EDA321/EDA322: Digital Design
Re-exam - August 2015

Date: August 27, 2015

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

Examiner: Ioannis Sourdis

Department: Computer Science and Engineering

Inquiries: Ioannis Sourdis (extension 1744); will visit the room at **15:30** and at **17:00**

Results and grading review: See me in my office 4110 on **September 21st at 10:00**.

Exam duration: 4 hours

Grading scale: 100 points in total

Chalmers:
0: 0%-49%, 3: 50%-64%, 4: 65%-84%, 5: 85%-100%
GU:
Fail (U): 0%-49%, Pass (G): 50%-79%, Pass with Distinction (VG): 80%-100%

Available references: Only 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)
Minimize the cost of function:

F(x4,x3,x2,x1,x0) = Σ(4,8,11,19,20,24,27) + D(1,7,9,15,16,17,23,25,31)

using Quine-McCluskey. 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:

Truth table for the minterms Σ(4,8,11,19,20,24,27):

i	x4	x3	x2	x1	x0	y
0	0	0	0	0	0	0
1	0	0	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	1	1	0
4	0	0	1	0	0	1
5	0	0	1	0	1	0
6	0	0	1	1	0	0
7	0	0	1	1	1	0
8	0	1	0	0	0	1
9	0	1	0	0	1	0
10	0	1	0	1	0	0
11	0	1	0	1	1	1
12	0	1	1	0	0	0
13	0	1	1	0	1	0
14	0	1	1	1	0	0
15	0	1	1	1	1	0
16	1	0	0	0	0	0
17	1	0	0	0	1	0
18	1	0	0	1	0	0
19	1	0	0	1	1	1
20	1	0	1	0	0	1
21	1	0	1	0	1	0
22	1	0	1	1	0	0
23	1	0	1	1	1	0
24	1	1	0	0	0	1
25	1	1	0	0	1	0
26	1	1	0	1	0	0
27	1	1	0	1	1	1
28	1	1	1	0	0	0
29	1	1	1	0	1	0
30	1	1	1	1	0	0
31	1	1	1	1	1	0

Truth table for the don't care terms D(1,7,9,15,16,17,23,25,31):

i	x4	x3	x2	x1	x0	h*
0	0	0	0	0	0	0
1	0	0	0	0	1	1
2	0	0	0	1	0	0
3	0	0	0	1	1	0
4	0	0	1	0	0	0
5	0	0	1	0	1	0
6	0	0	1	1	0	0
7	0	0	1	1	1	1
8	0	1	0	0	0	0
9	0	1	0	0	1	1
10	0	1	0	1	0	0
11	0	1	0	1	1	0
12	0	1	1	0	0	0
13	0	1	1	0	1	0
14	0	1	1	1	0	0
15	0	1	1	1	1	1
16	1	0	0	0	0	1
17	1	0	0	0	1	1
18	1	0	0	1	0	0
19	1	0	0	1	1	0
20	1	0	1	0	0	0
21	1	0	1	0	1	0
22	1	0	1	1	0	0
23	1	0	1	1	1	1
24	1	1	0	0	0	0
25	1	1	0	0	1	1
26	1	1	0	1	0	0
27	1	1	0	1	1	0
28	1	1	1	0	0	0
29	1	1	1	0	1	0
30	1	1	1	1	0	0
31	1	1	1	1	1	1

The minterms and don't care terms are put together:

Step 1: step2: step3:

Indexgruppe	Index
0	
1	1 * 4 * 8 * 16 *
2	9 * 17 * 20 * 24 *
3	7 * 11 * 19 * 25 *
4	15 * 23 * 27 *
5	31 *

Indexgruppe	Index
0/1	
1/2	9,1 (8) * 17,1 (16) * 20,4 (16) P1 9,8 (1) * 24,8 (16) * 17,16 (1) * 20,16 (4) P2 24,16 (8) *
2/3	11,9 (2) * 25,9 (16) * 19,17 (2) * 25,17 (8) * 25,24 (1) *
3/4	15,7 (8) * 23,7 (16) * 15,11 (4) * 27,11 (16) * 23,19 (4) * 27,19 (8) * 27,25 (2) *
4/5	31,15 (16) * 31,23 (8) * 31,27 (4) *

