EDA321: Digital Design
Re-Exam - January 2014

Date: January 13, 2014

Time: **8:30-12:30**

Examiner: Ioannis Sourdis

Department: Computer Science and Engineering

Inquiries: Ioannis Sourdis (extension 1744); will visit the room at **10:00** and at **11:30**

Results and grading review: See me in my office on **January 31 at 10 am**.

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: (15 points)

a) Find the minterms of the function F(x0,x1,x2,x3)= Π(3,4,11,12,13,15) (2 points)

Minimize the cost of function F.

b) Using Quine-McCluskey? (9 points)

c) using Karnaugh (4 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:

a) similar to Lecture 2 slide 13
b) Lecture 2 from slide 95 on
c) similar to Lecture 2 slides 19, 20, 21

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

Question 2: (5 points)

a) Describe the difference between an asynchronous and a synchronous reset in a D-flip-flop. Make a timing diagram showing the difference.

b) What is clock skew? Give an example where a clock skew may create wrong functionality of a synchronous circuit.

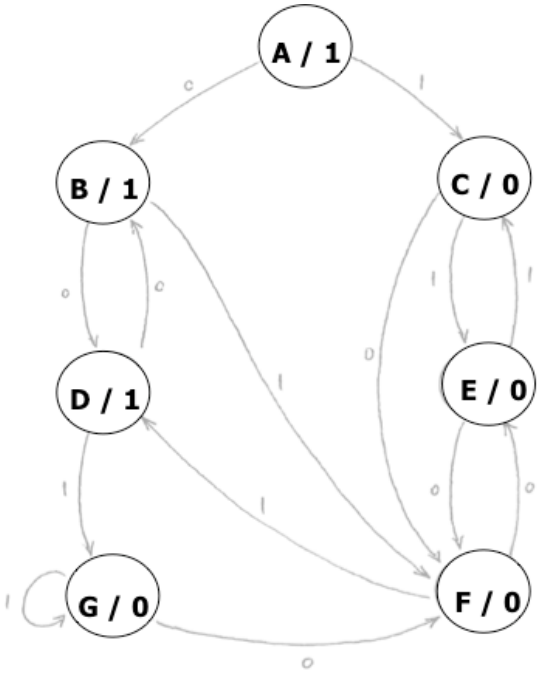
Answer:

a) Lecture 4 slide 25
b) Lecture 16, slide 23

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

Question 3: (8 points)

Minimize the states of the following FSM using partitioning. Draw the state diagram of the minimized FSM.



Answer:

a) Lecture 7 slide 49-57

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

Question 4: (15 points)

a) Describe an unsigned binary division between a 4-bit dividend x4x3x2x1 and a 4-bit divider y4y3y2y1. Draw the block diagram and the registers used, and explain the steps needed for each iteration. How many iterations are needed for a 4 bit division? (8 points)
b) Make the binary division between 8/3 showing each iteration and steps needed as you described them in (a)? (7 points)

Answer:

a) Lecture 11 slide 24 (modified for 4-bits instead of 32)
b) Lecture 11 slide 25-26

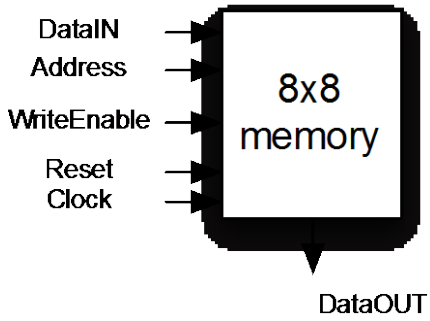
(Above mentioned lecture slides can be found at the end of this document)

Question 5: (7 points)

(a) (5 points)

Draw the block diagram of an 8x8 memory, using as "building blocks" D flip-flops and logic gates. The memory has the following inputs:

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



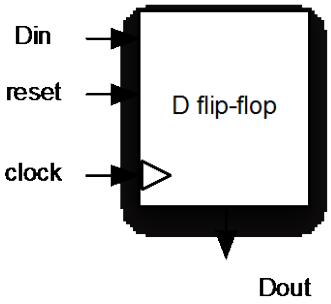
And the following output:

- DataOUT

Note: both "DataIN" and "DataOUT" are signals of 8 bits.

-How many bits is the "Address" signal?

The D flip-flop has the following interface:



(b) (2 points)
Taking the above 8x8 memory as a black box (not having the ability to change its internal design) what can we add to use it as a 64x1 memory?

Answer:

a) lecture 11, slide 34 (but for 4x4 instead of 4x3)
b) a column multiplexer as shown in lecture 11, slide 39

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

Question 6: (10 points)
Considering the following Boolean functions:

1. $F_1 = A_0A_1A_3 + A_1A_2\bar{A}_3 + \bar{A}_0\,\bar{A}_1\,\bar{A}_2$
2. $F_2 = A_0A_{1+}A_3\bar{A}_2$

Map the above functions to an FPGA that has:

a) 2-input Lookup Tables (LUT)
b) 4-input LUTs

Draw the block diagram for each case:
What is the minimum number of LUTs in each case (4 cases in total)?

