EDA321/EDA322: Digital Design
Re-Exam - January 2015

Date: January 3, 2015

Time: **14:00-18:00**

Examiner: Ioannis Sourdis

Department: Computer Science and Engineering

Inquiries: Stavros Tzilis (extension 5232); will visit the room at **14:30, 15:30** and at **17:00**

Results and grading review: See me in my office 4110 on **January 26 at 16:00**.

Duration: 4 hours

Grading scale: 100 points in total

Chalmers:
0: 0%-39%, 3: 40%-59%, 4: 60%-79%, 5: 80%-100%
GU:
Fail (U): 0%-39%, Pass (G): 40%-69%, Pass with Distinction (VG): 70%-100%

Available references: Blank paper and a calculator are allowed. No textbooks or lecture notes, etc. allowed.

General: Submit your solutions, in English, on blank paper sheets. Write legibly; feel free to use figures to get your point across.

The order of answering the questions does not matter (start with the easiest ones).

Please start the solutions for each problem on a new sheet. Please number the sheets so that the solutions are in numerical order.

Note that it is possible to receive partial credit for an answer even if it is not 100% correct.

Your personal identity code is required on each submitted sheet!

Good luck!

Question 1: (10 points)

a) Find the minterms of the function F(x3,x2,x1,x0)= Π(3,4,6,11,12,14) (1 points)

Minimize the cost of function F.

b) using Quine-McCluskey? (9 points)

Measure the cost of the minimized function by counting the total number of 2-input gates of the circuit (e.g. a*b + c*d, has cost of 3).

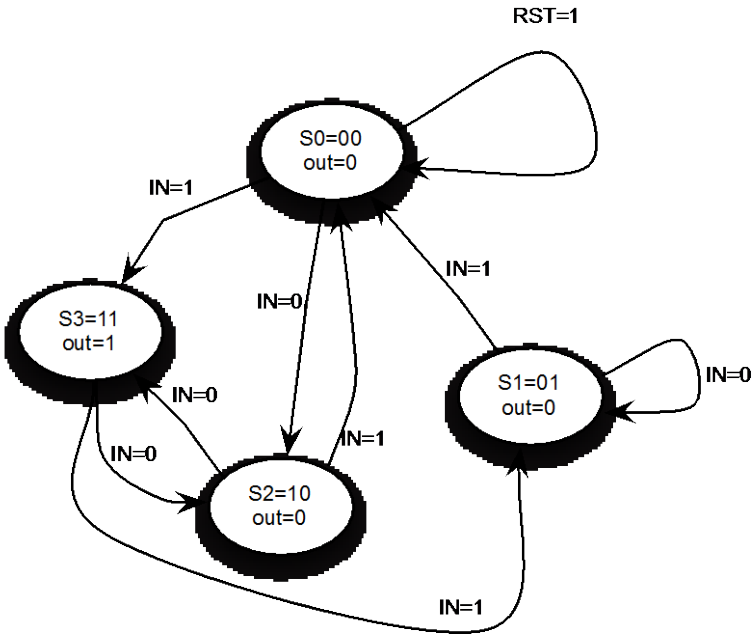
Answer:

Example on Lecture 2 slide 54, after applying the technique of slide 12

(Above mentioned lecture slides can be found at the end of this document)

Question 2: (10 points)

Make the state-assigned table of the FSM below and derive and draw the circuit that realizes the FSM using D-Flip-flops. Is it a Mealy or a Moore FSM?



Answer:

Book exercise 8.1 (solution available on solved exercises)

Question 3: (15 points)

a) Describe the steps of a **serial** unsigned binary multiplication between a 4-bit multiplicant x4x3x2x1 and a 4-bit multiplier y4y3y2y1. Draw and use in your explanations the hardware block diagram of a serial multiplier-module with a 4-bit ALU and its registers. Explain the steps needed for each iteration. How many iterations are needed for a 4-bit multiplication? (9 points)

b) Make the following binary division: 2*3 showing each iteration and steps needed as you described them in (a)? (6 points)

Answer:
Lecture11, slides 10-11.

