CAD for VLSI

References

- Algorithm for VLSI Design Automation
  - Sabih H Gerez
- High - Level Synthesis Introduction to Chip & System Design
  - Daniel D Gajski, Nikil D Dutt, Allien C-H Wu, Steve Y-L Lin
- Logic synthesis & Verification Algorithms
  - Gary D Hachtel, Fabio Somenzi
- Graph Theory with application to Engg & Computer Science
  - Narsingh Deo

Intro. to design methodologies
(Ch. 1 - Gerez)

- The VLSI Design Problem/Task
- Hierarchy & Abstraction
- The Design Domains
- Gajski's Y - Chart
- Design Issues & Tools

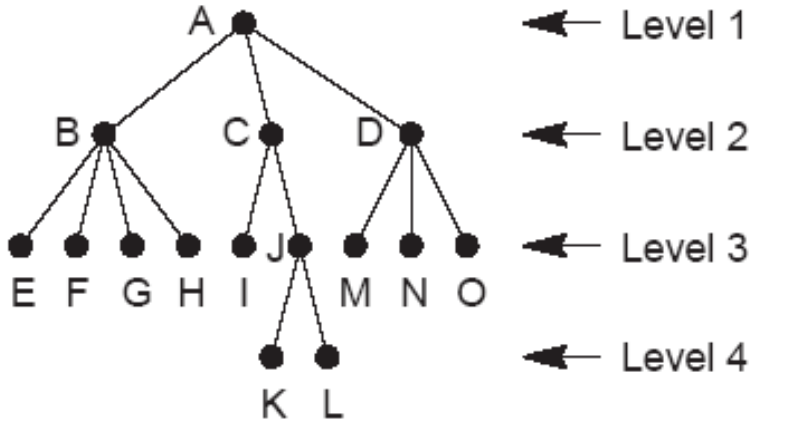
The VLSI Design Problem/Task

- Realize given specifications on silicon, optimizing the following entities:
  - Area
    - Impacts yield
  - Speed
  - Power dissipation
  - Design time
  - Testability
  - Cost
  - Optimization cannot be done in one step

Hierarchy & Abstraction

- Hierarchy: Shows the structure of a design at different levels of description.
- Abstraction: At a given level hides the details of lower levels.
- The use of abstraction makes it possible to deal with limited number of interacting parts

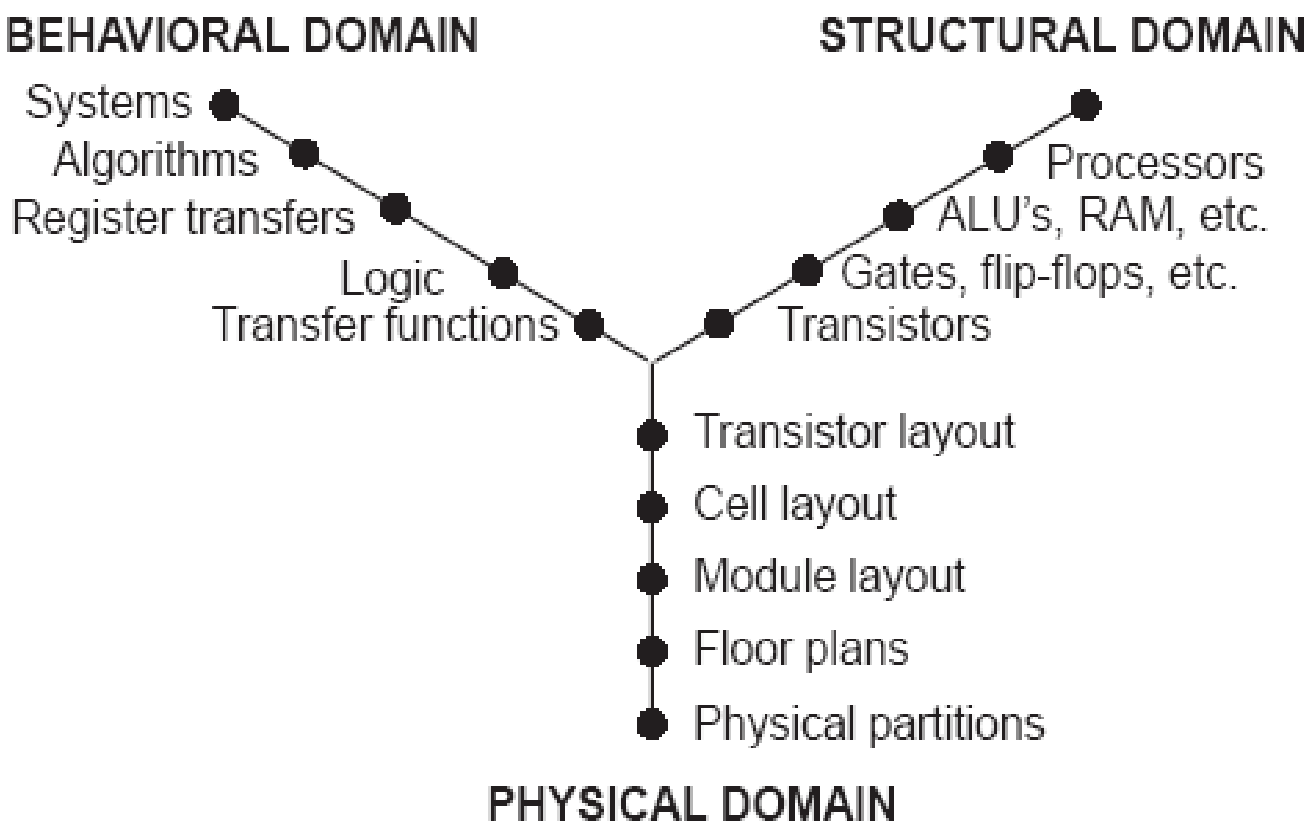
Decomposition tree



Design Domains

- Behavioral - Black box view
- Structural - Interconnection of sub blocks
- Physical - Layout properties
- Each design domain has its own hierarchy.