Indexgruppe	Index
0/1/2	
1/2/3	25,17,9,1 (8,16) P3 25,9,17,1 (16,8) 25,24,9,8 (1,16) P4 25,9,24,8 (16,1) 25,24,17,16 (1,8) P5 25,17,24,16 (8,1)
2/3/4	27,25,11,9 (2,16) P6 27,11,25,9 (16,2) 27,25,19,17 (2,8) P7 27,19,25,17 (8,2)
3/4/5	31,23,15,7 (8,16) P8 31,15,23,7 (16,8) 31,27,15,11 (4,16) P9 31,15,27,11 (16,4) 31,27,23,19 (4,8) P10 31,23,27,19 (8,4)

P1= x3' x2 x1' x0'
P2= x4 x3' x1' x0'
P3= x2' x1' x0
P4= x3 x2' x1'
P5= x4 x2' x1'
P6= x3 x2' x0
P7= x4 x2' x0
P8= x2 x1 x0
P9= x3 x1 x0
P10= x4 x1 x0

P/M1	4	8	11	19	20	24	27
P1	*				*		
P2					*		
P3							
P4		*				*	
P5						*	
P6			*				*
P7				*			*
P8							
P9			*				*
P10				*			*

There are 4 possible minimized functions:

F1 = P1 + P4 + P6 +P7 = x3' x2 x1' x0' + x3 x2' x1' + x3 x2' x0 + x4 x2' x0
F2 = P1 + P4 + P6 +P10 = x3' x2 x1' x0' + x3 x2' x1' + x3 x2' x0 + + x4 x1 x0
F3 = P1 + P4 + P7 +P9 = x3' x2 x1' x0' + x3 x2' x1' + x4 x2' x0 + x3 x1 x0
F4 = P1 + P4 + P9 +P10 = x3' x2 x1' x0' + x3 x2' x1' + x3 x1 x0 + x4 x1 x0

Cost is 12.

Question 2: (20 points)

Design a synchronous FSM (Finite State Machine) for an automatic machine that receives empty plastic bottles for recycling and gives back a supermarket-voucher (1 SEK for every bottle received). The machine stays idle and ready until somebody puts a bottle in it. Then, it checks whether the bottle received is "good" or "bad". If the bottle is good it accepts it otherwise it rejects it. After that it waits for more bottles or for the "finish" button to be pressed. It accepts up to 7 bottles. When the "finish" button is pressed or an 7th bottle is given then the machine prints the supermarket voucher with the correct amount of SEK to be given to the client (the number of SEK should be equal to the number of bottles received).

The FSM has the following inputs:

- **RST**: FSM reset. If '1' go to initial state.
- **GoodBottle**: if '1' a good bottle has been received
- **BadBottle**: if '1' a bad bottle has been received
- **Finish**: if '1' a client pressed the "finish" button

Note: the inputs **GoodBottle** and **BadBottle** cannot be '1' at the same time.

The FSM will produce the following outputs:

- **Print**: Print command (when '1' a supermarket-voucher should be printed)
- **SEK**: The number of SEK to be printed on the voucher / equal to the number of bottles inserted (this is a multiple bits output, the number of bits depends on the FSM implementation of your choice)
- **TryAgain**: When a bad bottle is received this output becomes '1' to inform the client and ask to try again.
- **RDY**: Ready (if '1' the machine is ready to serve the next client)

a) Draw the state diagram of your FSM. (8 points)
   - Values of Outputs should be written either in the states or in the transitions (arrows) depending on the FSM choice. *Note: omitted (not mentioned) outputs are assumed to be zero.*
   - On arrows that denote transition from one state to another indicate the input values that cause this transition. In case the

combination of multiple inputs cause a transition, write the (Boolean) expression that causes the transition:
e.g. Input1 or (Input2 and Input3)

- Explain your choice of FSM type. What would be different if you had chosen a different FSM type?

b) Draw the state table and assign values to the states (state assignment). Select a state encoding and justify your choice. (4 points)

