Name: ______________________________

First Name: __________________________

Student ID: __________________________

1st session examination

Design of Digital Circuits SS2016

(252-0014-00S)

Srdjan Capkun, Frank K. Gürkaynak

Examination Rules:

1. Written exam, 90 minutes in total.
2. No books, no calculators, no computers or communication devices. Six pages of hand-written notes are allowed.
3. Write all your answers on this document, space is reserved for your answers after each question. Blank pages are available at the end of the exam.
4. Put your Student ID card visible on the desk during the exam.
5. If you feel disturbed, immediately call an assistant.
6. Answers will only be evaluated if they are readable.
7. Write with a black or blue pen (no pencil, no green or red color).
8. Show all your work. For some questions, you may get partial credit even if the end result is wrong due to a calculation mistake.
9. Points for every part are indicated in the exam. They should correspond to the expected difficulty of the questions. Consider this when allocating your time.

Question:	1	2	3	4	5	6	7	Total
Points:	7	18	12	10	12	6	10	75
Score:								

This page intentionally left blank

1. (a) (4 points) Consider the following eight bit binary sequence of numbers:

```
1010 1110
```

Each of the following interprets this sequence differently. For each interpretation, state if the statement is true of false. If it is false, write the correct interpretation.

$(EA)_{16}$ in hexadecimal format: **False,** $(AE)_{16}$

-46 in decimal when using sign/magnitude representation: **True**

-81 in decimal when using two's complement representation: **False,-82**

174 when using unsigned representation: **True**

(b) (3 points) State whether the following statements about the binary representation of numbers are *true* or *false*. Give **brief** explanations for the statements that are *false*.

- Both two's complement and one's complement representation define two zeroes, one positive and one negative.

Solution: False, there's only one representation for two's complement.

- Using N bits it is only possible to represent $2^{N-1} - 1$ different numbers when a two's complement number system is used.

Solution: False. These are only the positive numbers, in total all numbers from -2^{N-1} to $2^{N-1} - 1$ can be represented.

- While there are methods to represent both positive and negative integers, it is not possible to represent fractions or real numbers using binary numbers.

Solution: False, fixed and floating point number systems can be used to represent such numbers.

2. For this question, use the following truth table for a 4-input logic function called Z.

Input				**Output**
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	X
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	X
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	X
1	1	1	1	1

(a) (1 point) What is the meaning of X in this truth table?

Solution: The output value is not important for the functionality of the circuit. It can be taken as '0' or '1' to simplify the equations

(b) (6 points) A friend of yours has determined the following Boolean equation for Z:

$$Z = (\overline{B} + D) \cdot (A + B + C) \cdot (A + B + \overline{C}) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{C} + D)$$

But he is not sure if this is correct. Verify whether or not the given equation matches the truth table given above. If not, please explain how the equation can be corrected.

Solution:
The equation is not correct. You can see this if you mark the min-terms on the truth table for each equation. The following are the problems:

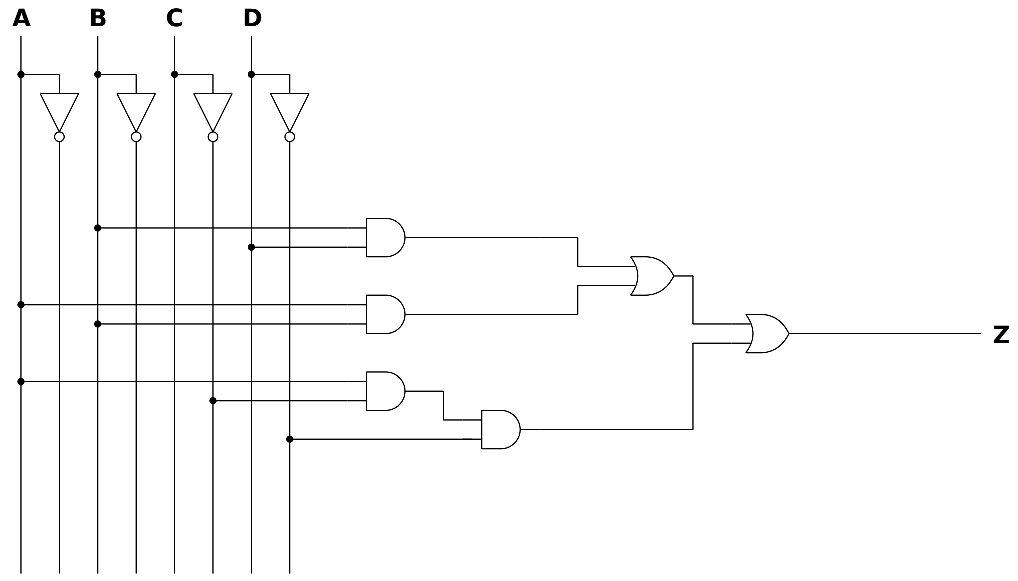
- $(\overline{B} + D)$ is wrong. Should have been $(B + \overline{D})$
- $(A + B + C)$ and $(A + B + \overline{C})$ can be merged to $(A + B)$
- min-term $(\overline{A} + B + C + \overline{D})$ is missing
- $(\overline{A} + \overline{C} + D)$ is redundant, especially if the X is taken as '1'