Considering that a 2-input LUT has area A, a 4-input LUT has area 4*A, and each wire area A/4, what is the best LUT size (2-input LUTs or 4-input LUTs) that gives the lower area for each function?

Answer:
a.1: 7 LUT, 15 wires, area = 7*A+15/4*A=10.25*A
a.2: 1 LUT, 5 wires, area = 4*A+5/4*A=**5.25*A**
b.1: 3LUTs, 7 wires, area = 3*A+7/4*A=**4.75*A**
b.2: 1 LUT, 5 wires, area = 4*A+5/4*A=5.25*A

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).

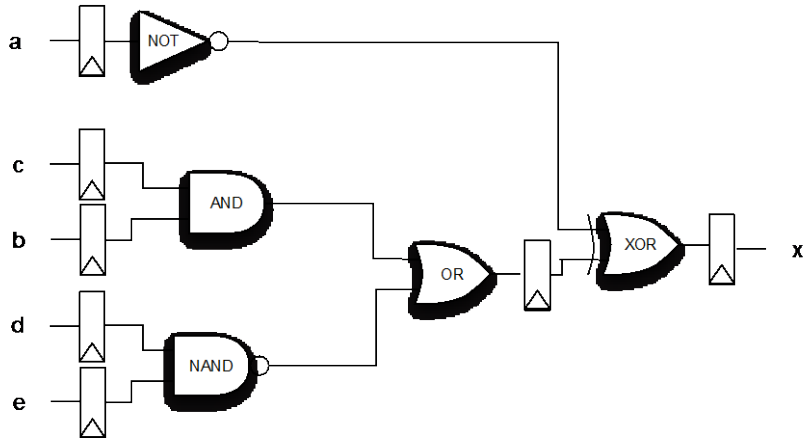
Question 8: (10 points)
Explain the timing constraints of a D flip-flop.

a) What is the propagation time, setup time and hold time? (6 points)
b) Make a timing diagram to show the above time constraints in a D flip-flop. (3 points)
c) Explain when a flip-flop may enter a metastable state. (3 points)

Answer:
Lecture 16 slides 18, 19, 31

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

- *a NOT gate has a delay of 1 ns*
- *an AND gate has a delay of 3 ns*
- *a NAND gate has a delay of 2 ns*
- *a XOR gate has a delay of 7 ns*
- *an OR gate has a delay of 3 ns*
- *Propagation time (clock to output) for a flip-flop 1ns*
- *Setup time for a flip-flop 2ns*
- *Wires have zero delay*
- *All inputs (a, b, c, d, e) have zero delay*



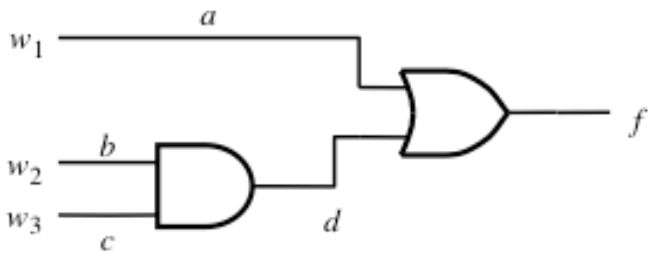
Answer:

The Critical path is from the output of the flip-flops connected to "a" to the input of the flip-flop at the output.

> Max Delay = (Propagation delay of the flip-flop) + (Delay of the NOT gate) + (Delay of the XOR gate) + (Setup-time of the flip-flop) ⇔ Max Delay = 1ns + 1ns + 7ns +2ns = 11ns

Question 10: (10 points)

Find the test-set of inputs that detect all the single stuck-at faults for the following circuit:



Answer:

Lecture 13, slide 24.

END of EXAM

Lecture Slides

Question 1:

Conversion between canonical forms

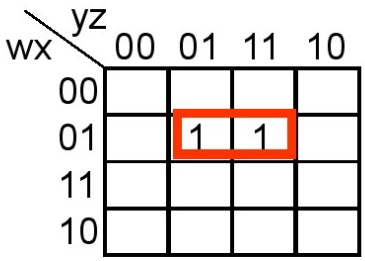
Relation between maxterms and minterms of a function

$$F(x,y,z) = \Sigma(1,3,5,6,7) = \Pi(0,2,4)$$

… and for the inverted function:

$$F'(x,y,z) = \Pi(1,3,5,6,7) = \Sigma(0,2,4)$$

Karnaugh-diagram

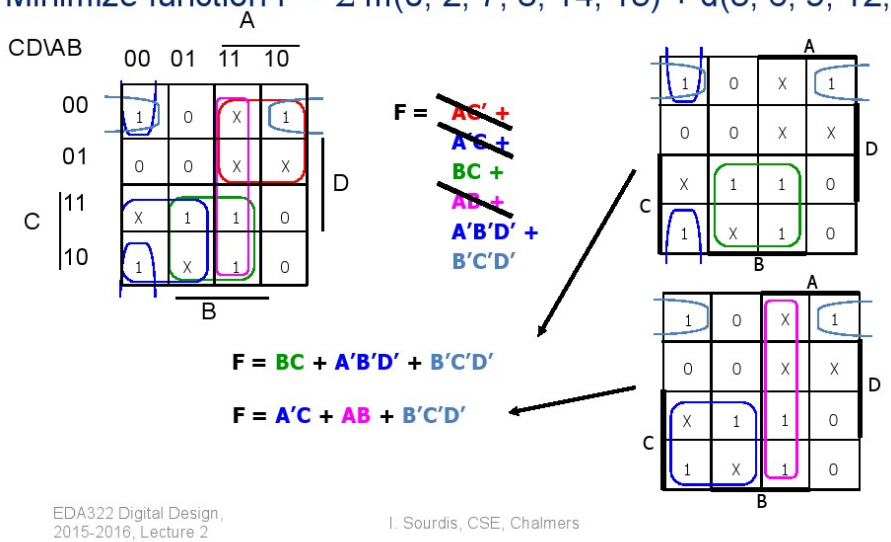


W'XY'Z + W'XYZ =
(W'XZ)(Y'+Y) = W'XZ

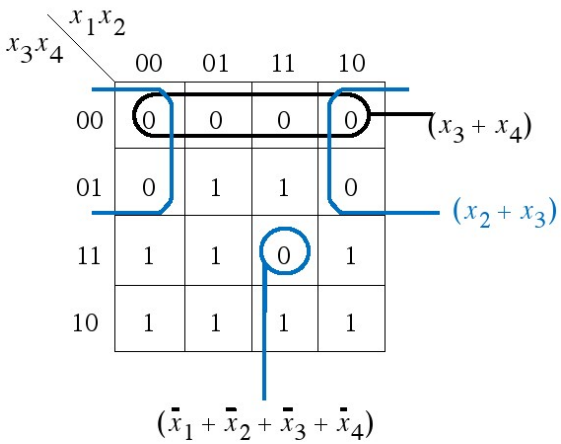
Note: (Y'+Y)=1

Karnaugh of 4 variables

Minimize function F = Σ m(0, 2, 7, 8, 14, 15) + d(3, 6, 9, 12, 13)



Product of Sums (maxterms)



$f = (x_3+x_4)(x_2+x_3)(\bar{x}_1+\bar{x}_2+\bar{x}_3+\bar{x}_4)$

$\text{POS } f(x_1,\ldots,x_4) = \Pi M(0, 1, 4, 8, 9, 12, 15).$

How would one choose between PoS and SoP?

Quine-McCluskey Method
An Example

1. Find all the prime implicants

$$f(a,b,c,d) = \sum m(0,1,2,5,6,7,8,9,10,14)$$

group 0	0	0000
group 1	1	0001
	2	0010
	8	1000
group 2	5	0101
	6	0110
	9	1001
	10	1010
group 3	7	0111
	14	1110

Group the minterms according to the number of 1s in the minterm.

This way we only have to compare minterms from adjacent groups.

Quine-McCluskey Method
An Example

Column I			Column II		Column III	
group 0	0	0000 ✓	0,1	000- ✓	0,1,8,9	-00-
group 1	1	0001 ✓	0,2	00-0 ✓	0,2,8,10	-0-0
	2	0010 ✓	0,8	-000 ✓	0,8,1,9	-00-
	8	1000 ✓	1,5	0-01	0,8,2,10	-0-0
group 2	5	0101 ✓	1,9	-001 ✓	2,6,10,14	--10
	6	0110 ✓	2,6	0-10 ✓	2,10,6,14	--10
	9	1001 ✓	2,10	-010 ✓		
	10	1010 ✓	8,9	100- ✓		
group 3	7	0111 ✓	8,10	10-0 ✓		
	14	1110 ✓	5,7	01-1		
			6,7	011-		
			6,14	-110 ✓		
			10,14	1-10 ✓		

No more combinations are possible, thus we stop here.

Quine-McCluskey Method
An Example

Column I		Column II	Column III
group 0	0 0000 ✓	0,1 000- ✓	0,1,8,9 -00-
group 1	1 0001 ✓	0,2 00-0 ✓	0,2,8,10 -0-0
	2 0010 ✓	0,8 -000 ✓	~~0,8,1,9 -00-~~
	8 1000 ✓	1,5 0-01	~~0,8,2,10 -0-0~~
group 2	5 0101 ✓	1,9 -001 ✓	2,6,10,14 --10
	6 0110 ✓	2,6 0-10 ✓	~~2,10,6,14 --10~~
	9 1001 ✓	2,10 -010 ✓	
	10 1010 ✓	8,9 100- ✓	
group 3	7 0111 ✓	8,10 10-0 ✓	
	14 1110 ✓	5,7 01-1	
		6,7 011-	
		6,14 -110 ✓	
		10,14 1-10 ✓	