(Above mentioned lecture slides can be found at the end of this document)

Question 4: (6 points)
Provide the definition and units (where applicable) of the following terms:

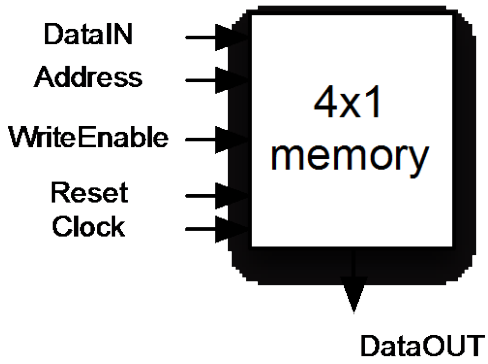
a) Memory Bandwidth
b) Memory latency
c) Asynchronous memory
d) Synchronous memory
e) Volatile
f) Non-volatile memory

Answer:
Lecture11, slides 40.

(Above mentioned lecture slides can be found at the end of this document)

Question 5: (9 points)
(a) Draw the block diagram of a 4x1 memory, using as "building blocks" D flip-flops and logic gates. The memory has the following inputs:

- DataIN,
- Write-enable,
- Address
- Reset, Clock



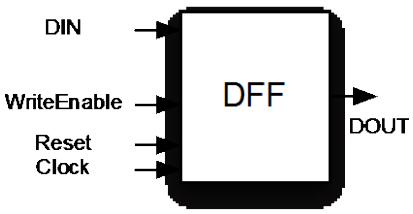
And the following output:

- DataOUT

Note: both "DataIN" and "DataOUT" are signals of 1 bit.

-How many bits is the "Address" signal?

The D flip-flop has the following interface:



(6 points)

(b) Taking the above 4x1 memory as a black box (not having the ability to change its internal design) what can we add to use it as a 8x2 memory? (3 points)

Answer:

a) *lecture 11, slide 34 (but for 4x1 instead of 4x3)*
b) *an 4 blocks in a 2x2 array using a column multiplexer for as shown in lecture 11, slide 39*

(Above mentioned lecture slides can be found at the end of this document)

Question 6: (10 points)

Describe and draw the block diagram of a 2x2 Field Programmable Gate Array (FPGA). Consider that each logic cell has 4 inputs and one output. Show the internals of a Logic cell, of its connections (connection block) to the reconfigurable interconnects, switch box.

Answer

Lecture 12, slides 9, 16, 30, 31

(Above mentioned lecture slides can be found at the end of this document)

Question 7: (10 points)

What is the difference between full custom ASICs and standard-cell ASICS? How each one is built and which one is expected to have better performance, lower power consumption, lower silicon area, and why?

Answer:

Standard cell ASICs use pre-fixed cells implementing a single gate or a flip-flip (or bigger blocks, then called block-based ASICs). Tools are then used to place and wire these standard cells. Cells are described in libraries of some vendor that the hardware designer uses.

In full-custom ASICs, every gate and flip-flop is designed in detail down to the transistor level and even lower. Full custom ASICs get better performance, lower power consumption, and better area efficiency compared to standard cell ASICs because designers can optimize them in more detail (down to the transistor level). However, they are more difficult and expensive to design. Full-custom ASICs are meant for critical blocks (e.g. blocks that cause performance bottlenecks).

Question 8: (10 points)