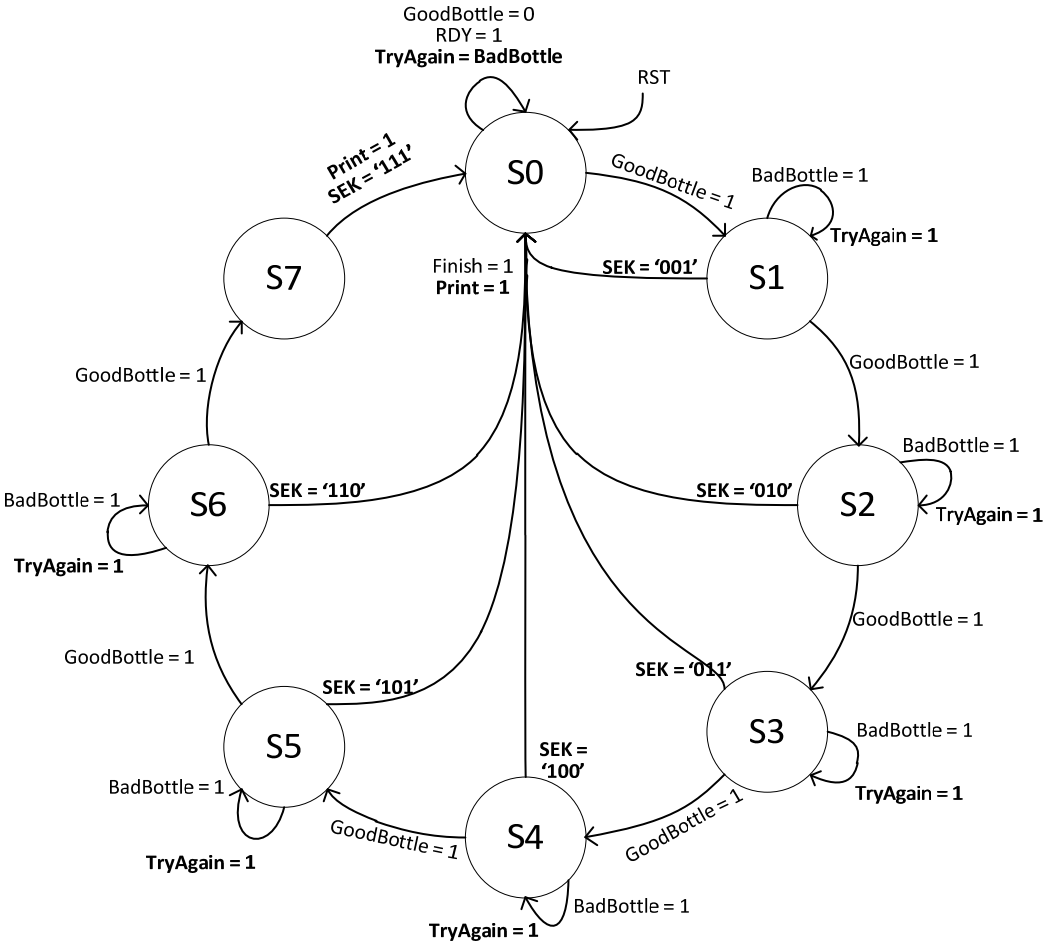
c) Retrieve the Boolean functions that implement the counter using D Flip-flops and draw the circuit of the counter at the gate-level (no need to show the internals of each flip-flop in your drawing). (8 points)

Note: if there is any information missing in the above description of the FSM, feel free to define it yourselves. In such case, explain it in your answer.

Asnwer:

a)

Mealy Machine state diagram



The Mealy choice produces fewer states than Moore. WE could optimize further the above Mealy machine by omitting S7. In a Moore extra states for the bad bottles and the finish cases would be needed.

b)

Present State		**Next State**			**Outputs**				
		Finish = 0		Finish = 1	Print	SEK		TryAgain	RDY
Name	$X_2X_1X_0$	GoodBottle=0	GoodBottle=1			Finish = 0	Finish = 1		
S0	000	S0	S1	Don't Care	0	Don't Care	000	BadBottle	Not GoodBottle
S1	001	S1	S2	S0	Finish	Don't Care	001	BadBottle	0
S2	010	S2	S3	S0	Finish	Don't Care	010	BadBottle	0
S3	011	S3	S4	S0	Finish	Don't Care	011	BadBottle	0
S4	100	S4	S5	S0	Finish	Don't Care	100	BadBottle	0
S5	101	S5	S6	S0	Finish	Don't Care	101	BadBottle	0
S6	110	S6	S7	S0	Finish	Don't Care	110	BadBottle	0
S7	111	S0	S0	S0	1	111	111	0	0

Binary encoding was chosen to economize on the number of bits (and D-flip-flops) need. Gray encoding may have resulted in less complex Boolean expressions (as the transition from one state to the next would require a change of a single bit).

c)

Boolean expression for the outputs can be obtained from Karnaugh Maps or just as follows:

Print = Finish . $(X_0 + X_1 + X_2) + (X_0.X_1.X_2)$
SEK(2:0) = X(2:0) . (Finish + $X_0.X_1.X_2$)
TryAgain = BadBottle(X_0.**Error! Bookmark not defined.**$X_1.X_2$)**'**
RDY = X_0**'**.X_1**'**.X_2**' .** GoodBottle**'**

Calculating Next State Boolean expressions:

If Finish = 0

GoodBottle	X_2	X_1	X_0	N_2	N_1	N_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	1	0	0
0	1	0	1	1	0	1
0	1	1	0	1	1	0
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	1
1	0	1	1	1	0	0
1	1	0	0	1	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	0	0	0

If Finish = 1, Next State = '000'
(so we need to AND all expressions with "NOT finish")

For N_0:

GB.X_2 \ X_1X_0	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	1

$N_0 = GB' X_0 \text{ Finish}'$

For N_1:

GB.X_2 \ X_1X_0	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	1	0	1
10	0	1	0	1

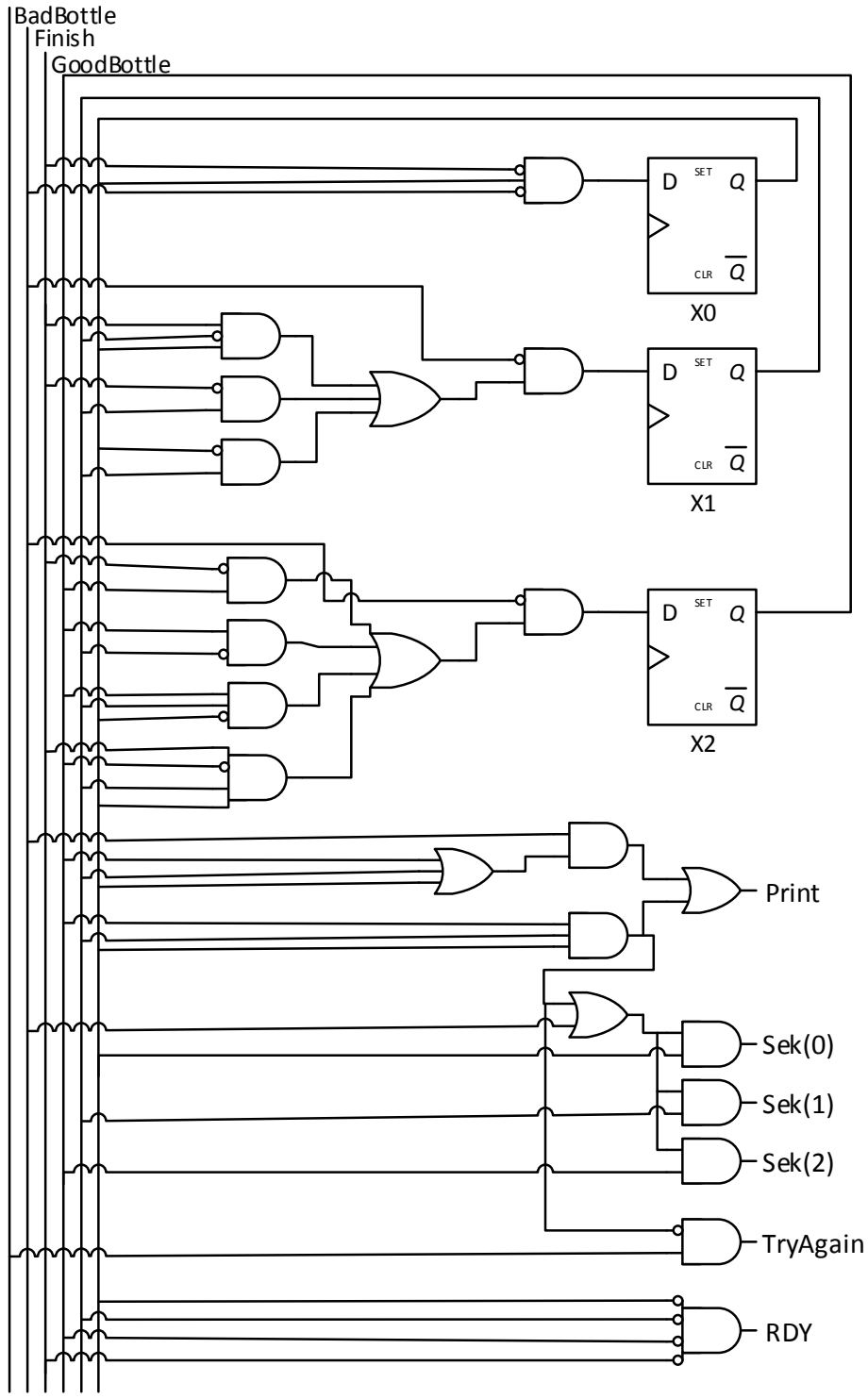
$N_1 = (GB' X_1' X_0 + GB' X_1 + X_0 X_1) \text{ Finish}'$