We can eliminate repeated combinations

Quine-McCluskey Method
An Example

Column I		Column II	Column III
group 0	0 0000 ✓	0,1 000- ✓	0,1,8,9 -00-
group 1	1 0001 ✓	0,2 00-0 ✓	0,2,8,10 -0-0
	2 0010 ✓	0,8 -000 ✓	2,6,10,14 --10
	8 1000 ✓	1,5 0-01 ←	
group 2	5 0101 ✓	1,9 -001 ✓	
	6 0110 ✓	2,6 0-10 ✓	
	9 1001 ✓	2,10 -010 ✓	
	10 1010 ✓	8,9 100- ✓	
group 3	7 0111 ✓	8,10 10-0 ✓	
	14 1110 ✓	5,7 01-1	
		6,7 011-	
		6,14 -110 ✓	
		10,14 1-10 ✓	

Now we form f with the terms not checked

f = a' c' d

Quine-McCluskey Method
An Example

Column I		Column II		Column III
group 0	0 0000 ✓	0,1	000- ✓	0,1,8,9 -00-
group 1	1 0001 ✓	0,2	00-0 ✓	0,2,8,10 -0-0
	2 0010 ✓	0,8	-000 ✓	2,6,10,14 --10
	8 1000 ✓	1,5	0-01	
group 2	5 0101 ✓	1,9	-001 ✓	
	6 0110 ✓	2,6	0-10 ✓	
	9 1001 ✓	2,10	-010 ✓	
	10 1010 ✓	8,9	100- ✓	
group 3	7 0111 ✓	8,10	10-0 ✓	
	14 1110 ✓	5,7	01-1 ⟵	f = a' c' d + a' bd
		6,7	011-	
		6,14	-110 ✓	
		10,14	1-10 ✓	

Quine-McCluskey Method
An Example

Column I		Column II		Column III
group 0	0 0000 ✓	0,1	000- ✓	0,1,8,9 -00-
group 1	1 0001 ✓	0,2	00-0 ✓	0,2,8,10 -0-0
	2 0010 ✓	0,8	-000 ✓	2,6,10,14 --10
	8 1000 ✓	1,5	0-01	
group 2	5 0101 ✓	1,9	-001 ✓	
	6 0110 ✓	2,6	0-10 ✓	
	9 1001 ✓	2,10	-010 ✓	
	10 1010 ✓	8,9	100- ✓	
group 3	7 0111 ✓	8,10	10-0 ✓	
	14 1110 ✓	5,7	01-1	f = a' c' d + a' bd + a' bc
		6,7	011- ⟵	
		6,14	-110 ✓	
		10,14	1-10 ✓	

Quine-McCluskey Method
An Example

	Column I	Column II	Column III
group 0	0 0000 ✓	0,1 000- ✓	0,1,8,9 -00- ⇐
group 1	1 0001 ✓	0,2 00-0 ✓	0,2,8,10 -0-0
	2 0010 ✓	0,8 -000 ✓	2,6,10,14 --10
	8 1000 ✓	1,5 0-01	
group 2	5 0101 ✓	1,9 -001 ✓	
	6 0110 ✓	2,6 0-10 ✓	
	9 1001 ✓	2,10 -010 ✓	
	10 1010 ✓	8,9 100- ✓	
group 3	7 0111 ✓	8,10 10-0 ✓	f = a' c' d + a' bd + a' bc + b' c'
	14 1110 ✓	5,7 01-1	
		6,7 011-	
		6,14 -110 ✓	
		10,14 1-10 ✓	

Quine-McCluskey Method
An Example

	Column I	Column II	Column III
group 0	0 0000 ✓	0,1 000- ✓	0,1,8,9 -00-
group 1	1 0001 ✓	0,2 00-0 ✓	0,2,8,10 -0-0 ⇐
	2 0010 ✓	0,8 -000 ✓	2,6,10,14 --10
	8 1000 ✓	1,5 0-01	
group 2	5 0101 ✓	1,9 -001 ✓	
	6 0110 ✓	2,6 0-10 ✓	
	9 1001 ✓	2,10 -010 ✓	
	10 1010 ✓	8,9 100- ✓	
group 3	7 0111 ✓	8,10 10-0 ✓	f = a' c' d + a' bd + a' bc + b' c'
	14 1110 ✓	5,7 01-1	+ b' d'
		6,7 011-	
		6,14 -110 ✓	
		10,14 1-10 ✓	

