ECE 385 Final Review

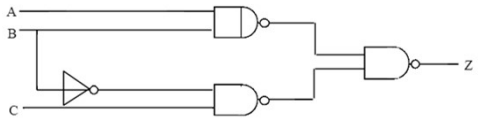
Yikuan Chen, 2018
Modified by Chushan Li 04/30/2024

Please take a piece of paper and a pen

ECE ILLINOIS

1

1.For the following circuit from Lab 1, will static-0 hazard happen when we switch in between A,B,C = 000 and 010?

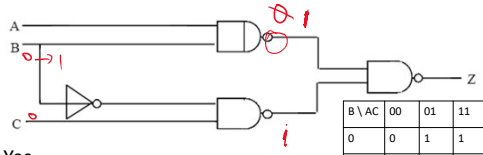


a) Yes
b) No

ECE ILLINOIS

2

1.For the following circuit from Lab 1, will static-0 hazard happen when we switch in between A,B,C = 000 and 010?



B \ AC	00	01	11	10
0	0	1	1	0
1	0	0	1	1

a) Yes
b) No ★

Transition from 000 to 010 (toggling B) will not cause the output to change from 0 to 1 because no matter what B is, as long as A and C remains 0, the NAND gate will always give a 1 and hence the output Z is always 0.

ECE ILLINOIS

3

11. In sLC3 design, what is the purpose of the provided tristate.sv?

a) To connect data from PC, ALU, MDR… to the BUS
b) To connect MEM2IO to the external SRAM
c) To connect BUS to SRAM
d) Both a and b
e) a, b and c

ECE ILLINOIS

4

11. In sLC3 design, what is the purpose of the provided tristate.sv?

a) To connect data from PC, ALU, MDR… to the BUS
b) To connect MEM2IO to the external SRAM ★
c) To connect BUS to SRAM
d) Both a and b
e) a, b and c

```
always_ff @(posedge Clk)
begin
    // Always read data from the bus
    Data_read_buffer <= Data;
    // Always updated with the data from Mem2IO
    Data_write_buffer <= Data_write;
end
```

ECE ILLINOIS

5

14. In Lab 6. If the instruction is 0001 000 001 100001 (Addi), ,and R0 has 0xFFFF, R1 has 0x0010, what is the condition code NZP after this operation?

a) 001
b) 101
c) 100
d) 010
e) 000

Instruction	Instruction(15 downto 0)						
ADD	0001	DR	SR1	0	00	SR2	
ADDi	0001	DR	SR1	1	imm5		
AND	0101	DR	SR1	0	00	SR2	
ANDi	0101	DR	SR1	1	imm5		
NOT	1001	DR	SR	111111			
BR	0000	n	z	p	PCoffset9		
JMP	1100	000	BaseR	000000			
JSR	0100	1	PCoffset11				
LDR	0110	DR	BaseR	offset6			
STR	0111	SR	BaseR	offset6			
PAUSE	1101	ledVect12					

ECE ILLINOIS

6

14. In Lab 6. If the instruction is 0001 000 001 100001 (Addi), ,and R0 has 0xFFFF, R1 has 0x0010, what is the condition code NZP after this operation?

a) 001 ★ (R0 <= 0x0010 + 0x0001 = 0x0011)
b) 101
c) 100
d) 010
e) 000

7

15. In Lab 6, what does the following code intend to do?

```
0 AND R0, R0, 0
1 ADD R0, R0, 1
2 ADD R1, R0, 1
3 STR R1, R0, -2
```

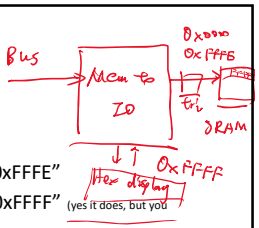
a) store value “1” to SRAM at address “0xFFFE”
b) store value “2” to SRAM at address “0xFFFF”
c) store value “1” to HEX Display
d) store value “2” to HEX Display
e) store value “2” to SRAM at address “0xFFFE”

8

15. In Lab 6, what does the following code do?

```
0 AND R0, R0, 0
1 ADD R0, R0, 1
2 ADD R1, R0, 1
3 STR R1, R0, -2
```

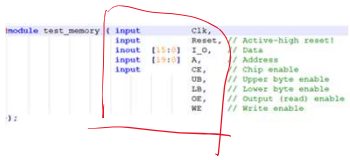
Line 3 use memory mapped IO to store 2 to “0-2 = 0xFFFF”, which is Hex Display.