For N_2:

X_1X_0

GB.X_2	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	0	1
10	0	0	1	0

$N_2 = (GB' X_2 + X_2 X_1' + X_2 X_1 X_0' + GB X_2' X_1 X_0)$ Finish'



Question 3: (10 points)

a) Consider a 3-input FPGA logic cell. This logic cell includes a memory block to implement a Lookup Table (LUT). How many rows and columns does this memory block have?
b) Draw the block diagram (gatelevel) of the above memory block. *
c) Where are the inputs (Data-in and Address) and output (Data-out) of this memory block connected in the logic cell? Draw the block diagram of the FPGA logic cell to show the connections.
d) How can one use the above memory block to implement a 4-input FPGA logic cell? Draw the block diagram of the 4-input FPGA logic cell that uses multiple copies of this memory block. **

**Note: D-flip-flops and 1-bit memory cells can be drawn as a box (no need to show their circuit in gate-level)*
*** Note: There is no need to redraw the internals of the memory block again.*

Answer:

a) 1 column (wide), 8 rows (memory elements)
b) similar to slide 33 in Lecture on Arithmetic and memories.
c) The address of the memory block is connected to the 3-bits input of the Logic Cell. The Data-in (1 bit) is used to program the LUT (store inputs to the memory block). Data-out (1-bit) is connected to the D-flipflop of the logic cell and the output multiplexer of the logic cell (which selects either the DFF output or the LUT output).
d) Similar to Lecture on Arithmetic and Memories, slide 37. Two 8x1 memory blocks are needed to form a memory (LUT) of 16x1 required for the 4-input logic cell. The connections are then similar to the 3-input logic cell.

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

Question 4: (20 points)

a) Draw the block diagram (in gate-level) of a 4-bit Array Multiplier *Note: parts of the design that are repeated multiple times can be shown in gate-level once and then copy their box multiple times.* (12 points)

b) Show the critical path of the array multiplier and calculate its delay considering that a XOR gate has a delay of 2 nsec, an AND gate has a delay of 1 nsec and an OR gate a delay of 1 nsec. (8 points)

Answer:

a) based on Lecture on "Arithmetic Units & Memories" slide 18

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

b) 29 nsec

Question 5: (10 points)
What are the three main reasons to use Reconfigurable devices (FPGAs)?

Answer:
(see also Lecture on Reconfigurable Hardware, slides 6-7)

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

1) As components of systems with low or medium production volume and/or with short time-to-market requirements. For low or medium production volume, it is more expensive to fabricate ASICs chips instead of using FPGAs due to the higher NRE (Non recurring Engineering) costs of ASICs. In addition, Time-to-market would be much shorter for FPGAs compared to ASICs chips.
2) To prototype an ASIC. We can first prototype a ASIC chip on FPGA before fabricating it. This improves and speeds-up debugging.
3) For accelerating or in general executing more efficiently (in terms of energy and power) various algorithms and applications. Many applications and algorithms run more efficiently in Reconfigurable Hardware (compared to software running on general purpose machines) because it offers parallelism, customization and adaptation.

Question 6: (10 points)
Fill the 3 gaps of the following statements:

"If we reduce the clock frequency of a particular digital circuit from 100MHz to 50 MHz and its voltage from 1 Volt to 0.9 Volts, then its power consumption (i) (*increases by/decreases to*) 40%.

Its energy consumption for reevaluating its output for a new input (ii) (*increases by/decreases to*) (iii) ___ %."

Note: "increases by 40%" means that it becomes 1.4 times the original value. "decreases to 40%" means it becomes 0.4 of the original value.

Answer:

If we reduce the clock frequency of a particular digital circuit from 100MHz to 50 MHz and its voltage from 1 Volt to 0.9 Volts, then its power consumption (i) *decreases to* 40%.

