MALASPINA UNIVERSITY-COLLEGE
FINAL EXAMINATION DECEMBER 2006
COMPUTING SCIENCE CSCI 355

Instructor: Dr. P Walsh **Duration:** 3 Hours.

TOTAL VALUE	100				
QUESTION	1	2	3	4	5
VALUE	10	25	25	10	30

Instructions

- Students must count the number of pages in this examination paper before beginning to write, and report any discrepancy immediately to the invigilator.
- This examination paper consists of 5 pages not including this cover page.
- Answer all questions in an answer booklet. All questions relate to material covered in class or in the lab. Show all your work.
- This is a CLOSED BOOK - examination. Students may consult notes written on one 8in. by 11 in. sheet of paper as described in class.
- Calculators are NOT permitted.

QUESTION 1.

Let

$$F(A,B,C,D,E) = \Pi M(0,1,2,3,6,7,8,9,10,11,14,15,16,17,24,25,26,27,30,31)$$

(a) Find the minimal SOP expression for F.
(b) Find the minimal POS expression for F.
(c) Implement F in CMOS (minimize the number of transistors).

QUESTION 2.

(a) Consider the following state table (with state encodings). It has one input X and four states. Your task is implement this system using the following counter component. To this end, develop SOP equations for each of the counter inputs. Note, you may assume that initially, the counter is in state `00` and 'x' means dont-care.

Present State	Next State	
	X=0	X=1
00	01	10
01	10	11
10	10	11
11	11	00

```
component counter
        port (
        Q: OUT std_logic_VECTOR(15 downto 0);
        CLK: IN std_logic;
        LOAD: IN std_logic;
        L: IN std_logic_VECTOR(15 downto 0);
        CE: IN std_logic;
        SCLR: IN std_logic);
end component;
```

LOAD	CE	SCLR	Operation
1	x	x	load
0	1	0	count up
0	1	1	clear

(b) Consider the situation where you are given two D flip flops and two 8-to-1 multiplexors. The D flip flops are clocked edge-triggered with no preset or clear. Your task is to design a two bit register using only the D flip flops and multiplexors. The register must operate according to the following function table:

$S_0\ S_1\ S_2$	Function	Pseudocode
0 0 0	no change	$Q_A^+ := Q_A \quad Q_B^+ := Q_B$
0 0 1	circular shift right	$Q_A^+ := Q_B \quad Q_B^+ := Q_A$
0 1 0	load most significant bit	$Q_A^+ := I_A \quad Q_B^+ := Q_B$
0 1 1	load least significant bit	$Q_A^+ := Q_A \quad Q_B^+ := I_B$
1 0 0	parallel load	$Q_A^+ := I_A \quad Q_B^+ := I_B$
1 0 1	complement	$Q_A^+ := \overline{Q_A} \quad Q_B^+ := \overline{Q_B}$
1 1 0	clear	$Q_A^+ := 0 \quad Q_B^+ := 0$
1 1 1	set	$Q_A^+ := 1 \quad Q_B^+ := 1$

Notation Current flip flop states: Q_A, Q_B. Next flip slop states: Q_A^+, Q_B^+. Inputs (least significant first): I_B, I_A. Function designator bits: S_0, S_1, S_2.

QUESTION 3.

(a) Consider the FSM specified by the following state table.

Present State $QA\ QB\ QC$	Next State Input=X		Output Z	
	X=0	X=1	X=0	X=1
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	x	x
001	xxx	xxx	x	x

Obtain excitation equation(s) for N-K flip flops and output equation(s). An N-K flip-flop has two inputs (N, K) and two outputs (Q, Q_{BAR}). Input K behaves like the K input to a J-K flip flop. Input N behaves like the complement of the J input to a J-K flip flop.

Note 'x' means dont-care.

(b) Consider the following state table. It has one input X and one output Z. Your task is to identify any equivalent states.

Present State	Next State Input=X		Output Z	
	X=0	X=1	X=0	X=1
a	f	b	0	0
b	d	c	0	0
c	f	e	0	0
d	a	a	1	0
e	d	c	0	0
f	f	b	1	1

QUESTION 4.

Consider the entity declaration below. The component is a simple serial to parallel converter with the following behaviour. After initialization, wait for `EN` to be asserted. Once `EN` has been asserted, read 4 data bits from `DIN` serially; use `DCLK` as the *data clock* (to differentiate the data bits on the the data line). Once all four bits have been received, output them in parallel on `DOUT` and wait for `EN` to be asserted again.

Your task is to develop the FSM that implements `s2p`. You may do this by writing VHDL code or by developing a state transition/output diagram.

```
entity s2p is
    port (CLK, EN, DIN, DCLK: in std_logic;
          DOUT: out std_logic_vector(3 downto 0));
end 22p;
```

QUESTION 5.

Consider a processor ISA where the data-bus is 16 bits wide and the address bus is 8 bits wide. The instruction format splits a 16 bit word into an 8 bit opcode followed by an 8 bit address. There are three programmer-visible registers:

- Accumulator ACC (16 bits wide)
- Program Counter PC (8 bits wide)
- Index register IR (16 bits wide)

The processor's instruction set is as follows:

Opcode	Mnemonic	Pseudocode
0x0	LDA S	ACC := [S]
0x1	STO S	[S] := ACC
0x2	ADD S	ACC := ACC + [S]
0x3	SUB S	ACC := ACC - [S]
0x4	JMP S	PC := S
0x5	JGE S	if ACC $\geq$ 0, PC := S
0x6	JNE S	if ACC $\neq$ 0, PC := S
0x7	LDI S	IR := [S]
0x8	STI S	[S] := IR
0x9	ADI S	ACC := ACC + [S + IR]
0xa	SUI S	ACC := ACC - [S + IR]

Your task is to design the processor's datapath and part of its control unit. Your design must be totally synchronous (control unit and datapath) and your datapath can only contain the components employed in the MU0 assignment (see Question 2 and below). Question deliverables are as follows:

- Develop a data flow diagram depicting the processor's datapath.
- Identify, label and enumerate (write down) all the control lines that will be required.
- Show the path through the control unit FSM that occurs when an ADI instruction is fetched and executed.

```
component busmux is
        port (
        MA: IN std_logic_VECTOR(15 downto 0);
        MB: IN std_logic_VECTOR(15 downto 0);
        S: IN std_logic_VECTOR(0 downto 0);
        O: OUT std_logic_VECTOR(15 downto 0));
end component;

component alu is
        port (
        A: IN std_logic_VECTOR(15 downto 0);
        B: IN std_logic_VECTOR(15 downto 0);
        ADD: IN std_logic;
        S: OUT std_logic_VECTOR(15 downto 0));
end component;
```

```
component ram IS
        port (
        addr: IN std_logic_VECTOR(7 downto 0);
        clk: IN std_logic;
        din: IN std_logic_VECTOR(15 downto 0);
        dout: OUT std_logic_VECTOR(15 downto 0);
        we: IN std_logic);
end component;
```