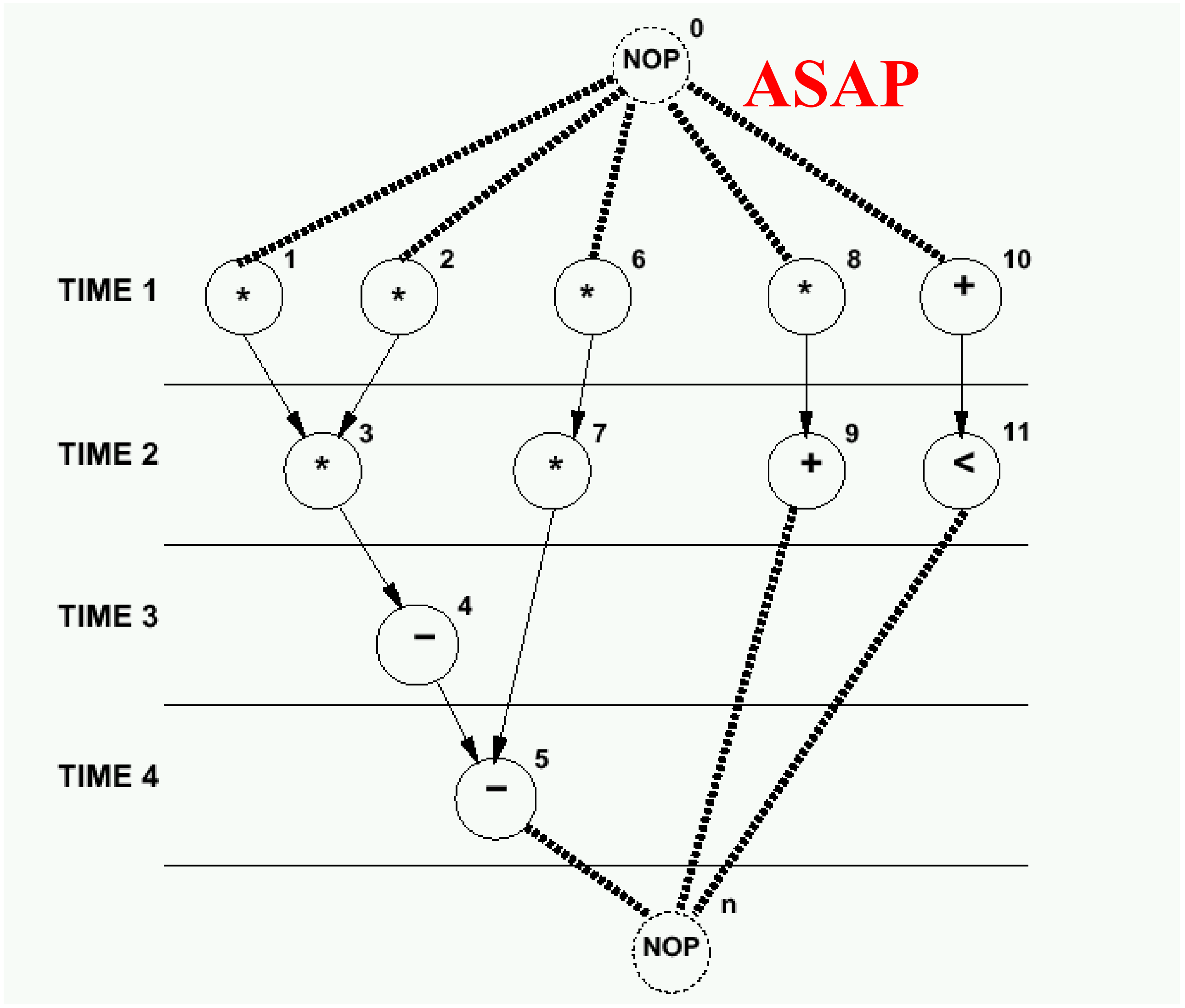
- Scheduling:
  - Associate a start-time with each operation.
  - Determine latency and parallelism of the implementation.

- Scheduled sequencing graph:
  - Sequencing graph with start-time annotation.