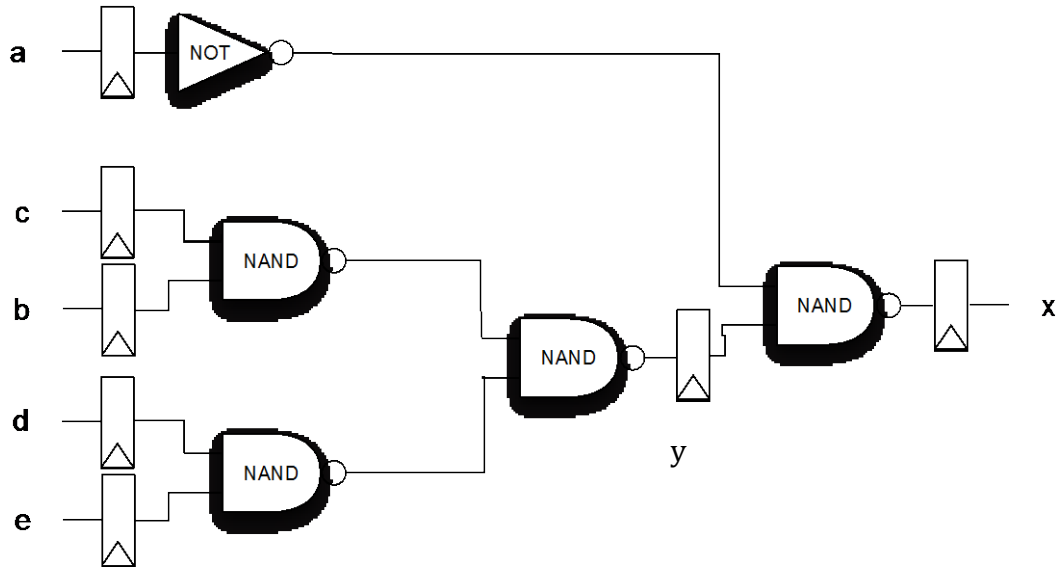
a) What is metastability and when can it happen? (5 points)
b) How can it be avoided? (5 points)

Answer:
Lecture 16, slides 30, 31, 32

(Above mentioned lecture slides can be found at the end of this document)

Question 9: (10 points)
Calculate the maximum delay of the circuit (critical path delay), considering the following:

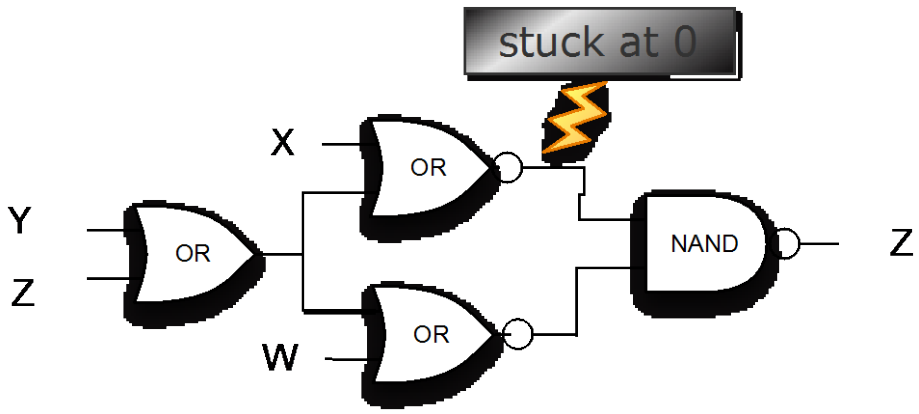
- *a NAND gate has a delay of 2 ns*
- *a NOT gate has a delay of 1 ns*
- *Propagation time (clock to output) for a flip-flop 0.5 ns*
- *Setup time for a flip-flop 0.5ns*
- *Hold time 0.2ns*
- *Wires have zero delay*
- *All inputs (a, b, c, d, e) have zero delay*



Answer:
The Critical path is from the output of the flip-flops connected to "b-c" or "d-e" to the input of the intermediate flip-flop (y).

> Max Delay = (Propagation delay of the flip-flop) + (Delay of the NAND gate) *2 + (Setup-time of the flip-flop) ⇔ Max Delay = 0.5ns + 2ns*2 +0.5ns = 5ns

Question 10: (10 points)
Find a test vector for detecting the stuck-at zero fault in the following circuit using the fault activation and fault propagation method:



Answer:
Similar to Lecture 13, slide 26

(Above mentioned lecture slides can be found at the end of this document)

END of EXAM

Lecture Slides

Question 1:

Canonical Forms

SoP canonical form

F= Σ(1,3,5,6,7)

F = 001 011 101 110 111

F = A'B'C + A'BC + AB'C + ABC' + ABC

A	B	C	F	F'	#
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	0	1	2
0	1	1	1	0	3
1	0	0	0	1	4
1	0	1	1	0	5
1	1	0	1	0	6
1	1	1	1	0	7

F= Π(0,2,4)

PoS canonical form

F = (A + B + C) (A + B' + C) (A' + B + C)