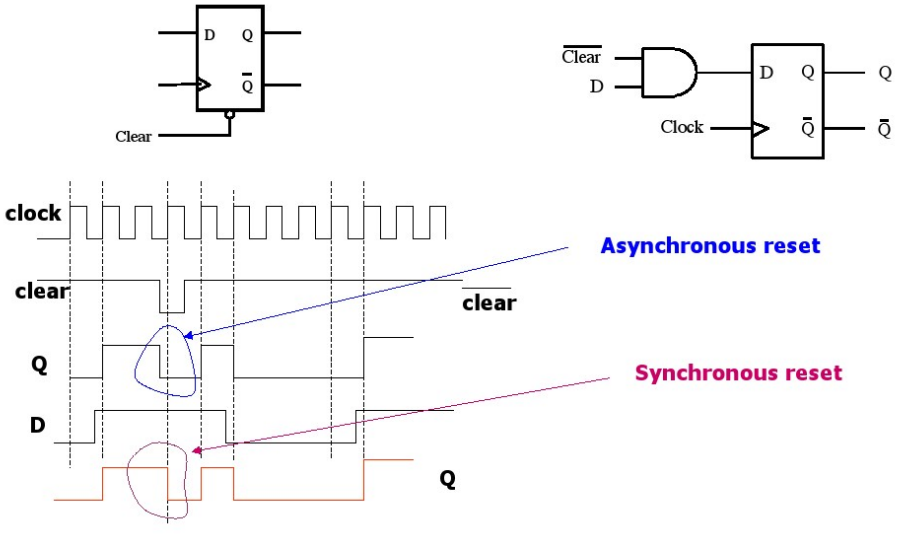
Quine-McCluskey Method
An Example

Column I		Column II	Column III
group 0	0 0000 ✓	0,1 000- ✓	0,1,8,9 -00-
group 1	1 0001 ✓	0,2 00-0 ✓	0,2,8,10 -0-0
	2 0010 ✓	0,8 -000 ✓	2,6,10,14 --10 ⇐
	8 1000 ✓	1,5 0-01	
group 2	5 0101 ✓	1,9 -001 ✓	
	6 0110 ✓	2,6 0-10 ✓	
	9 1001 ✓	2,10 -010 ✓	
	10 1010 ✓	8,9 100- ✓	
group 3	7 0111 ✓	8,10 10-0 ✓	
	14 1110 ✓	5,7 01-1	
		6,7 011-	
		6,14 -110 ✓	
		10,14 1-10 ✓	

f = a' c' d + a' bd + a' bc + b' c' + b' d' + cd'

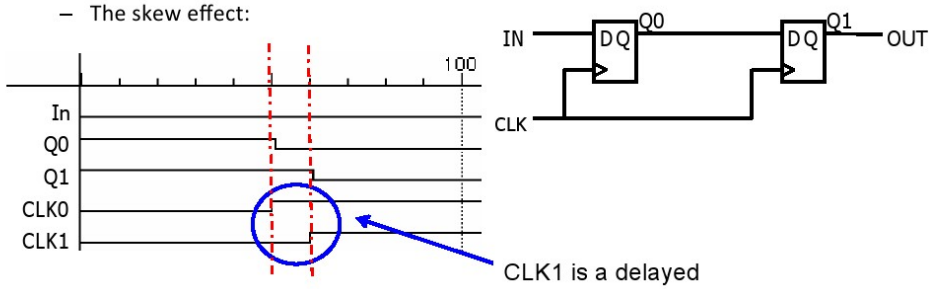
Question 2:

Reset of D flip-flop



Clock Skew

- The Problem
  - correct behavior requires that the next state of all memory elements are determined by all the memory elements thus at the same time
  - This is difficult in high performance systems since the time it takes for the clock to arrive, are of the same magnitude as the delay through the logic
  - The skew effect:



Initial state: IN = 0, Q0 = 1, Q1 = 1
Due to skew next state is : Q0 = 0, Q1 = 0,
instead of Q0 = 0, Q1 = 1

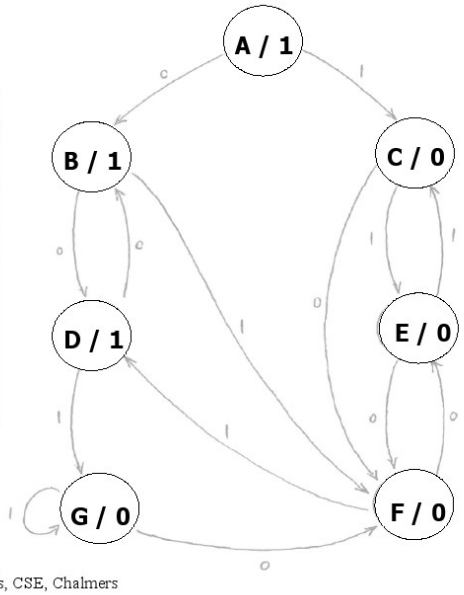
Question 3:

State Minimization: Partitioning

- State Minimization through Partitioning:
  - Form an initial partition (P_1) that includes **all** states.
  - Form a second partition (P_2) by separating the states into blocks based upon their output values.
  - Form a third partition (P_3) by separating the states into blocks corresponding to the next state values.
  - Continue partitioning until two successive partitions are the same (i.e. $P_{N-1} = P_N$).
  - All states in any one block are equivalent.
    - Equivalent states can be combined into a single state.

State Minimization: Partitioning

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0



State Minimization: Partitioning

Initial Partition:

P_1 = (ABCDEFG)

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

The initial partition contains all states in the state diagram / table.