Gajski's Y - Chart



Behavioral domain

- The design is seen as a black box
- Input – Output relationships are given without implementation details
- At transistor level – Id as a function of Vgs and Vds
- At gate level – Boolean equations / Truth tables
- Register Transfer level – Sequential logic, State registers, functions that compute NS and output
- System level – Algorithms, Applications

Structural domain

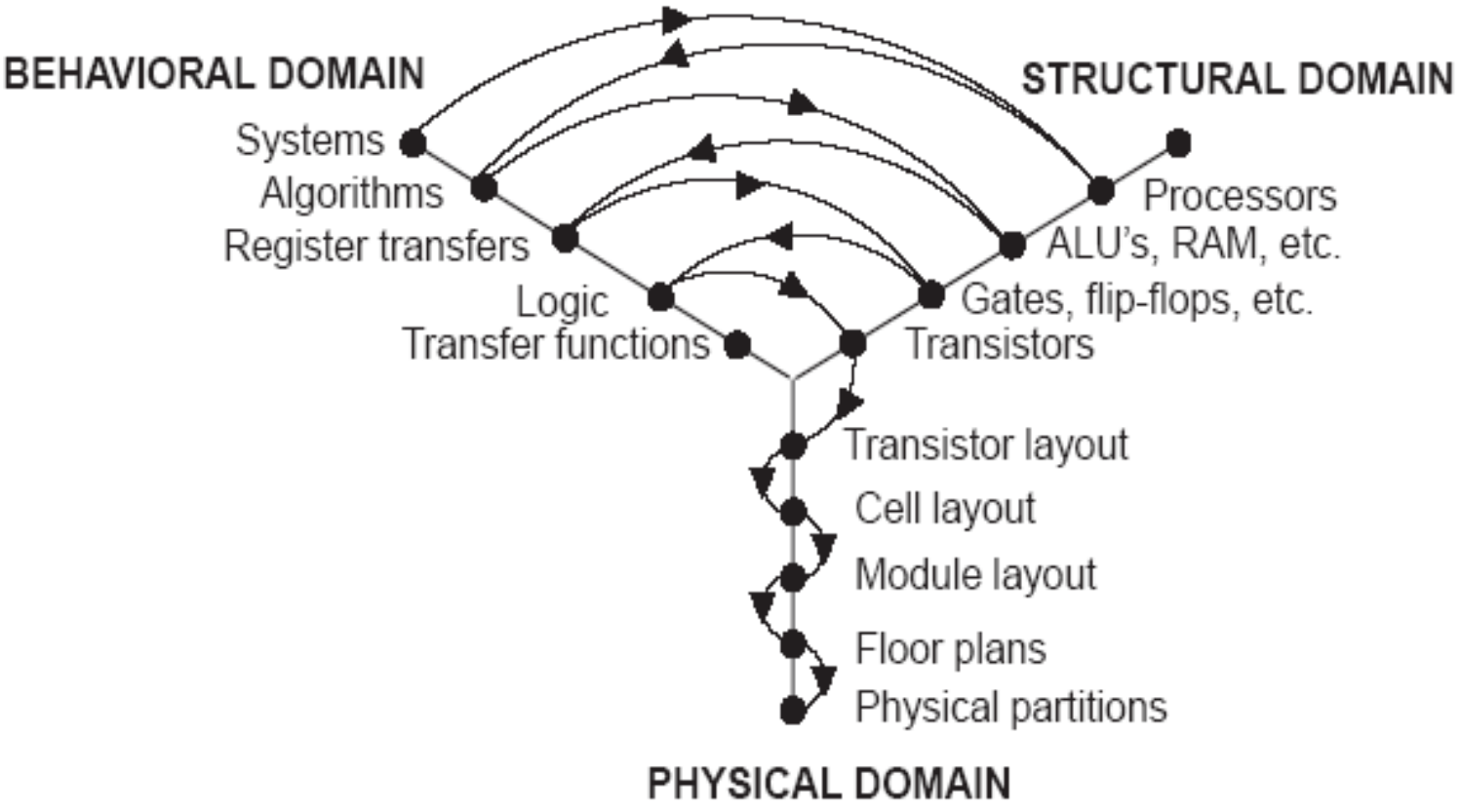
- The design is seen as composition of subcircuits
- Description indicates the subcircuits used and the way they are interconnected
- Each subcircuit may have description at behavioral and structural level or both
- Example – A schematic diagram showing how the transistors are to be connected to form a gate

Physical domain

- A VLSI circuit is realized on a 2/3 dimensional plane
- Physical domain description gives information on the way various subparts are located on the silicon surface
- These subparts are obtained from the structural description
- A layout of a logic gate can indicate
  - Orientations of transistors
  - Their physical sizes
  - Interconnections within the gate
- All these information is used to generate mask patterns

Design methodology

Top down structural decomposition and Bottom-up layout



Design actions

- Synthesis
  - Transition within a domain or from one to another
  - Adds detail to the current state of the design
- Verification
  - To ensure that implementation reflects/matches the intention in every stage of a design
  - Even fully automatic tasks need to be verified
- Analysis
  - Provide data on the quality of the design
  - Hints on how to optimize
- Optimization
  - Improving the quality of the design without making a transition to another level of abstraction
- Design management
  - Data management, storage, tool flow/communication

Design methods and technologies

- Full custom
  - Maximal freedom
  - The designer can fix shapes of every device at the mask level
  - Higher performance
  - Large search space, longer design time
  - Difficult to do VLSI designs this way
  - Mostly manual. Eg. Analog design
  - Custom manufacturing

Design methods and technologies

- Semi-custom
  - The designer has limited freedom
  - Not too many choices, smaller search space hence shorter design time
  - Performance depends on implementation technology
  - VLSI designs can be done
  - Design automation tools can be used
  - Custom manufacturing or usage of pre-manufactured chips

Design methods and technologies

- Semi custom design
  - Gate arrays (FPGA's, MPGA's)
  - Standard cells (ASIC cell library)
  - Parameterized modules
  - Any combination of the above

Design methods and technologies

Gate Arrays - MPGAs

- Manufacturers pre-fabricate wafers with transistors placed in regular patterns without final metallization
- The designers need to specify the wiring/metal layer patterns to interconnect these transistors – customization
- Post customization, manufacturer completes fabrication

- These types are called mask programmable(programmed) gate arrays – MPGAs

- Older types have separate logic and interconnect areas called wiring channels.
- Modern versions have wiring patterns on top of the logic or transistors – sea of gates architecture.
- Lower cost, turnaround time compared to full custom design

Design methods and technologies

Gate Arrays – FPGAs

- Chips are fully fabricated
- Interconnects can be programmed by applying electrical inputs to some pins
- Interconnections can be temporary or permanent
- These can be programmed in the field/lab/designer's place
- Unit cost is higher, lower cost for low volumes

Design methods and technologies

Standard Cells

- A VLSI circuit is realized by combining elementary circuits called 'standard cells'
- Standard cells are predesigned and have been made available to the designers in a library called 'standard cell library'
- A complete design is built from these cells, which saves low level design steps and hence faster design time
- Custom manufacturing or FPGA implementation
- Higher performance
- Higher cost for low volumes

Design methods and technologies

- Module generators
  - There are sub-circuits in a design that have regular structures – module
  - Entire module can be described by one or two parameters – no. of bits of an adder, word length