F = 000 010 100

F' = A'B'C' + A'BC' + AB'C'

canonical form ≠ minimal form

Quine-McCluskey: the binary notation

- **Step 1:** Describe individual minterms of the given expression by their equivalent binary numbers.
- **Step 2:** Form a table by grouping numbers with equivalent number of 1's in them, i.e. first numbers with no 1's, then numbers with one 1, and then numbers with two 1's, … etc.
- **Step 3:** Compare each number in the top group with each minterm in the next lower group. If the two numbers are the same in every position but one, place a check sign (✓) to the right of both numbers to show that they have been paired and covered. Then enter the newly formed number in the next column (a new table). The new number is the old numbers but where the literal differ, an "x" is placed in the position of that literal.
- **Step 4:** Using (3) above, form a second table and repeat the process again until no further pairing is possible. (On second repeat, compare numbers to numbers in the next group that have the same "x" position.
- **Step 5:** Terms which were not covered are the prime implicants and are ORed and ANDed together to form final function.

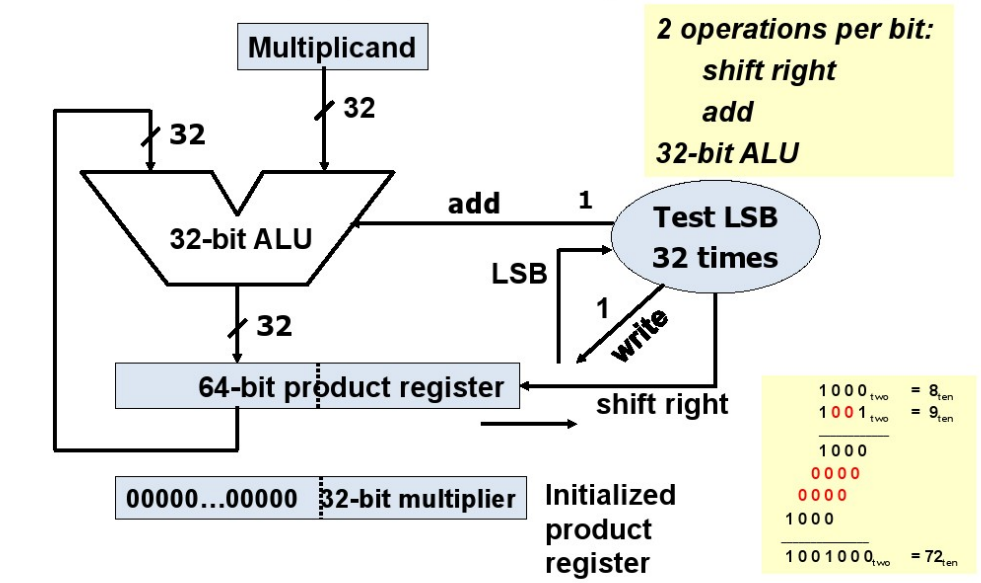
Quine-McCluskey the decimal notation

- **Step 1:** List the minterms grouped according to the number of 1's in their binary representation in the decimal format.
- **Step 2:** Compare each minterm with larger minterms in the next group down. If they differ by a power of 2 then they pair-off. Check both minterms and form a second table by the minterms paired and substitute the decimal difference of the corresponding minterms in the bracket, i.e. mx, my (y-x).
- **Step 3:** Compare each element of the group in the new table with elements of the next lower group and select numbers that have the same numbers in parenthesis. If the lowest minterm number of the table formed in the lower group is greater than the corresponding number by a power of 2 then they combine; place a ✓ ☞on the right of both elements.
- **Step 4:** Form a second table by all four minterms followed by both powers of 2 in parentheses, i.e. the previous value (the difference) and the power of 2 that is greater.
- **Step 5:** Select the common literals from each prime implicant by comparison.
- **Step 6:** Write the minimal SOP from the prime implicant that are not checked ✓.