Reevaluating its output will take double the time (since frequency is reduced to half). Energy is the product of time and power so if for 100 MHz the circuit has

Energy consumption E=P*T then for 50MHz the energy consumption is E'=(P*0.4)*(2*T) = 0.8*E

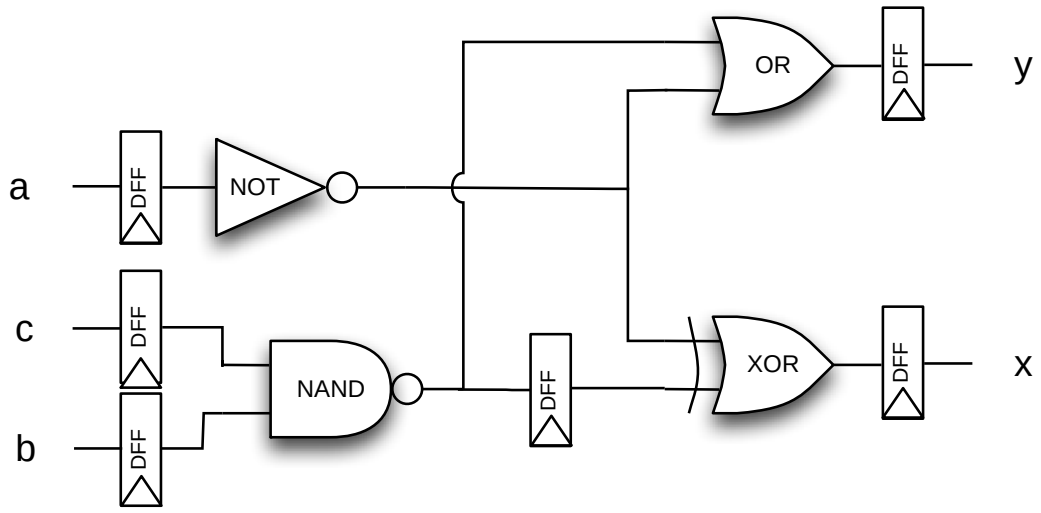
Consequently:

its energy consumption for reevaluating its output for a new input (ii) *decreases to* (iii) 80%.

Question 7: (10 points)

Calculate the maximum delay of the circuit (critical path delay), considering the following:

- a NOT gate has a delay of 1 ns
- a NAND gate has a delay of 2 ns
- an OR gate has a delay of 2.5 ns
- a XOR gate has a delay of 4 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) have zero delay



Answer:

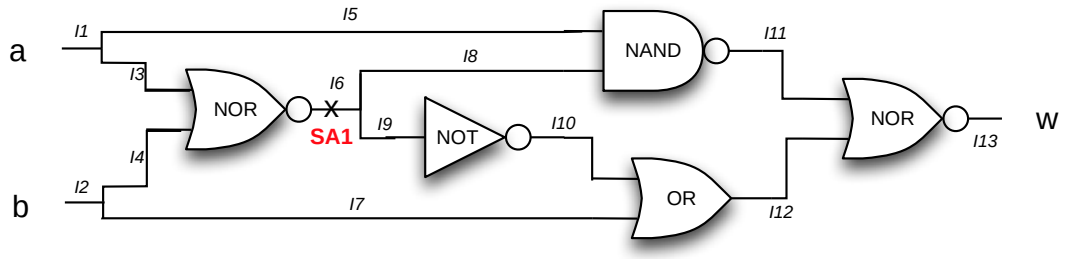
After checking all paths (from inputs to the next DFF or output, from DFF to the next DFF or output), the critical path of the circuit is:

From the output of the DFF after input "a" through the NOT gate, the XOR gate to the DFF before output x.

Delay = (DFF prop. time) + NOT_delay + XOR_delay + (DFF setup-time)
= 1+1+4+2=
= 8 nsec

Question 8: (10 points)

Find the test vector for detecting the stuck-at zero fault in the following circuit using the fault activation and fault propagation method:



Answer:

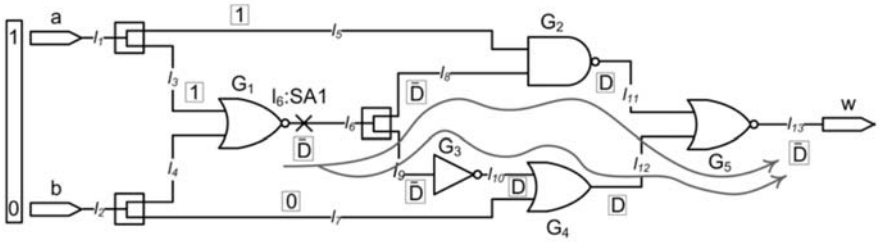


Figure 8: Circuit with reconvergent fanout, trying multiple sensitized paths.