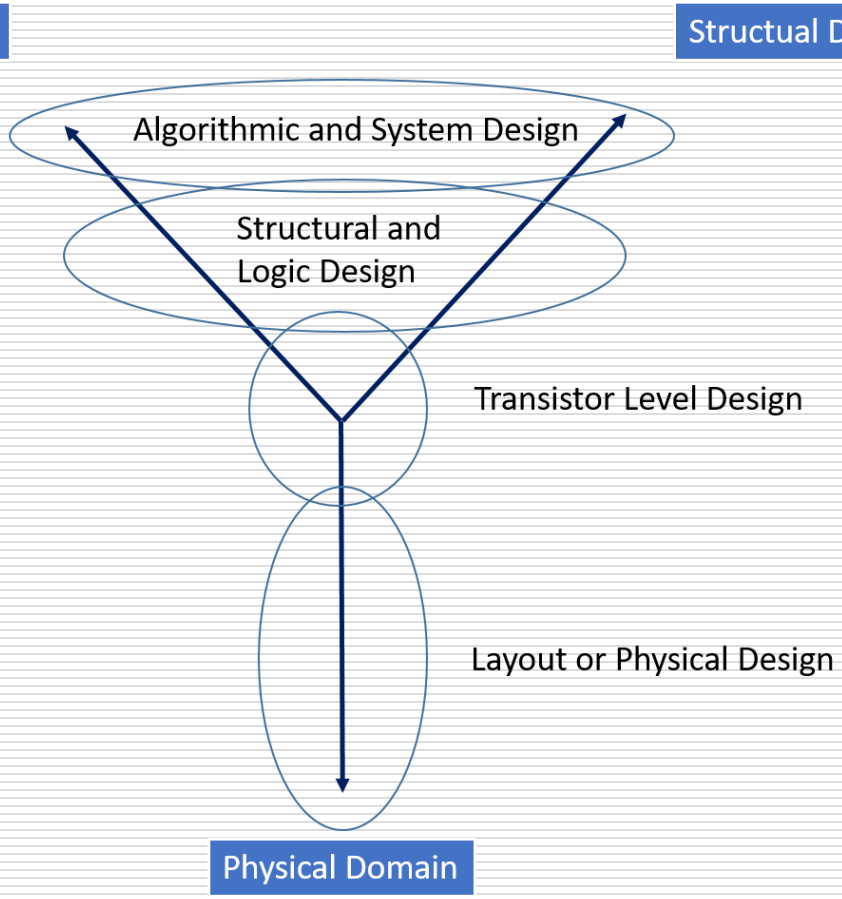
- 'Technology' refers to the semiconductor process used to fabricate an IC.
  - Type of semiconductor (GaAs, CMOS, Bipolar..)
  - Details of a certain technology like gate length.. etc.

- Design methods and Technology have consequences on CAD tools

VLSI design automation tools – Quick Tour (Ch. 2 - Gerez)

- The CAD tools can be grouped into different levels of design and support activities
  - Algorithmic and System Design
  - Structural and Logic Design
  - Transistor Level Design
  - Layout Design
  - Verification
  - Design Management

VLSI design automation tools



Algorithmic and System Design

- Design-space exploration
- Experimentation with the specifications, Algorithms to be used
- Behavioral description using C, C++, Matlab etc.
- Hardware description using HDL's, simulation, synthesis, formal verification
- Graphical design entry – FSM editors
- Partitioning a system specification into hardware and software
- Co-design, co-simulation

Structural and Logic design:

- Schematic entry
- Gate-level simulation, Fault simulation
- Register-transfer level Synthesis
- Logic synthesis - Synthesis at the level of Boolean logic
  - Two level – SOP/POS forms – PLA implementation
  - Multi level – Random logic –Standard cell implementation
  - Sequential circuits – FSM's
- Technology mapping
- Timing analysis

Transistor-level design:

- Most tools are schematic capture and simulation tools
- Depending on accuracy required, types of simulations:
  - Switch level – digital
  - Timing level - analog PWL models
  - Circuit level – analog accurate models
- Extraction – post layout
- Simulation of the extracted circuit
- Characterization of standard cells

Layout Design

- Floor planning
- Placement and Routing
- Module and Cell generation
- Layout editing: symbolic and at mask level
- Layout compaction
- Design-rule checking
- Layout extraction

Design management:

- Data bases, frame works, etc.

Floor planning

- Simultaneous development of structure and the layout
- When making behavioral to structural transition, relative positions of sub blocks are fixed
- Overall layout information becomes available early
- Closely related to placement problem, it mainly deals with the estimations.
- Placement gives detailed layout information

Placement

- Suppose that layouts of sub blocks are available, along with the interconnections to be made
- Task is to compose the layout of the entire IC from these
- First task is to assign a position for each sub block in the plane – Placement
- Attempt is made to minimize the area occupied by the interconnections and chip

Routing

- Generation of wiring patterns to realize the interconnections between the sub blocks as specified in the netlist.

- Goal of placement and routing
  - Minimization of chip area
  - Meeting of timing constraints
    - By minimizing the wire length/propagation delay
    - Called timing driven layout
  - Signal integrity, Cross talk

Partitioning

- Grouping of the sub blocks in a structural description such that tightly connected blocks are put in the same group
- Number of connections between the groups is kept low
- Not strictly a layout problem

Cell compiler

- Generates the layout of a circuit with few transistors
- Mostly targets regular arrangement of transistors
- Similar to module generation

Module generation

- Module generator generates layout of a module/hardware block such as adders, memories etc. from elementary cells
- These elementary cells are called microcells
- Microcells have a complexity around 10 transistors
- Examples of microcells – full adder circuit, single bit memory cell
- Given the number of bits/word length, module generator composes the layout of the entire module

Layout editor

- Can be used to edit/modify the layout at the level of mask patterns
- Full custom tool set usually includes it
- Allows the designer to insert/modify/delete patterns in specific layers

Design rule checking (DRC)

- Mask patterns should obey some rules – like minimal distances/width etc.
- These are called design rules

- Tools that analyze a layout to detect violations of these rules are – Design Rule Checkers
- Interactive or offline

- Circuit extractor
- It takes mask patters as input.
- It constructs a circuit of transistors, resistors and capacitors that can be simulated

Symbolic layout

- Porting issues when technology changes
- Symbolic layout fixes the topology of the layout - relative positions, orientations
- Exact widths and distances are irrelevant

- **Layout compactor** tool takes the symbolic layout as input and assigns widths and distances such that design rules are satisfied

Verification Methods

- Prototyping
  - Bread-boarding – in early days of IC design
  - FPGA's – Rapid system prototyping, used in audio/video processing
  - Shows the effect of algorithms in real time
- Simulation
  - Construct a computer model, execute it for a set of input signals and observe the output signals
  - It is difficult to have an exhaustive test of a complex circuit
- Formal Verification
  - Using mathematical methods to prove that a circuit is correct
  - One can reason about large/infinite sets without enumerating all the elements of the set
  - Gives higher certainty than simulation

Design management tools

- CAD tools consume/produce enormous amount of data
- Data management techniques are required
- Version management
- Tool integration
- Framework
  - Uniform user interface
  - Standardized procedure calls
  - Methodology management

