EE-677:
Foundation of CAD for VLSI
Summary

Virendra Singh

Computer Architecture and Dependable Systems Lab

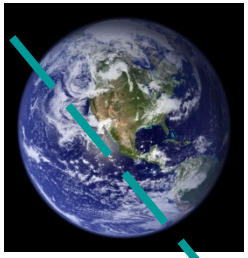
Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

Wanted: CUSTOMERS, who breathe, eat, and live in…..



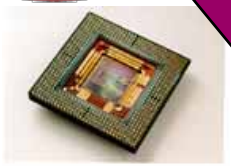
Global & Regional Political & Macro-Economic Environments

~$ 50,000B

Customer Demand

Electronic End Equipment

~$ 1050B



Semiconductors

~$ 400B

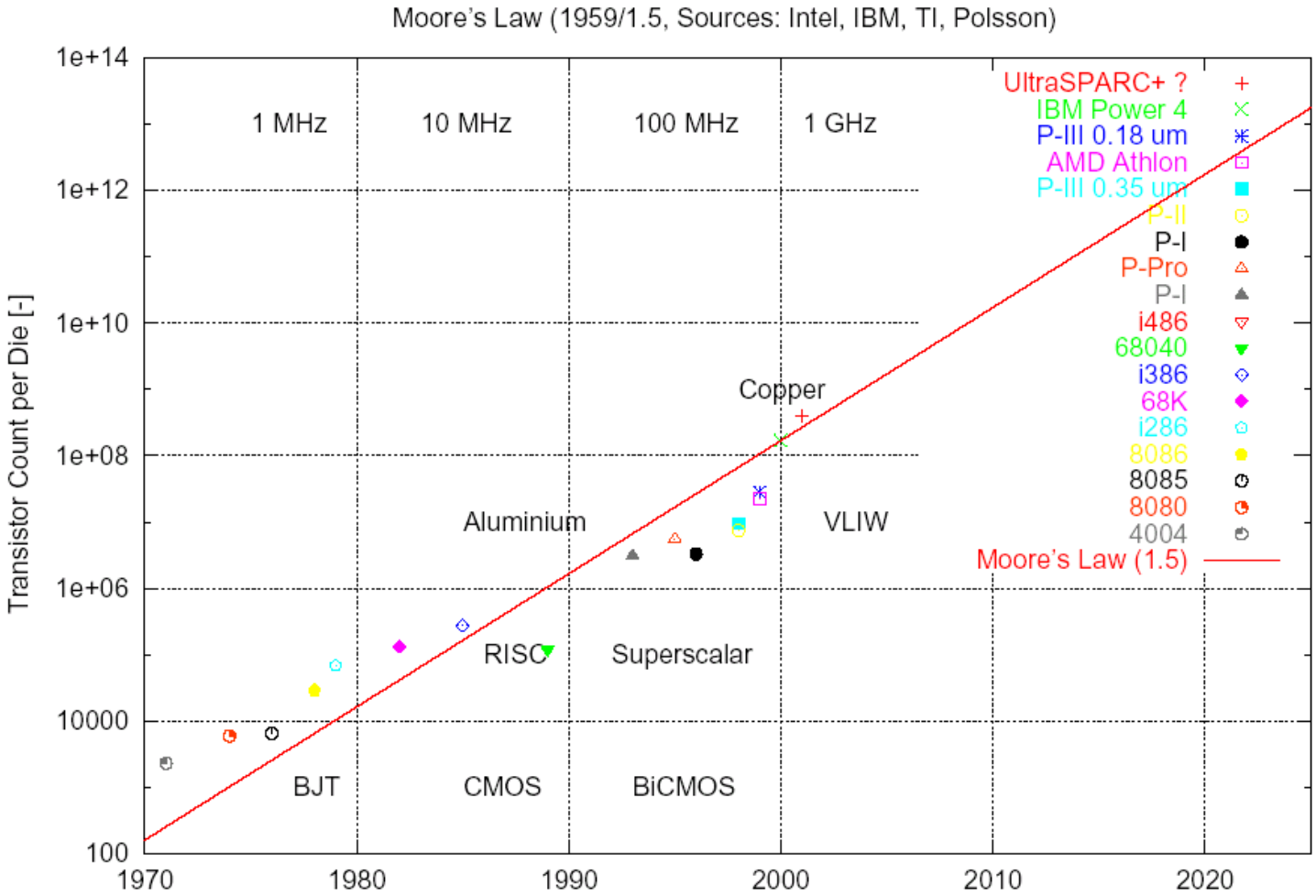
Semiconductor Equipment & Materials

~$ 100B

ITRS International Technology Roadmap for Semiconductors

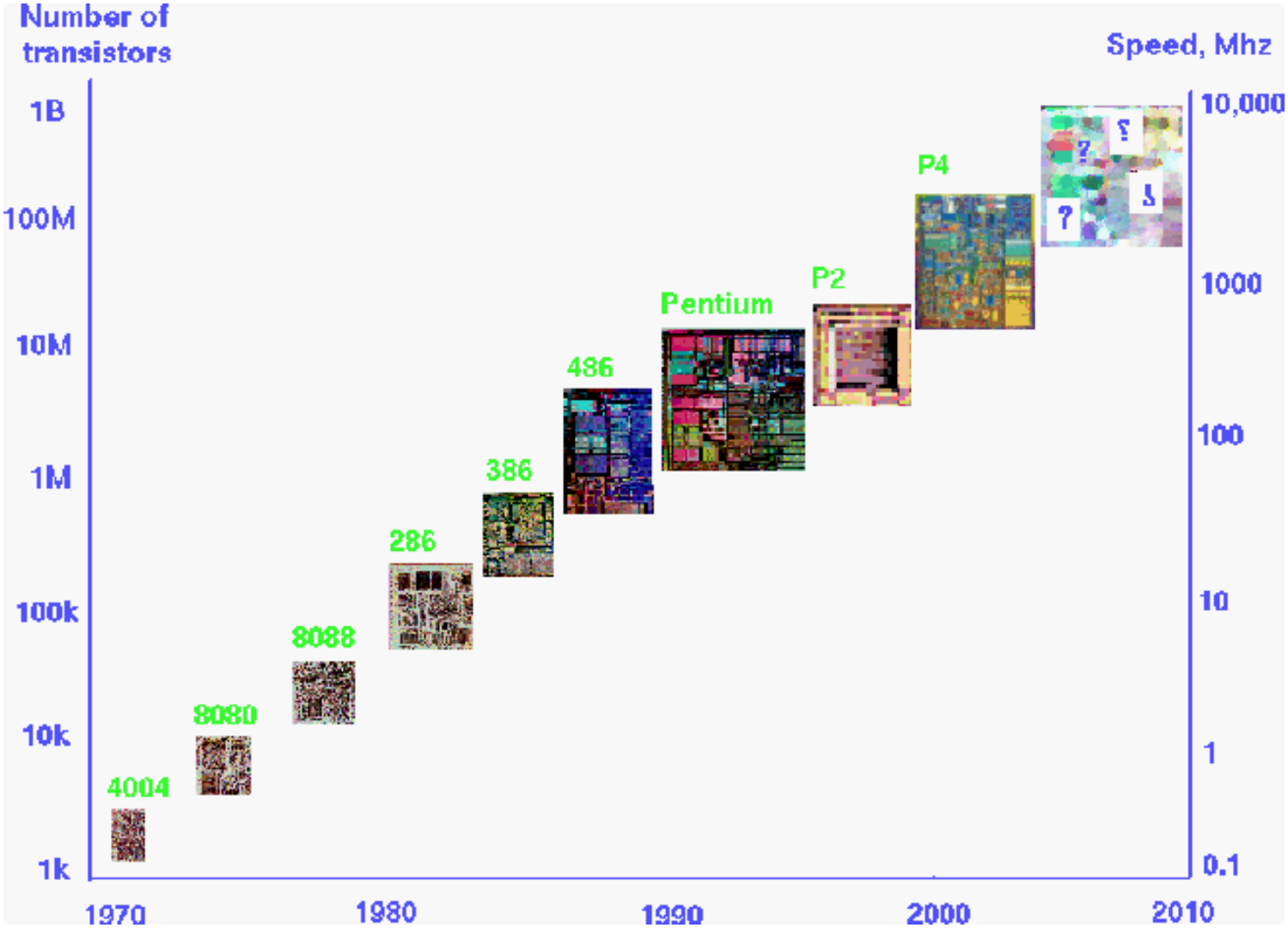
Sources: NASA Gov. ; SEMI

Motivation: Moore's Law
Complexity Growth of VLSI circuits

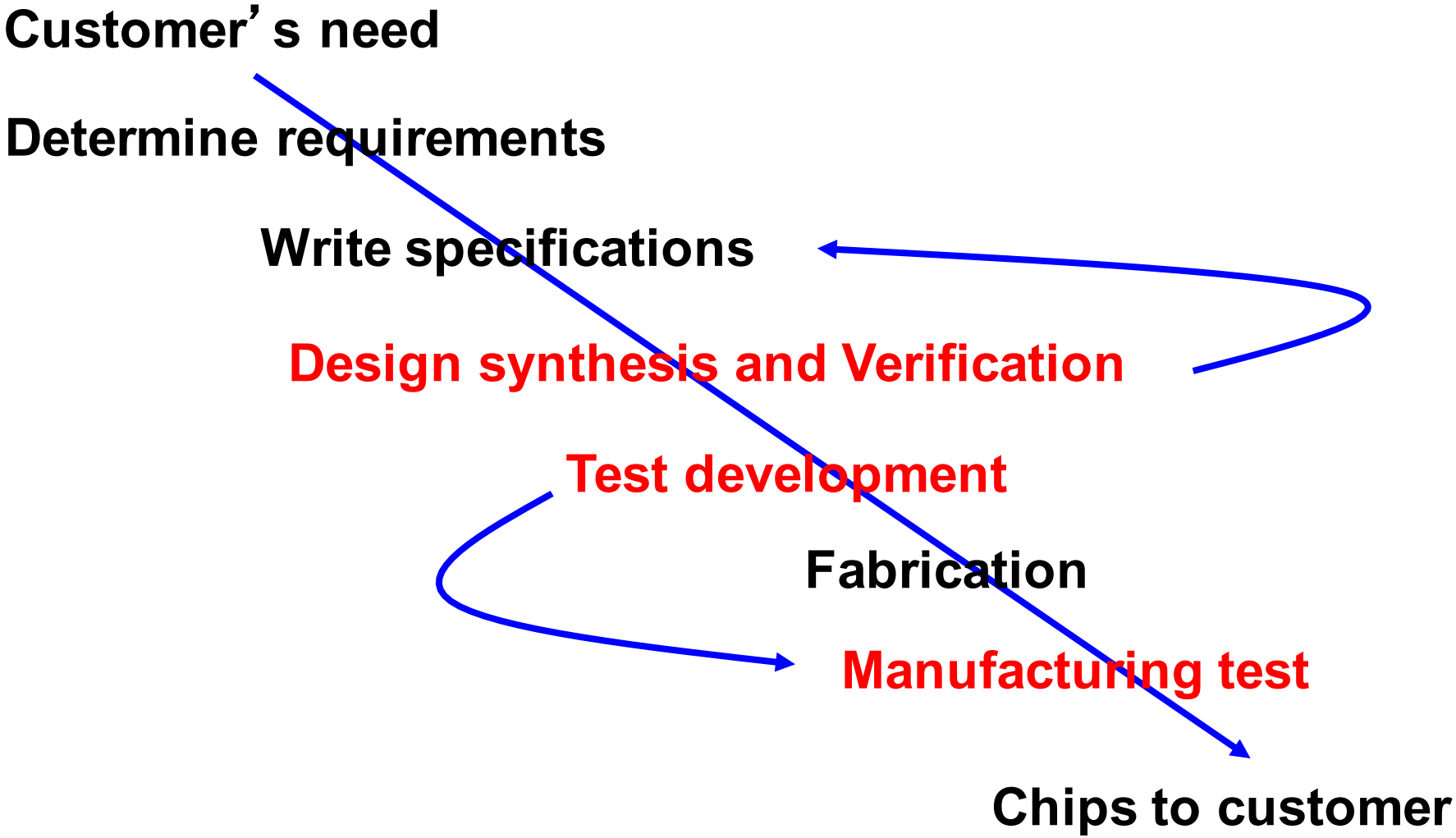


Source (Copp, *Int. AOC EW Conf., 2002*)