a) store value “1” to SRAM at address “0xFFFE”
b) store value “2” to SRAM at address “0xFFFF”
c) store value “1” to HEX Display
d) store value “2” to HEX Display ★
e) store value “2” to SRAM at address “0xFFFE”

9

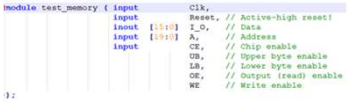
13. The storage element that test_memory.sv emulates is:



a) Register file
b) SDRAM
c) SRAM
d) Flash
e) On-chip-Memory

10

13. The storage element that test_memory.sv emulates is:



a) Register file
b) SDRAM
c) SRAM ★
d) Flash
e) On-chip-Memory

11

6. What type of architecture does NIOS IIe belong to?

a) A Moore Machine
b) A Mealy Machine
c) A Von-Neumann Machine
d) A Harvard Machine
e) A Modified Harvard Machine

12

6. What type of architecture does NIOS IIe belong to?

a) A Moore Machine
b) A Mealy Machine
c) A Von-Neumann Machine
d) A Harvard Machine
e) A Modified Harvard Machine ★

The original Harvard Machine uses entirely separate memory system to store instructions and data.

13

9. Start from the circuit in Lab 7, if we want bit zero to be constantly blinking (with 50% duty cycle) and bit 7:1 keep displaying the accumulation value, what should we add to the code AFTER the accumulation part? Assuming all code are inside a while(1) loop, and other code (accumulation, not shown) are correctly written.

a.
```
for(i = 0; i < 1000000; i ++); //delay
 *LED = *accumulator & (0x01);
```

b.
```
for(i = 0; i < 1000000; i ++); //delay
*accumulator = *accumulator ^ (0x01);
*LED =*accumulator;
for(i = 0; i < 1000000; i ++); //delay
accumulator = *accumulator ^ (0x01);
*LED = *accumulator;
```

c.
```
for(i = 0; i < 1000000; i ++); //delay
*accumulator = *accumulator & (0xFE);
*LED = *accumulator;
for(i = 0; i < 1000000; i ++); //delay
*accumulator = *accumulator | (0x01);
*LED = *accumulator;
```

d.
```
for(i = 0; i < 1000000; i ++); //delay
*LED = *accumulator & (0xFE);
for(i = 0; i < 1000000; i ++); //delay
*LED = *accumulator | (0x01);
```

e) more than one solution above

14

9. Start from the circuit in Lab 7, if we want bit zero to be constantly blinking (with 50% duty cycle) and bit 7:1 keep displaying the accumulation value, what should we add to the code AFTER the accumulation part? Assuming all code are inside a while(1) loop, and other code (accumulation, not shown) are correctly written.

a.
```
for(i = 0; i < 1000000; i ++); //delay
 *LED = *accumulator & (0x01);
```

b.
```
for(i = 0; i < 1000000; i ++); //delay
*accumulator = *accumulator ^ (0x01);
*LED =*accumulator;
for(i = 0; i < 1000000; i ++); //delay
accumulator = *accumulator ^ (0x01);
*LED = *accumulator;
```

c.
```
for(i = 0; i < 1000000; i ++); //delay
*accumulator = *accumulator & (0xFE);
*LED = *accumulator;
for(i = 0; i < 1000000; i ++); //delay
*accumulator = *accumulator | (0x01);
*LED = *accumulator;
```

d. ★★
```
for(i = 0; i < 1000000; i ++); //delay
*LED = *accumulator & (0xFE);
for(i = 0; i < 1000000; i ++); //delay
*LED = *accumulator | (0x01);
```

e) more than one solution above

A will clear bit 7:1; b does not guarantee the last bit to be always toggling (if accumulator value changes from even to odd, you will see the last bit of LED to be 1,0,0,1); c would corrupt the value of the accumulator; only d is correct.