Introduction to High Level Synthesis

Reference : Introduction to High Level Synthesis

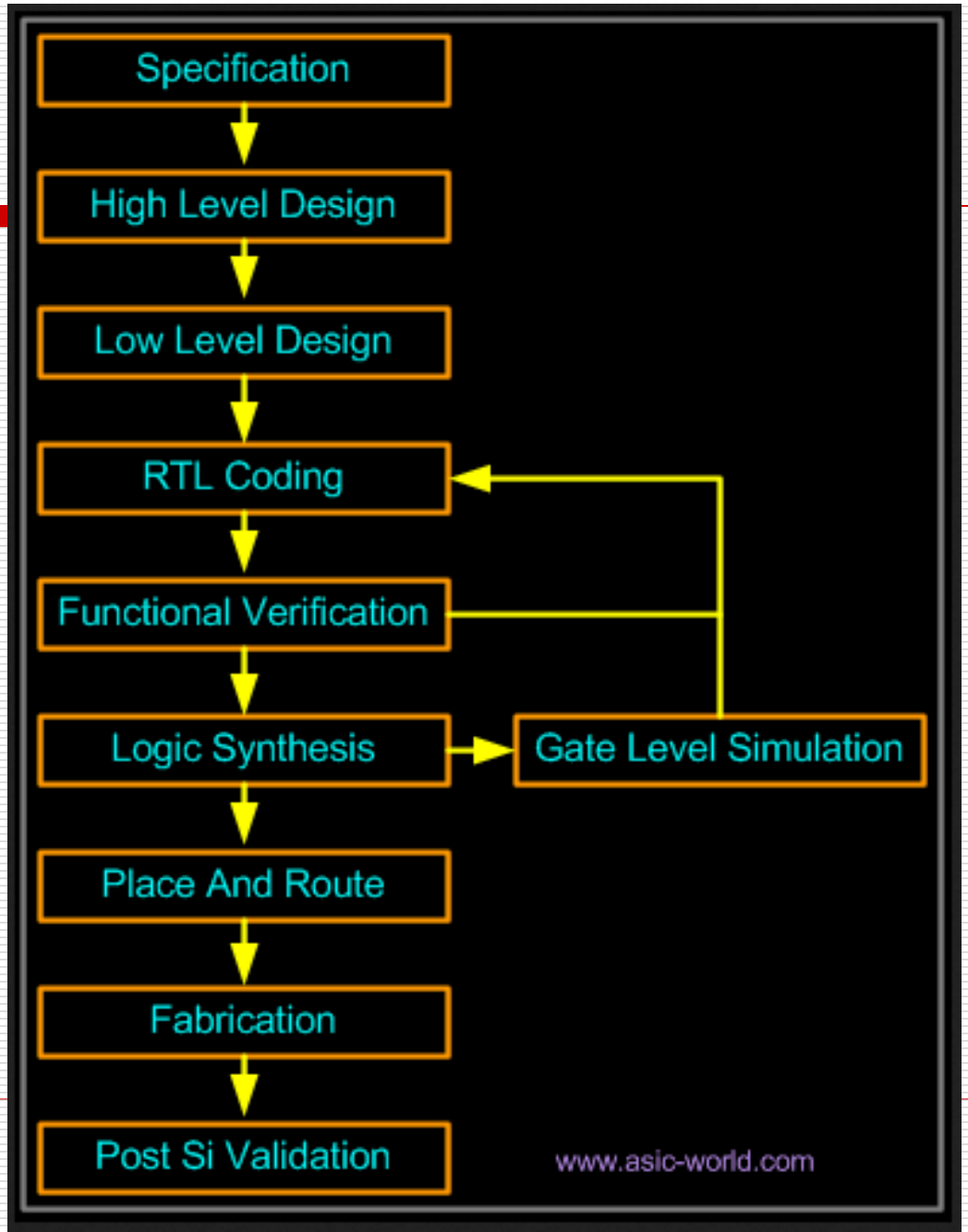
By

Gajski, Nikil, Allen, Steve

Contents

- The VLSI design flow
- Definition of synthesis
- Types of synthesis
- Design representations and transformations

Simplified VLSI Design Flow

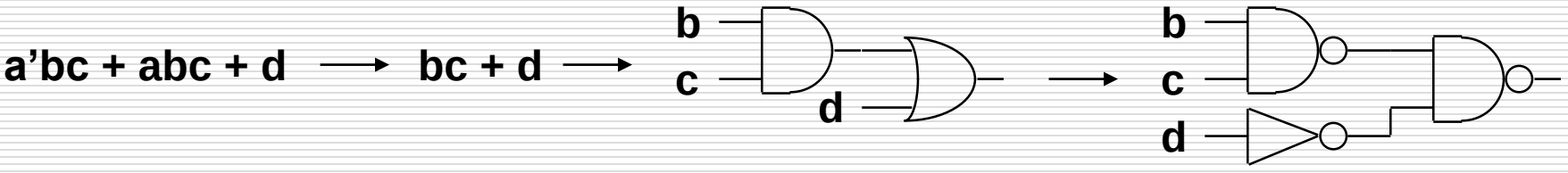


Definition of Synthesis (1.3)

- **Synthesis**
  - It is a translation from behavioral description into a structural description
  - Also called Design Refinement, adds an additional level of detail that provides information needed for the next level of synthesis or for manufacturing of the design
- Synthesis process consists of the following tasks
  - Compilation
  - Minimization
  - Technology mapping
  - Optimization
  - Transistor sizing

Types of synthesis

- **Circuit synthesis**
  - Generates a transistor schematic from a set of input-output current, voltage and frequency characteristics or equations
- **Logic synthesis**
  - Translates Boolean expression into a netlist of components from a given library of logic gates



Types of synthesis

- **Register transfer (RT) synthesis**
  - Generates two parts
    - → A **datapath,** which is a set of storage elements and functional units that perform the given RT
    - → A **control unit** that controls the sequencing of the states in the RT descriptions.
- **System synthesis**
  - Generates a structure of processors, memories, controllers and interface adapters from a set of system components.