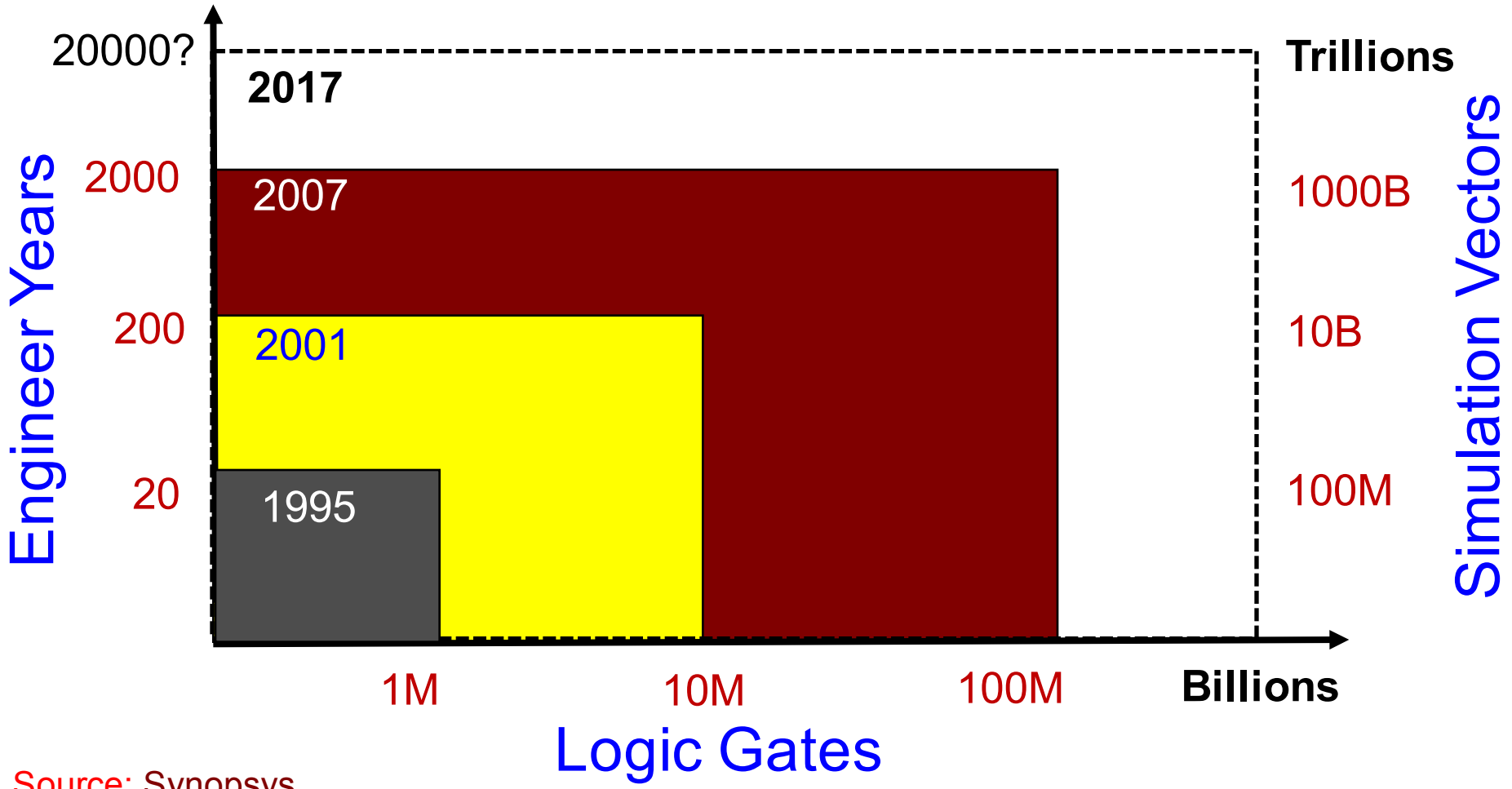
Design Complexity



VLSI Realization Process

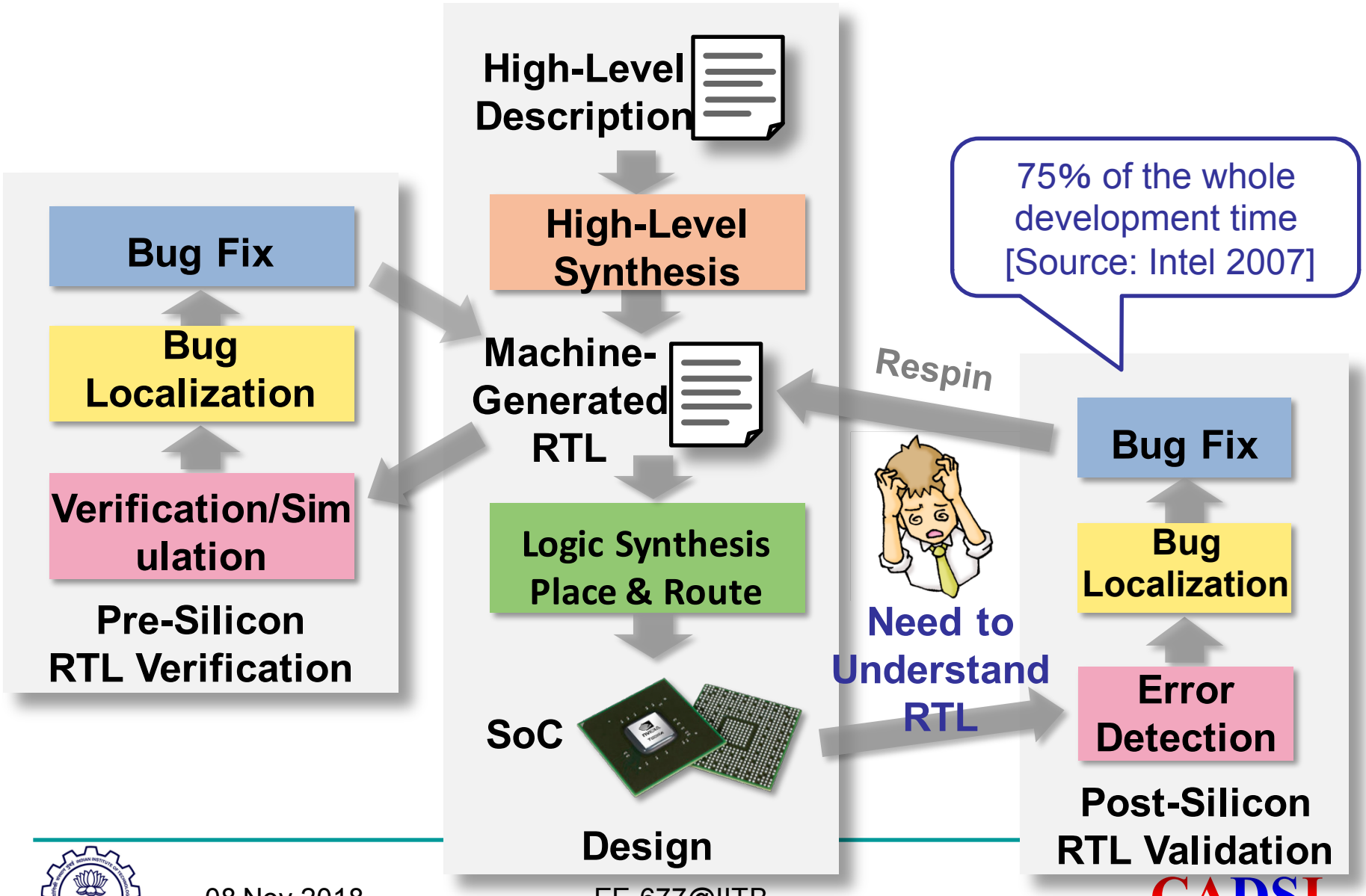


Design Validation Complexity

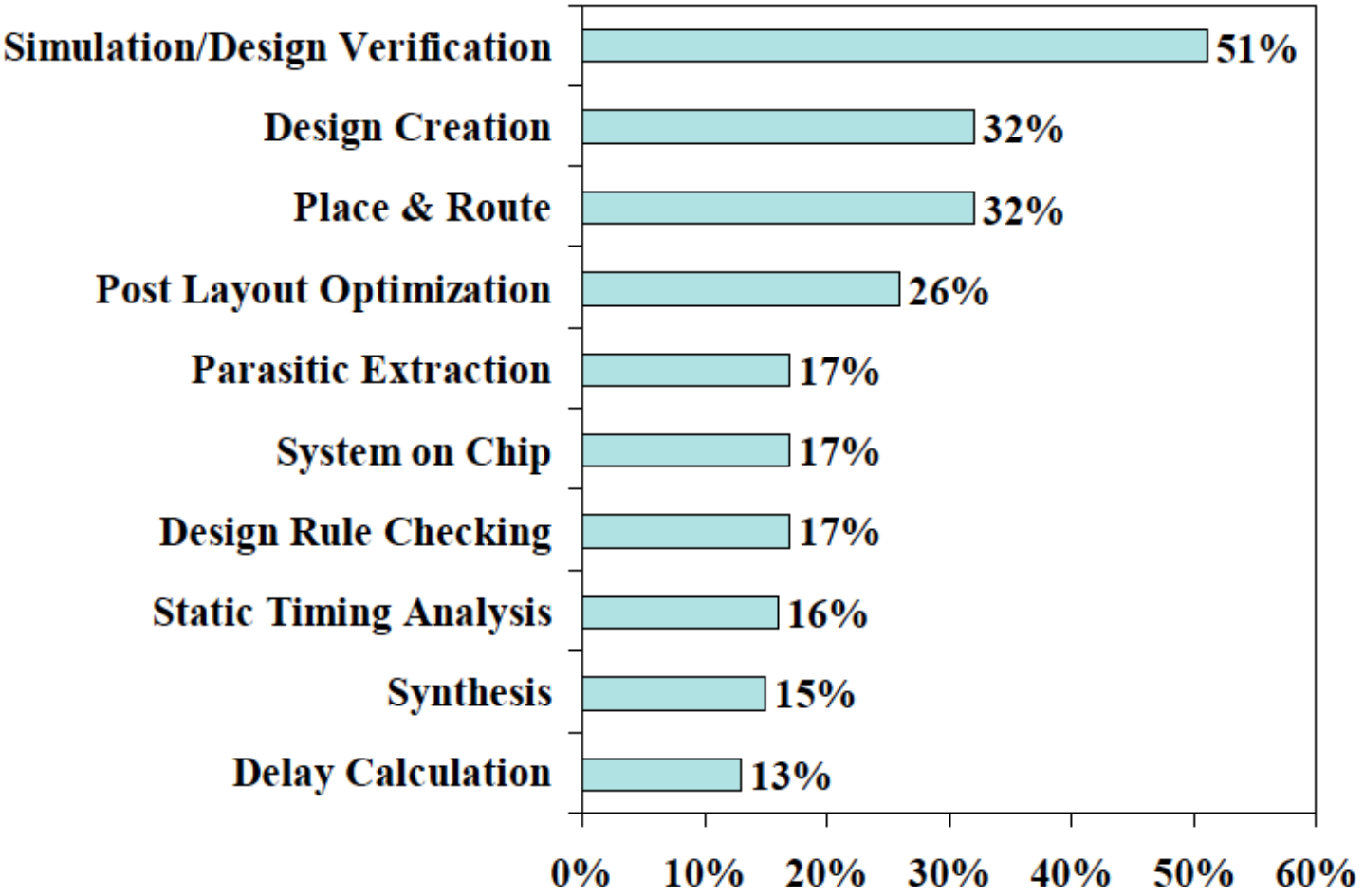


Source: Synopsys

Conventional SoC Design Flow

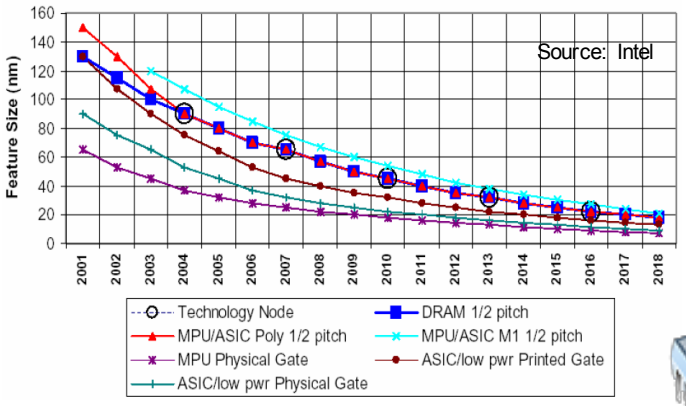


Verification challenge

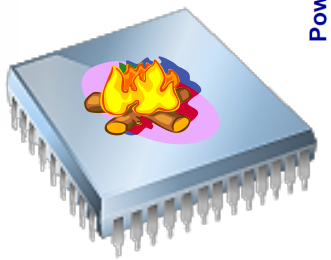


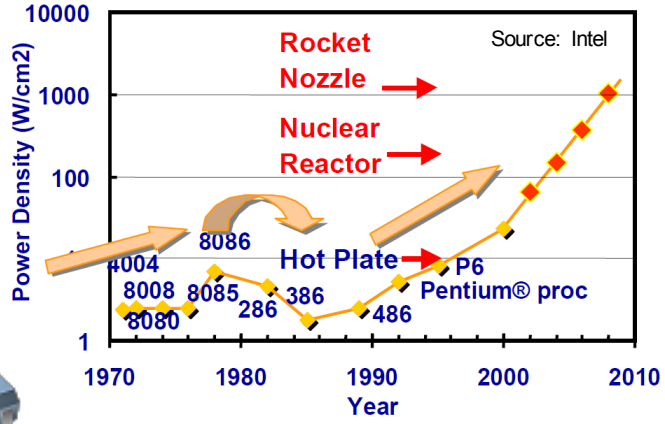
Bottlenecks in Design Cycles:
Survey of 545 engineers by EETIMES 2000

Challenges under deep submicron technologies

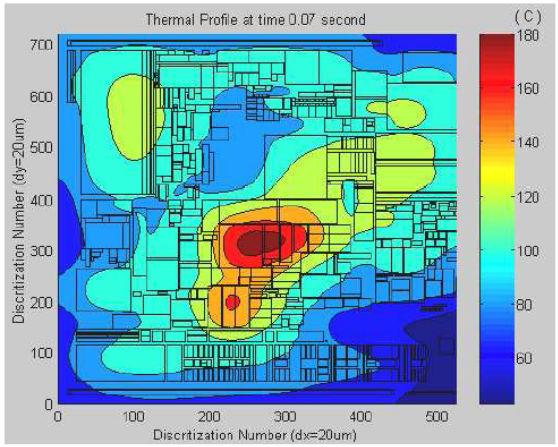


Chip size decreases



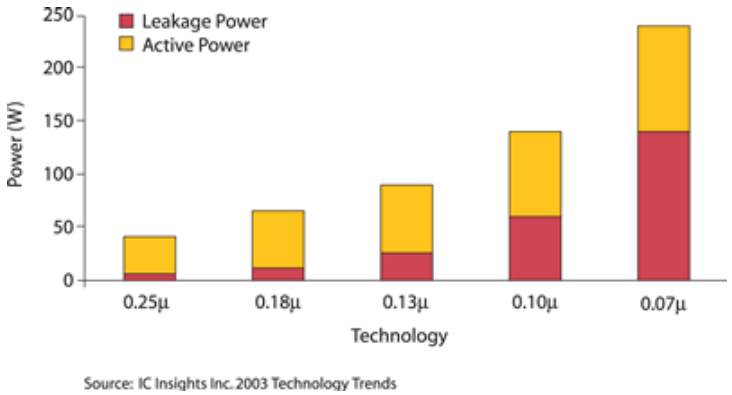


Power density increases



Source: Wang et al. ISPD2003

Chip becomes hotter



Source: IC Insights Inc. 2003 Technology Trends

Leakage power make it worse

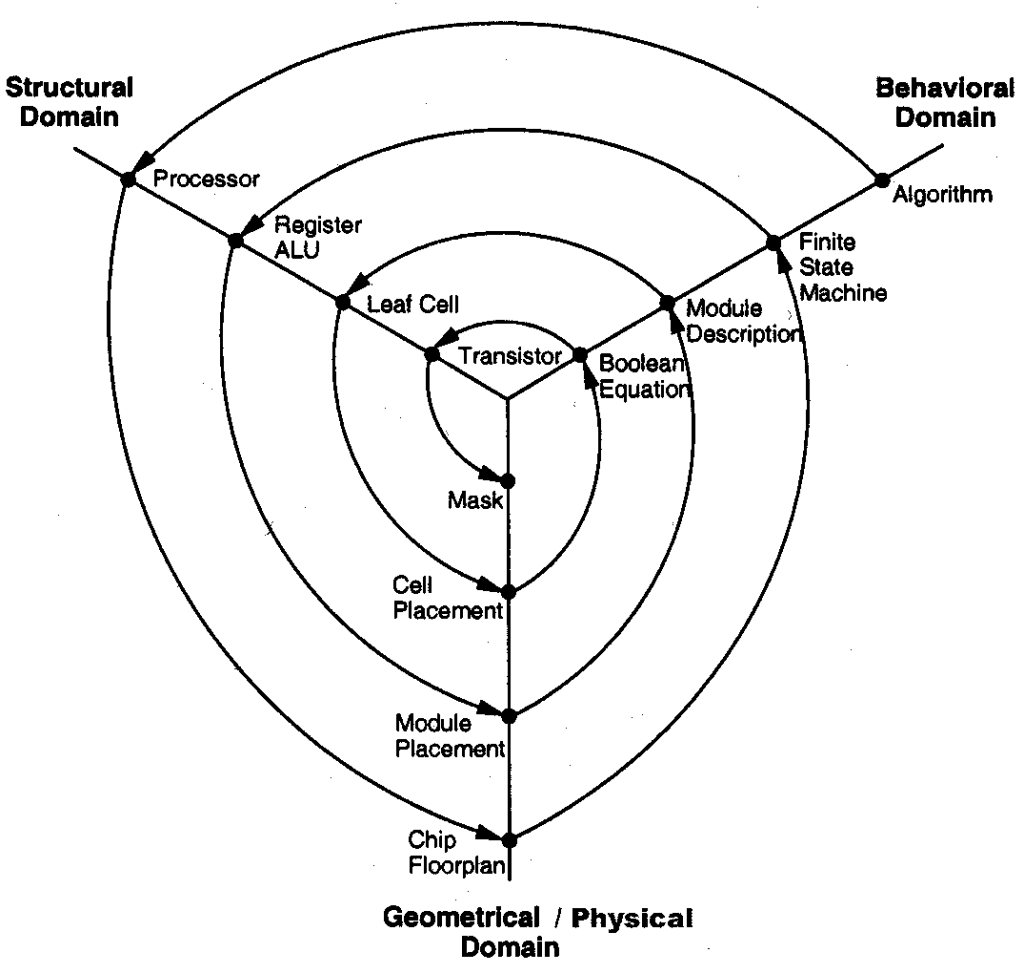
Coping with Complexity

- How to design System-on-Chip?
  - Billions of transistors
  - Tens to hundreds of engineers
- Structured Design
- Design Partitioning

Structured Design

- **Hierarchy**: Divide and Conquer
  - Recursively system into modules
- **Regularity**
  - Reuse modules wherever possible
  - Ex: Standard cell library
- **Modularity**: well-formed interfaces
  - Allows modules to be treated as black boxes
- **Locality**
  - Physical and temporal

Gajski Y-Chart



Architectural Synthesis

Architectural Level Abstraction

- Datapath
- Controller

Architectural Synthesis

- Constructing the macroscopic structure of a digital circuit starting from behavioural models that can be captured from Data flow or Sequencing Graph

