IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

Lecture 62: OTHER LOW POWER DESIGN TECHNIQUES

PROF. INDRANIL SENGUPTA

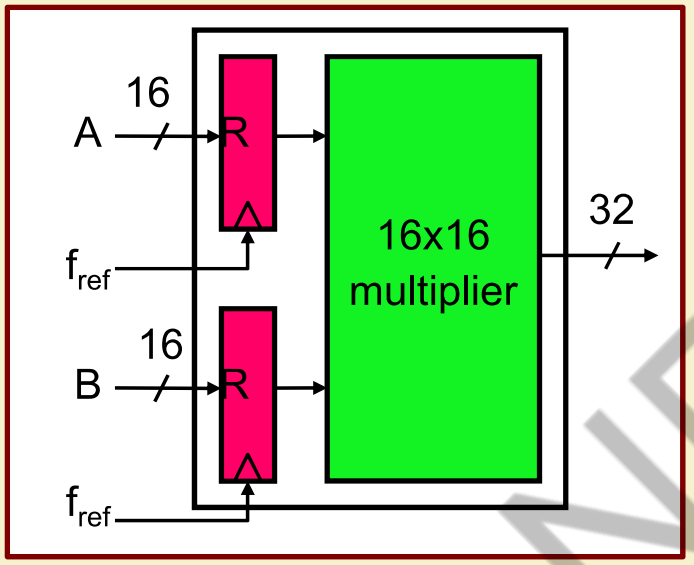
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

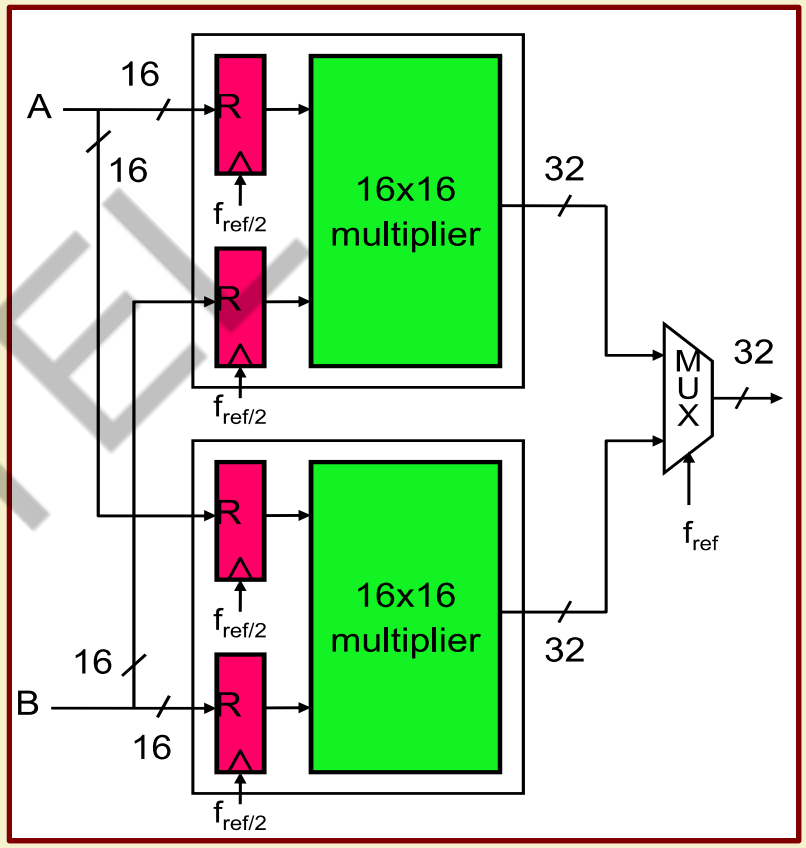
Architectural Level Approaches

- Several architectural level approaches can be used to reduce the power dissipation.
- Some of the common approaches:
  - Use parallelism
  - Use pipelining
  - Use retiming
  - Bus segmentation

(a) Using Parallelism

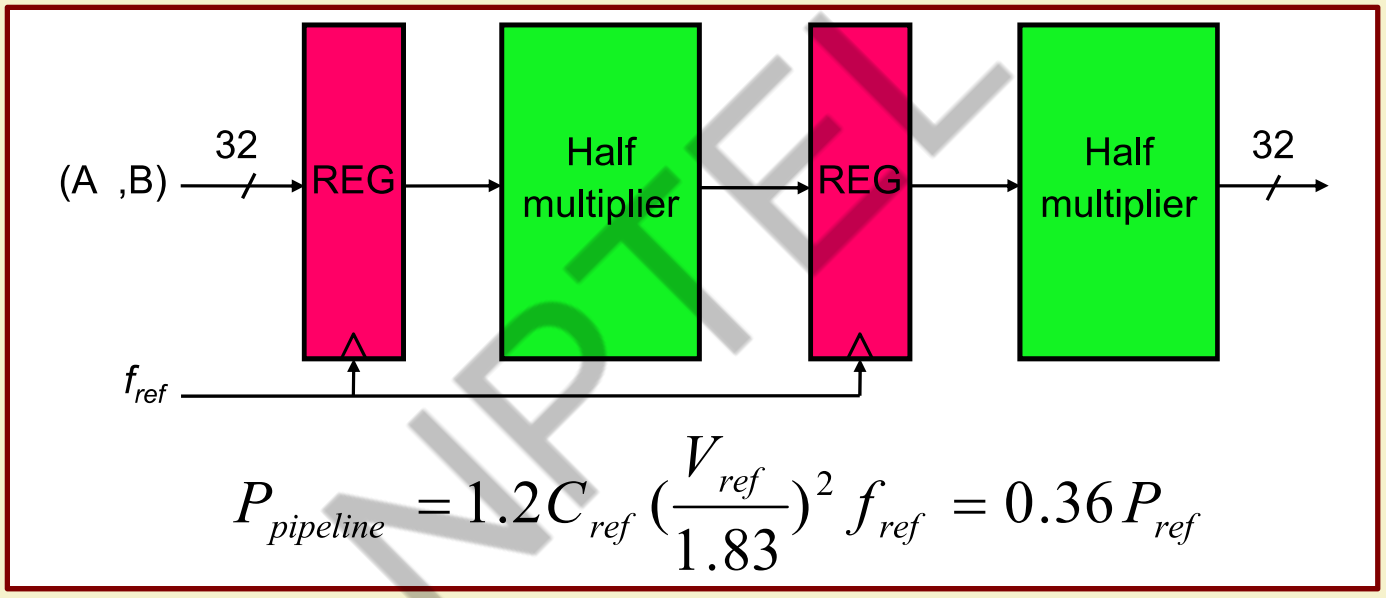
- We illustrate the concept using the example of a 16-bit multiplier.
  - Instead on one multiplier running at f_{ref}, we use two multipliers that run at half the clock rate, i.e. $f_{ref}/2$.
  - Overall throughput remains the same.
  - Output multiplexer selects the multiplier outputs alternately and produce outputs at the rate f_{ref} .