    (Each Components can be RT descriptions)

Types of synthesis

- Layout Synthesis (Physical design)
  - Translates a structural description into layout information ready for mask generation
- Cell synthesis
  - Generates a cell layout from a given transistor schematic with specified transistor sizes

High level synthesis (1.5)

- Deals with RT components
  - Operators : ALU's, shifters, multipliers..
  - Storage elements : registers, RAM's, associative memories..
  - Interconnection units : selectors, buses, signals..
- RT components are connected in different configurations to form 'system components' such as controllers, processors, memories
- System design deals with a large number of 'system components'

High level synthesis

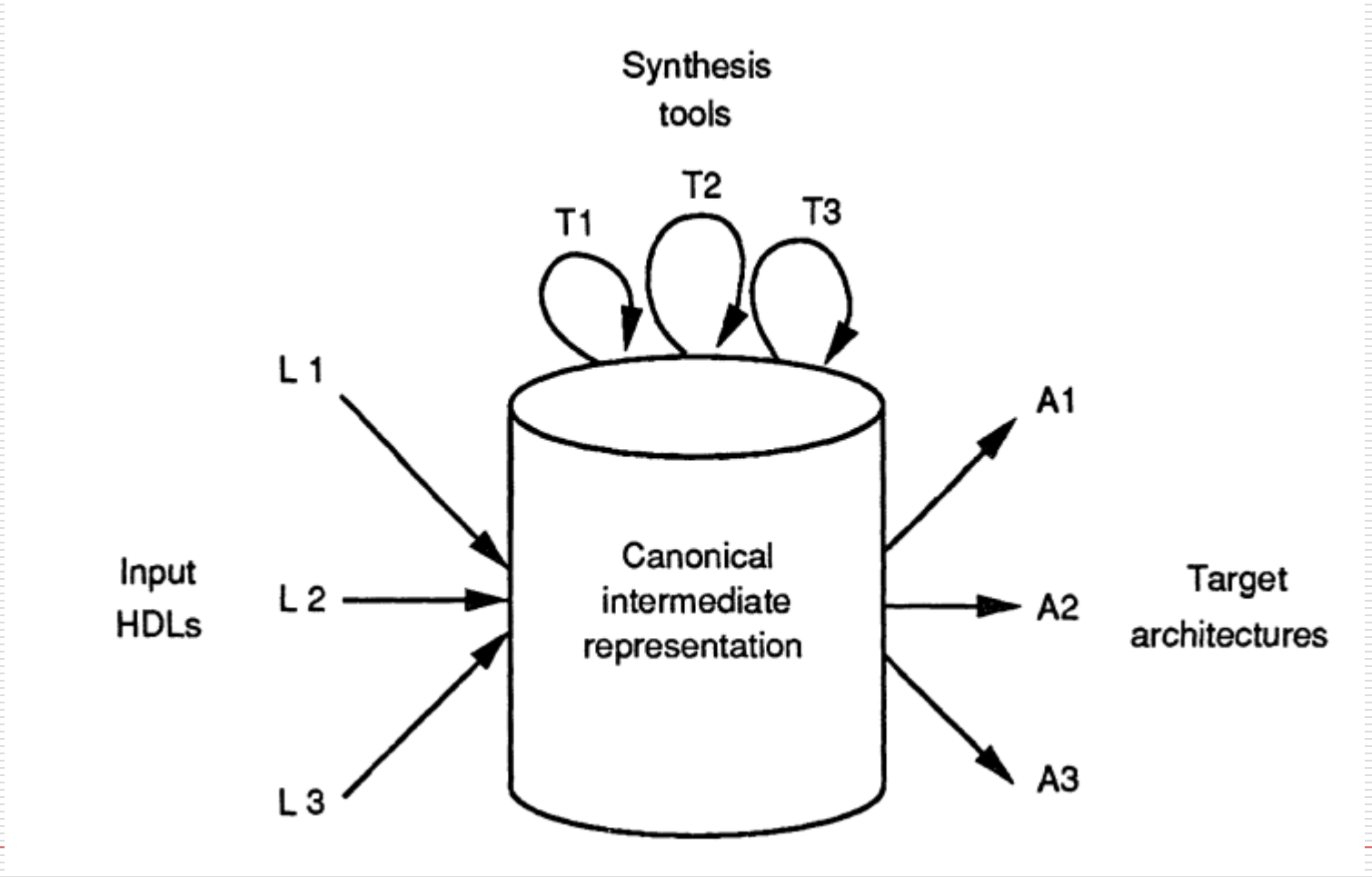
- Types of information needed to describe 'system components' (at structural level) are,
  - Specification of RT elements
  - Connectivity of declared RT elements
  - Set of register transfers for each state
  - Sequencing of states for each combination of inputs
  - Timing relationships between inputs/outputs
  - Physical characteristics of the components – area, power, speed
  - Definition of external environment such as temperature, process..

Design Representations and Transformations (Ch. 5 - Gajski)

Intermediate Representation –requirements

- We need a canonical intermediate representation that facilitates efficient mapping of input HDL description into different target architectures using different tools
- It needs to preserve the original behavior (HDL)
- Should allow addition of synthesis results/information in different design domains/levels of abstraction to allow for multi-level simulation and debugging
- Must provide uniform view of the design across the tools/users
- Should be HDL independent
- Must be powerful enough to support various architectural styles

Design Representations and Transformations



Control and Data Flow graphs - CDFG

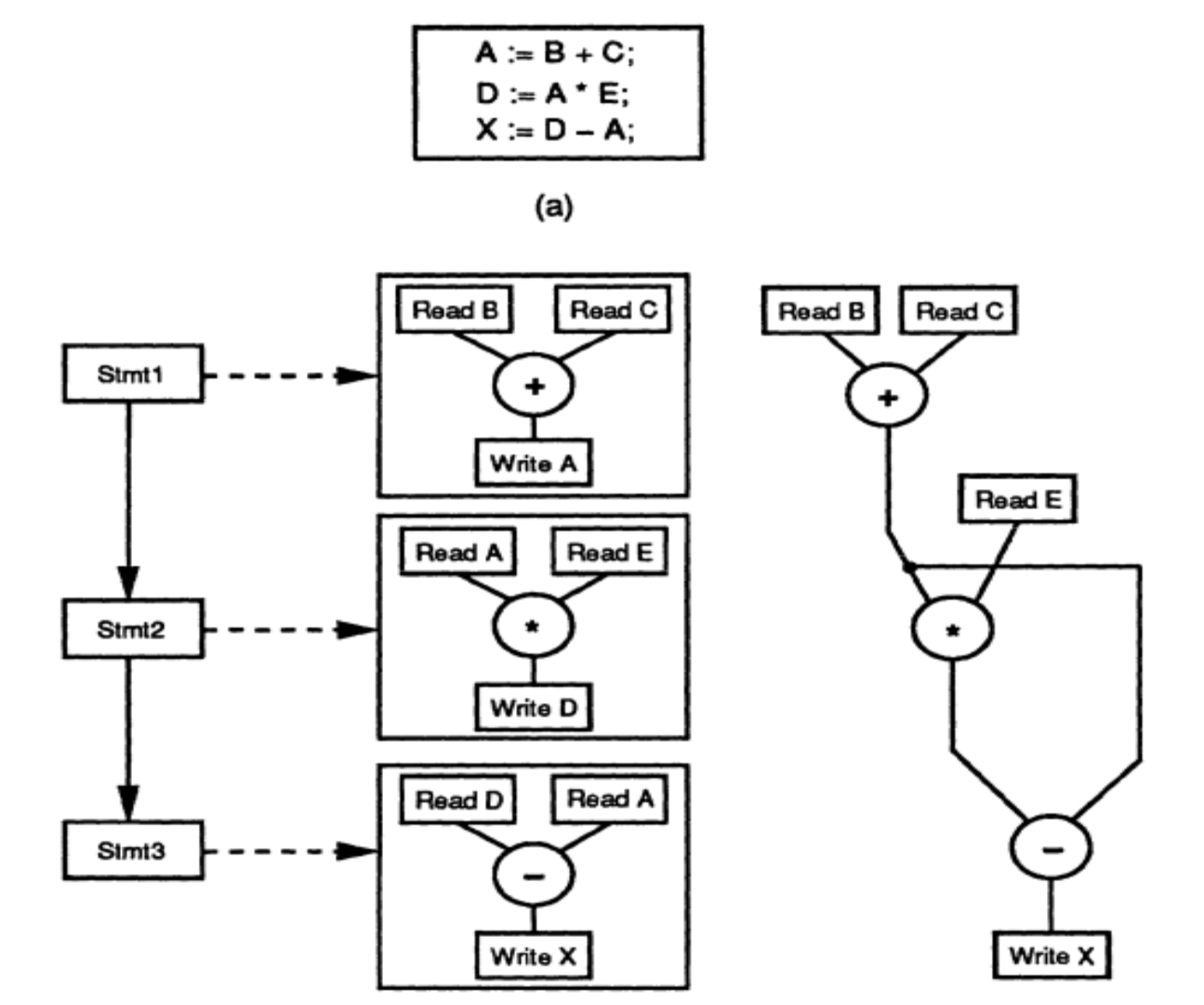
- First step in synthesis is compilation of input behavior in into a intermediate graph representation