Architectural Synthesis

Computation: Differential Equation Solver

$Y'' + 3 x y' + 3y = 0$

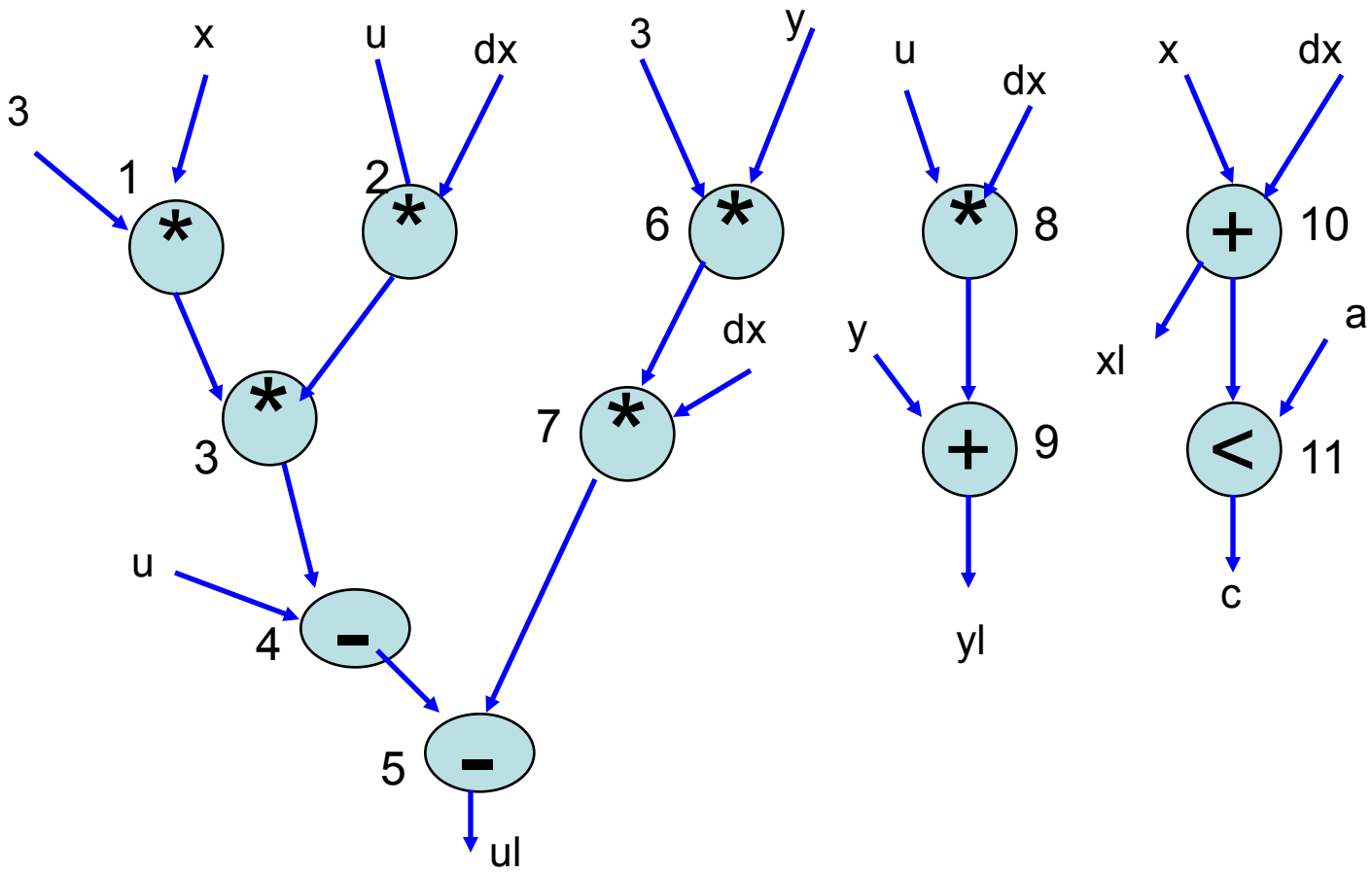
$X(0) = 0$

$y(0) = y$

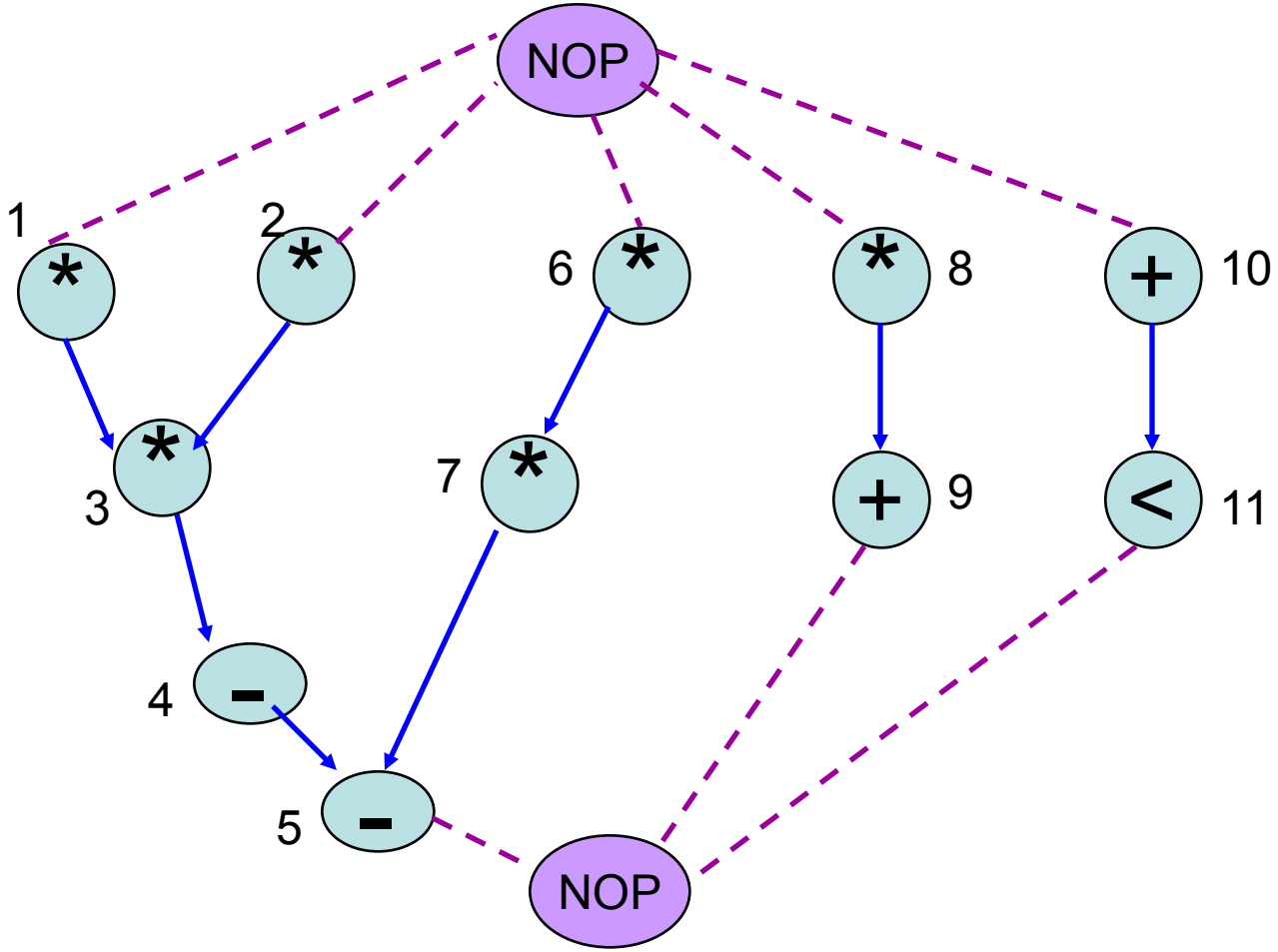
$y'(0) = u$

```
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 = xl < a
            x = xl; u = ul; y = yl;
       until (c);
       write (y)
}
```

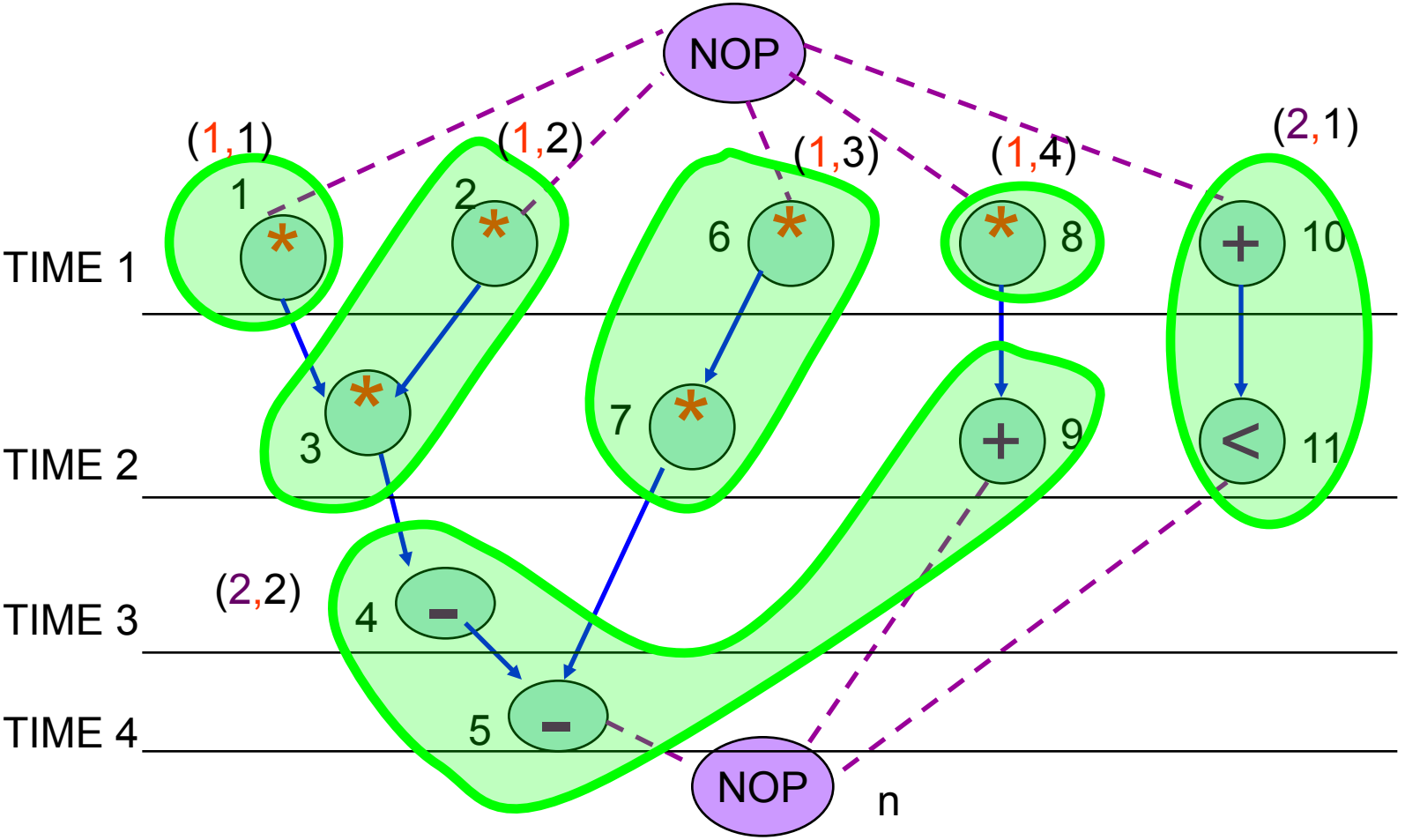
Data Flow Graph



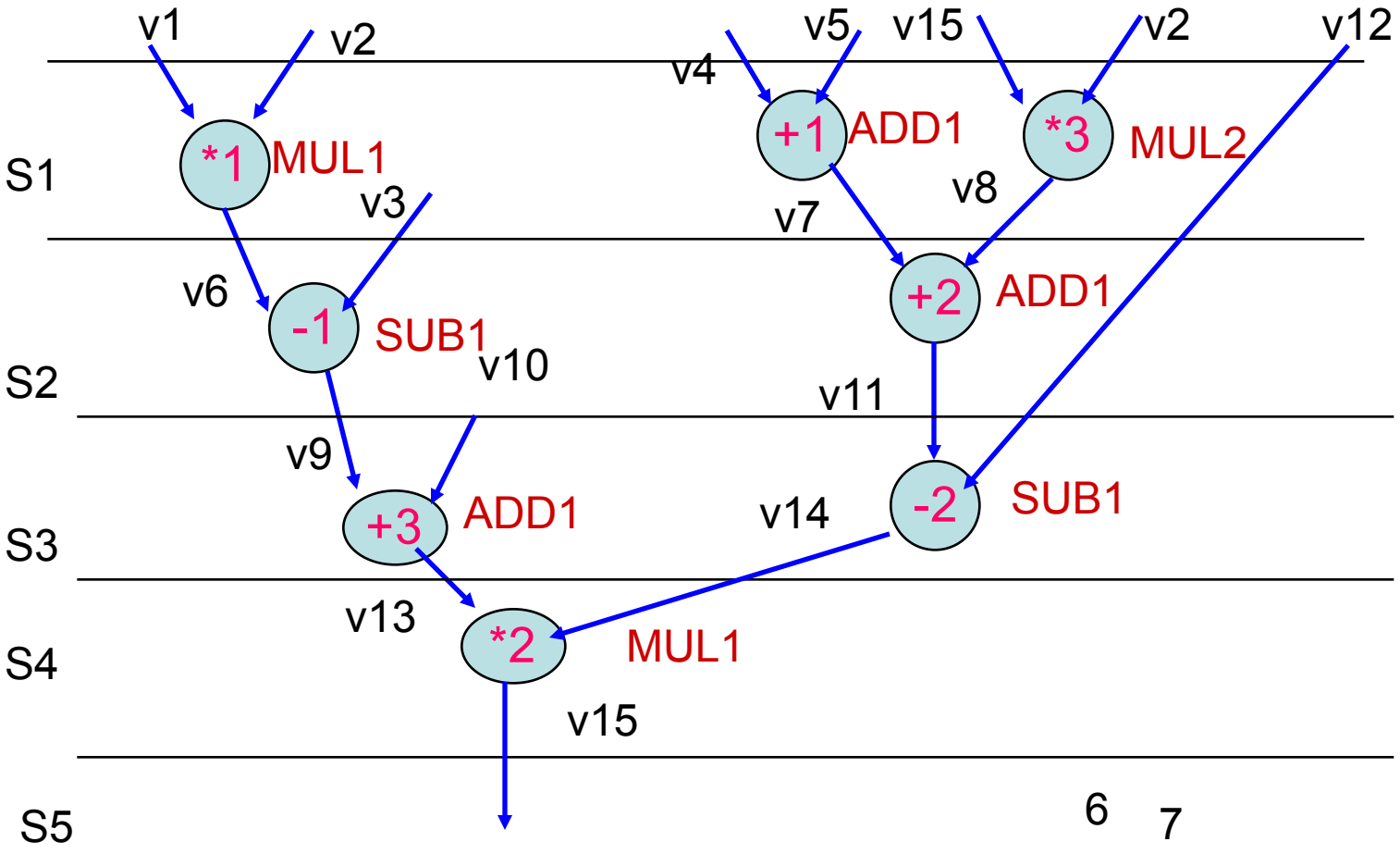
Sequencing Graph



Spatial Domain: Binding



Scheduled DFG



Variable Assignment

Register	Assignment1
R1	V1, V7, V11, V13
R2	V2, V8, V10, V14
R3	V3, V5, V9
R4	V4, V6
R5	V12
R6	V15

Switching Activity

S1	V1	*1	V2 V15	*3	V2	X	V12 V4	+1	V5
S2	V7	X	V8	X	V8 V6	-1	V3 V7	+2	V8
S3	V11	X	V10	X	V10 V11	-2	V12 V9	+3	V10
S4	V13	*2	V14	X	V14 V13	X		X	V14
S5									
		MUL 1		mu l2		Sub1		Add 1	