15

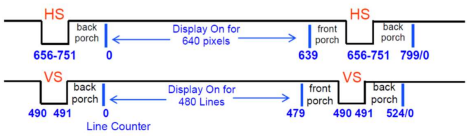
12. In Lab 8, which signal could be used as the clock for ball module if we want the data refreshing rate to be the same as the frame rate?

a) CLOCK_50
b) VGA_Clk
c) VGA_HS
d) VGA_VS
e) Draw_X or Draw_Y

16

12. In Lab 8, which signal could be used as the clock for ball module if we want the data refreshing rate to be the same as the frame rate?

a) CLOCK_50
b) VGA_Clk
c) VGA_HS
d) **VGA_VS ★**
e) Draw_X or Draw_Y



The vertical sync signal becomes low for a short period at the end of each frame, and stays high for the rest of the time, which is a good choice as the clock of ball module

Picture from course lecture slides

17

1. In Lab8, in order to complete a USB_Read Operation, in what sequence do you need to call io_read and io_write? (warm-up)

a) Io_read → io_write→ io_read→io_write
b) Io_write→io_read
c) Io_write→io_write→io_read
d) Io_read
e) Io_read→io_read

18

1. In Lab8, in order to complete a USB_Read Operation, in what sequence do you need to call io_read and io_write? (warm-up)

a) Io_read → io_write→ io_read→io_write

b) Io_write→io_read ★

c) Io_write→io_write→io_read

d) Io_read

e) Io_read→io_read

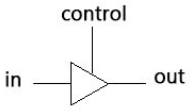
ECE ILLINOIS

19

Problem 8 – Additional Notes

An *inout* logic must be connected to a tristate buffer, otherwise the behavior is undefined. By the way, FPGA only supports inout logic to be mapped to i/o pins and does not internally support inout.

```
assign out = (control)? in : 16'bZZZZZZZZZZZZZZZZ;
```

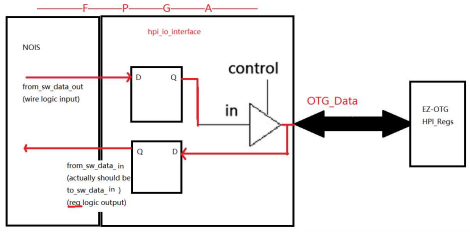


ECE ILLINOIS

20

Problem 8 – Additional Notes

A more complete picture



ECE ILLINOIS

21

- In lab 9, if 80x30 text mode is applied, which part on display will be modified if writing data to VRAM[0x25]?
  a. Row 0, Column 37
  b. Row 37, Column 0
  c. Row 1, Column 71
  d. Row 68, Column 1
-

ECE ILLINOIS

22

- In lab 9, if 80x30 text mode is applied, which part on display will be modified if writing data to VRAM[0x25]?
  a. Row 0, Column 37
  b. Row 37, Column 0
  c. Row 1, Column 71
  d. Row 68, Column 1
-

ECE ILLINOIS

23

- In lab 9, if in the palette design, each single color (R/G/B) uses 5 bits to represent, there are still 2 colors per each word, and 32 32-bit words of palette registers are used. How many different kinds of colors are supported in this palette?
  a. 32
  b. 64
  c. 128
  d. 192
-

ECE ILLINOIS

24

- In lab 9, if in the palette design, each single color (R/G/B) uses 5 bits to represent, there are still 2 colors per each word, and 32 32-bit words of palette registers are used. How many different kinds of colors are supported in this palette?
  a. 32
  b. 64
  c. 128
  d. 192
-

25

Speed Round - (Cont.)

3. Which one of these modules' output is asynchronous in experiment 3?
a) Computation Unit
b) Routing Unit
c) Both
d) Neither
Both the routing unit and the computation unit are asynchronous. The only synchronous elements are the flip flops in the state machine (or counter) and the shift registers.

4. True or false: In a carry look-ahead adder, generation of P/G bits (P_n, G_n) requires the value of the previous P/G bits (P_{n-1}, G_{n-1}) to be known.
a) True
b) False
The performance savings from the CLA come from not having dependencies between the various bits for generating the carry, note that P and G can be bitwise computed from the A and B inputs.

26

Speed Round - (Cont.)

5. In the multiplier lab (experiment 5), how is the 9th bit of the adder's sum generated?
a) S8 = S7
b) S8 = C_out7
c) S8 = A7 + B7
d) S8 = A7 + B7 + C_out7
The adder has to be a true 9-bit adder with A and B sign extended as inputs. Otherwise an overflow may occur in an add stage (e.g. 127 + 1).

6. In experiment 6, what is the purpose of the IR?
a) The IR tells the CPU what address it should fetch its next instruction from
b) The IR contains the data that are either written to or read from memory
c) The IR contains the address of memory the CPU is currently operating on
d) The IR contains the current execution cycle's instruction
The instruction register contains the current instruction, which is decoded into the proper control signals.