- We perform high level synthesis tasks such as
  - Scheduling
  - Unit Selection
  - Functional, Storage and Interconnection Binding
  - Control Generation

Data Flow Graph (5.3)

- Captures operational activity
- Data flow graph generation steps :
- Generation of parse trees
- Interpret the execution ordering
  - Sequential
  - Concurrent
- Dataflow analysis
- Fusing of parse trees

Data Flow Graph - Example



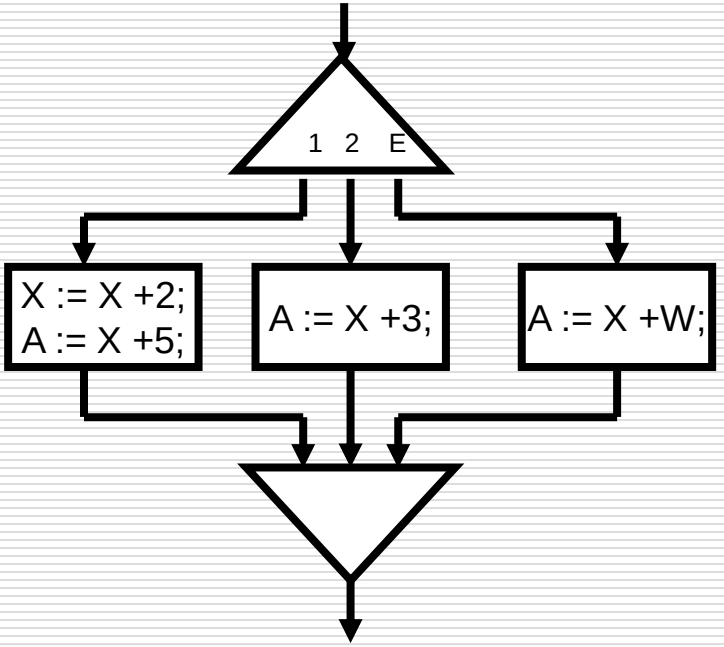
Control flow graph (5.4)

- Consists of
  - Conditional branches
  - Conditional join
  - Loops
  - Statement assignment blocks

- Each of these are represented by symbols – Triangles, Inverted triangles, rectangles

Control flow Representation

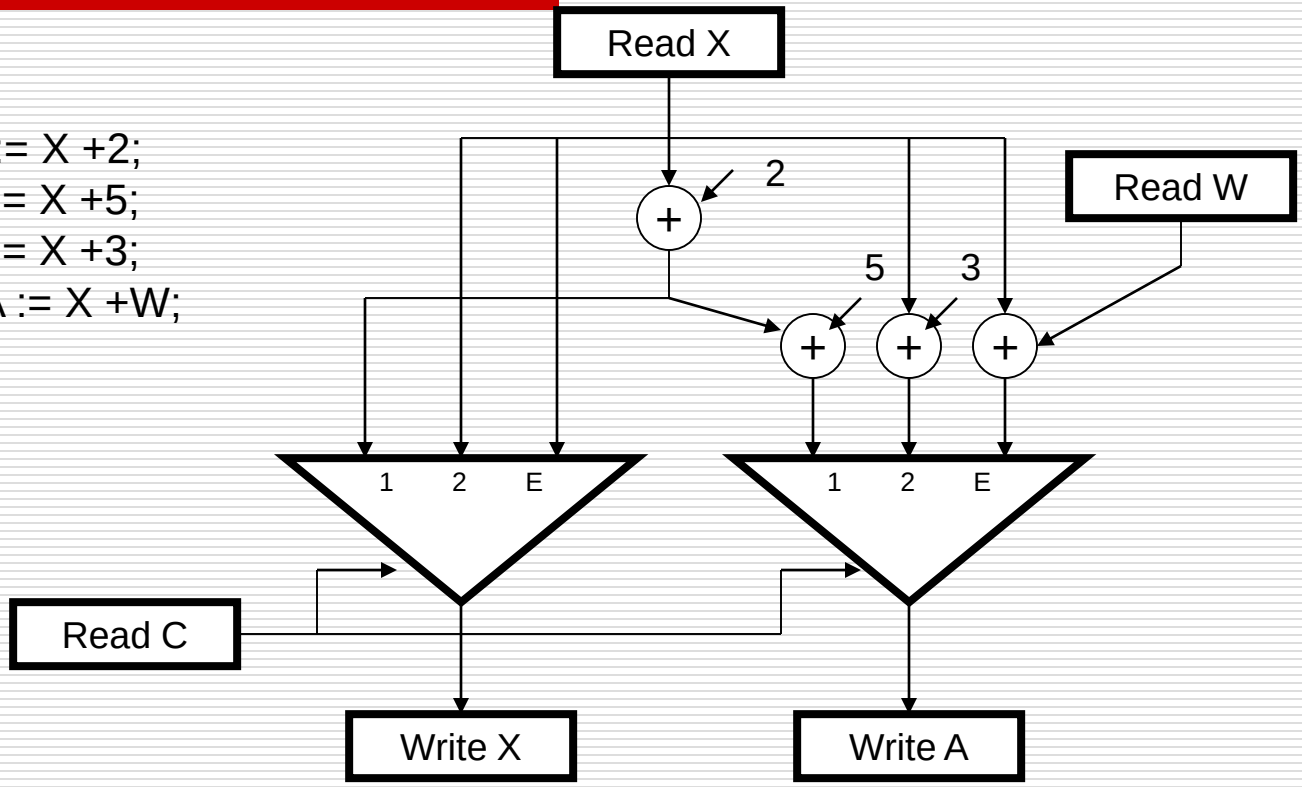
```
Case c is
      When 1 =>      X := X +2;
                     A := X +5;
      When 2 =>      A := X +3;
      When others => A := X +W;
End case;
```



