**Ryerson University**
**Department of Electrical & Computer Engineering**
**ELE328 — Digital Systems**

**F2004 Final Examination**

**Name:** _____ **ID No.:** _____ **Section:** _____

**Time Limit:** 2 hour and 50 minutes **Professors:** Chen, Y.C., Mekhiel, N., Sedaghat, R.

- Closed-book examination, no calculator or any other aids allowed.
- Answer all questions in the space provided and show all steps for full credit.
- Circle the name of your Professor shown above.

1. The state table of a finite-state machine (FSM) with one input $w$ and 2 outputs $z_1$ and $z_0$ is given below:

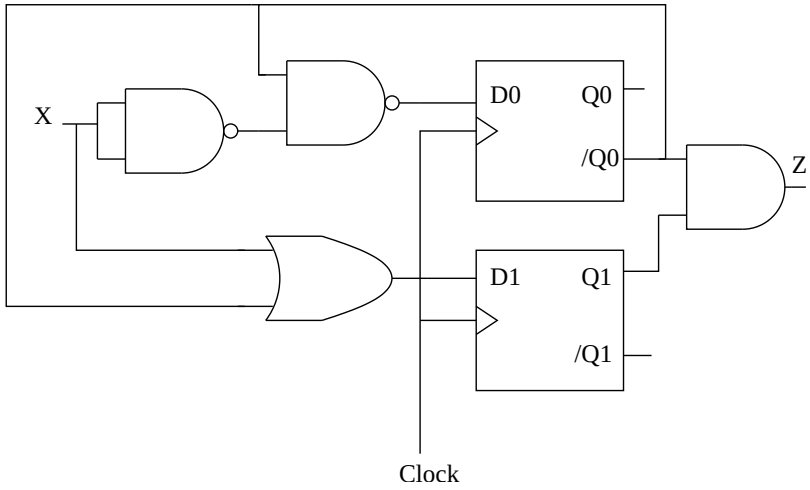
| Present State | Next State | | Outputs | |
|---|---|---|---|---|
| | $w=0$ | $w=1$ | $z_1$ | $z_0$ |
| S0 | S2 | S1 | 0 | 0 |
| S1 | S2 | S1 | 0 | 1 |
| S2 | S2 | S3 | 1 | 0 |
| S3 | S1 | S3 | 0 | 0 |

(a) (1 mark) Explain whether the given FSM is a Moore-type or Mealy-type state machine?

(b) (9 marks) The given FSM is to be implemented as a synchronous sequential circuit with T flip-flops using the state assignments: S0=00, S1=01, S2=10, S3=11. Derive the equations for the inputs to the T flip-flops, and the equations for the outputs $z_1$ and $z_0$.

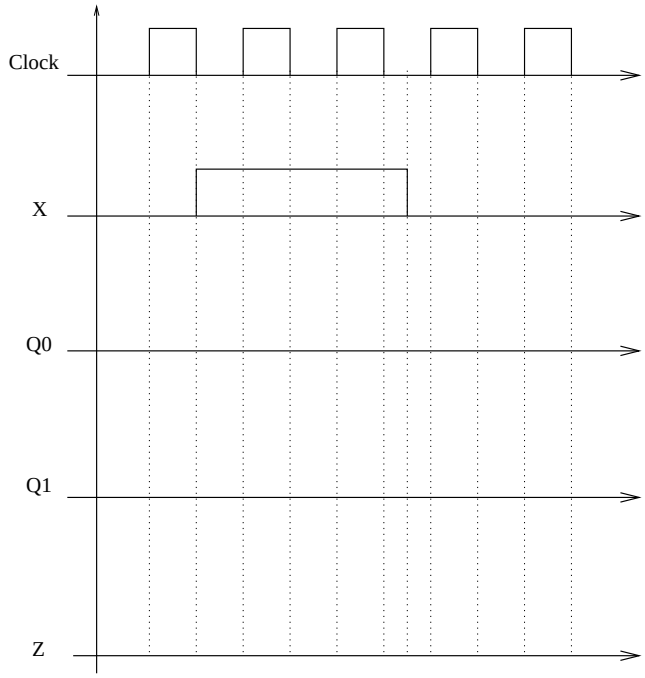
**(cont'd)**

2. Consider the following circuit which implements a finite-state machine (FSM):




(a) (4 marks) Derive the state assigned table for the FSM.