7. Inside an `always_comb` procedure block, which type of assignment should be used?
a) blocking
b) non-blocking
Combinational logic has implicit ordering, due to their dependencies when drawn out as a schematic.

27

15. What is the correct way to connect port in SystemVerilog?

If I have: `module M(input a, input b, output logic c);`
in top level, I have `logics A, B, C`

a) M m0(.*);
b) M m0(.A(a),.B(b),.C(c));
c) M m0(.a(A),b(B),.c(C));
d) a and c
e) a and b

28

15. What is the correct way to connect port in SystemVerilog?

If I have: `module M(input a, input b, output logic c);`
in top level, I have `logics A, B, C`

a) M m0(.*); //(A != a, case sensitive)
b) M m0(.A(a),.B(b),.C(c));
c) M m0(.a(A),.b(B),.c(C)); ★
d) a and c
e) a and b

29

5. Which way of using parameter in SystemVerilog is **wrong?**

```
module regfile (input clk, input [ADDR_WIDTH-1:0]addr, inout wire [DATA_WIDTH-1:0]data);
parameter ADDR_WIDTH=8;
parameter DATA_WIDTH=32;
… (implementation code)
Endmodule
```

a) `regfile #(8,16) myreg(.*);`
b) `regfile #(.ADDR_WIDTH(8),.DATA_WIDTH(16)) myreg(.*);`
c) `regfile (#ADDR_WITH =8, #DATA_WITH =16) myreg(.*);`
d) All of them are wrong
e) None of them are wrong

30

5. Which way of using parameter in SystemVerilog is **wrong?**

```
module regfile (input clk, input [ADDR_WIDTH-1:0]addr, inout wire [DATA_WIDTH-1:0]data);
parameter ADDR_WIDTH=8;
parameter DATA_WIDTH=32;
… (implementation code)
Endmodule
```

a) `regfile #(8,16) myreg(.*);`
b) `regfile #(.ADDR_WIDTH(8),.DATA_WIDTH(16)) myreg(.*);`
c) `regfile (#ADDR_WITH =8, #DATA_WITH =16) myreg(.*);`
d) All of them are wrong
e) None of them are wrong
Note: # comes first.

31

11. Which of the following SystemVerilog code will cause "always_comb does not infer purely combinational logic"?

a.//b and c are input
logic [5:0] a;
always_comb begin
if(b == c) a = ~b;
end
	b.//b and c are input
	logic [5:0] a = 5'b0;
	always_comb begin
	if(b == c) a = ~b;
	end

c.//b and c are input
logic [5:0] a;
always_comb begin
if(b != c) a = ~b;
else a = b;
end
	d.//b and c are input
	logic [5:0] a;
	always_comb begin
	a = b;
	if(b == c) a = ~b;
	end

e. More than one will cause

32

11. Which of the following SystemVerilog code will cause "always_comb does not infer purely combinational logic"?

if statement must have else or initial value!

(case must have default)

a.//b and c are input
logic [5:0] a;
always_comb begin
if(b == c) a = ~b;
end ★★
	b.//b and c are input
	logic [5:0] a = 5'b0;
	always_comb begin
	if(b == c) a = ~b;
	end

c.//b and c are input
logic [5:0] a;
always_comb begin
if(b != c) a = ~b;
else a = b;
end
	d.//b and c are input
	logic [5:0] a;
	always_comb begin
	a = b;
	if(b == c) a = ~b;
	end

e. More than one will cause

33

4. A pure combinational circuit with feedback is a latch circuit. Which one(s) of the following code will result in "**latch inferred**" warning in Quartus? (assume all starts with some valid value)

1.
always_comb begin
state = next_state;
if(state == A) next_state = C;
else next_state = B;
end
	2.
	always_ff @ (posedge Clk) begin
	if(state == A) next_state <= C;
	else next_state <= B;
	state <= next_state;
	end

3.
assign a = (a == 1)? 0 : 1;
	4.
	always_comb
	if(a == 1) a = 0;
	else a = 1;
	end

a)only 1 b)only 1,2 c)only 3 d)only 1,3,4 e)only 3,4

34

4. A pure combinational circuit with feedback is a latch circuit. Which one(s) of the following code will result in "**latch inferred**" warning in Quartus? (assume all starts with some valid value)