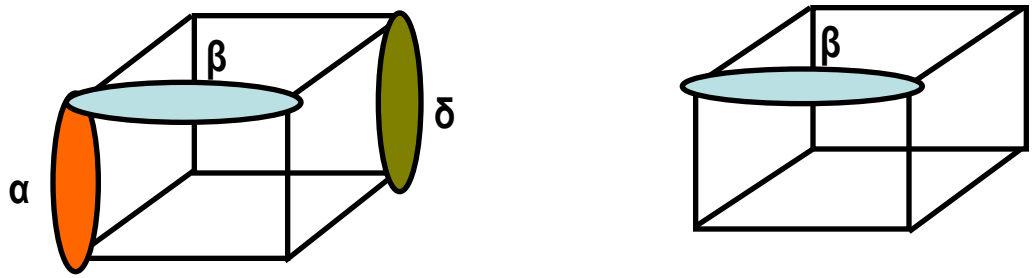
Variable Assignment

Register	Assignment2
R1	V1, V13
R2	V2
R3	V4, V8, V10
R4	V5, V7, V9
R5	V12
R6	V3
R7	V6, V11
R8	V14
R9	V15

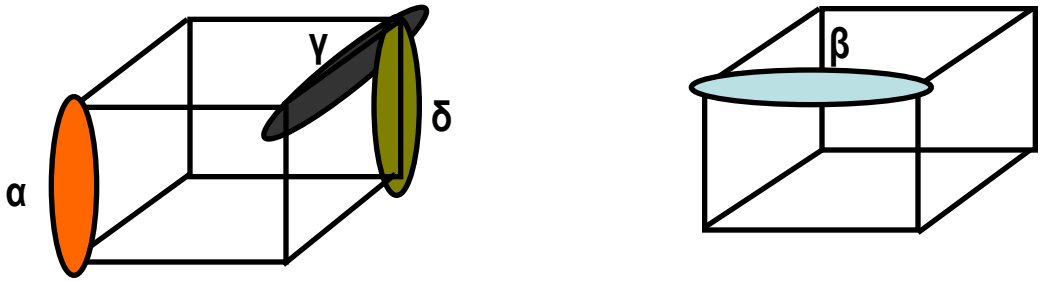
2-Level Logic Optimization

- $f_1 = a'b'c' + a'b'c + ab'c + abc + abc'$; $f_2 = a'b'c + ab'c$

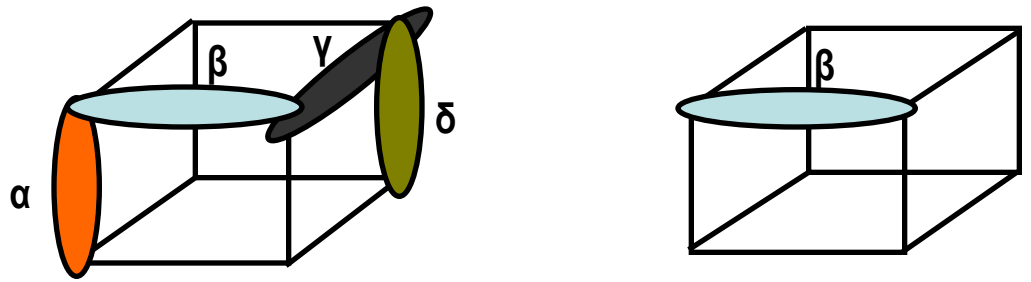
Minimum cover



Irredundant cover



Minimal cover w.r. to single implicant containment



f1 f2

Heuristic minimization - operators

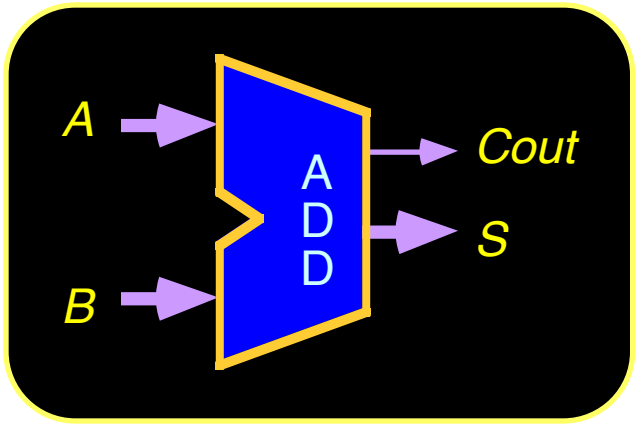
- Expand
  - Make implicants prime
  - Removed covered implicants
- Reduce
  - Reduce size of each implicant while preserving cover
- Reshape
  - Modify implicant pairs: enlarge one and reduce the other
- Irredundant
  - Make cover irredundant

Rough comparison of minimizers

- MINI
  - Iterate EXPAND, REDUCE, RESHAPE
- Espresso
  - Iterate EXPAND, IRREDUNDANT, REDUCE
- Espresso guarantees an irredundant cover
  - Because of the irredundant operator
- MINI may return irredundant covers, but can guarantee only minimality w.r.to single implicant containment

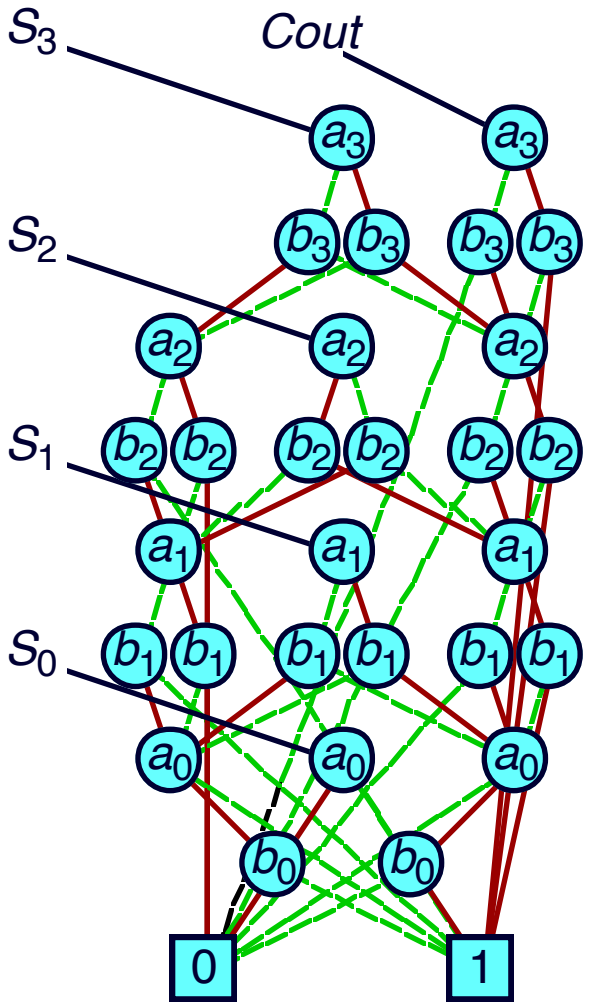
Representing Circuit Functions

- Functions
  - All outputs of 4-bit adder
  - Functions of data inputs

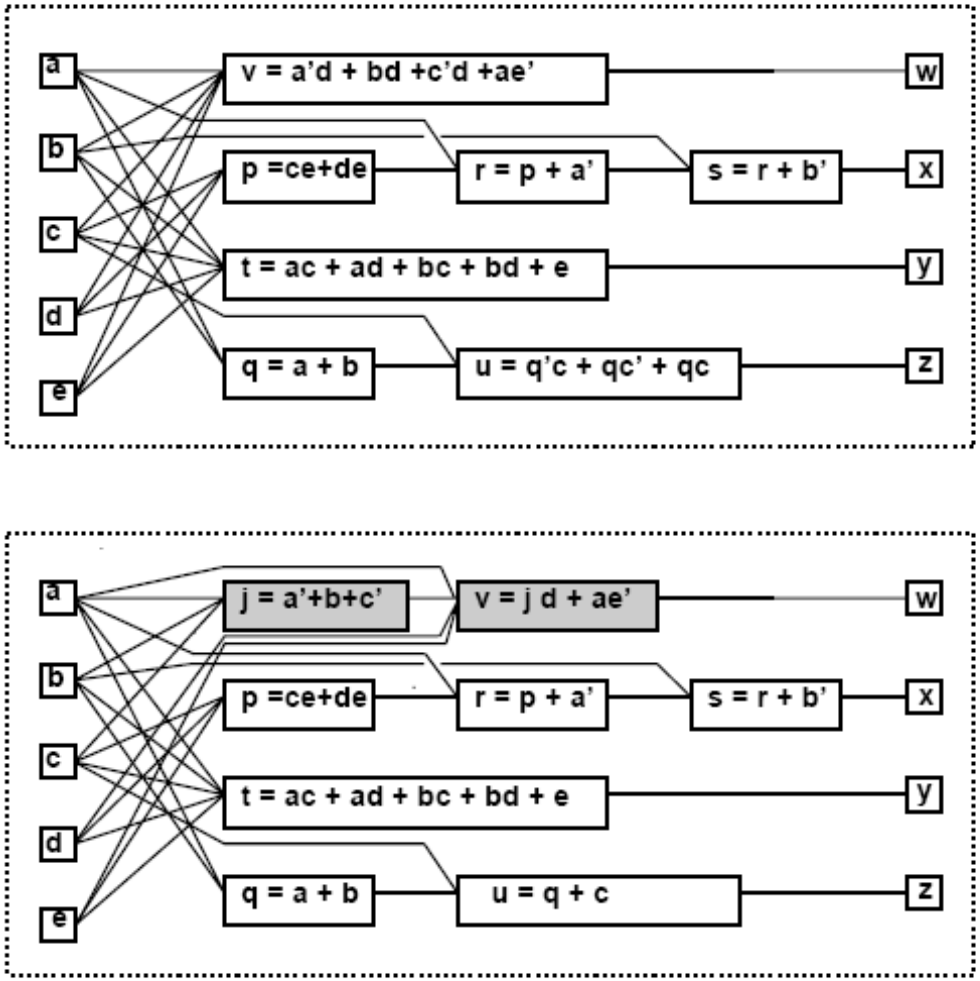


- Shared Representation
  - Graph with multiple roots
  - 31 nodes for 4-bit adder
  - 571 nodes for 64-bit adder

☛ *Linear growth*



Multi-level logic minimization



Boolean and algebraic methods

- Boolean methods for multilevel synthesis
  - Exploit properties of Boolean functions
  - Use *don't care* conditions
  - Computationally intensive
- Algebraic methods
  - Use polynomial abstraction of logic function
  - Simpler, faster, weaker
  - Widely used

Theorem: Brayton and McMullen

- Two expressions f_a and f_b have a common multiple-cube divisor f_d if and only if
  - There exist kernels k_a in $K(f_a)$ and k_b in $K(f_b)$ such that f_d is the sum of two (or more) cubes in $k_a \cap k_b$
- Consequences
  - If kernel intersection is void, then the search for common sub-expression can be dropped
  - If an expression has no kernels, it can be dropped from consideration
  - The kernel intersection is the basis for constructing the expression to extract

State Minimization Procedure

1. Partion the states of M into subsets s.t. all states in same subset are *1-equivalent*
2. Two states are 2-equivalent iff they are 1-equivalent and their I_i successors, for all possible I_i, are also 1-equivalent

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

P0 = (ABCDEF)

P1 = (ACE), (BDF)

P2 = (ACE), (BD), (F)

P3 = (AC), (E), (BD), (F)

P4 = (AC), (E), (BD), (F)

FSM Traversal - Algorithm

- Traverse the product machine M(X,S,δ, λ,O)
  - start at an initial state S_0
  - iteratively compute symbolic image $Img(S_0,R)$ (set of *next states*):

$$\mathrm{Im}\, g(S_0, R) = \exists_x \exists_s S_0(s).R(x,s,t)$$

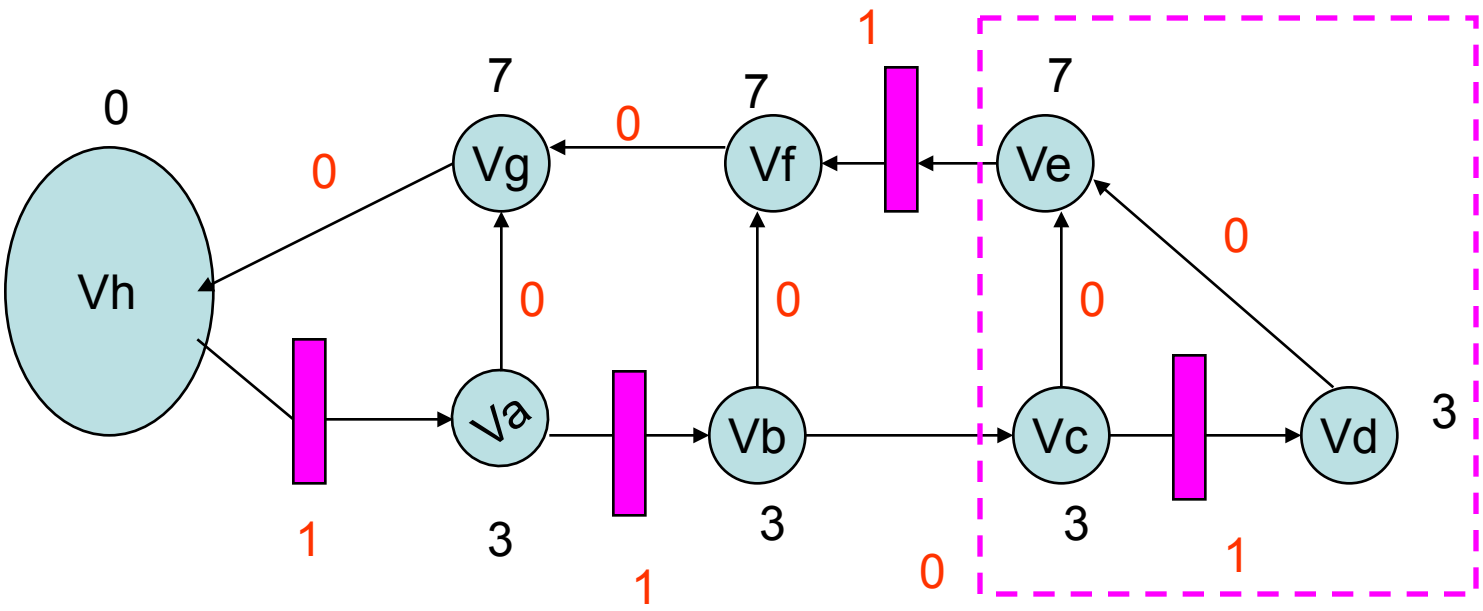
$$R = \prod_i R_i = \prod_i (t_i \equiv \delta_i(s,x))$$

until an *error state* is reached

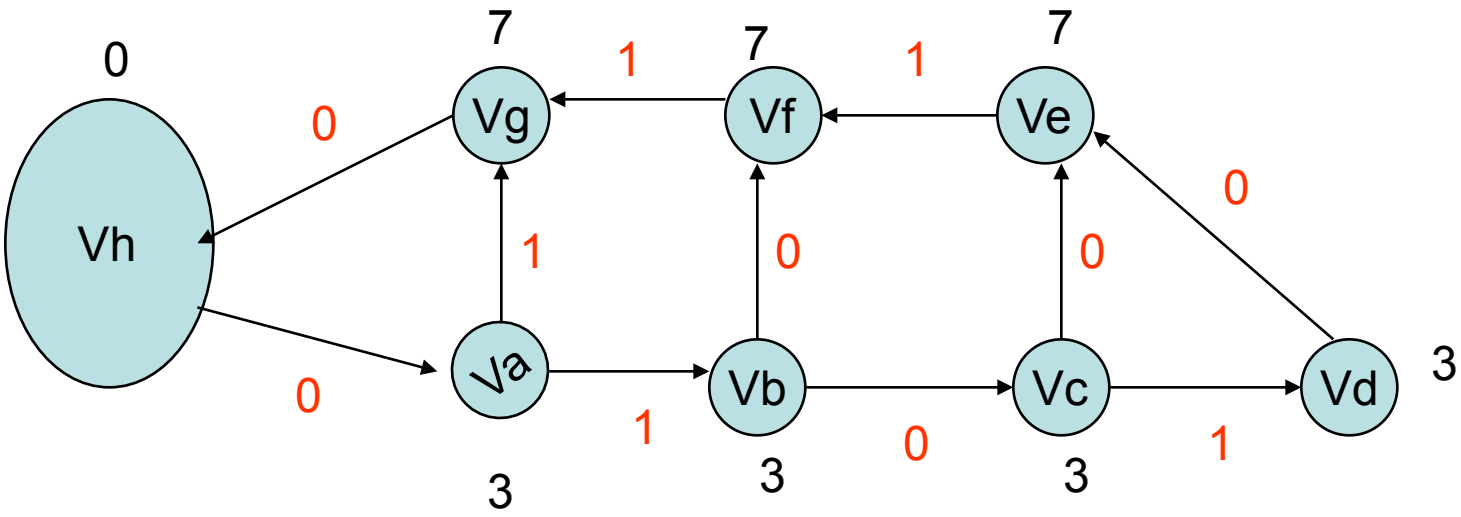
  - transition relation R_i for each next state variable t_i can be computed as $t_i = (t \otimes \delta(s,x))$