EDA322 Digital Design, 2015-2016, Lecture 2 — I. Sourdis, CSE, Chalmers — 55

Question 3:

Serial Multiplication (Improved)



EDA322 Digital Design, 2015-2016, Lecture 11 — I. Sourdis, CSE, Chalmers — 10

Example: $0010_{two} \times 0011_{two}$

$0010_{two} \times 0011_{two} = 0110_{two}$, i.e., $2_{ten} \times 3_{ten} = 6_{ten}$

Iteration	Step	Multiplicand	Product
0	Initial values	0010	0000 0011
1	LSB=1 => Prod=Prod+Mcand	0010	0010 0011
	Right shift product	0010	0001 0001
2	LSB=1 => Prod=Prod+Mcand	0010	0011 0001
	Right shift product	0010	0001 1000
3	LSB=0 => no operation	0010	0001 1000
	Right shift product	0010	0000 1100
4	LSB=0 => no operation	0010	0000 1100
	Right shift product	0010	0000 0110

Question 4:

Definitions

- **Bandwidth:**
  Total amount of data across out of a device or across an interface per unit time. (usually Bytes/sec)
- **Latency:**
  A measure of the time from a request for a data transfer until the data is received.

Memory Interfaces for Accessing Data

- **Asynchronous** (unclocked):
  A change in the address results in data appearing
- **Synchronous** (clocked):
  A change in address, followed by an edge on CLK results in data appearing. Sometimes, multiple requests may be outstanding.
- **Volatile:**
  Looses its state when the power goes off. (the opposite: **non-volatile**)

Question 5:

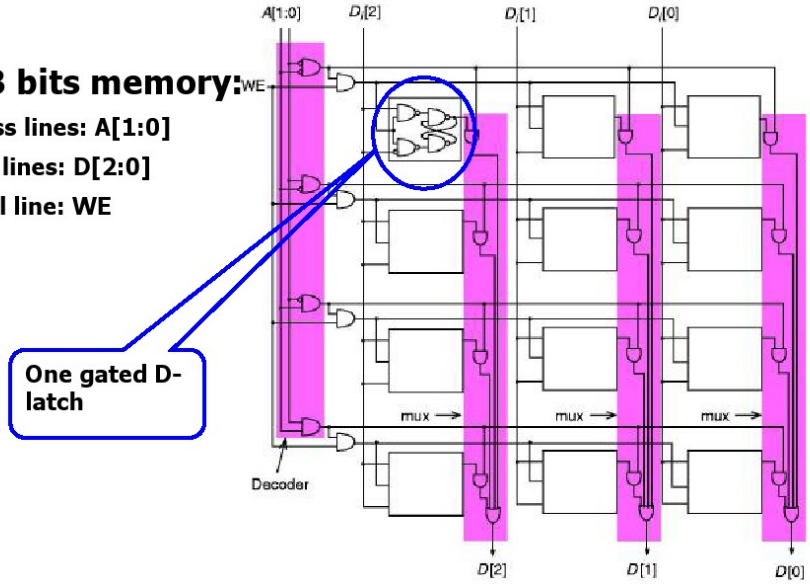
Memory Example

A 2^2 by 3 bits memory:

two address lines: A[1:0]

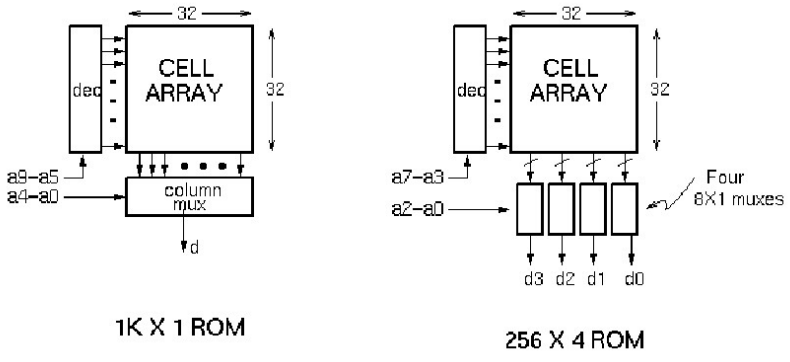
three data lines: D[2:0]