State Minimization: Partitioning

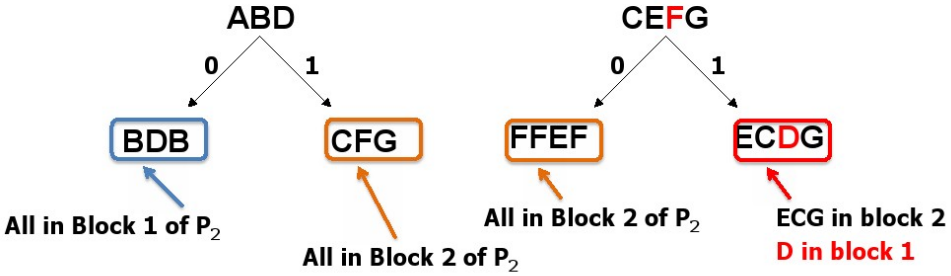
- Separate states based on output value.
  - P_2 = (ABD)(CEFG)

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

State Minimization: Partitioning

- Separate states based on next state values.
- P_2 = (ABD)(CEFG)

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

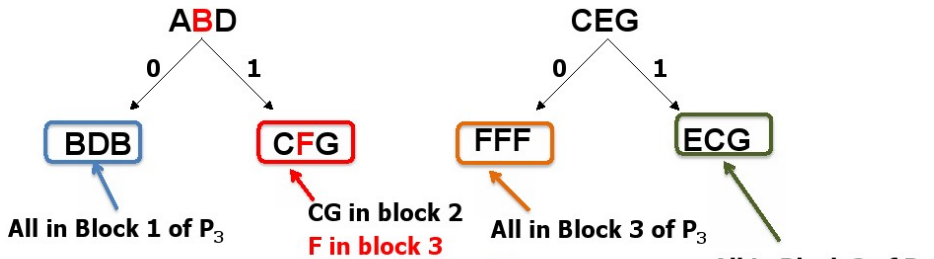


- P_3 = (ABD)(CEG)(F) → unique state

State Minimization: Partitioning

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

- Separate states based on next state values.
- P_3 = (ABD)(CEG)(F)



- P_4 = (AD)(CEG)(F)(B)

unique states

State Minimization: Partitioning

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

- Separate states based on next state values.

- P_5 = (AD)(CEG)(F)(B)

P_5 same as previous partition (P_4) => state minimization is completed

State Minimization: Partitioning

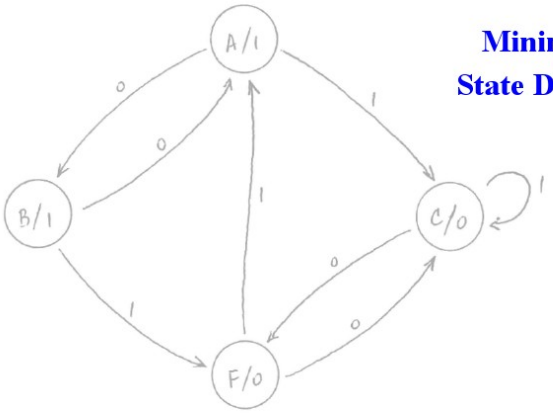
- Since $P_4 = P_5$, state minimization is complete.
- The equivalent states are:
  - A = D
  - C = E = G
  - B
  - F
- Thus, the FSM can be realized with just 4 states.

FSM: State Minimization

Present state	Next state		Output
	w = 0	w = 1	z
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Minimized State Table