16
A
R
16x16
multiplier
32
f_{ref}
16
B
R
f_{ref}


16
A
16
R
$f_{ref/2}$
16x16
multiplier
32
R
$f_{ref/2}$
M
U
X
32
f_{ref}
R
$f_{ref/2}$
16x16
multiplier
16
B
16
R
32
$f_{ref/2}$

(b) Using Pipelining

- Basic concept:
  - Suppose the pipeline stages are made simpler, such that the stage delays are reduced.
  - We can reduce the original reference voltage V_{ref} to a lower value V_{new} to maintain the same worst-case delay.
- An example:
  - Suppose we split a 50 MHz multiplier into two equal parts.
  - The delay between the pipeline stages can remain at 1/50MHz, when $V_{new} = V_{ref} / 1.83$.



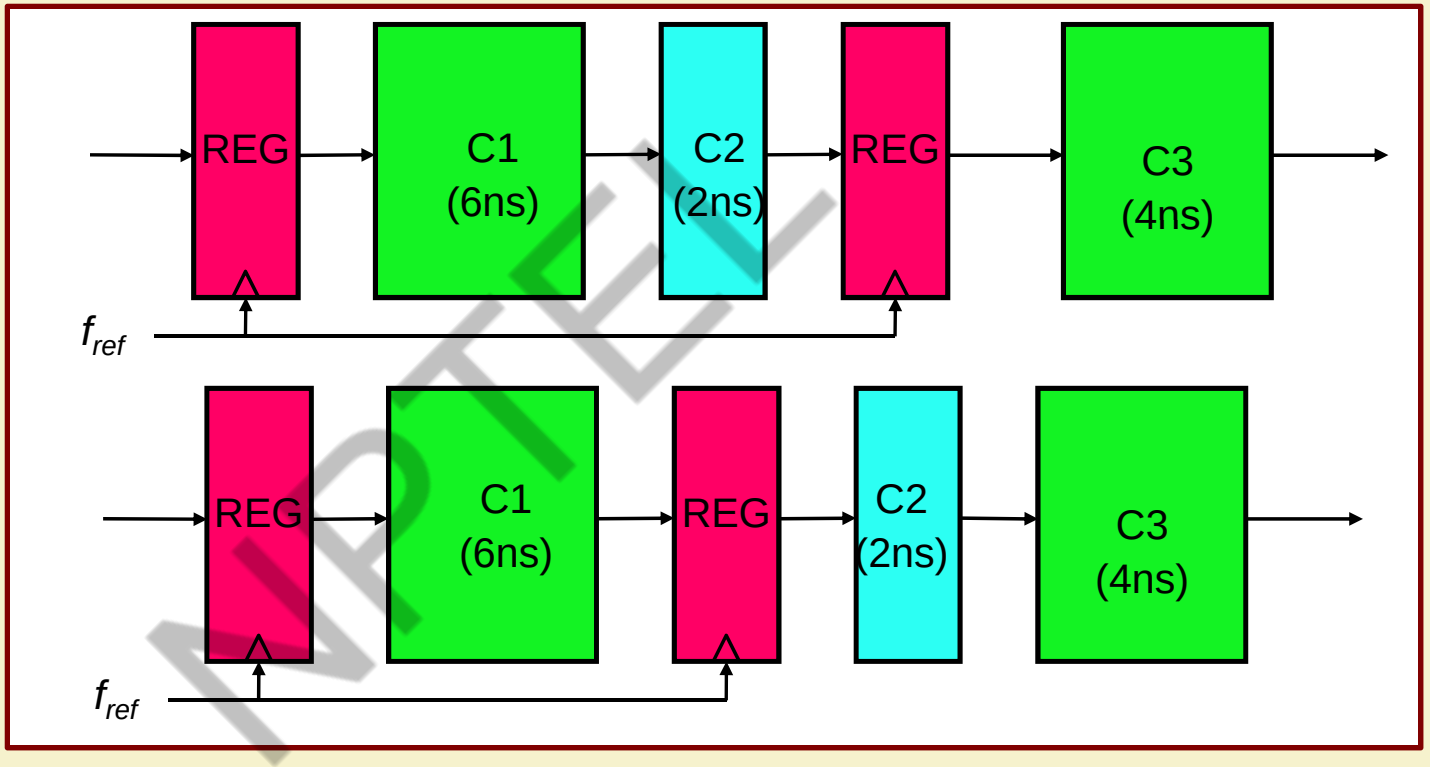
$$P_{pipeline} = 1.2 C_{ref} \left(\frac{V_{ref}}{1.83}\right)^2 f_{ref} = 0.36 P_{ref}$$

(c) Using Retiming

- Retiming is a design transformation technique used to change the locations of the delay elements in a circuit without affecting the input/output characteristics.
- Two approaches shall be explained:
  a) Retiming for pipeline design
  b) Retiming for gate-level and flip-flop-level design