The X values have not been optimally used, this results in a more complex equation, if X values are chosen carefully a SOP form would probably be better, in addition there are redundant terms, the equation is not simplified

(c) (5 points) Derive your own *optimized* Boolean equation corresponding to the same truth table using *sums-of-products* form. Try to take advantage of the 'X' values to minimize the equation as much as possible. (*Hint: use a Karnaugh map*)

Solution: If you take $\overline{AB}C\overline{D}$ as 0 and all other X's as '1', you can derive:
$Z = (B \cdot D) + (A \cdot \overline{C} \cdot \overline{D})$

(d) (4 points) Draw a gate-level schematic that realizes the function Z using **only** 2-input AND, OR gates. Assume that you have all variables (A, B, C, D) available as input. Their complements $(\overline{A}, \overline{B}, \overline{C}, \overline{D})$ are already drawn for you.



(e) (2 points) Assume that all the gates (AND, OR, NOT) in the previous diagram have a propagation delay of 100 ps and a contamination delay of 50 ps. What is the delay of the longest (*critical*) path and the *shortest* path of this circuit?

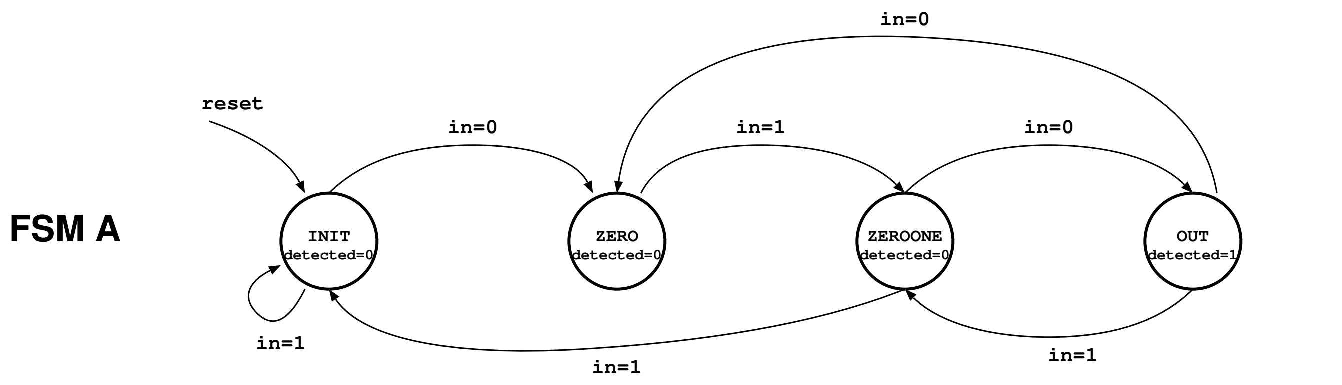
Solution: In the solution above, the *critical* path goes through 1 inverter, 2 AND gates, and 1 OR gate and is $(4 \times t_{pd} ==)$ 400 ps. The *short* path goes through one AND gate and two OR gates and equals to $(2 \times t_{cd} ==)$100 ps.