one control line: WE



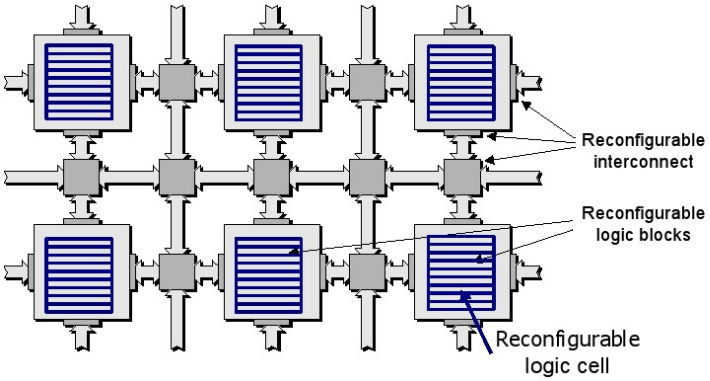
Column MUX in ROMs and RAMs:

- Controls physical aspect ratio
- In DRAM, allows reuse of chip address pins



Question 6:

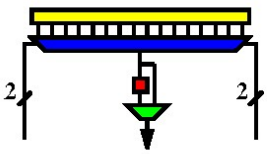
FPGA devices

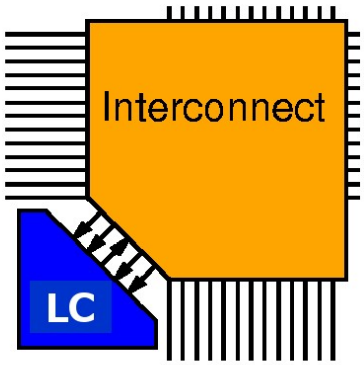


EDA322 Digital design, 2015-2016, Lecture 12 I. Sourdis, CSE, Chalmers 9

Logic Cells: Conventional FPGA Tile

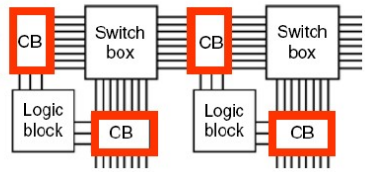
K-LUT (typical k=4) w/ optional output Flip-Flop

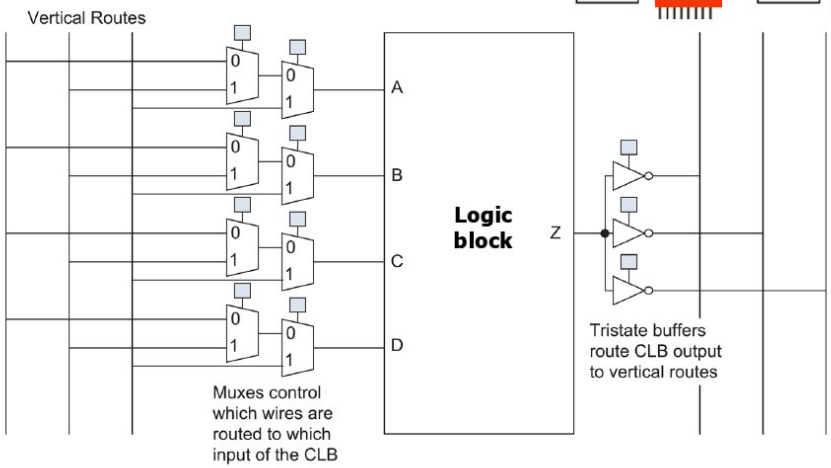




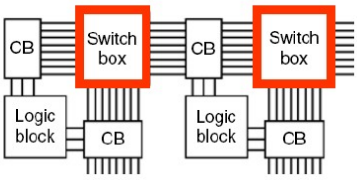
EDA322 Digital design, 2015-2016, Lecture 12 I. Sourdis, CSE, Chalmers 16