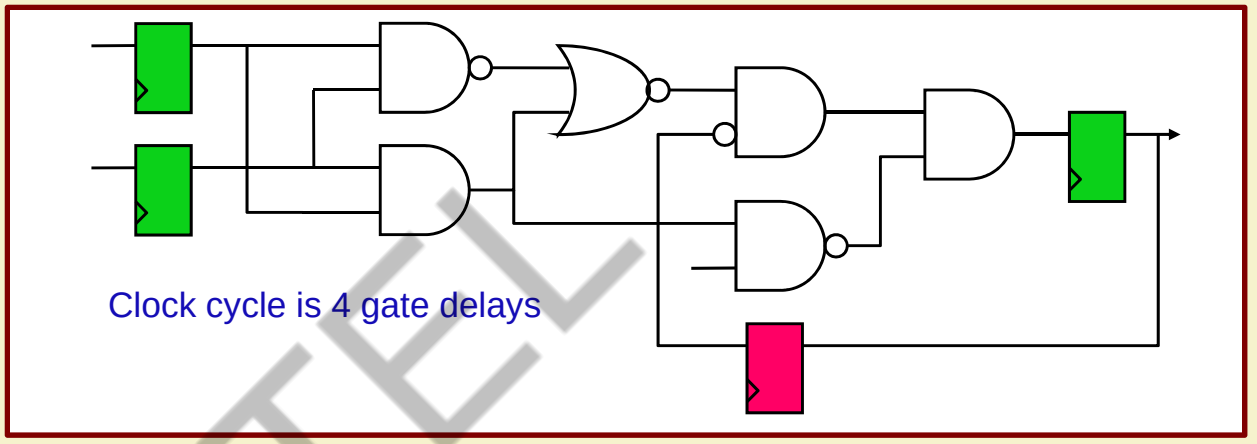
Pipeline Design

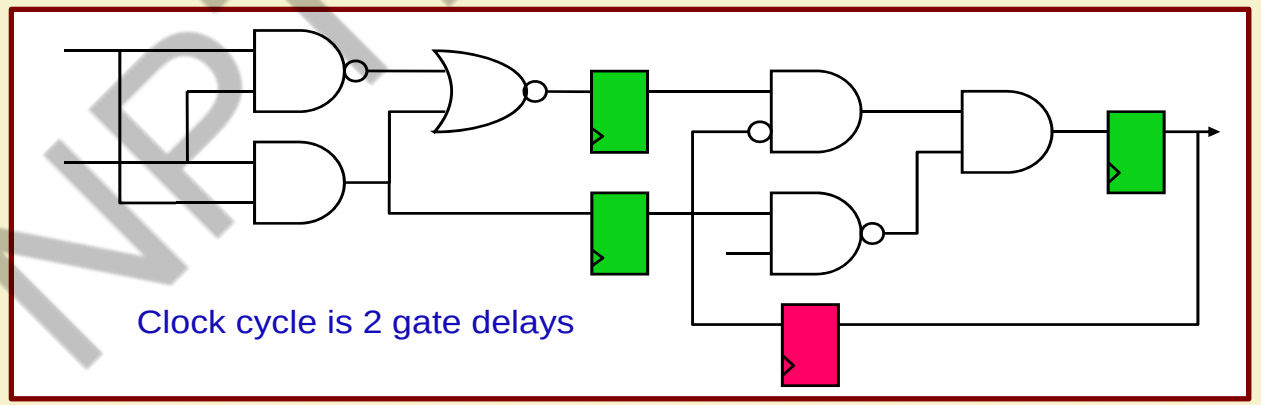
- The first pipeline has maximum stage delay of 8ns.
- The second pipeline has maximum stage delay of 6 ns.



Gate-level Design

- The first design has a maximum path length of 4 gates between flip-flops.
- The second design has a maximum path length of 2 gates between flip-flops.
  - We moved two flip-flops across two levels of gates.



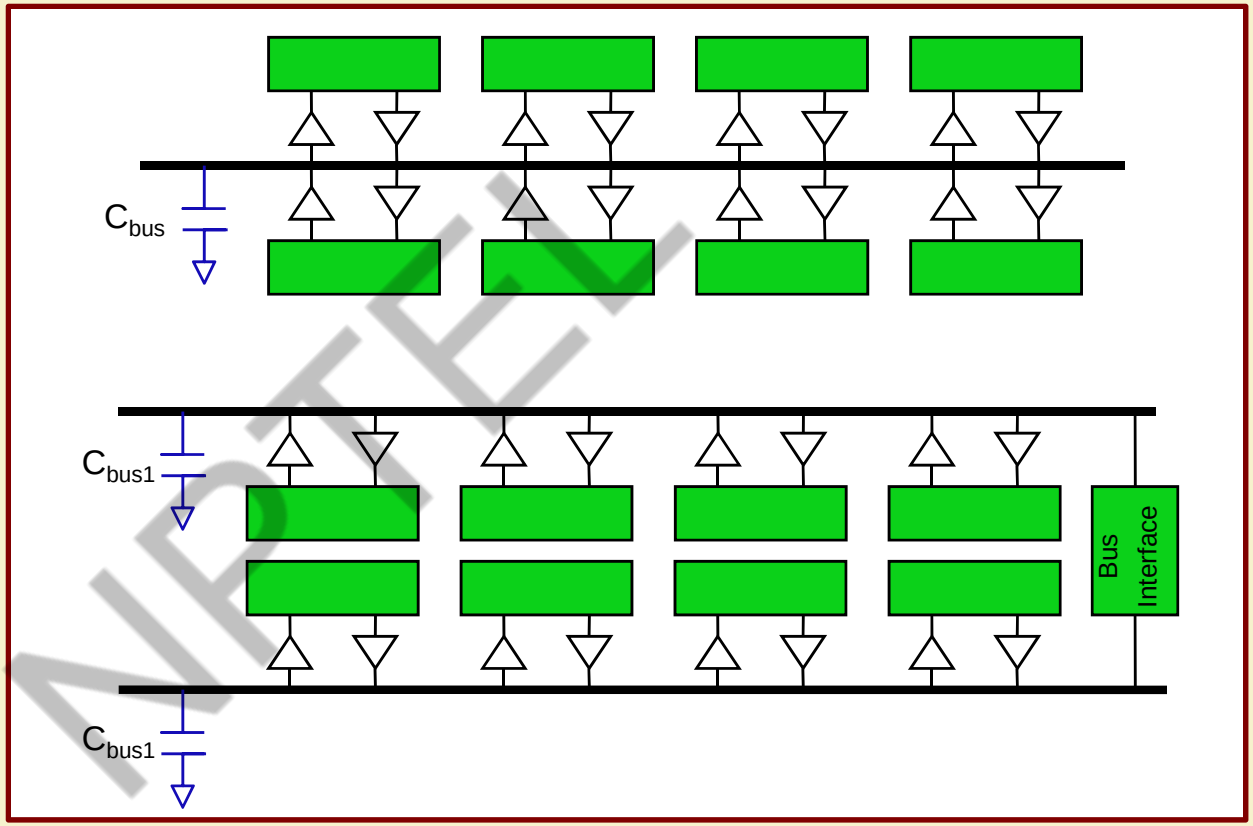


(d) Bus Segmentation

- Here we try to avoid the sharing of resources.
  - Leads to smaller switched capacitance (due to reduction in fanout).
- Consider an example of a system bus – two alternatives:
  1) A single shared bus is connected to all the modules. This results in large bus capacitance due to:
     - Large number of drivers and receivers sharing the same bus.
     - The parasitic capacitance of the long bus line.
  2) A segmented bus structure – may increase the routing area.
     - Switched capacitance during each bus access is significantly reduced.