Control Flow Representation

Data flow Representation

```
Case c is
      When 1 =>      X := X +2;
                     A := X +5;
      When 2 =>      A := X +3;
      When others => A := X +W;
End case;
```



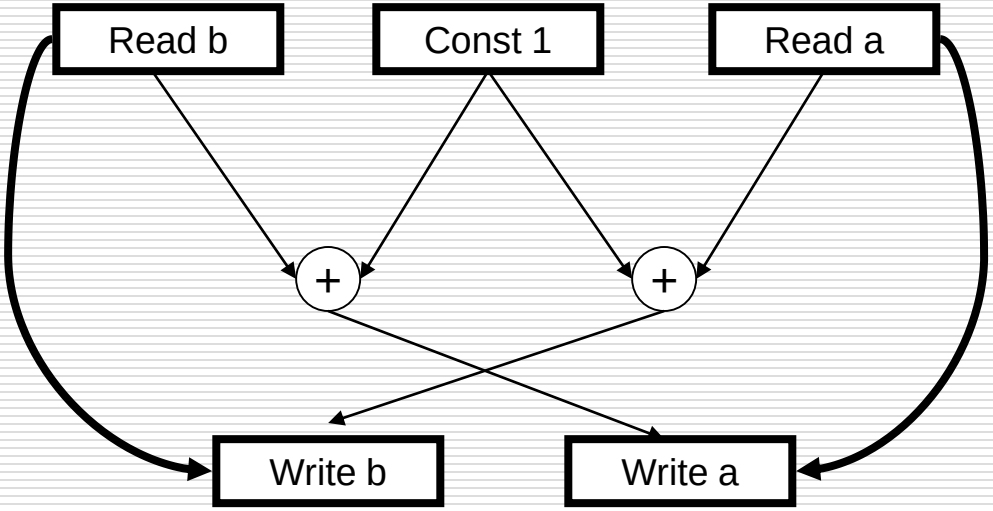
Data Flow Representation

Sequencing & Timing

Signal statement

```
b <= a + 1;
a <= b + 1;
```

Statements execute concurrently



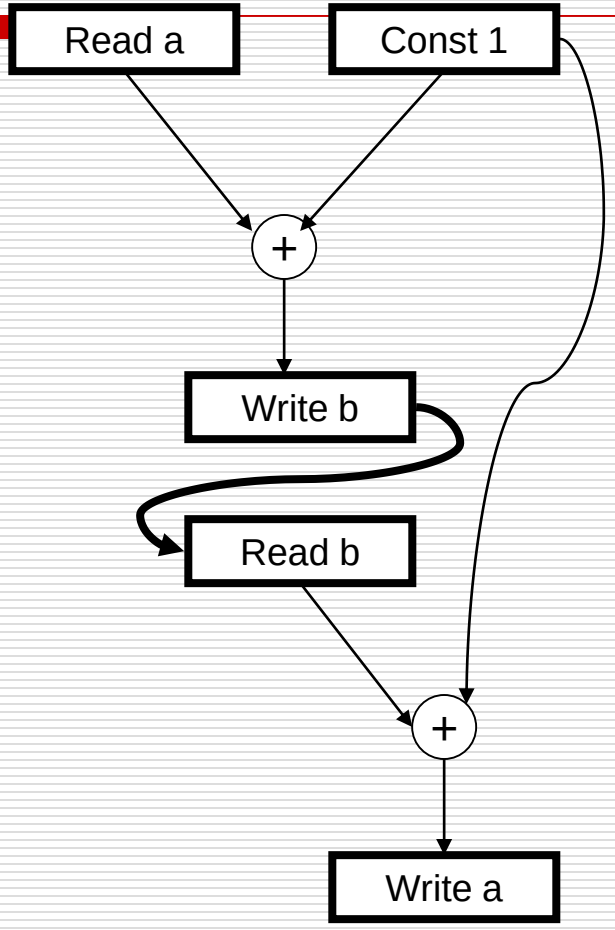
Note:

- We must ensure read operation precedes the write operation
- We must ensure that variable values are read before a new value is defined
- Conversely, a read access of a variable should not be executed before its value is defined.