(this is an alternative way to compute transition relation, when design is specified at gate level)

Retiming

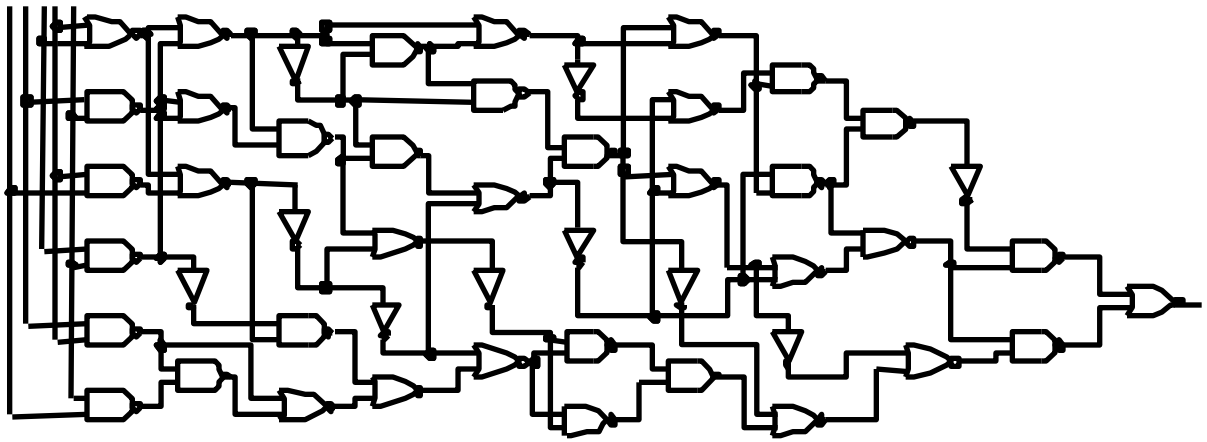


Retiming



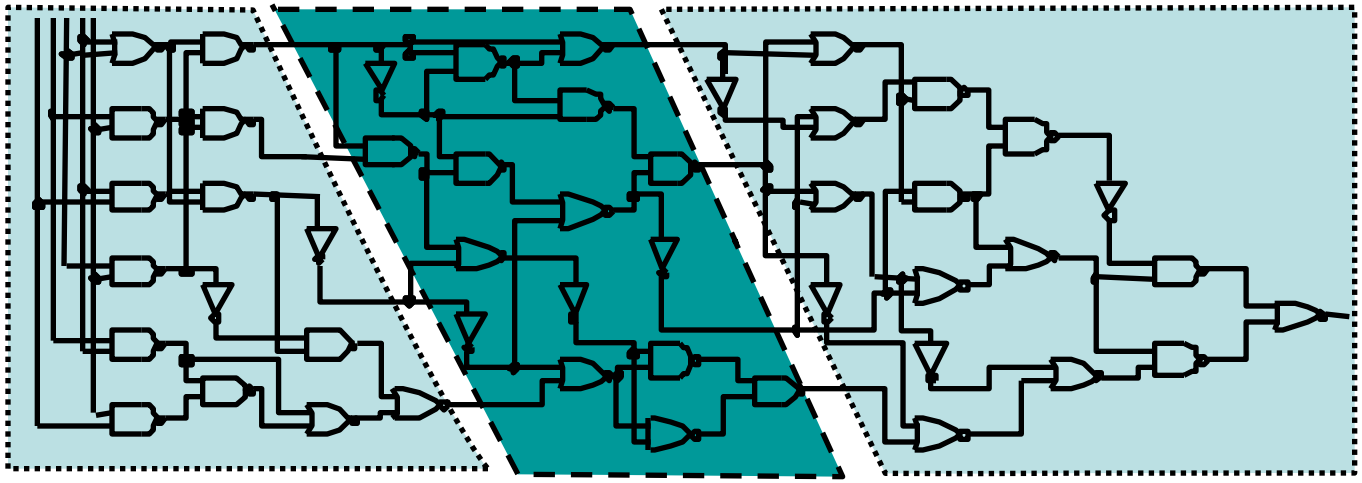
Partitioning of a circuit

Input size: 48



Cut 1=4 Cut 2=4

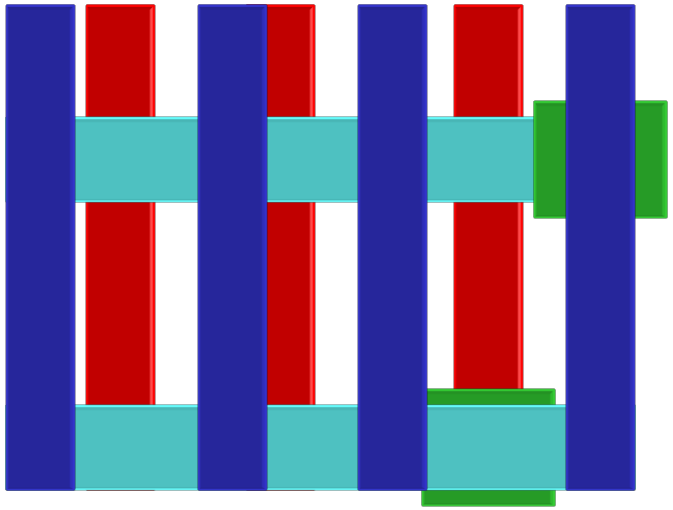
Size 1=15 Size 2=16 Size 3=17



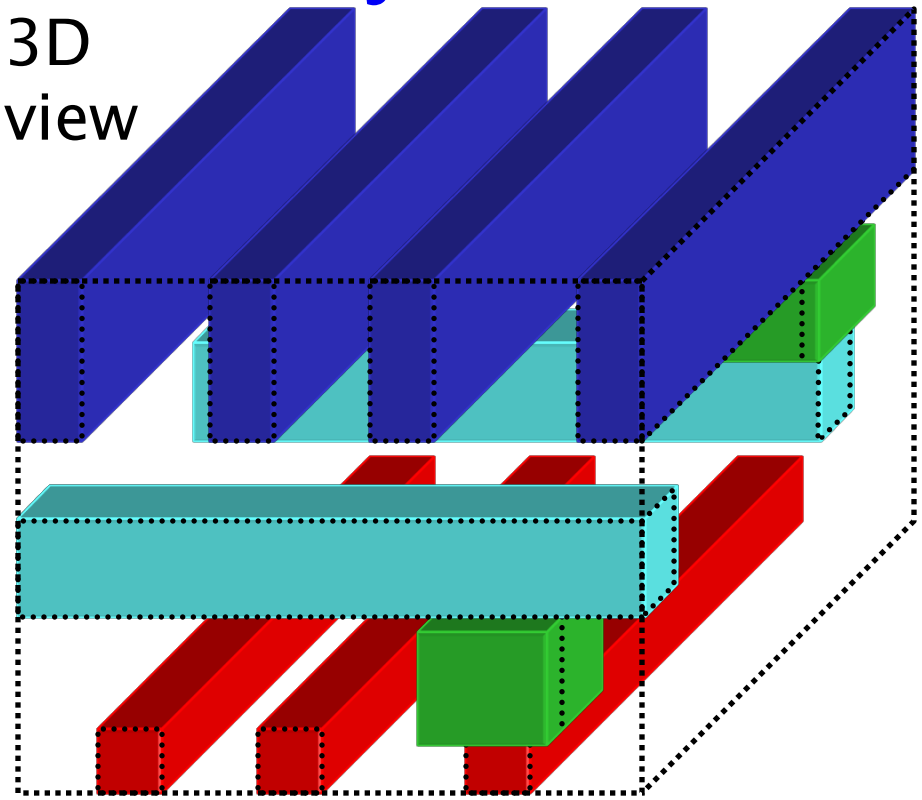
[©Sherwani]

Routing Anatomy

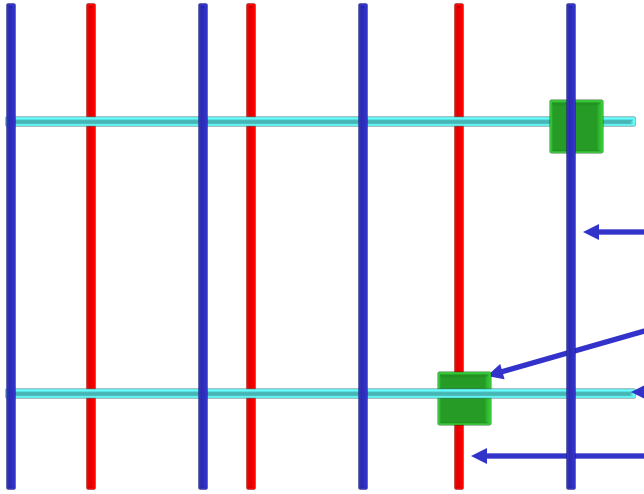
Top view



3D view



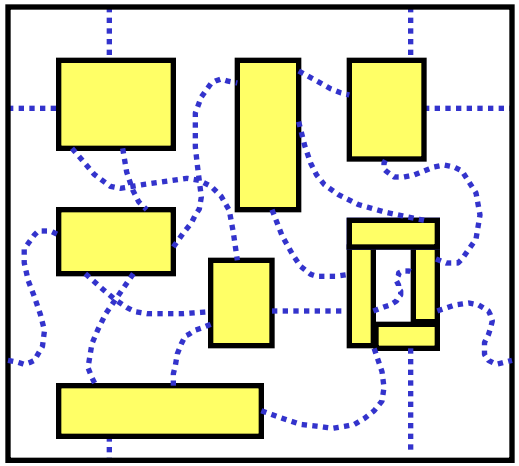
Symbolic Layout



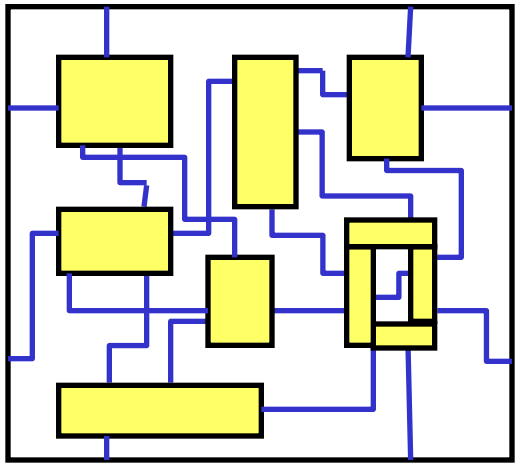
Note: Colors used in this slide are not standard

Global vs. Detailed Routing

- Global routing
  - Input: detailed placement, with exact terminal locations
  - Determine "channel" (routing region) for each net
  - Objective: minimize area (congestion), and timing (approximate)



- Detailed routing
  - Input: channels and approximate routing from the global routing phase
  - Determine the exact route and layers for each net
  - Objective: valid routing, minimize area (congestion), meet timing constraints
  - Additional objectives: min via, power



[©Sherwani]