Single shared bus →

Segmented bus →



END OF LECTURE 62

IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

Lecture 63: ALGORITHMIC LEVEL TECHNIQUES FOR LOW POWER DESIGN

PROF. INDRANIL SENGUPTA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Algorithmic Level Approaches

- Several approaches exist that exploit changes in the algorithmic level to achieve power reduction.
- Some approaches:
  a) State encoding for binary counters
  b) State encoding for general Finite State Machines (FSMs)
  c) Clock gating in FSMs

(a) State Encoding for Binary Counters

- Instead of working at the circuit level, we can also minimize the switching activity at higher level.
- One popular method:
  - Use an appropriate coding for the signals rather that the straight binary code.
  - Can help in the minimization of switching activity.
- For circuits where the states are traversed in a particular sequence (e.g. *counter*), we can use *Gray Code* encoding.

3-bit Representations of Binary and Gray Codes

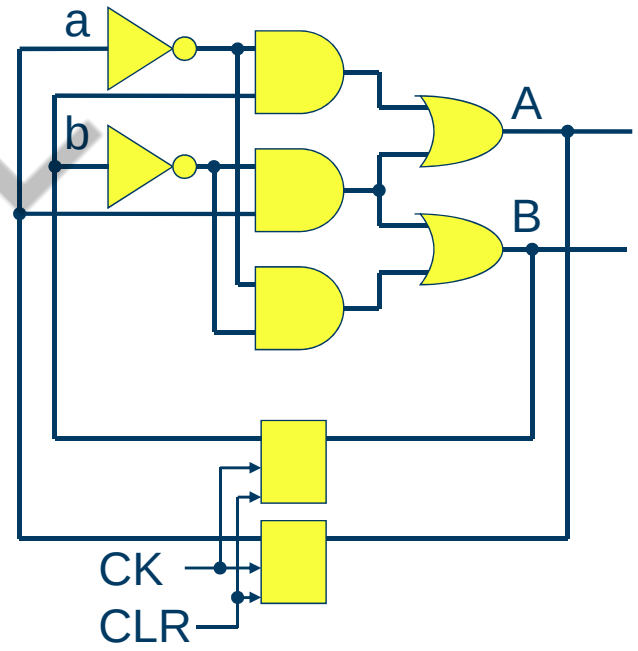
Binary Code	Gray Code	Decimal
000	000	0
001	001	1
010	011	2
011	010	3
100	110	4
101	111	5
110	101	6
111	100	7

- State encoding for a 2-bit binary counter:
  - State sequence: *00 → 01 → 10 → 11 → 00 …*
  - 6 bit transitions in 4 clock cycles
  - 6/4 = *1.5 transitions per clock*
- State encoding for a 2-bit Gray code counter:
  - State sequence: *00 → 01 → 11 → 10 → 00 …*
  - 4 bit transitions in 4 clock cycles
  - 4/4 = *1.0 transitions per clock*
- Hence Gray code counter is more power efficient.

2-bit binary counter design with original encoding.

Present state		Next state	
a	b	A	B
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

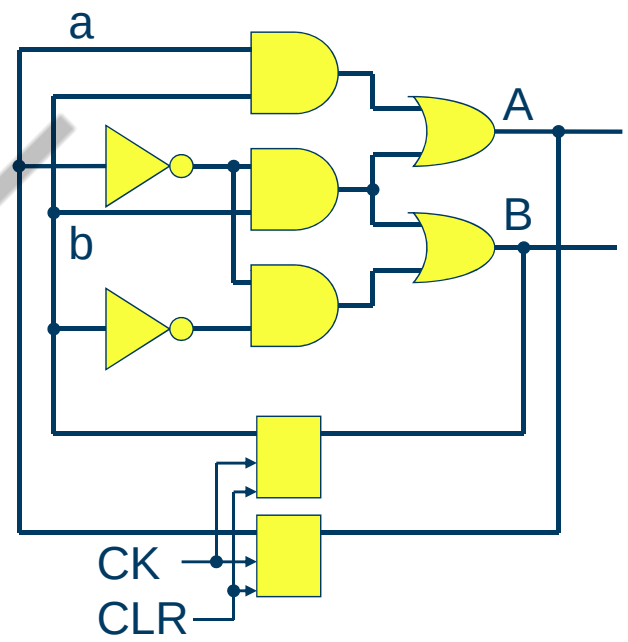
A = a'b + ab'
B = a'b' + ab'



2-bit binary counter design with Gray code encoding.

Present state		Next state	
a	b	A	B
0	0	0	1
0	1	1	1
1	0	0	0
1	1	1	0

A = a'b + ab
B = a'b' + a'b



Comparison for 3-bit counter.

75% more toggles in binary counter.