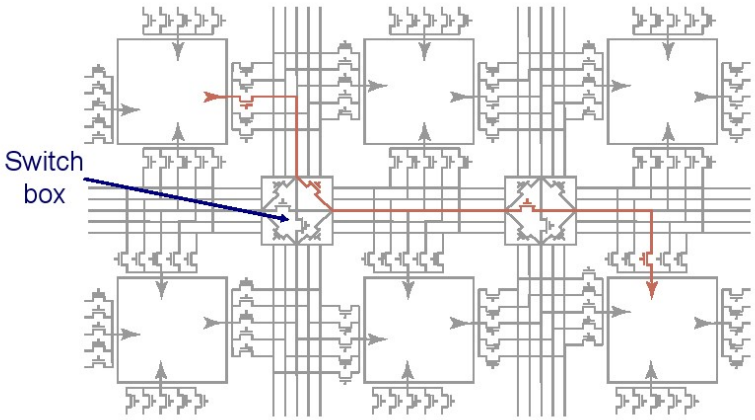
Connection





Switch blocks

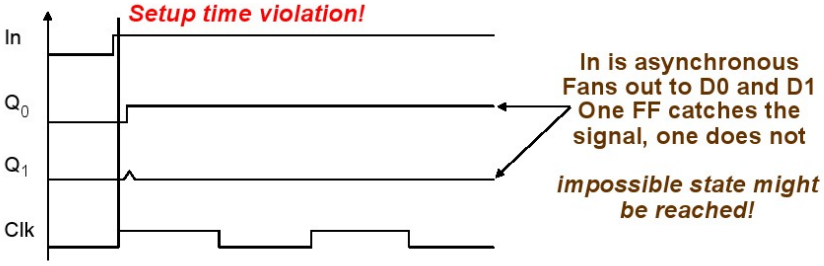




Question 8:

Metastability and Asynchronous Inputs

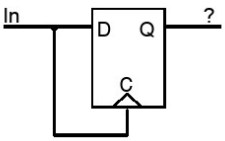
What Can Go Wrong



Single FF that receives the asynchronous signal is *a synchronizer*

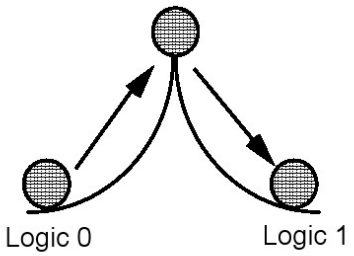
Metastability and Asynchronous Inputs

Synchronizer Failure

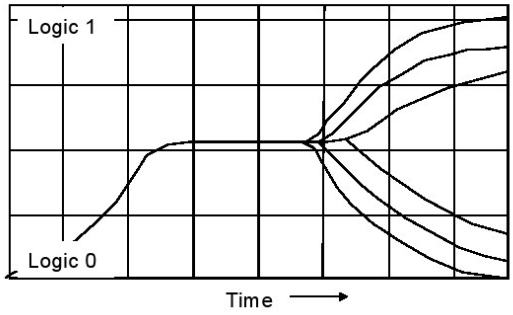


When FF input changes close to clock edge, the FF may enter the *metastable* state: neither a logic 0 nor a logic 1

It may stay in this state an indefinate amount of time, although this is not likely in real circuits



Small, but non-zero probability that the FF output will get stuck in an in-between state

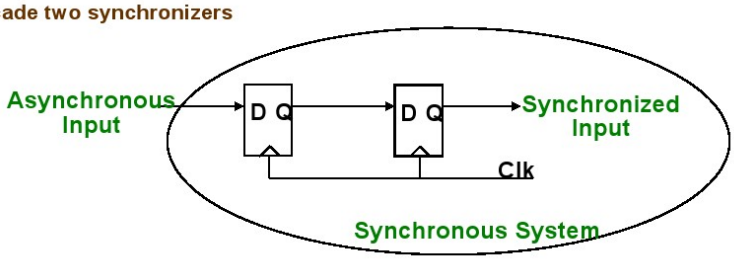


Oscilloscope Traces Demonstrating Synchronizer Failure and Eventual Decay to Steady State

Metastability and Asynchronous Inputs

Solutions to Synchronizer Failure

- the probability of failure can never be reduced to 0, but it can be reduced
- slow down the system clock
  this gives the synchronizer more time to decay into a steady state
  synchronizer failure becomes a big problem for very high speed systems
- use fastest possible logic in the synchronizer
  this makes for a very sharp "peak" upon which to balance
  S or AS TTL D-FFs are recommended
- cascade two synchronizers



Question 10:

Example1, cont'd

- For **fault propagation** it is required that node b=1, giving (t·a)' = t' at the output. Put w=0 to get b=1.
- The **fault activation** is achieved by putting x=0 and a=0, which requires y= z = 0.
- The testvector for the SA 0 is <xyzw> = 0000