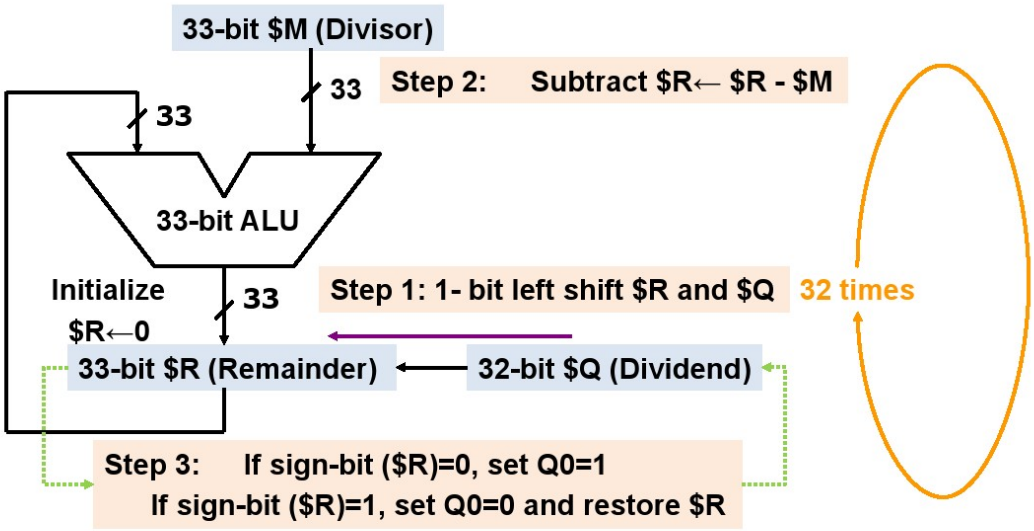
FSM: State Minimization



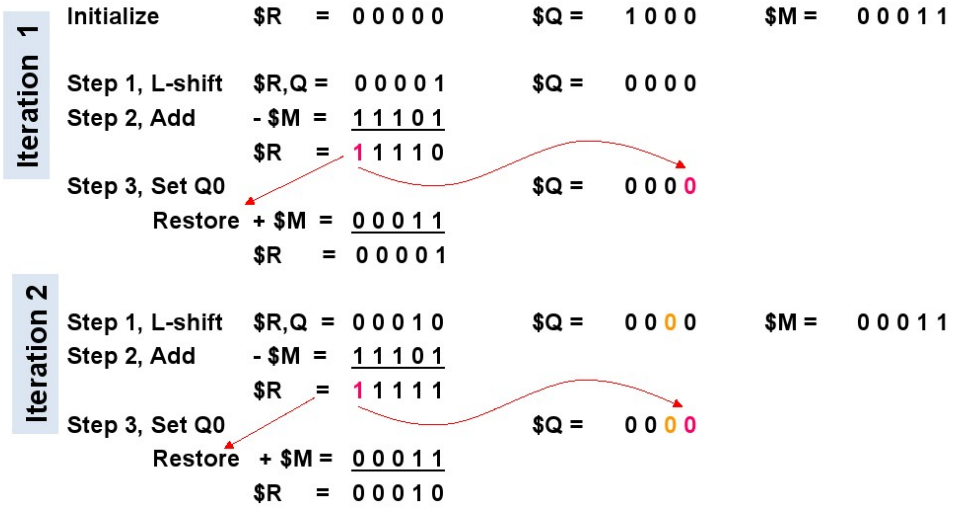
Minimized State Diagram

Question 4:

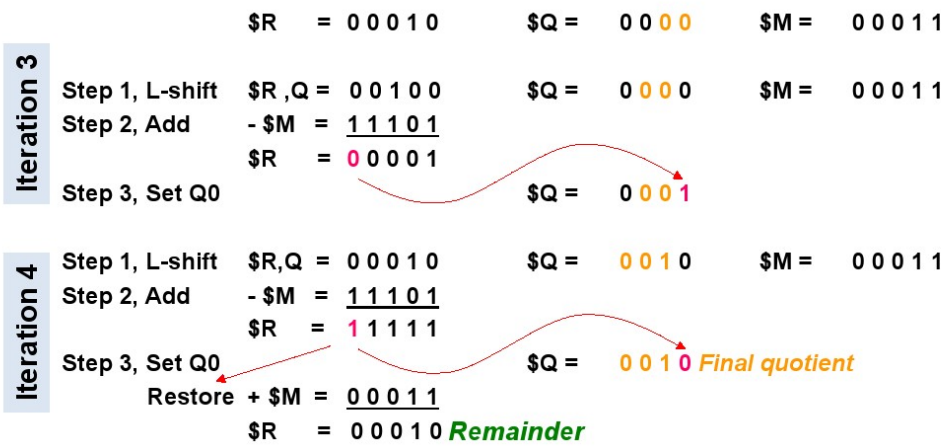
Division



Ex: 8/3 = 2, Remainder = 2



Ex: 8/3 = 2 (Remainder = 2) (Con'd)



Note "Restore $R" in Steps 1, 2 and 4. This method is known as *the RESTORING DIVISION. An improved method, NON-RESTORING DIVISION, is possible (Hamacher, et al.)*

Question 5:

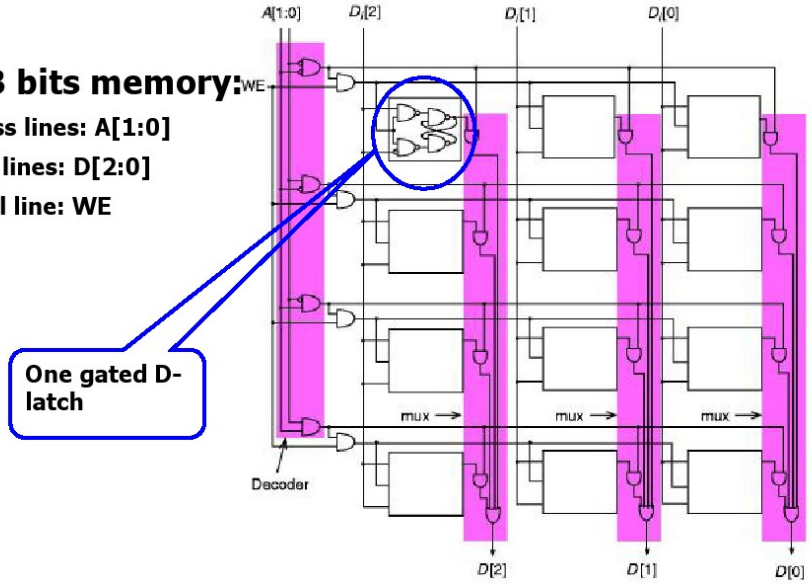
Memory Example

A 2^2 by 3 bits memory:

two address lines: A[1:0]