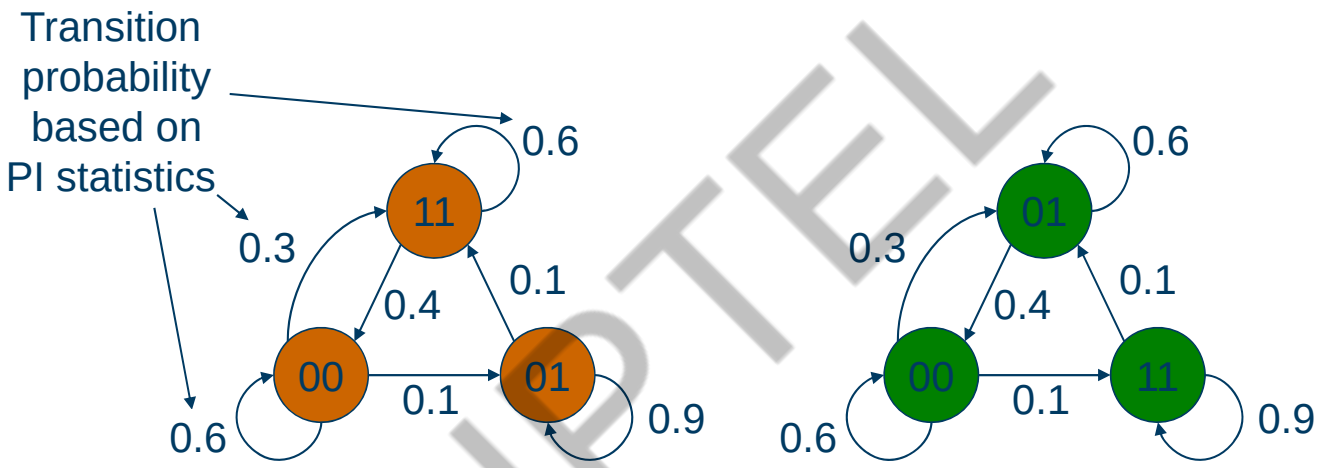
Binary		Gray-code	
State	No. of toggles	State	No. of toggles
000	-	000	-
001	1	001	1
010	2	011	1
011	1	010	1
100	3	110	1
101	1	111	1
110	2	101	1
111	1	100	1
000	3	000	1
Av. Transitions/clock = 1.75		Av. Transitions/clock = 1	

Toggles for *N*-bit counter design:

- Number of toggles for binary counter $T_{bin} = 2(2^N - 1)$
- Number of toggles for Gray counter $T_{gray} = 2^N$
- Thus :: $T_{gray} / T_{bin} = 2^{N-1} / (2^N - 1) \rightarrow 0.5$

Bits	T(binary)	T(gray)	T(gray)/T(binary)
1	2	2	1.0
2	6	4	0.6667
3	14	8	0.5714
4	30	16	0.5333
5	62	32	0.5161
6	126	64	0.5079
∞	-	-	0.5000

(b) State Encoding for FSMs



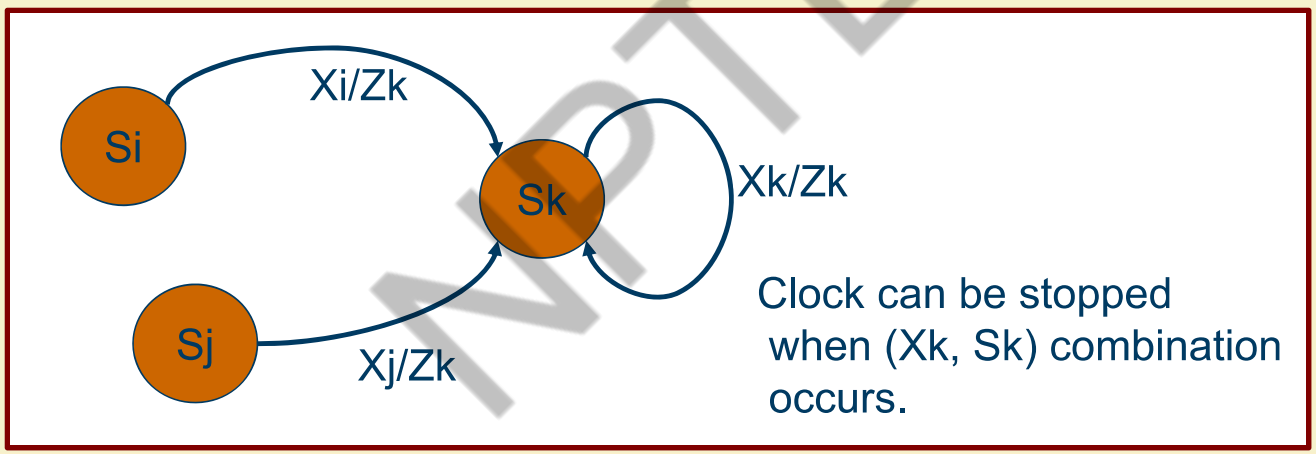
Expected number of state-bit transitions:

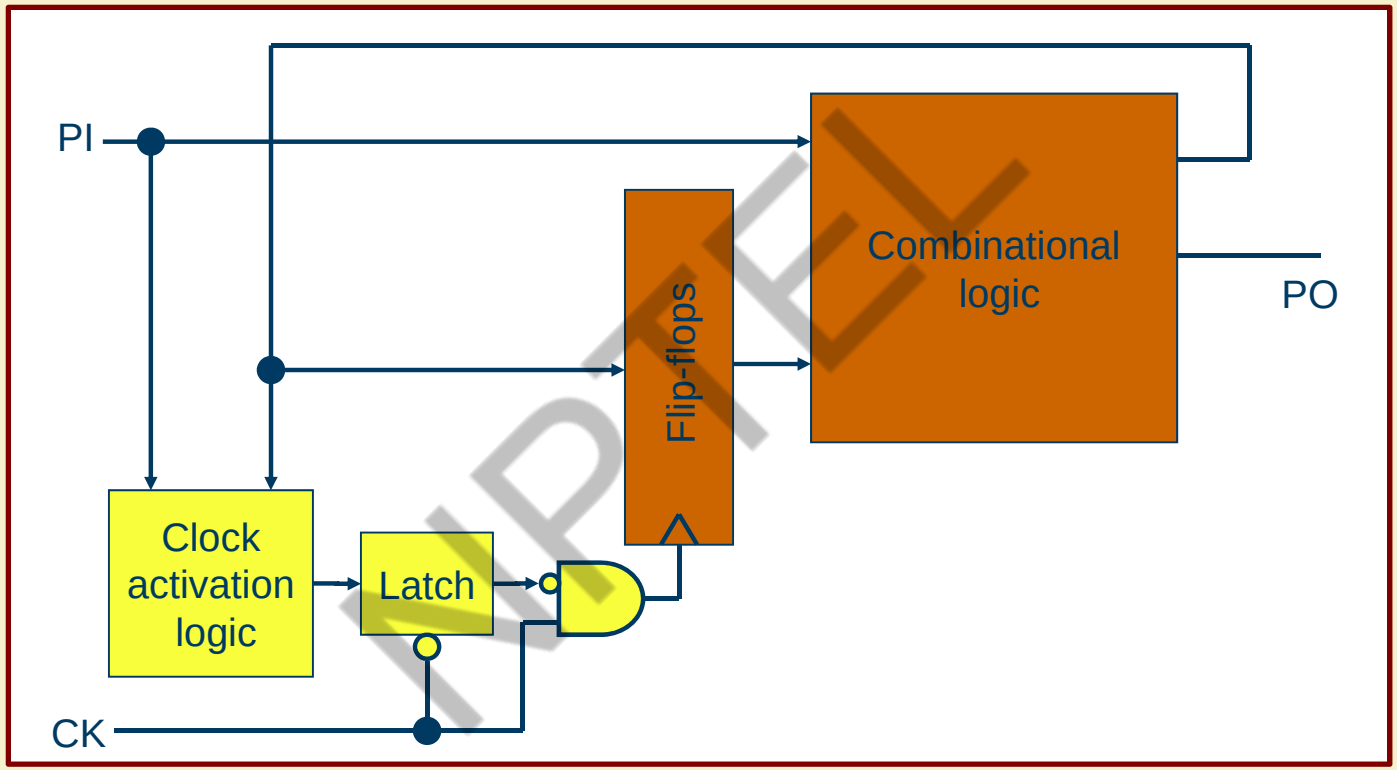
2(0.3+0.4) + 1(0.1+0.1) = 1.6

1(0.3+0.4+0.1) + 2(0.1) = 1.0

(c) Clock Gating in FSMs

- In a Moore machine, the output(s) depend only on the state variables.
- If a state has a self-loop in the state transition diagram, then clock can be stopped whenever the self-loop is to be executed.

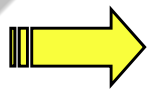



PI
Combinational logic
PO
Flip-flops
Clock activation logic
Latch
CK

RTL Level Design (Signal Gating)

Simple Decoder

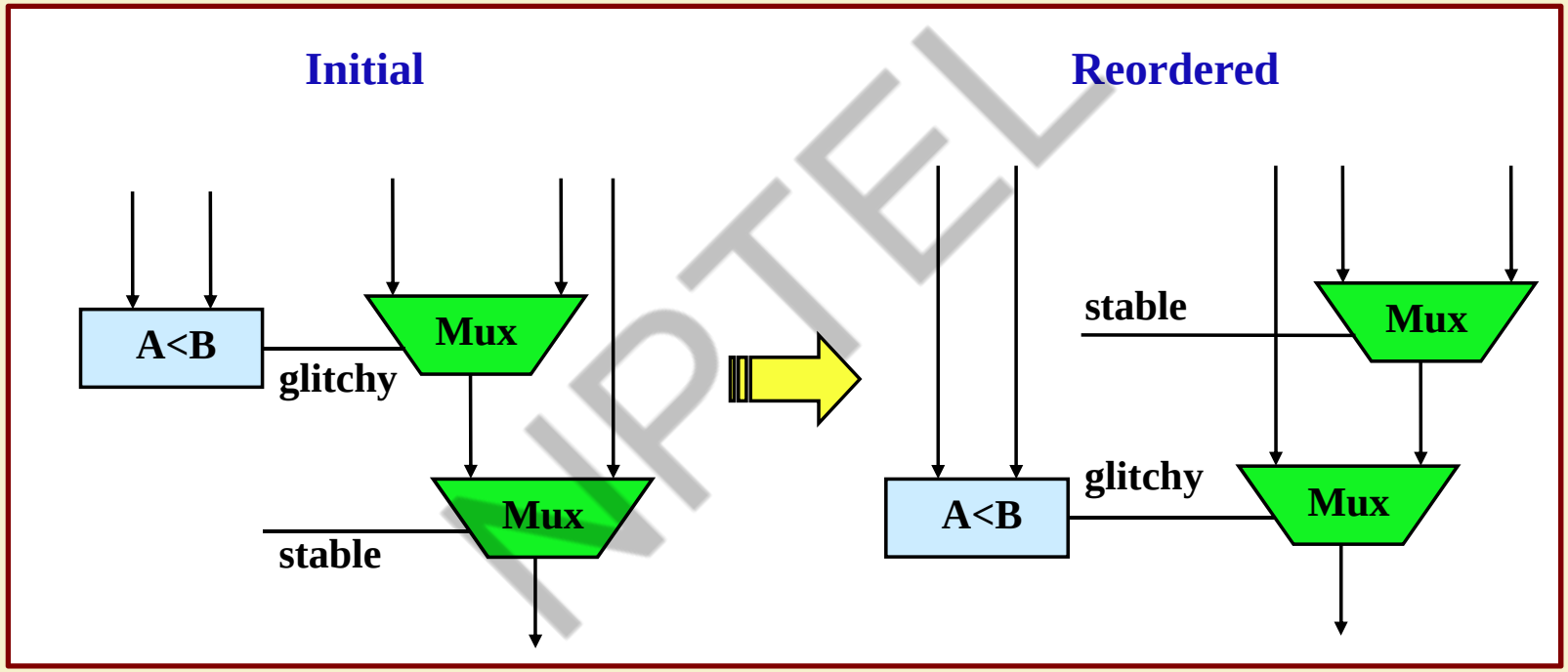
```
module decoder (a, sel);
    input  [1:0[ a;
    ouput [3:0] sel;
    reg     [3:0] sel;
    always @(a) begin
        case (a)
            2'b00: sel=4'b0001;
            2'b01: sel=4'b0010;
            2'b10: sel=4'b0100;
            2'b11: sel=4'b1000;
        endcase
    end
endmodule
```