Computer Aided Design of VLSI Systems

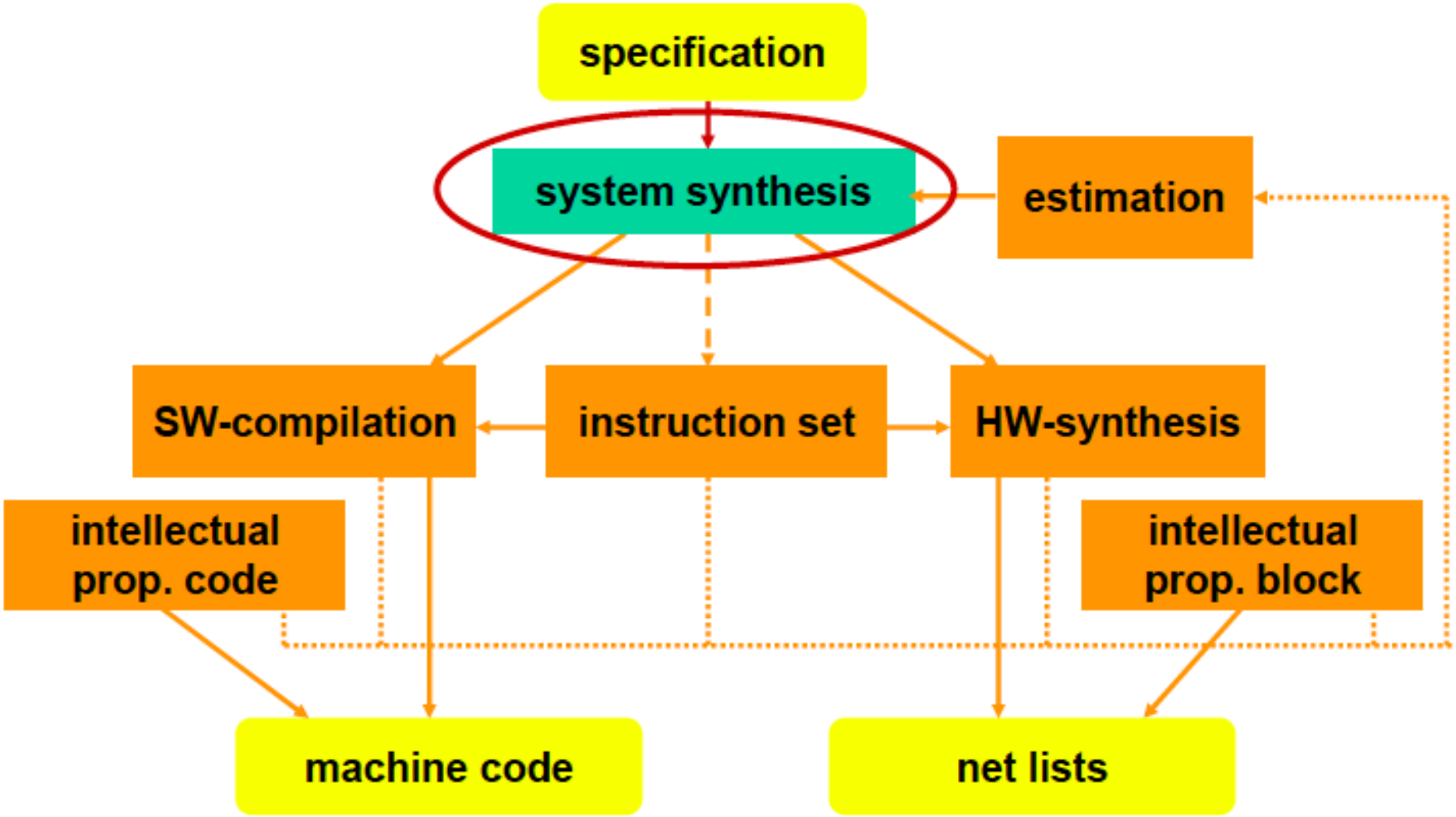
Current Topics

Courses

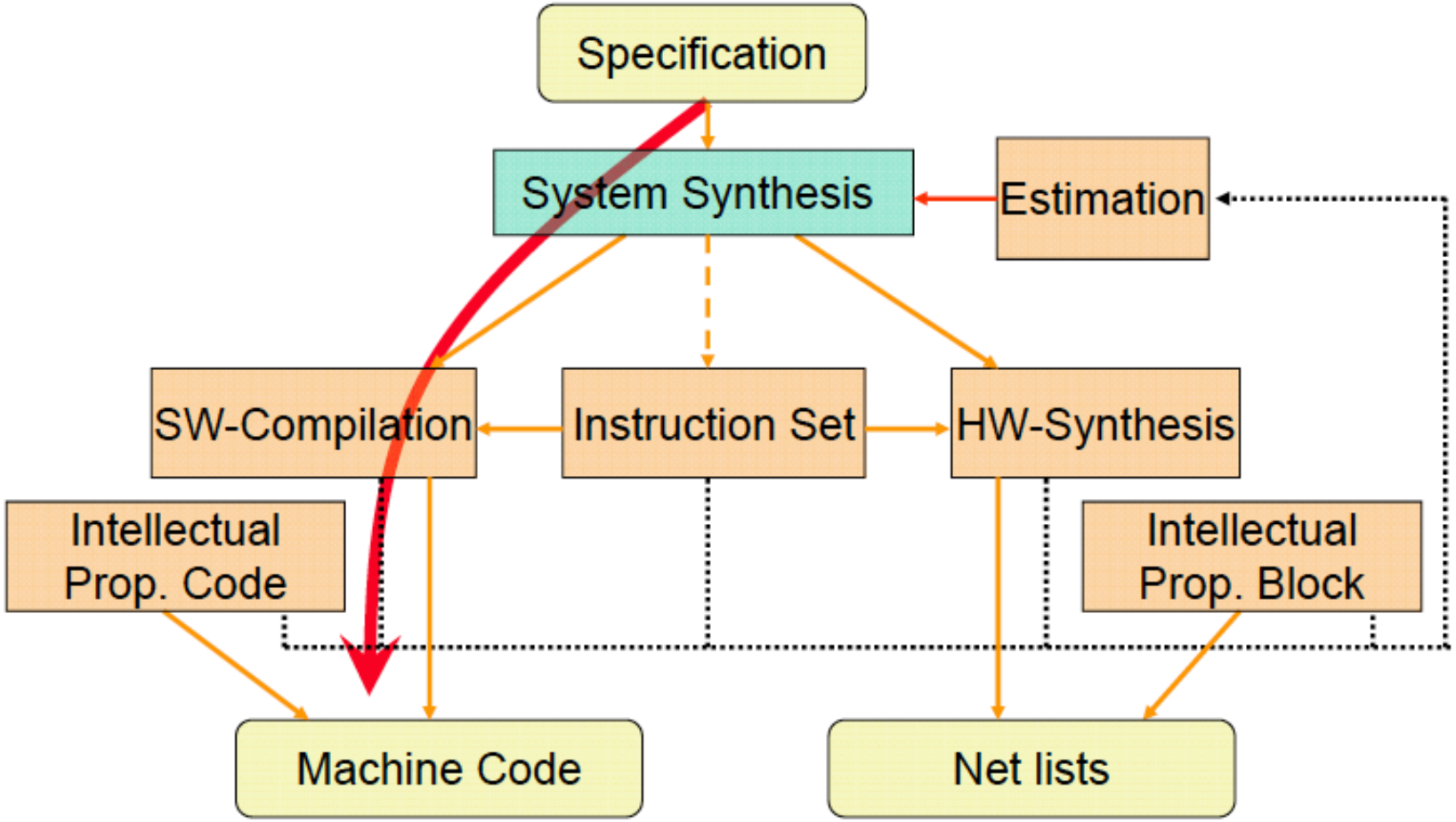
- EE-709: Testing and Verification of VLSI Circuits

- CS-654: Current Topics in VLSI and System Design

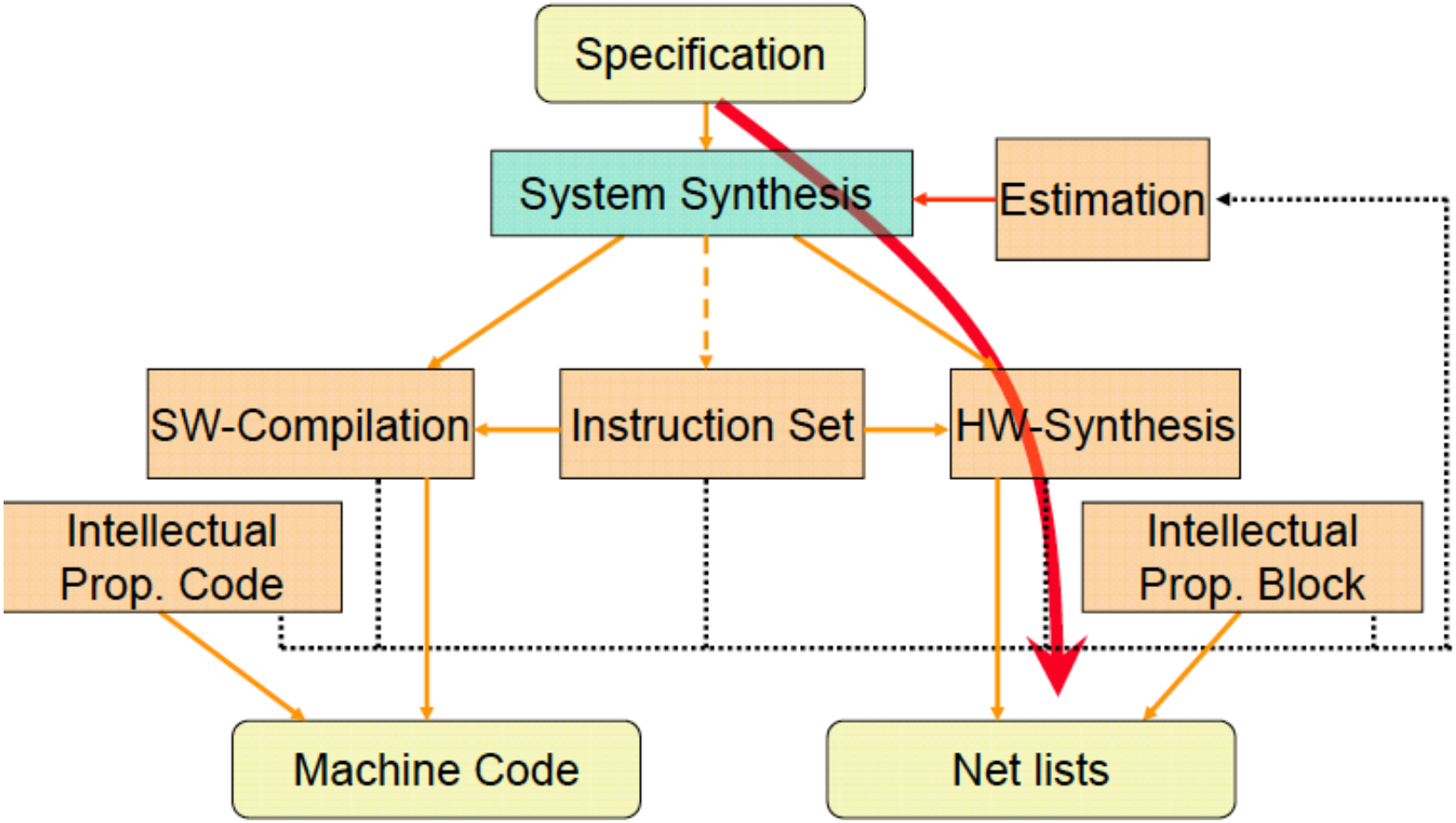
System Level Design



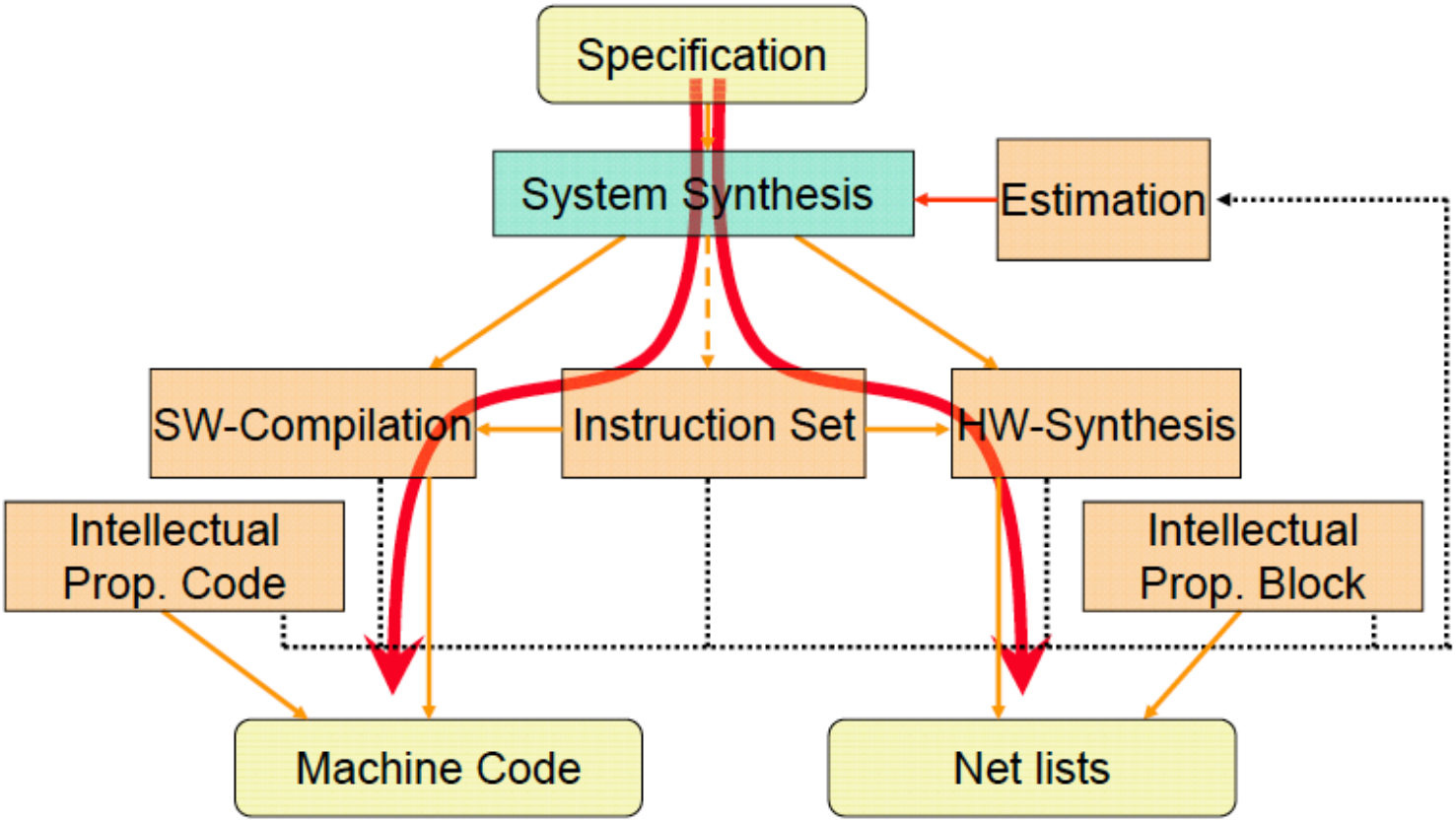
Fixed Processor Architecture



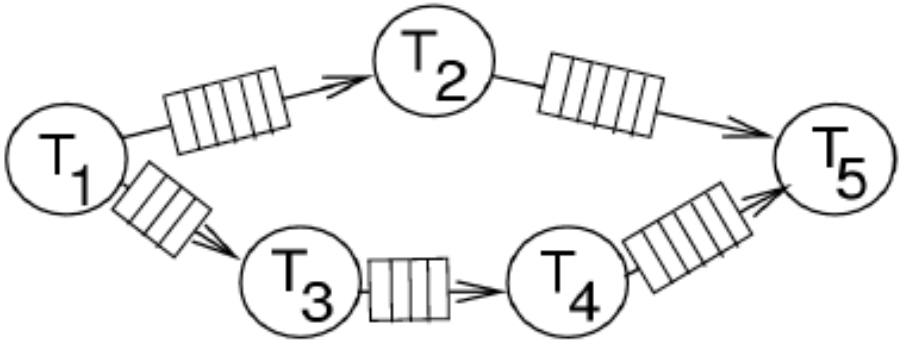
Application Specific HW Design



HW/SW Co-design



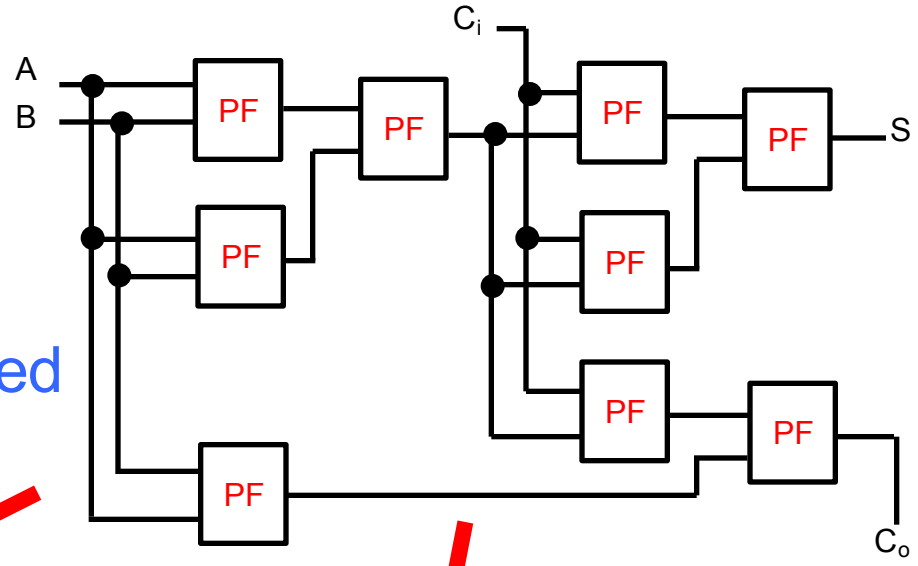
Kahn Process Network



- Proposed by Kahn in 1974 as a general-purpose scheme for parallel programming:
  - ***read***: destructive and blocking (reading an empty channel blocks until data is available)
  - ***write***: non-blocking
  - ***FIFO***: infinite size
- Unique attribute: ***determinate***

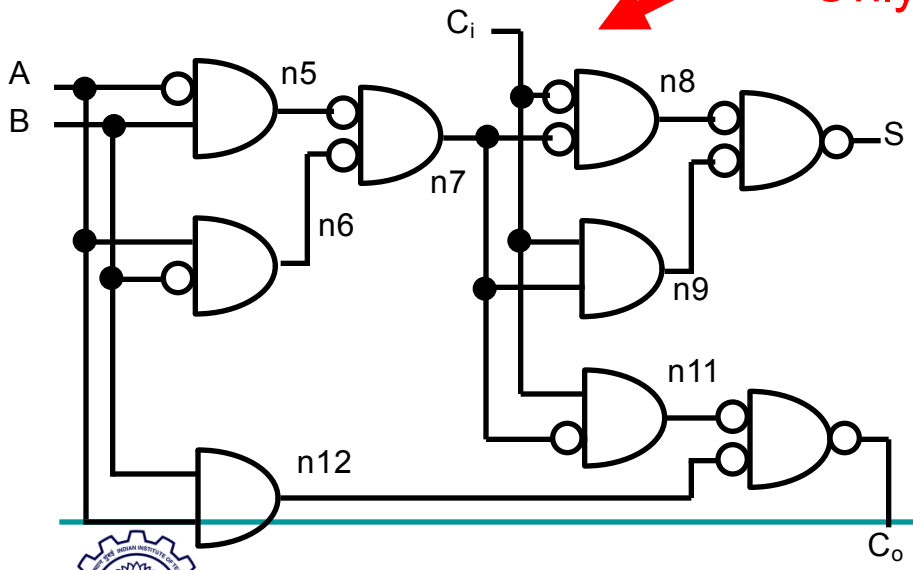
Logic synthesis on fixed topology

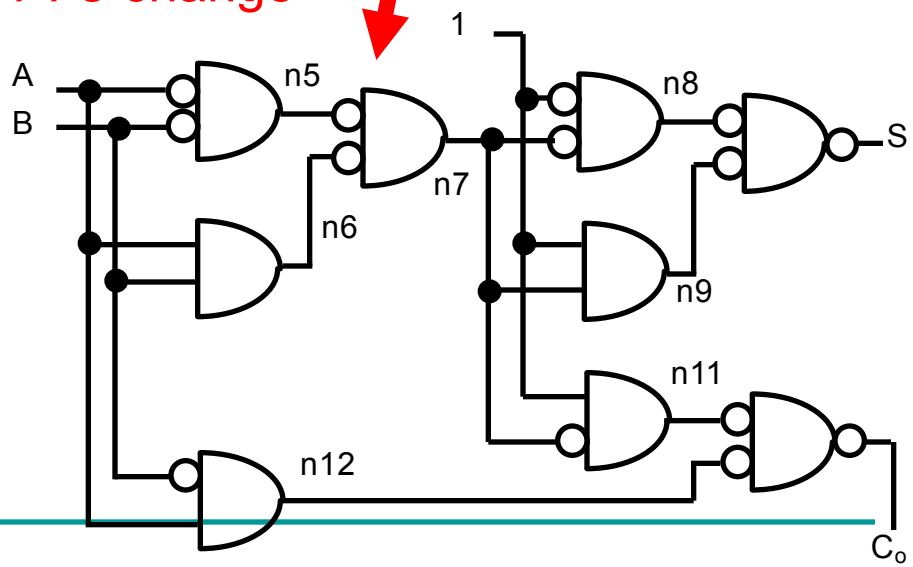
- Reconfigurable circuits with fixed topology