- Fault activation: $I_6 = \overline{D} \longrightarrow a = 1 \longrightarrow I_5 = 1$
- For $\overline{D}$ propagation, we select both paths shown in figure
  $\overline{D}$ propagation in two directions $\xrightarrow{\text{justified by}} I_5 = 1, I_7 = 0 \longrightarrow b = 0$
- $(I_{11} = D)\ NOR\ (I_{12} = D) \longrightarrow (I_{13} = \overline{D})$
- Test: $ab = 10$

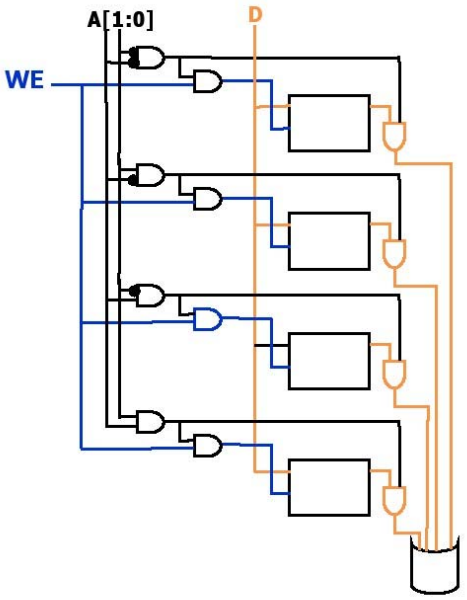
END of EXAM

Lecture Slides

Question 3:

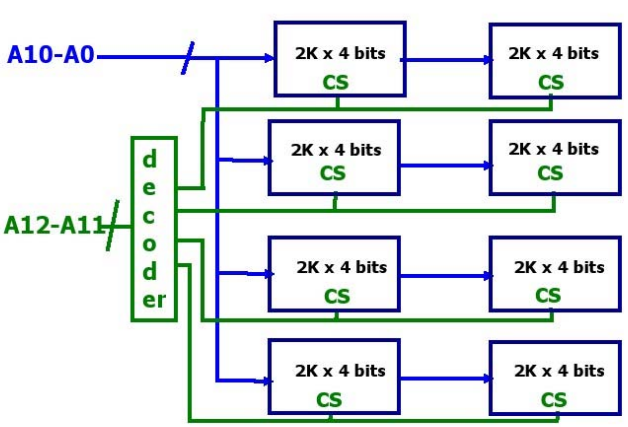
Building a Memory

- Each bit
  - is a gated D-latch
- Each location
  - consists of w bits (here w = 1)
  - w = 8 if the memory is byte addressable
- Addressing
  - n locations means **$\log_2 n$** address bits (here 2 bits => 4 locations)
  - decoder circuit translates address into 1 of n locations