1.
always_comb begin
state = next_state;
if(state == A) next_state = C;
else next_state = B;
end
	2.
	always_ff @ (posedge Clk) begin
	if(state == A) next_state <= C;
	else next_state <= B;
	state <= next_state;
	end

3.
assign a = (a == 1)? 0 : 1;
	4.
	always_comb
	if(a == 1) a = 0;
	else a = 1;
	end

a)only 1 b)only 1,2 c)only 3 d)only 1,3,4 ★★ e)only 3,4

3 and 4 are completely equivalent, and there is a latch since a depend on it self (think about how SR latch work). Although 2 is incorrect in terms of acting as a FSM, it is syntactically correct (no latch). 1 is a little bit more subtle, but state actually depend on its own value, which results in a ***latch.***

35

3. In SystemVerilog, if we want to monitor a signal inside a sub-module without putting it in the output list, what could we do? Say we have:

```
module testbench()
//code neglected
 adder a0(.*);
endmodule
```

```
module adder(…)
logic a;
//code neglected
endmodule
```

//and we want to look at signal "a" in a0 (which is not an output)

I. (in testbench module): `logic a_monitor;`
`always_comb begin a_monitor = a0.a; end`
II. (in testbench module): `logic a_monitor = a0.a;`
III. (in testbench module):
`logic a_monitor;`
`always_ff @ (posedge Clk) a_monitor <= adder(a0).a;`
IV. (in testbench module):
`logic a_monitor;`
`always_ff @ (posedge Clk) a_monitor <= a0.a;`

a) all will do
b) only III will do
c) only IV will do
d) both III and IV
e) both I and IV

36

3. In SystemVerilog, if we want to monitor a signal inside a sub-module without putting it in the output list, what could we do? Say we have:

```
module testbench()
//code neglected
adder a0(.*);
endmodule
```

```
module adder(…)
logic a;
//code neglected
endmodule
```

//and we want to look at signal "a" in a0 (which is not an output)

I. (in testbench module): `logic a_monitor; always_comb begin a_monitor = a0.a; end`

II. (in testbench module): `logic a_monitor = a0.a;`

III. (in testbench module):
```
logic a_monitor;
always_ff @ (posedge Clk) a_monitor <= adder(a0).a;
```

IV. (in testbench module):
```
logic a_monitor;
always_ff @ (posedge Clk) a_monitor <= a0.a;
```

a) all will do
b) only III will do
c) only IV will do
d) both III and IV
e) both I and IV ★★

We should refer the name of the instance (i.e. a0) and use always_ff to constantly refresh the value. By the way, you may only assign a constant value to a logic if you combine declaration and assignment in one line (e.g. logic a = 0;)

ECE ILLINOIS ILLINOIS

37

14. True or False: SDRAM is faster than SRAM (since SDRAM is Dynamic RAM)

a) True
b) False

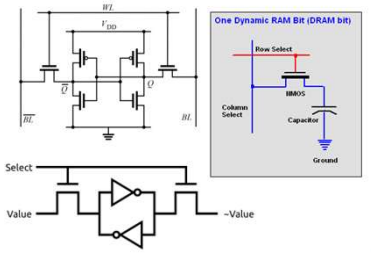
ECE ILLINOIS ILLINOIS

38

14. True or False: SDRAM is faster than SRAM (since SDRAM is Dynamic RAM)

a) True
b) False ★

This refresh operation is where dynamic RAM gets its name. Dynamic RAM has to be dynamically refreshed all of the time or it forgets what it is holding. The downside of all of this refreshing is that it takes time and slows down the memory. A flip-flop for a memory cell takes 4 or 6 transistors along with some wiring, but never has to be refreshed. This makes static RAM significantly faster than dynamic RAM. Take more space though



ECE ILLINOIS ILLINOIS

39

16. Which one of memories on DE2 board has the largest storage capacity?

a) SRAM
b) b) Flash Memory
c) On Chip Memory
d) SDRAM
e) Hard Disk

ECE ILLINOIS ILLINOIS

40

16. Which one of memories on DE2 board has the largest storage capacity?

a) SRAM (2MB)
b) Flash Memory (8MB)
c) On Chip Memory (<500 KB)
d) SDRAM (128MB) ★
e) Hard Disk