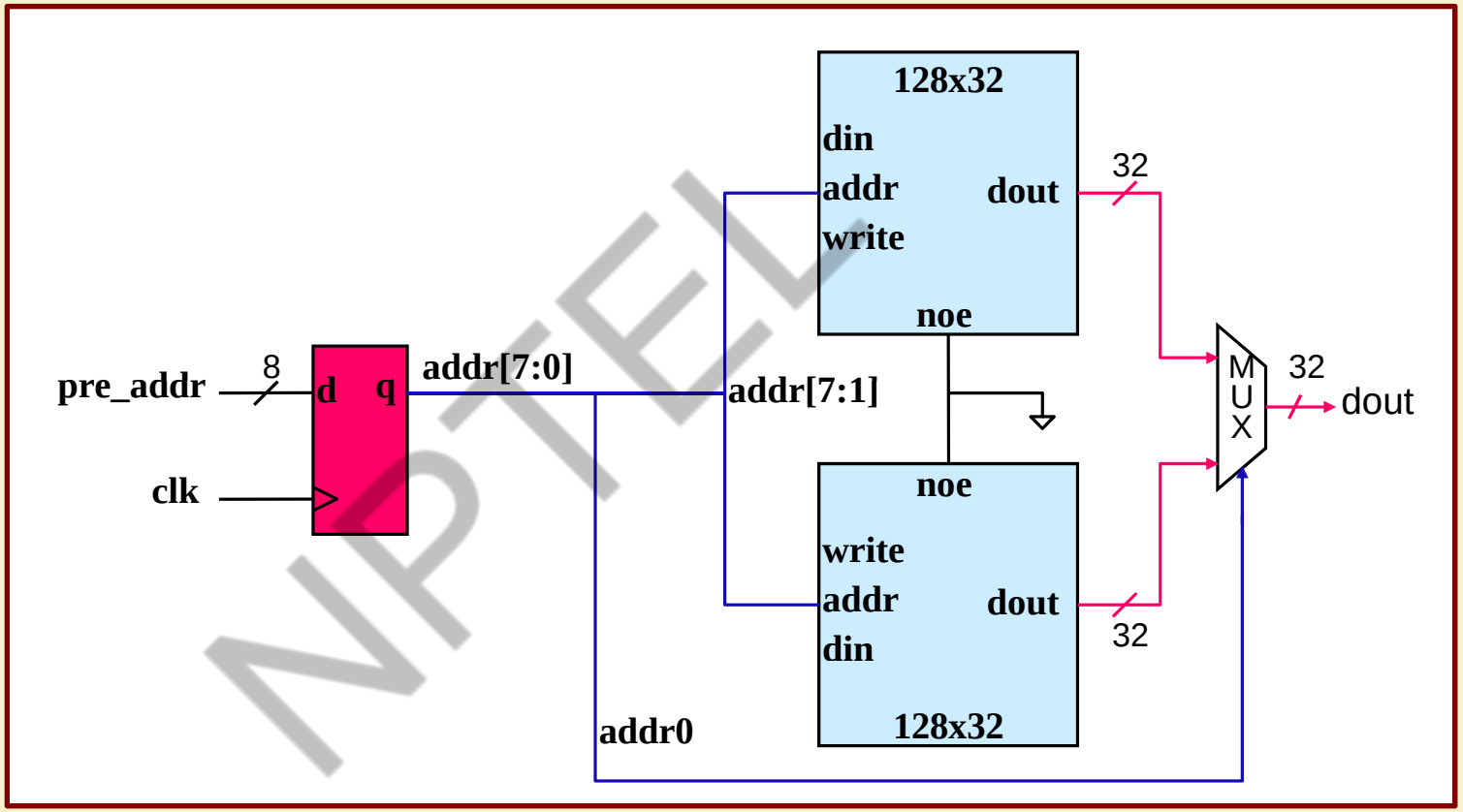
Decoder with enable

```
module decoder (en,a, sel);
    input          en;
    input  [1:0[ a;
    ouput [3:0] sel;
    reg     [3:0] sel;
    always @({en,a}) begin
        case ({en,a})
            3'b100: sel=4'b0001;
            3'b101: sel=4'b0010;
            3'b110: sel=4'b0100;
            3'b111: sel=4'b1000;
            default: sel=4'b0000;
        endcase
    end
endmodule
```

