NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2012-2013

CE1005/CZ1005 – DIGITAL LOGIC

CPE104/CSC104 – LOGIC DESIGN

Nov/Dec 2012 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 5 pages.
2. Answer **ALL** questions.
3. This is a closed-book examination.
4. All questions carry equal marks.

---

1. (a) Perform the following number conversions. Show all steps clearly.

   (i) Convert decimal 74 to binary.

   (ii) Convert hexadecimal DFA to decimal.

   (iii) Convert octal 1654 to hexadecimal.

   (iv) Convert decimal 8.4375 to binary.

   (10 marks)

   (b) Obtain the minimum-cost sum-of-product (SOP) expression of the following Boolean function using algebraic manipulations. Show all steps clearly.

   $$F(a, b, c, d) = (a + d)(a' b + c' d)(a c + b d)'$$

   (8 marks)

Note: Question No. 1 continues on Page 2

(c) Implement the following logic function using NAND gates only. Illustrate with a clearly-labelled logic circuit diagram.

$$F(x, y, z) = x' y' z' + x y' z + x y z'$$

(7 marks)

2. (a) Figure Q2 shows a CMOS logic circuit with inputs A, B, C, D and output Z. Obtain its truth table.

(6 marks)

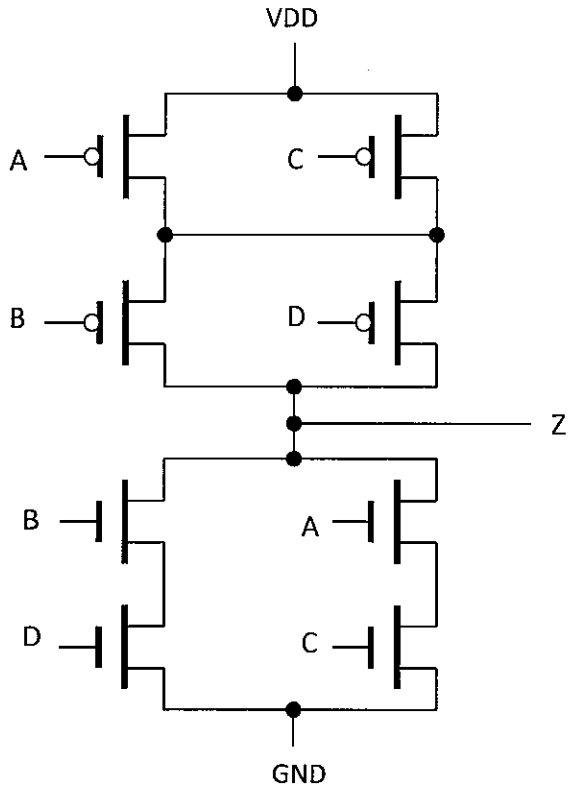

Figure Q2

(b) Showing all steps clearly, represent the signed decimal value **-51** using

(i) 8-bit sign-magnitude representation.

(ii) 8-bit two's complement representation.

(4 marks)

Note: Question No. 2 continues on Page 3

(c) Perform the following operations using 8-bit two's complement arithmetic. Show all the steps clearly. In each case, state whether or not arithmetic overflow has occurred.

(i) 01010101 + 00111101

(ii) 11001100 - 00111101

(6 marks)

(d) Given the following canonical product-of-maxterm logic expression and "don't care" expression d, obtain the minimum-cost sum-of-product (SOP) expression for F using the Karnaugh map technique. All loops must be clearly shown.

F (a, b, c, d) = $\prod$ M (0, 2, 5, 10, 13)

d (a, b, c, d) = $\prod$ M (7, 8, 11, 12, 14)

(9 marks)

3. (a) A 1-bit full adder is a combinational circuit with three inputs (*Ain*, *Bin* and *Cin*) and two outputs (*Sum* and *Cout*), where *Sum* represents the 1-bit summation of the three inputs and *Cout* is the 1-bit carry. A minimised expression for each of the full adder outputs is:

$$Sum = Ain \oplus Bin \oplus Cin$$

$$Cout = Ain.Bin + Ain.Cin + Bin.Cin$$

(i) Draw a circuit for the full adder using only **2-input gates**.