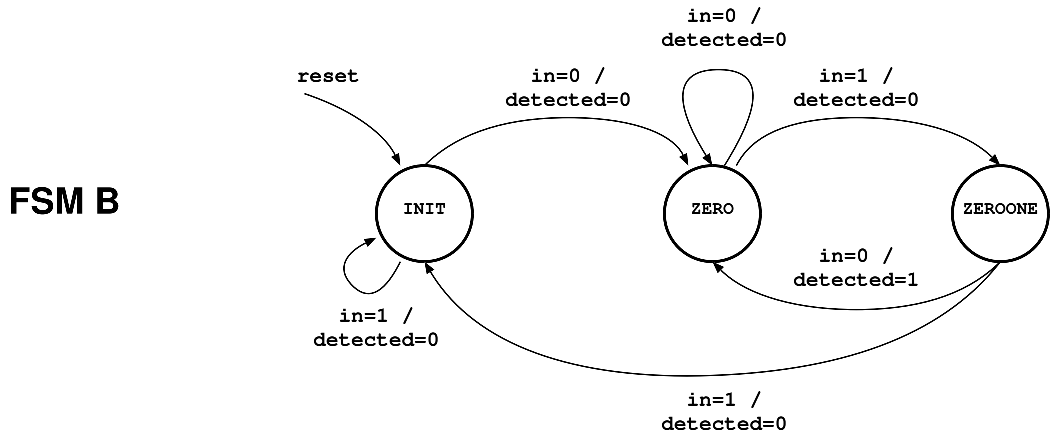
Note: Depending on how the circuit is drawn the numbers could change slightly, it is possible that the longest path is 5 gates.

This page intentionally left blank

3. We want to design a Finite State Machine (FSM) that has a one bit input (`in`) and will detect the sequence 0-1-0. If this sequence is detected, the one bit output (`detected`) will be set to 1, otherwise this output will remain at 0.

   Two of your colleagues have designed different state transition diagrams given below.





(a) (1 point) Which one of the state diagrams depicts a Moore and which one a Mealy type of FSM

> **Solution:**
> FSM A is a Moore type FSM, the output depends only on the state. FSM B is a Mealy type FSM, the output depends on both the state and inputs

(b) (4 points) For both state transition diagrams state whether or not they are correct.

> **Solution:**
> FSM A has a small mistake, for state `ZERO` it is not clear what will happen when `in=0`. FSM B is correct.

(c) (7 points) Complete the following Verilog module that would implement the state machine as described in the question. You can implement one state transition diagram of your colleagues if that one is correct.

```
module fsm (input in, input clk, input reset, output reg detected);

reg [1:0] next_state, present_state;

parameter INIT    = 2'b11;
parameter ZERO    = 2'b00;
parameter ZEROONE = 2'b01;

always @ (*)
   begin
     next_state <= present_state;   // default
     detected <= 1'b0;
     case (present_state)
       INIT:     next_state <= in ? INIT   : ZERO;
       ZERO:     next_state <= in ? ZEROONE: ZERO;
       ZEROONE:  if (in)
                   next_state <= INIT;
                 else
                   begin
                     next_state <= ZERO;
                     detected <= 1'b1;
                   end
       default: next_state <= present_state;
     endcase
   end

always @ (posedge clk, posedge reset)
  if (reset)  present_state <= INIT;
  else        present_state <= next_state;

endmodule
```

4. In this question for each part there will be two Verilog code snippets. For each part you will have to say whether both, only one, or none of the code snippets fulfill what is being asked. All code snippets are syntactically correct. They will compile and produce either a sequential circuit or a combinational circuit.

   (a) (2 points) Which code snippet generates a sequential circuit?