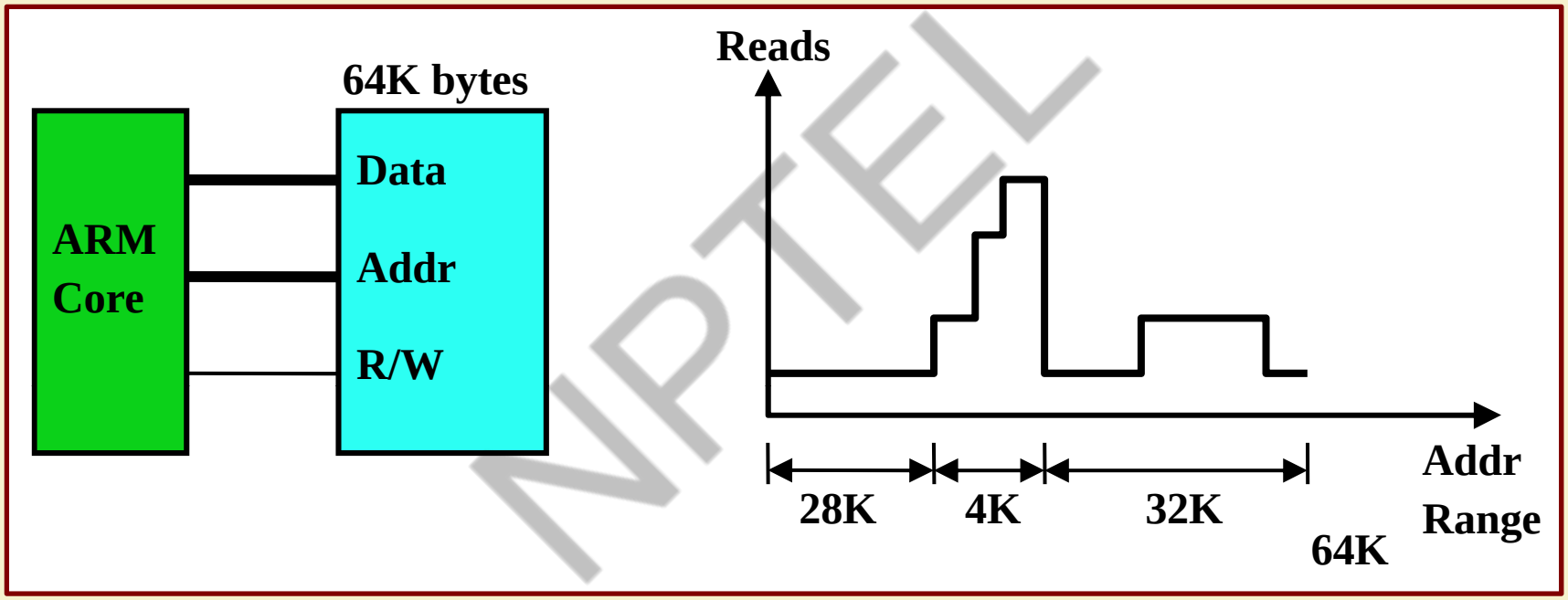
RTL Level Design : Datapath Reordering



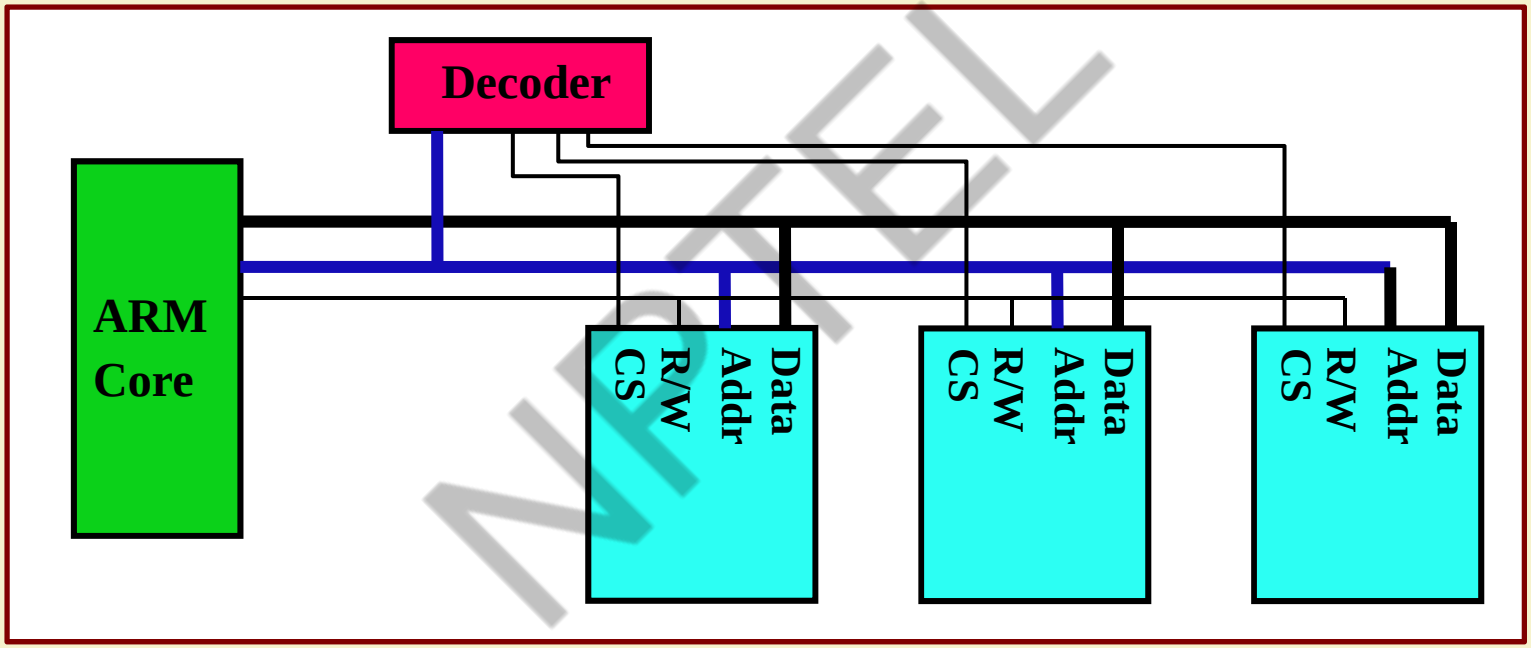
RTL Level Design : Memory Partition



Application-driven Memory Partition



Power Optimal Partitioned Memory Organization



END OF LECTURE 63

IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

Lecture 64: SUMMARIZATION OF THE COURSE

PROF. INDRANIL SENGUPTA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

What has been Covered?

- WEEK 1:
  - Introduction to design automation and design representation
  - VLSI design styles
  - Overview of VLSI Physical Design Automation

- WEEK 2:
  - Circuit partitioning
  - Floorplanning representation and algorithms
  - Placement problem and algorithms
  - Design style specific issues

- WEEK 3:
  - Grid routing problem and algorithms
  - Line search algorithms
  - Global routing problem and algorithms

- WEEK 4:
  - Detailed routing problem
  - Channel routing algorithms
  - Clocking issues and clock design

- WEEK 5:
  - Clock network synthesis problem
  - Various clock routing architectures and skew minimization issues
  - Power and ground routing

- WEEK 6:
  - Introduction to timing closure problem
  - Static Timing Analysis and variations
  - Slack analysis and identification of critical paths
  - False path identification techniques
  - Timing driven placement techniques

- WEEK 7:
  - Timing driven routing techniques
  - Physical synthesis techniques (gate sizing, buffering, netlist restructuring, etc.)
  - Performance driven design flow
  - Miscellaneous approaches to timing optimization

- WEEK 8:
  - Interconnect modeling and delay analysis
  - Design rule check
  - Layout compaction techniques

- WEEK 9:
  - Introduction to VLSI circuit testing
  - Fault modeling approaches and techniques
  - Fault equivalence and fault dominance
  - Fault simulation algorithms (parallel, deductive, concurrent, etc.)

- WEEK 10:
  - Automated test pattern generation techniques
  - Design for testability techniques
  - IEEE 1149.1 boundary scan standard
  - Built-in-Self-Test (BIST)

- WEEK 11:
  - Low power VLSI design concepts
  - Dynamic power, static power, leakage power
  - General techniques to reduce power
  - Gate level design techniques for power reduction

- WEEK 12:
  - Architectural level techniques for reducing power
  - Algorithmic level techniques for reducing power

Supplementary Materials

- Demonstration of physical design automation processes on commercial CAD tools.

Main References

1. N.A. Sherwani, “Algorithms for VLSI Design Automation”, Kluwer Academic Publishers, 1999.
2. S.M. Sait and H. Youssef, “VLSI Physical Design Automation: Theory and Practice”, World Scientific Pub. Co., 1999.
3. M.L. Bushnell and V.D. Agrawal, “Essentials of Electronic testing”, Kluwer Academic Publishers, 2000.
4. A.B. Kahng, J. Lienig, I.L. Markov and J. Hu, “VLSI Physical Design: From Graph Partitioning to Timing Closure”, Springer, 2011.

END OF THE COURSE

THANK YOU FOR ATTENDING