(4 marks)

(ii) Write a structural gate-level Verilog module which implements the full adder described by the circuit given in Q3(a)(i).

(6 marks)

Note: Question No. 3 continues on Page 4

(b) You are required to implement a circuit which produces a 5-bit output (*Y*). If the select input (*Sel*) is a logic '0', *Y* is the sum of two 4-bit values (*Xin* and *Yin*). If the select signal is a logic '1', *Y* is the negative of *Xin*.

(i) Sketch a schematic diagram to implement the required circuit using four 1-bit full adder blocks and any other necessary components.

(7 marks)

(ii) Briefly explain why you would **not** implement the circuit using structural Verilog as per the schematic diagram given in Q3(b)(i).

(2 marks)

(iii) Write a Verilog module which implements the required circuit and uses just a single **assign** statement.

(6 marks)

4. (a) Describe the behaviour of the following sequential components, and the difference between them:

- Transparent D-type Latch
- D-type flip-flop

Which is preferred in modern digital design and why?

(6 marks)

(b) You are required to implement a finite state machine (FSM) that implements an alarmed locking mechanism. It has two 1-bit inputs, *correct* and *wrong*, and two 1-bit outputs, *lock* and *alarm*:

- The FSM starts in the *idle* state, and moves to that state whenever reset is asserted.
- In the *idle* state, if *correct* is asserted, the FSM moves to the *unlock* state, where it remains.
- In the *idle* state, if *wrong* is asserted, the FSM moves to a *failone* state.
- In *failone*, if *wrong* is asserted once more, the FSM moves to the *failtwo* state; if *correct* is asserted, it moves to the *unlock* state.
- In *failtwo*, if *wrong* is asserted the FSM moves to the *failalarm* state, where the *alarm* output is asserted, and it stays in that state until *correct* is asserted, at which point it returns to the *idle* state.
- In *failtwo*, if *correct* is asserted, the FSM enters the *unlock* state.

Note: Question No. 4 continues on Page 5

- In the *unlock* state, the *lock* output is low; it is high in all other states.
- In the *failalarm* state, the *alarm* output is asserted. It is deasserted in all other states
- You should assume that *correct* and *wrong* are not asserted at the same time.

(i) Draw a state transition diagram that captures the above behavior.

(6 marks)

(ii) Implement the FSM in Verilog. Use a combinational **always** block for the state transitions, and a separate synchronous **always** block for the register process.

(8 marks)

(iii) Draw a timing diagram of the state transitions and FSM outputs of your top-level module given the input sequence in Figure Q4.

(5 marks)

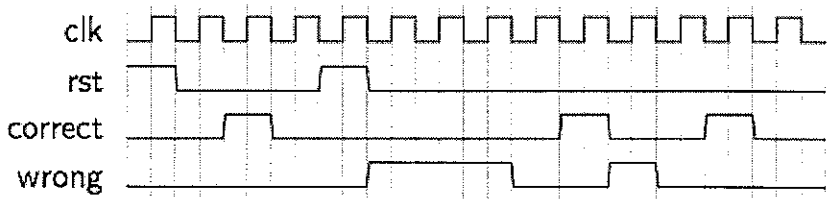

Figure Q4

END OF PAPER

CE1005 DIGITAL LOGIC
CPE104 LOGIC DESIGN
CSC104 LOGIC DESIGN
CZ1005 DIGITAL LOGIC

Please read the following instructions carefully:

1. **Please do not turn over the question paper until you are told to do so. Disciplinary action may be taken against you if you do so.**

2. You are not allowed to leave the examination hall unless accompanied by an invigilator. You may raise your hand if you need to communicate with the invigilator.

3. Please write your Matriculation Number on the front of the answer book.

4. Please indicate clearly in the answer book (at the appropriate place) if you are continuing the answer to a question elsewhere in the book.