Wiring is fixed

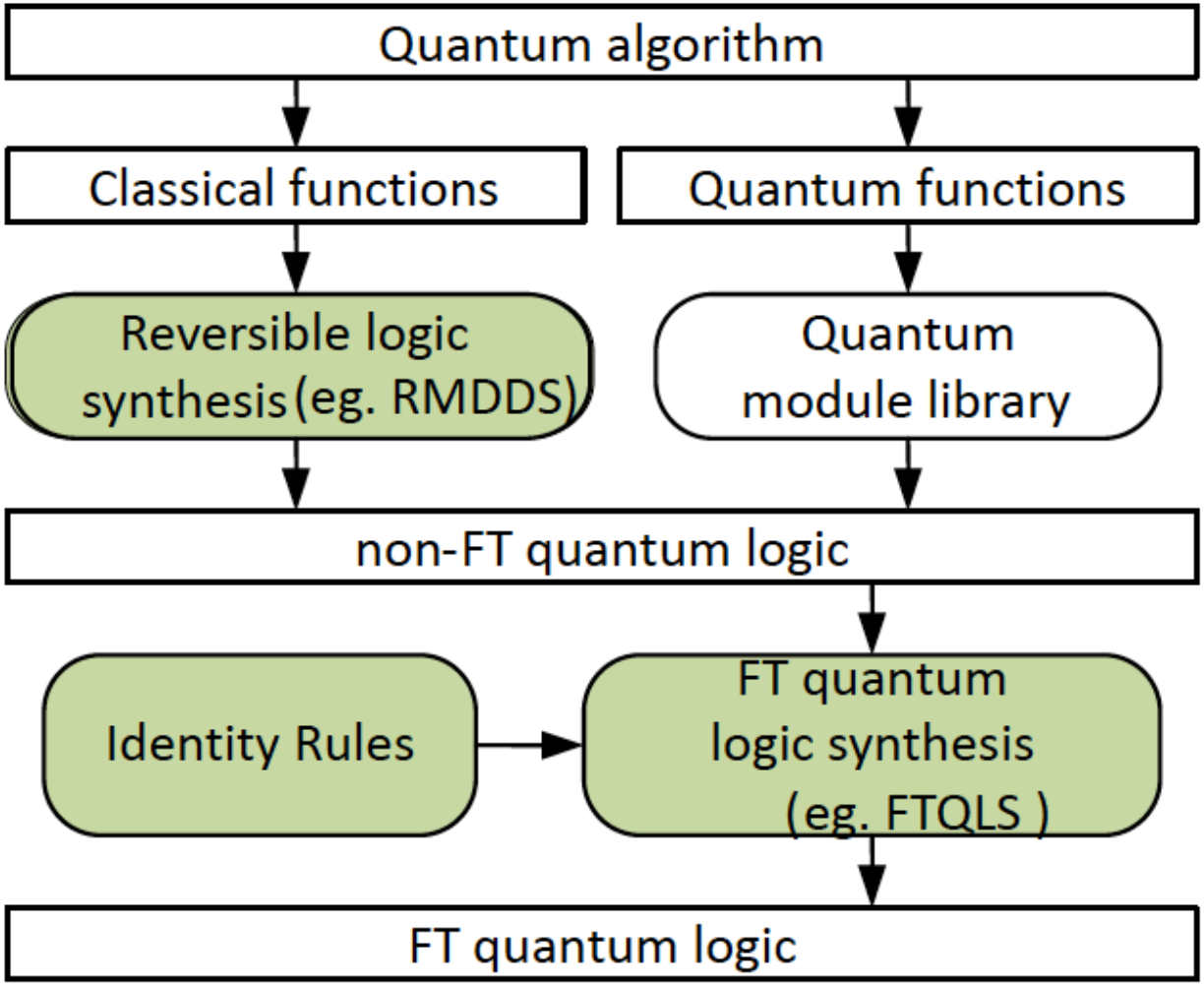
Only PFs change





CADSL

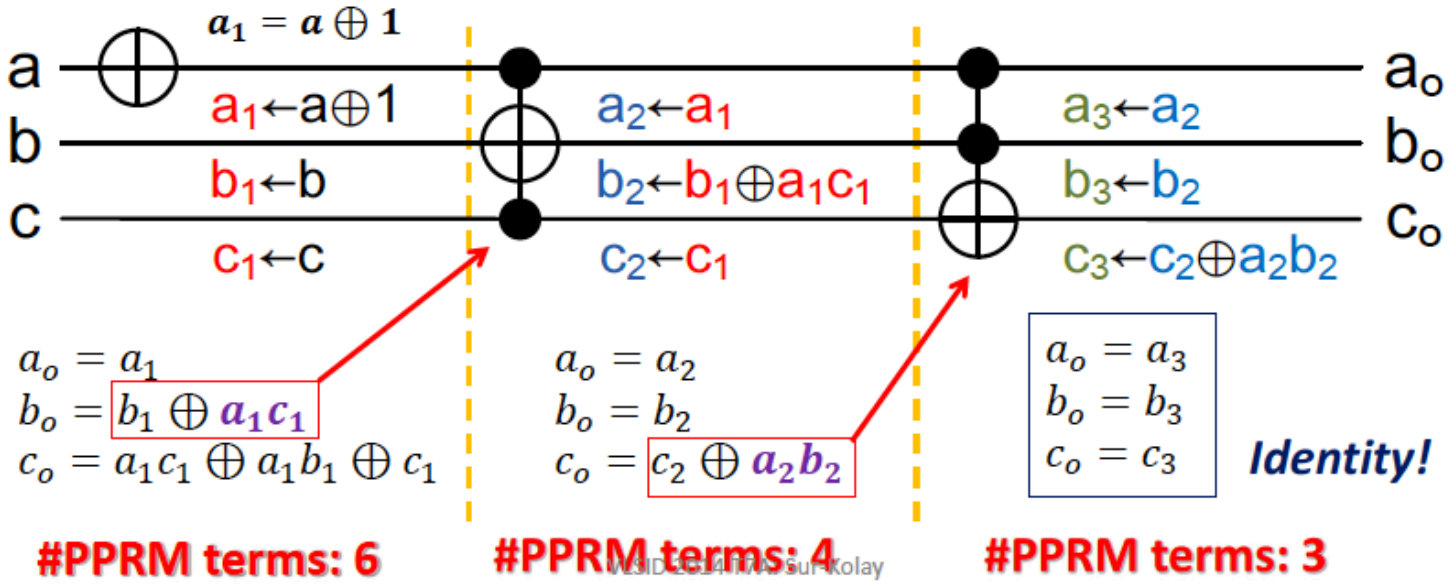
Quantum circuit synthesis



RM Reversible Circuit Synthesis

- Synthesis complete when:
  - Input/output mapping reduces to ***identity***
  - #PPRM terms = #inputs

$$a_o = a \oplus 1$$
$$b_o = ac \oplus b \oplus c$$
$$c_o = ac \oplus ab \oplus b$$



Approximate Computing

Task: Division

$$is \frac{923}{21} > 1.75?$$

$$is \frac{923}{21} > 45.27?$$

$$\frac{923}{21} = ?$$

Computation Accuracy Required to answer

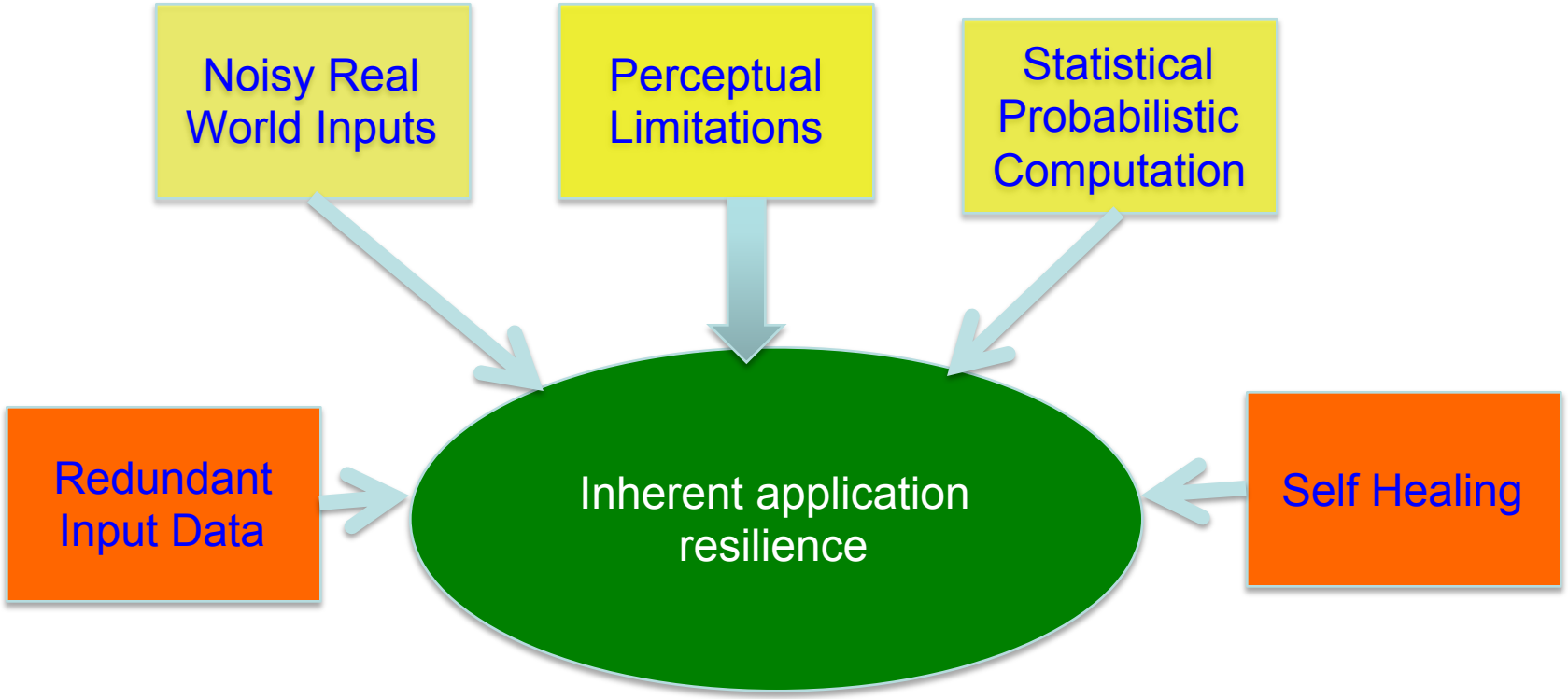
Different Context

Different Effort

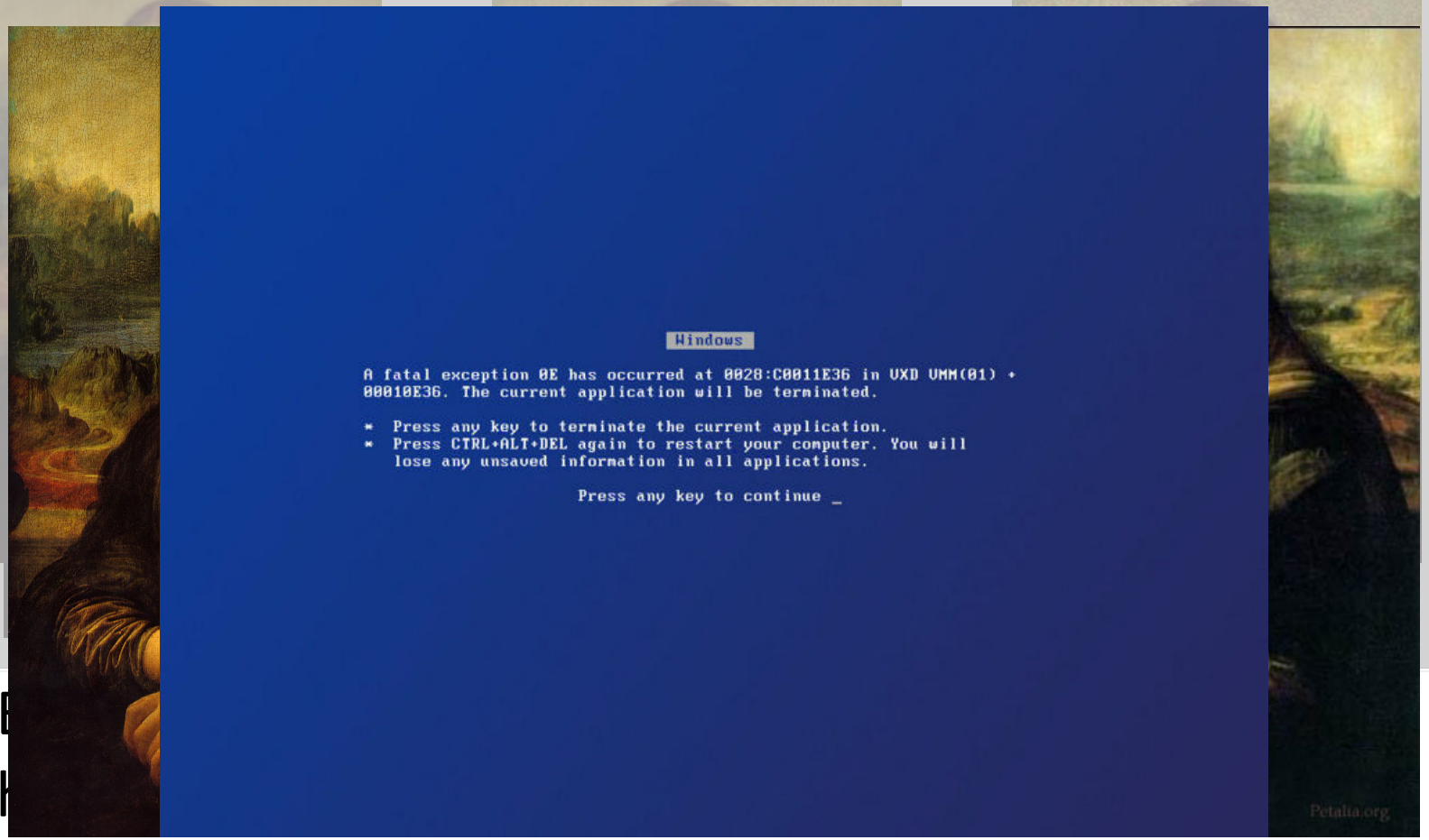
High effort → more accurate
Low effort → less accurate