ECE ILLINOIS ILLINOIS

41

17. Which one of the following statements about SDRAM is correct?

a) SDRAM can write to successive rows (with column address fixed) faster than write to successive columns (within the same row)
b) SDRAM can write to successive columns (with row address fixed) faster than write to successive rows (within the same column)
c) Both types of operations above have the same latancy

ECE ILLINOIS ILLINOIS

42

17. Which one of the following statements about SDRAM is correct?

a) SDRAM can write to successive rows (with column address fixed) faster than write to successive columns (within the same row)

b) SDRAM can write to successive columns (with row address fixed) faster than write to successive rows (within the same column)

c) Both types of operations above have the same latancy

In order to read into another closed row, currently open row must be pre-charged to close, which slows down the whole operation

43

18. What is the correct way to initialize the On-chip Memory?

a) Use an always_ff block to assign values (like regfile in Lab 6)

b) Use $readmemh/readmemb (not synthesizable) and initial block ★★

c) Both a and b achieves the same functionality

d) Instantiate many register modules and reset them to wanted values (like testmemory.sv in Lab 4)

e) There is no way to initialize On-Chip Memory

44

18. What is the correct way to initialize the On-chip Memory?

a) Use an always_ff block to assign values (like regfile in Lab 6)

b) Use $readmemh/readmemb (not synthesizable) and initial block ★★

c) Both a and b achieves the same functionality

d) Instantiate many register modules and reset them to wanted values (like testmemory.sv in Lab 4)

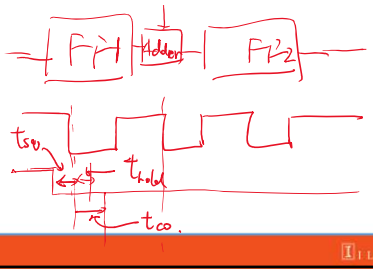
e) There is no way to initialize On-Chip Memory

Although initial block can't be synthesized, it actually doesn't need to synthesize it at all. The synthesis tool will read from the specified file and initialize the On-Chip Mem. The method in (a) and (c) will produce a huge matrix of combinational logic, using a lot of resource and make compilation time forever.

45

12. If a flip-flop has setup time 2ns, hold time 3ns, clock-to-output time 5ns, and the clock frequency is 50MHz. What is the longest allowed combinational path delay in one cycle?

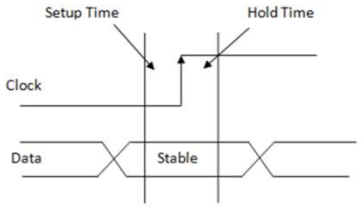
a) 20 ns
b) 18 ns
c) 15 ns
d) 13 ns
e) 10 ns



46

12. If a flip-flop has setup time 2ns, hold time 3ns, clock-to-output time 5ns, and the clock frequency is 50MHz. What is the longest combinational path delay in one cycle?

a) 20 ns
b) 18 ns
c) 15 ns
d) 13 ns ★ ★ ★
e) 10 ns



13 = period – setuptime – clock-to-output-time (hold time is irrelevant for this cycle)

47

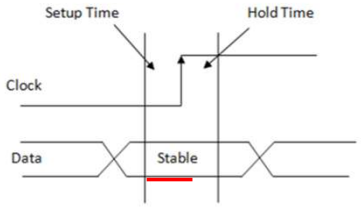
13. In a hypothetical 50MHz system. If Flipflop 1 has setup time 1ns, hold time 1ns, clk-to-out time 3ns, Flipflop2 has setup time 2ns, hold time 5ns, clk-to-out time 7ns, what's MINIMUM allowed path delay?

a) 1 ns
b) 2 ns
c) 3 ns
d) 5 ns
e) 8 ns

48

13. In a hypothetical 50MHz system. If Flipflop 1 has setup time 1ns, hold time 1ns, clk-to-out time 3ns, Flipflop2 has setup time 2ns, hold time 5ns, clk-to-out time 7ns, what's MINIMUM allowed path delay?

a) 1 ns
b) 2 ns ★ ★ ★
c) 3 ns
d) 5 ns
e) 8 ns



2 = 5(hold time of FF2)-3(c-to-o time of FF1)

- Know the high-level description (with a reasonable level of detail, of course) of each Lab is the most important thing.
- Know the things discussed in lecture about Final Project will be helpful.
- Know the properties of all the storages on the board (on chip memory, SRAM, SDRAM, Flash)