Using Memory Building Blocks

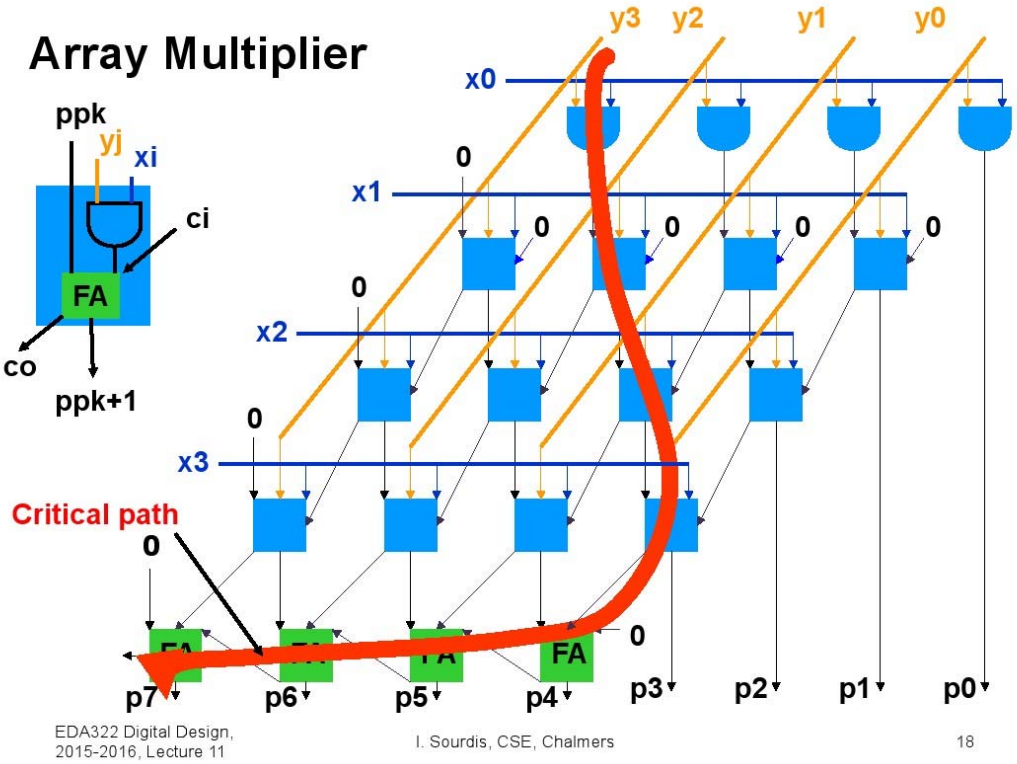
– Building an 8K byte memory using chips that are 2K by 4 bits.



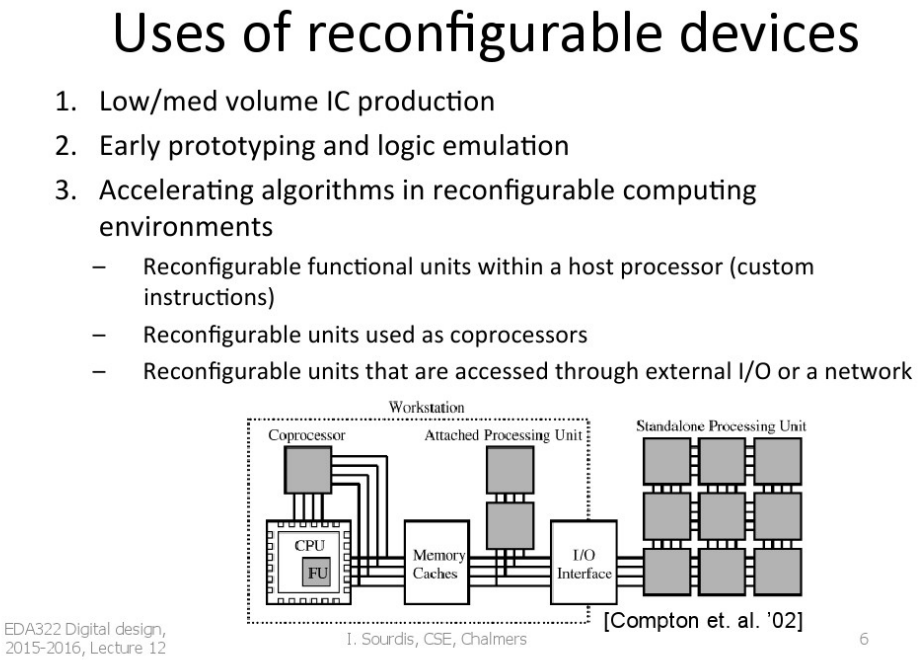
- **CS = chip select: when set, it enables the addressing, reading and writing of that chip.**

This is an 8KB byte addressable memory

Question 4:



Question 5:



Why reconfigurable?

- **Performance, Performance/Power:** Many applications run more efficiently in Reconfigurable Hardware :
  - Streaming applications
  - Parallelism,
  - When GPPs and ASICs don't match Applications requirements (datapath, ISA, etc.)
    - **Customization required**
  - Apps with changing requirements (**adaptation**)
- **FPGAs provide:**
  - the spatial computational resources to implement massively-parallel computations directly in hardware
  - Customization
  - Adaptation

  More:
  - Reduced Time to market vs. ASIC
  - No NRE (Non recurring Engineering) cost vs. ASIC
    - **NRE costs: one-time cost to research, develop , design and test a new product**