Sequencing & Timing

Variable statement

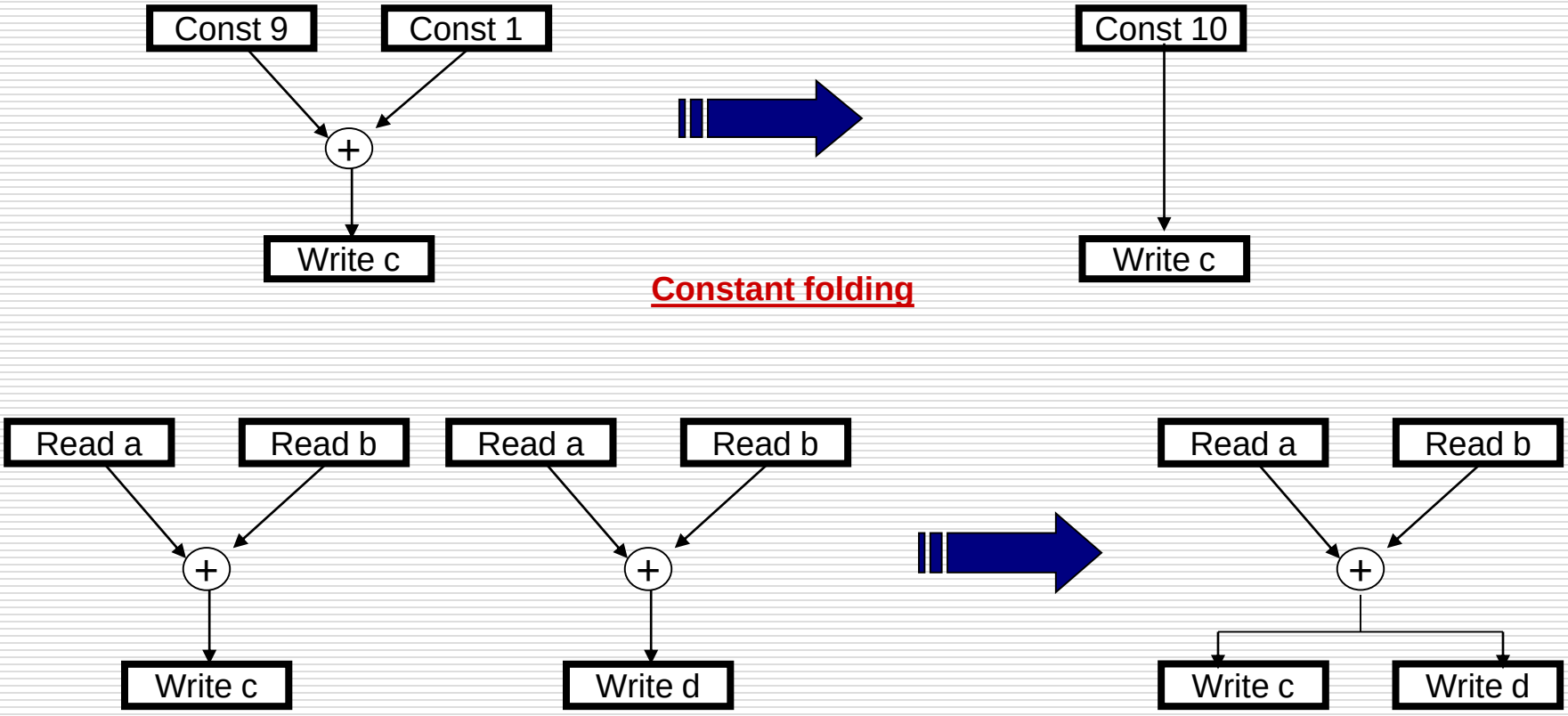
```
b := a + 1;
a := b + 1;
```



Transformations (5.7)

- Initial flow graph representation closely matches input description – language syntax and constructs
- These may not be relevant or useful to synthesis
- The original flow graph is transformed into a form, which is more amenable for high level synthesis

Compiler Transformation

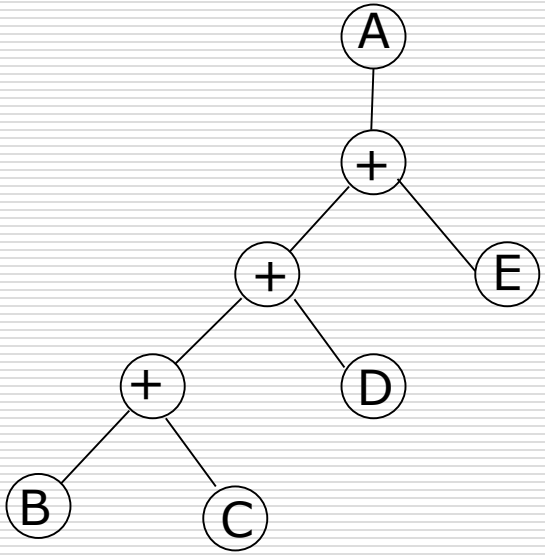


Flow Graph Transformation

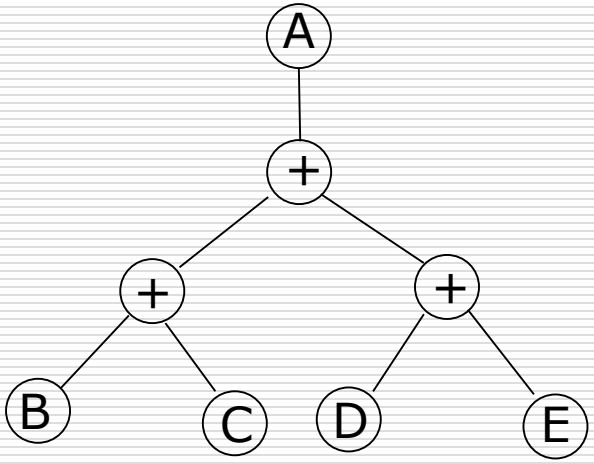
- Flow graph level transformations are done to improve parallelism in the design
- Tree height reduction
- CF - DF Transformation / Flattening of hierarchical CDFG graph into flat DFG

Flow Graph Transformation

- Tree height reduction
  - Tree height reduction uses the commutative and distributive properties of operators to decrease the height of a long expression tree
  - Exposes the potential parallelism within a complicated dataflow graph
  - Eg: A :=B + C + D + E



After tree height reduction
A := (B+C) + (D+E)

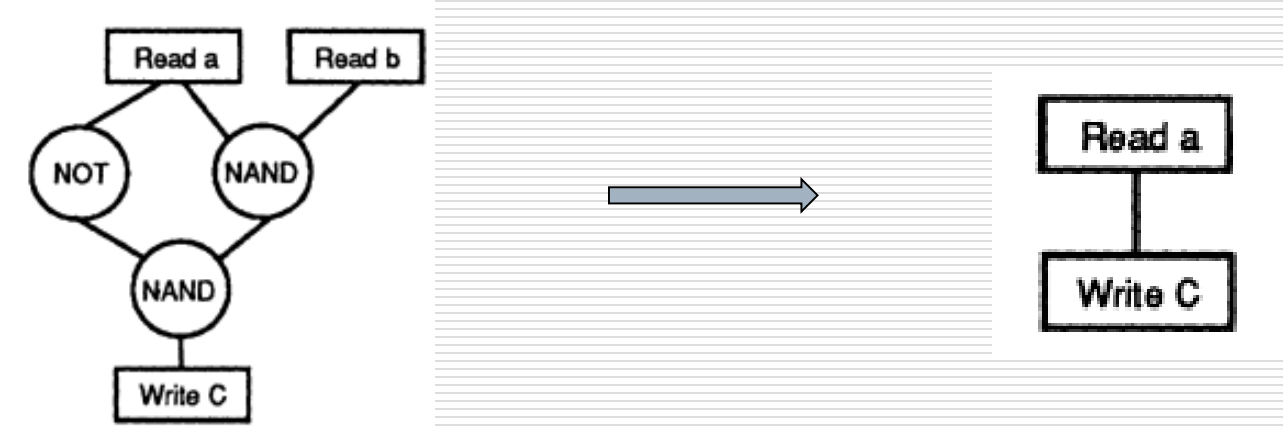


Flow Graph Transformation

- CF - DF Transformation / Flattening of hierarchical CDFG graph into flat DFG
  - Useful when Boolean variables are tested
  - Eliminates redundant states
  - Data flow representation exploits parallelism better
  - Loops in the control flow can be unrolled into DFG which exposes parallelism

Hardware - Specific Transformations

- These make use of properties of hardware to optimize the intermediate representation
- At the logic level, we can apply local Boolean optimization techniques to locally optimize parts of a flow graph



Hardware - Specific Transformations

- At the RT level, we can use pattern matching to detect and replace portions of the flow graph with simpler flow-graph segments corresponding to functional units available in a RT library