(A)

```
module a1 (input b,
           output reg c);
  reg a;
  always @ (*)
  begin
    a = b;
    c = ~a;
  end
endmodule
```

(B)

```
module a2 (input clk,
           input rst,
           input d,
           output reg q);
always @ (clk, rst, d)
   if (rst)
      q <= 1'b0;
   else if (clk)
      q <= d ;
   else
      q <= ~d;
endmodule
```

[✓ **none**] ○ A ○ B ○ Both A and B

   (b) (2 points) Which code snippet properly instantiates the module mux2?

```
module mux2 (input d1, input d2, input s, output out);
 assign out = s? d1:d2;
endmodule
```

(A)

```
module b1 (input a,
           input b,
           input sel,
           output q);
  mux2 first (a,b,sel,q);
endmodule
```

(B)

```
module b2 (input a,
           input b,
           input sel,
           output q);
  mux2 second (.d1(a),
               .d2(b),
               .s(sel),
               .out(q)
              );
endmodule
```

○ none ○ A ○ B [✓ **Both A and B**]

(c) (2 points) Which code snippet results in a 2-input multiplexer ?

(A)

```
module c1 (input sel,
           input a,
           input b,
           output z);
 assign z = (sel) ? a : 0;
endmodule
```

(B)

```
module c2 (input sel,
           input a,
           input b,
           output z);
 assign z = ~sel & b | sel & a;
endmodule
```

○ none ○ A [✓ **B**] ○ Both A and B

(d) (2 points) Which code snippet(s) will produce a 8-bit value which is composed of (from MSB to LSB), $c_3c_2c_1d_6d_6110$ (c and d are both 8-bit values)?

(A)

```
module d1 (input [7:0] c,
           input [7:0] d,
           output [7:0] z);
  assign z = {c[3:1],
              {2{d[6]}},
              3'b110    };
endmodule
```

(B)

```
module d2 (input [7:0] c,
           input [7:0] d,
           output reg [7:0] z);
always @ (c,d)
  begin
    z<= 8'b00000110;
    z[7:5] <= c[3:1];
    z[4] <= d[6];
    z[3] <= d[6];
  end
endmodule
```

○ none ○ A ○ B [✓ **Both A and B**]

(e) (2 points) Which code snippets produce a combinational circuit?

(A)

```
module e1 (input clk,
           input a,
           input b,
           output reg [1:0] q);
  always @ (clk)
    if (clk)
      q <= 2'b10;
    else
      q <= 2'b01;
endmodule
```

(B)

```
module e2 (input clk,
           input a,
           input b,
           output reg [1:0] q);
 always @ (*)
   if (a)
      q <= 2'b01;
   else if (b)
      q <= 2'b10;

endmodule
```

○ none [✓ **A**] ○ B ○ Both A and B

---

5. (12 points) Consider the following MIPS program. For clarity the addresses have been written using only 4 hexadecimal digits. Leading hexadecimal digits are all zeroes (the real start address is `0x00003000`).

```
0x3000  start:     addi $s0, $0,    4
0x3004             xor  $s1, $s1,   $s1
0x3008             addi $s2, $0,    24
0x300C             sw   $s2, 0($s1)
0x3010             addi $s2, $s2,   10
0x3014             add  $s1, $s1,   $s0
0x3018             sw   $s2, 0($s1)

0x301C             addi $a0, $0,    -9
0x3020             jal  func
0x3024             sw   $v0, 4($s1)

0x3028             lw   $a0, 0($0)
0x302C             jal  func
0x3030             lw   $t2, 0($s0)
0x3034             sub  $t3, $t2,   $v0

0x3038  done:      j    done

0x303C  func:      add  $t1, $a0,   $0
0x3040             slt  $t2, $t1,   $0
0x3044             beq  $t2, $0,    pos
0x3048             sub  $t1, $0,    $t1
0x304C  pos:       add  $v0, $0,    $t1
0x3050             jr   $ra
```

We are interested in determining the value of some registers at the end of the program execution when the program reaches line `0x3038`. Fill in the following table, writing the value of the indicated registers at the end of the program, and at which line these values have been written into these registers.

As an example: at the end of execution the register `$s0` will have the value `4`. This value has been written into the register while executing line `0x3000`.