2

Inherent Application Resilience: Source



Quality is Important

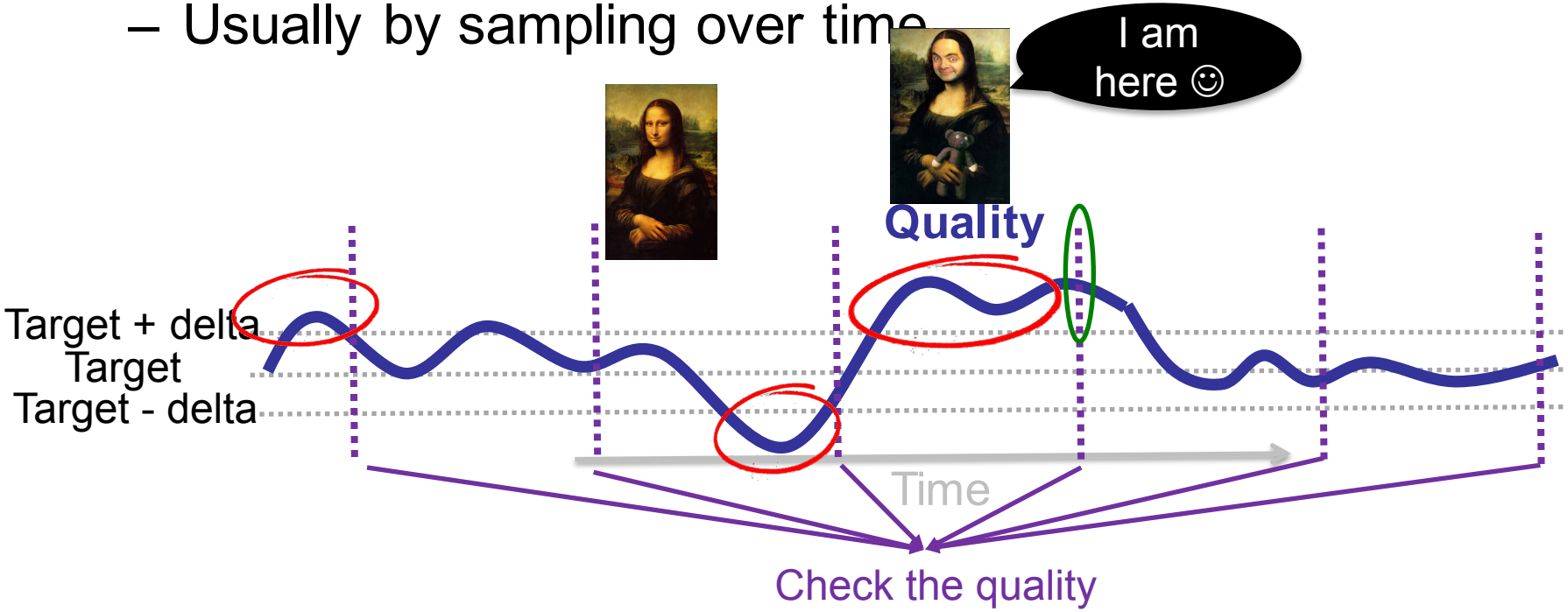


- B
  K

CADSL

Approximation Challenge

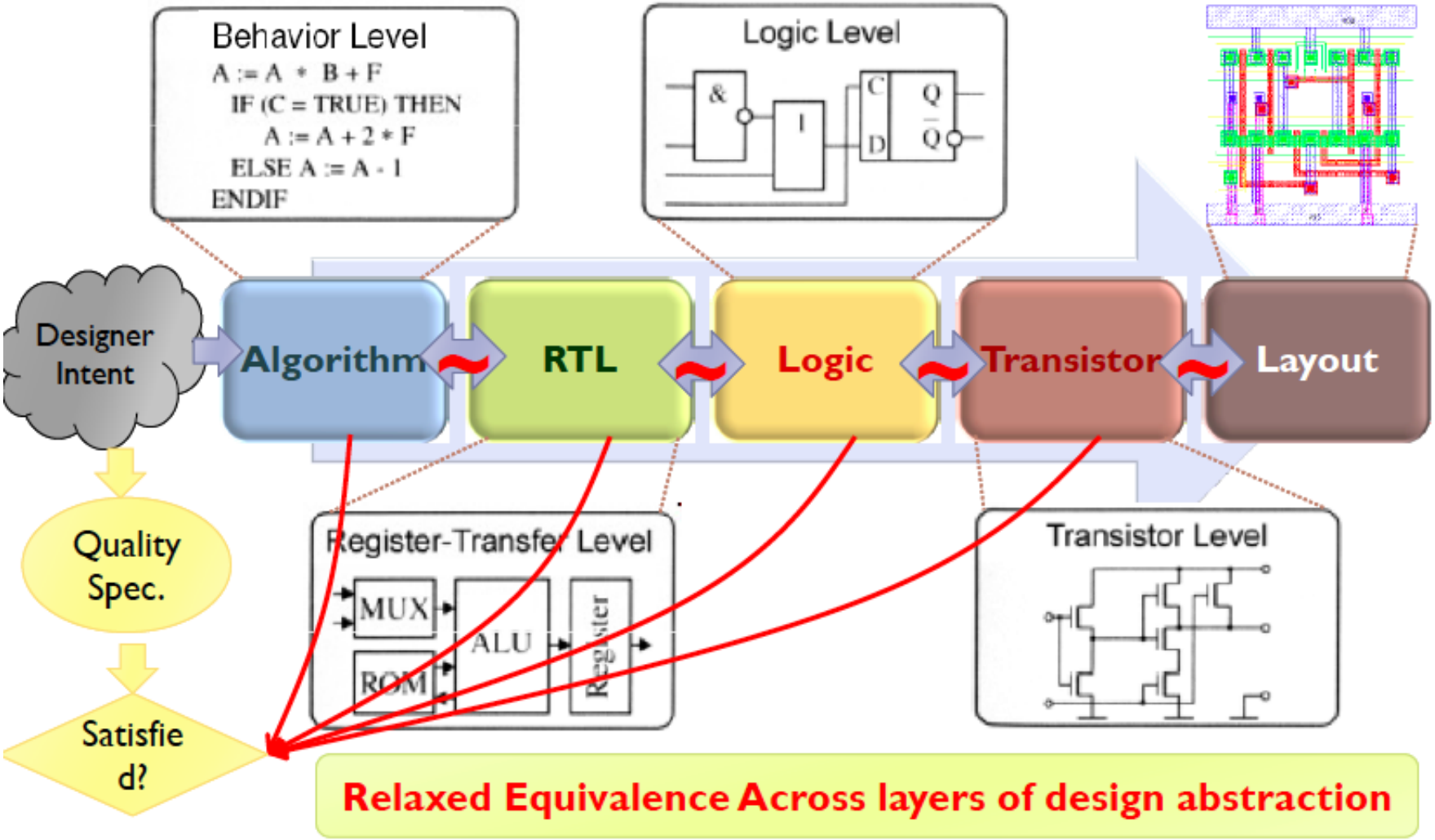
- Measuring output quality is expensive
  - Usually by sampling over time



Requirement: Inexpensive continuous monitoring

CADSL

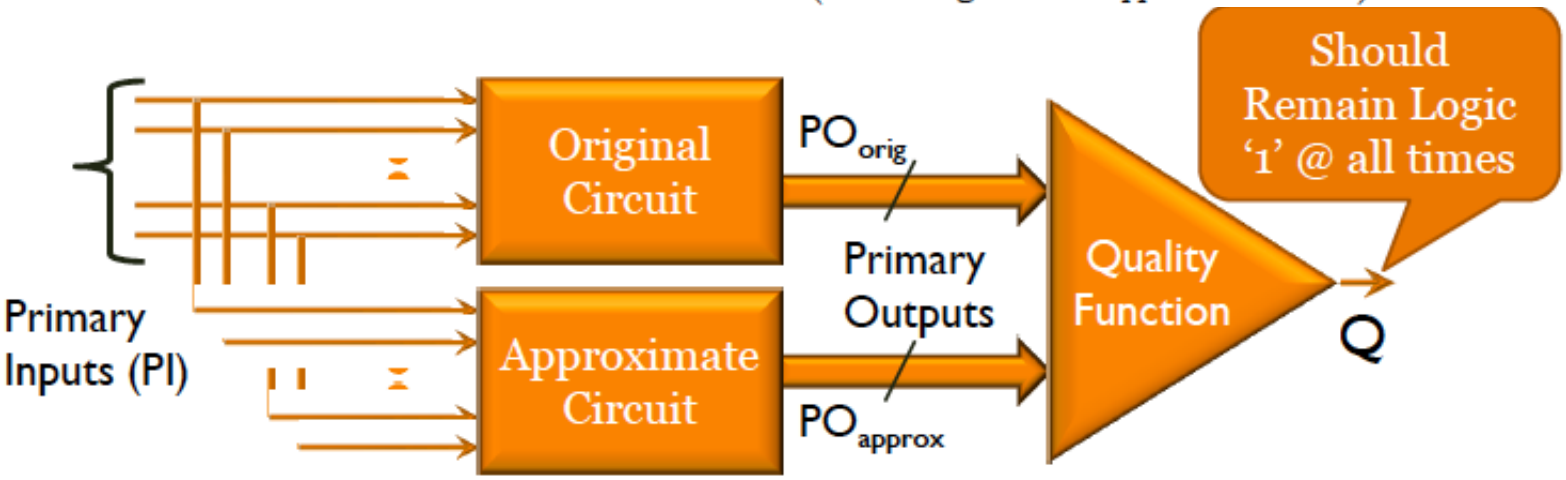
Traditional Design vs Approximate Design



Quality Constraint Circuit

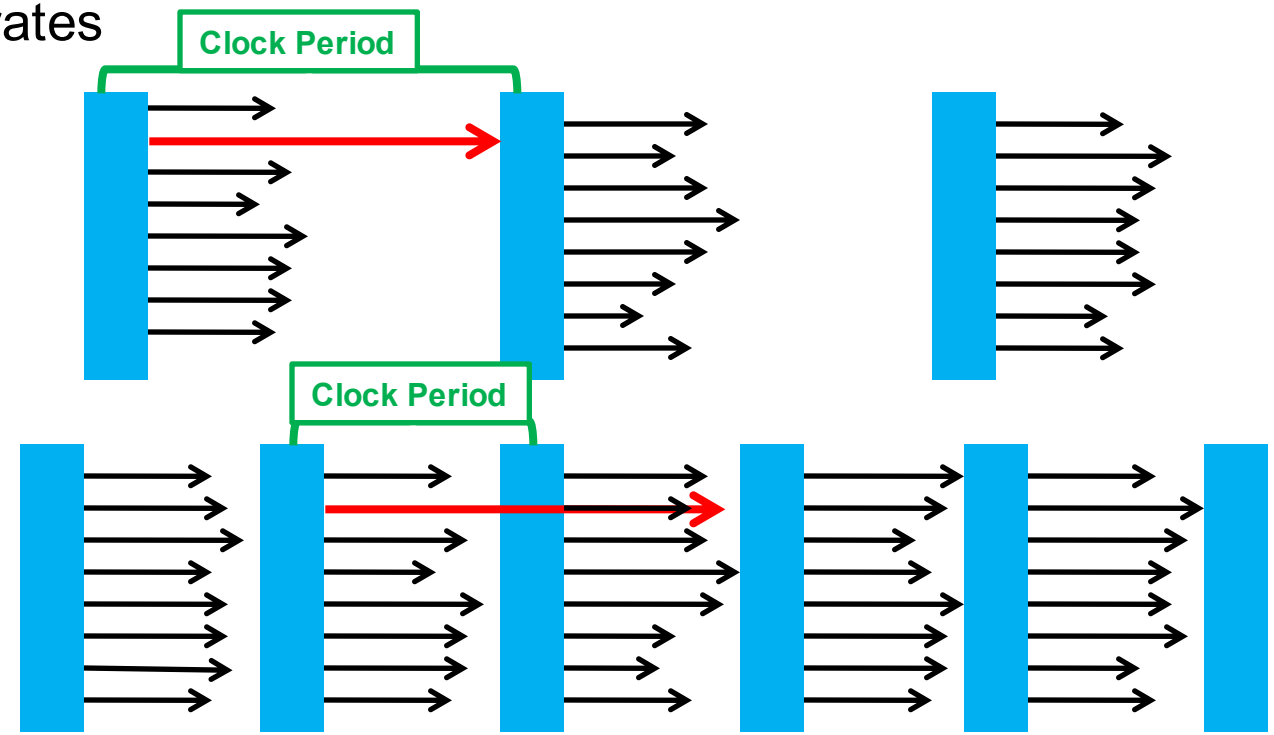
- Quality constraint circuit: Enforce quality constraint during synthesis
- Quality function: Quality constraint encoded as Boolean circuit
  - Output is 1 when quality is acceptable
- Approximate circuit can be very different provided QCC remain tautology

$$Q = \left(\left| PO_{orig} - PO_{approx} \right| \leq K \right) \ ? \ 1 : 0$$



Better-than-Worst-Case Design

- Traditional design sets clock rates to accommodate worst case delay
  - Guaranteed error free operation with speed binning
  - Significant performance loss due to random path/process variability
- **Better-than-worst-case design** aims to allow *occasional* errors to achieve faster clock rates



- Challenge is to design efficient **error detection and recovery** techniques that ensure performance gains beyond the error recovery overhead

Property Aware Design Methodology

Property Checking is actually more like *design* than like *verification* !

white box verification

(property checking)

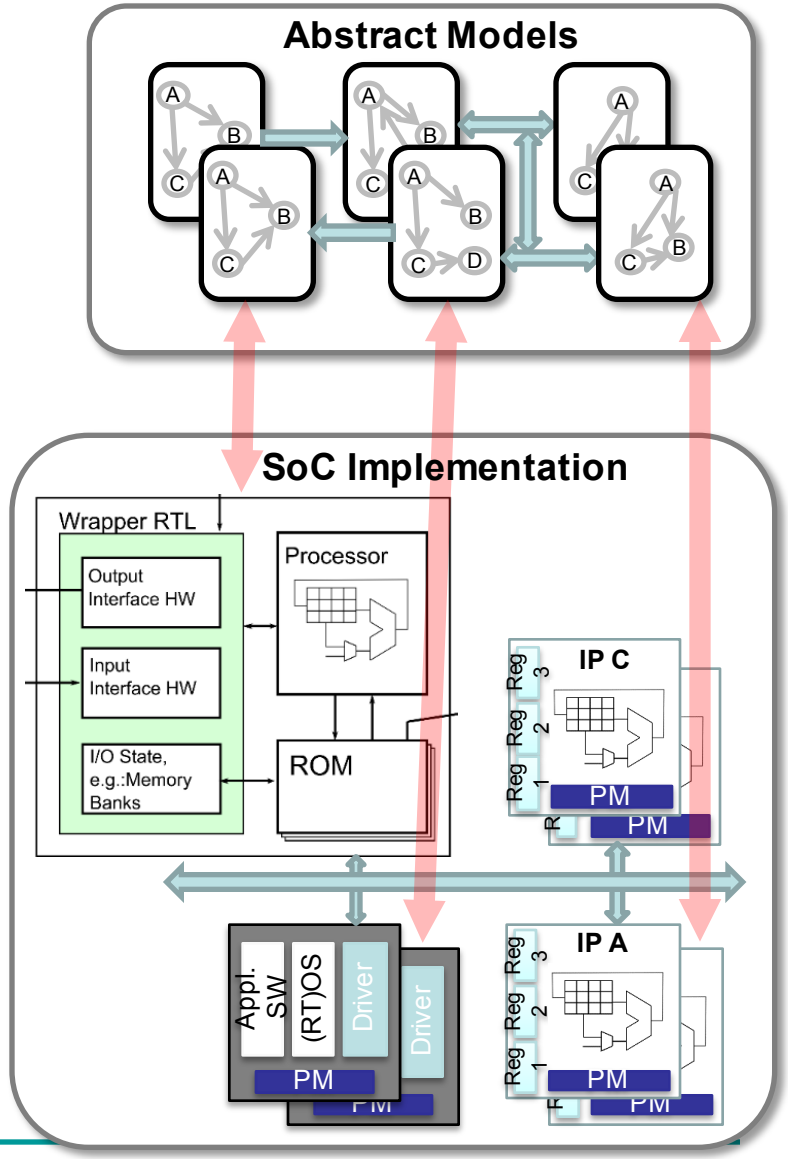
black box verification

(simulation)

The biggest hurdle

ESL-based Flows: the "Semantic Gap"

- Models of SoC designs at the Electronic System Level (ESL) gain popularity
- Yet, sign-off verification still largely relies on RTL as the point of reference
- No formal relationship between high-level models and RTL implementation: "semantic gap"



Closing the Semantic Gap

How to do the abstraction?

Idea:

define the semantics of the system model in terms of objects of the RTL description

by

formulating properties for the RTL model in a standard property language, e.g., SVA

Thank you for your kind cooperation through out the semester