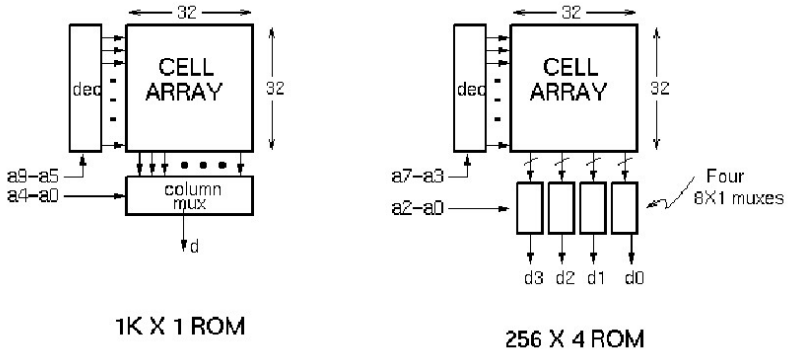
three data lines: D[2:0]

one control line: WE



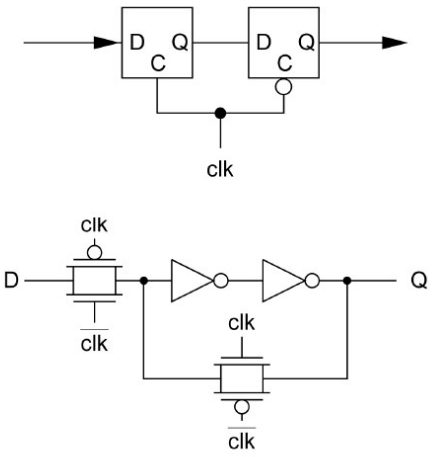
Column MUX in ROMs and RAMs:

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



Question 8:

Delay in Flip-flops



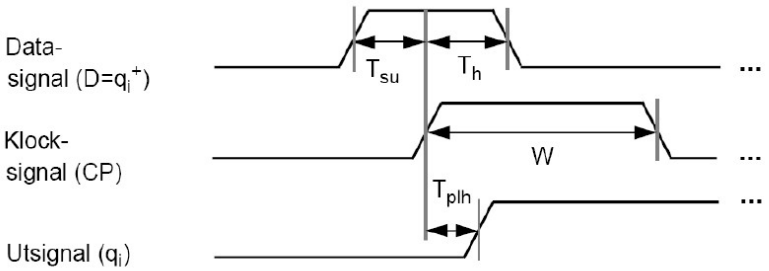
T_{su}= setup time
(the time the input of a flip flop needs to be stable before the clock edge to be stored correctly)

T_h= hold time
(the time the input of a flip flop needs to be held after the clock edge to be stored correctly)

T_p=propagation time (clock to Q)
(the time a flip-flop needs to propagate a value stored to the output)

EDA322 Digital design, 2015-2016, Lecture 16 I. Sourdis, CSE, Chalmers Page 18

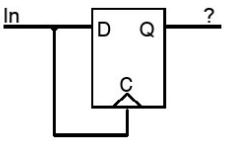
Time parameters for clocked memory elements



EDA322 Digital design, 2015-2016, Lecture 16 I. Sourdis, CSE, Chalmers 19

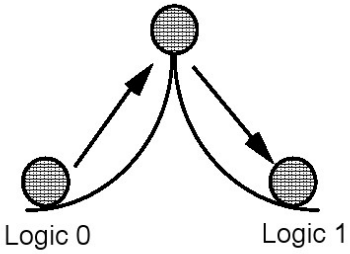
Metastability and Asynchronous Inputs

Synchronizer Failure

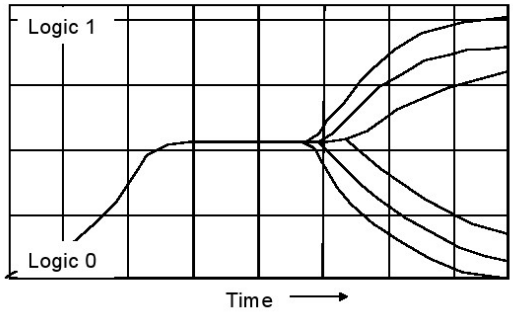


When FF input changes close to clock edge, the FF may enter the *metastable* state: neither a logic 0 nor a logic 1

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

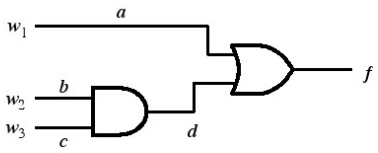


Oscilloscope Traces Demonstrating Synchronizer Failure and Eventual Decay to Steady State

EDA322 Digital design, 2015-2016, Lecture 16 — I. Sourdis, CSE, Chalmers — Page 31

Question 10:

Testset



One possible approach but unrealistic for big circuits!

Test $w_1 w_2 w_3$	Fault detected									
	a/0	*a*/1	*b*/0	*b*/1	*c*/0	*c*/1	*d*/0	*d*/1	*f* /0	*f* /1
000		√						√		√
001		√		√				√		√
010		√				√		√		√
011			√		√		√		√	
100	√								√	
101	√								√	
110	√								√	
111									√	