Register	**Value**	**Assigned on line**
`$s0`	`4`	`0x3000`
`$s2`	`34`	`0x3010`
`$t1`	`24`	`0x303C`
`$t2`	`34`	`0x3030`
`$t3`	`10`	`0x3034`
`$ra`	`0x3030`	`0x302C`

6. We are interested in how long it takes for a specific program to run. The program has 200 million instructions and is being executed on a single-cycle processor running at a clock of 400 MHz.

   (a) (2 points) How long (in seconds) will it take for this program to run on this architecture?

   Solution:

$$\begin{aligned} Time\ to\ execute &= N \cdot CPI \cdot \frac{1}{f} \\ Time\ to\ execute &= 200.000.000 \cdot 1 \cdot \frac{1}{400.000.000\,Hz} \\ Time\ to\ execute &= 0.5\ seconds \end{aligned}$$

   (b) (2 points) As an alternative, you consider a multi-cycle architecture that can run at 1.2 GHz, what is the minimum CPI that the multi cycle architecture has to achieve so that we can be faster?

   Solution:

$$\begin{aligned} Time\ to\ execute &= N \cdot CPI \cdot \frac{1}{f} \\ 0.5\ seconds &= 200.000.000 \cdot CPI \cdot \frac{1}{1.200.000.000\,Hz} \end{aligned}$$

   CPI has to be at least 3

   (c) (2 points) As yet another alternative, there is a different architecture for which the program can be compiled more efficiently into 120 million instructions. The architecture has a CPI of 2 and runs at 500 MHz. Is this option faster than the single cycle architecture from 6a?

   Solution:

$$\begin{aligned} Time\ to\ execute &= N \cdot CPI \cdot \frac{1}{f} \\ Time\ to\ execute &= 120.000.000 \cdot 2 \cdot \frac{1}{500.000.000\,Hz} \\ Time\ to\ execute &= 0.48\ seconds \end{aligned}$$

   It is marginally faster

7. In this question we will examine the cache performance of a computing system when running a specific program. A profile of the program has determined that the memory locations accessed by the program is in the following order:

```
0x1000 4000
0x1000 4020
0x1000 4004
0x1000 4028
0x1000 4024
0x1000 4020
0x1000 400C
0x1000 402C
0x1000 4040
0x1000 4000
0x1000 400C
0x1000 402C
0x1000 4020
0x1000 4024
0x1000 4004
0x1000 4000
```

Assume a memory system with the following specifications:

- Memory word: 4 bytes (memory is byte-addressable).
- Cache type: direct mapped cache
- Cache size: 8 words.
- Cache block size: 1 word.
- Cache access time: $t_{cache} = 2$ cycles.
- Main memory access time: $t_{MM} = 40$ cycles.

(a) (2 points) Consider the program with the memory accesses listed above. How many cache misses and how many cache hits will you generate?

Solution: 14 misses and 2 hits.

(b) (2 points) What is the total memory access time for the program in question

Solution: 14 misses each require 1 cache access + 1 memory access (42 cycles), and the 2 cache hits require 2x2 = 4 additional cycles. So the total is:

$$Total = 14 \times (40 + 2) + 2 \times 2 Total = 588 + 4 Total = 592$$

(c) (3 points) How many of these misses are compulsory, how many are conflict and how many are capacity misses

Solution: There are 4 compulsory misses, 10 conflict misses, and no capacity misses. Half the capacity of the cache is not used during the execution of the program.

(d) (3 points) For each of the following changes to the cache organization, state whether it will or not increase the cache hit ratio and explain why.

1. Increasing the cache capacity from 8 to 16 words
2. Making a 2-way set associative cache instead of direct mapped cache
3. Using a block size of 4.

Solution:

1. Will not work directly, unless used in combination with the other two methods.
2. Will work, 2-way set associative cache reduces conflict misses. However, there will be more compulsory misses. (5 hit/ 11 miss)
3. Will work, using a block size of 4 reduces the compulsory misses. Still there would be several conflict misses (6 hit/ 10 miss)

A better solution would be to use 2-way set associative with a block size of 4 (12 hit, 4 miss). Or increase the size to 16, and use 4 way set associative with block size 4 (13 hit / 3 miss).