## Example of description of architecture

```
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 of Synthesis in the temporal domain



Here we use sequencing graph

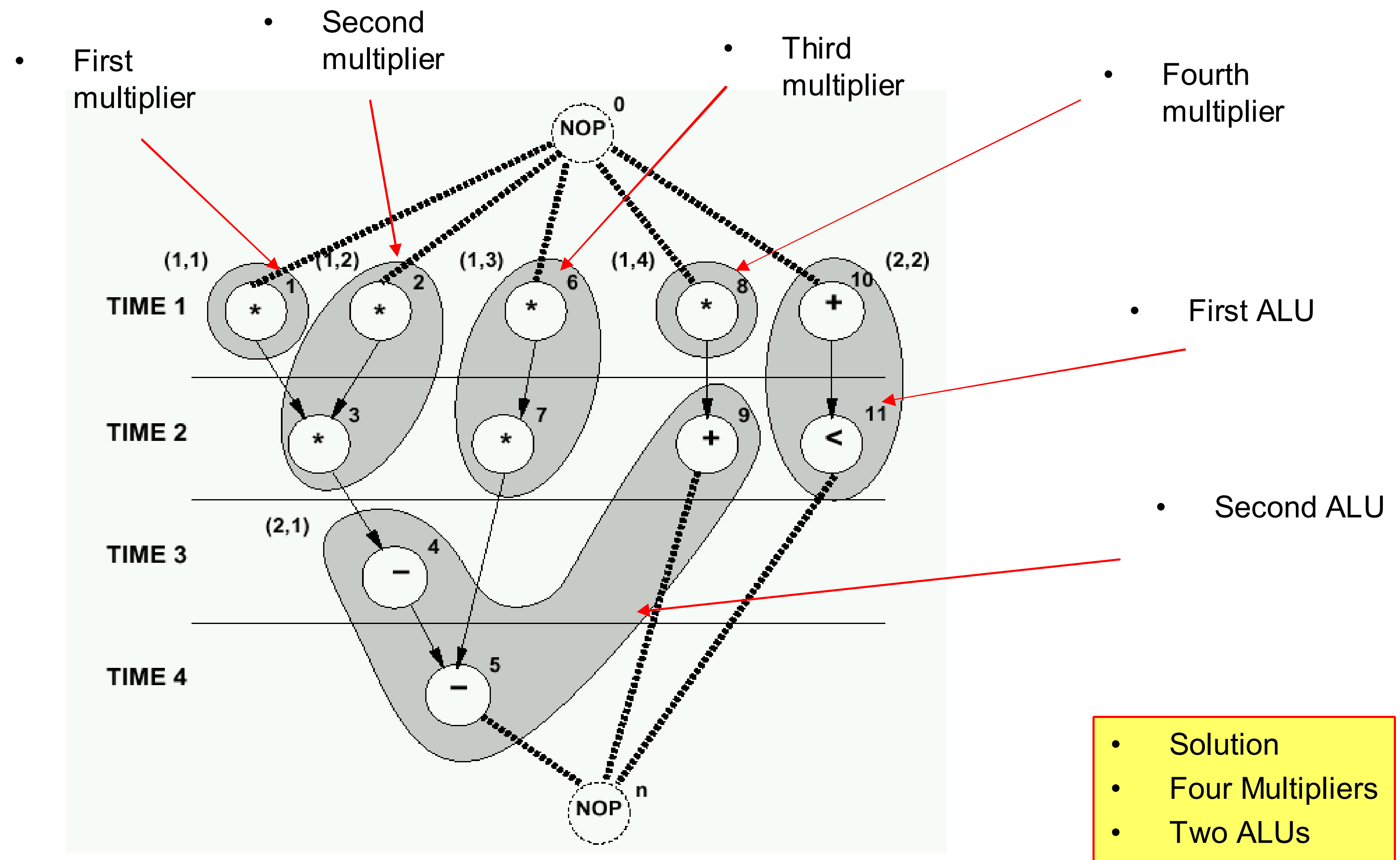
## Synthesis in the spatial domain

1. Binding:
   - Associate a resource with each operation with the same type.
   - Determine area of the implementation.

2. Sharing:
   - Bind a resource to more than one operation.
   - Operations must not execute concurrently.

3. Bound sequencing graph:
   - Sequencing graph with resource annotation.

# Example of Synthesis in the spatial domain



- Solution
- Four Multipliers
- Two ALUs
- Four Cycles

## Area-latency trade-off

- Rationale:
  - Cycle-time dictated by system constraints.
- Resource-dominated circuits:
  - Area is determined by resource usage.
- Approaches:
  1. Schedule for minimum latency under resource constraints
  2. Schedule for minimum resource usage under latency constraints

## Objective function 1

Main goals in classical approach

1. Minimum area
   - Functional units, registers, memory, interconnect
2. Maximum speed
   - Number of clock cycles

Generally one parameter is set as a constrained and the other one is optimized

# Essential Current and Open Issues in Design Automation

- Behavioral Specification Languages
  - From Matlab to chip, from Prolog to chip, etc.
- Target Architectures
  - Network on a chip, sensors and motion control integrated.
- Intermediate Representation
  - For users to exchange, to understand the design better
- Operation Scheduling
  - On the level of complex operations such as transforms or filters.
- Allocation/Binding
  - On many levels of operations and processors
- Control Generation
  - State machine optimization for large controllers
  - New technologies , integrate FSM-logic-layout

Still areas of active research

## Questions?