(b) (4 marks) Complete the following timing diagram for the circuit by assuming $Q_1 = Q_0 = 0$ at the beginning.

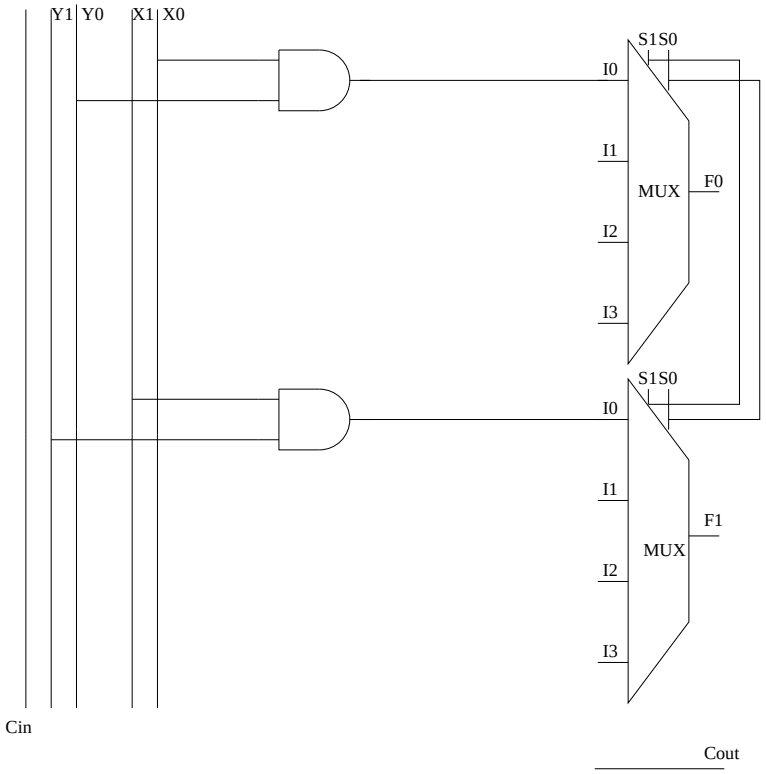



(c) (4 marks) Derive the state assigned table if JK flip-flops (instead of D flip-flips) are to be used to implement the FSM.

3. (10 marks) The circuit below is used as a part of an ALU to perform logical and arithmetic operations on the 2-bit inputs $X = X_1X_0$ and $Y = Y_1Y_0$. The operation to be performed is determined by the function select input $S_1S_0$ to produce the output $F = F_1F_0$ according to the following table:

| $S_1S_0$ | Operations |
|---|---|
| 00 | F = X AND Y |
| 01 | F = X OR Y |
| 10 | F = X XOR Y |
| 11 | F = X + Y (addition) |

Complete the implementation of the circuit shown below according to the given requirements:



4. Consider the following VHDL code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY fsm IS
  PORT( Clock,Resetn,w   : IN STD_LOGIC;
        Z   : OUT STD_LOGIC );
END fsm;

ARCHITECTURE Behavior OF fsm IS
  TYPE State_type IS (A,B,C);
  SIGNAL y: State_type;
BEGIN
  PROCESS(Resetn, Clock)
  BEGIN
     IF Resetn = '0' THEN
              y<=A;
     ELSEIF (Clock'EVENT AND Clock = '1') THEN
       CASE y IS
           WHEN A =>
              IF w='0' THEN
                  y<=A;
              ELSE
                  y<=B;
              END IF;
           WHEN B =>
              IF w='0' THEN
                  y<=A;
              ELSE
                  y<=C;
              END IF;
           WHEN C =>
              IF w='0' THEN
                  y<=A;
              ELSE
                  y<=C;
              END IF;
       END CASE;
     ENDIF;
  END PROCESS;
       z<= '1' WHEN y=C ELSE '0';
END Behavior;
```

(a) (3 marks) Describe the behavior of the given FSM using a state diagram.

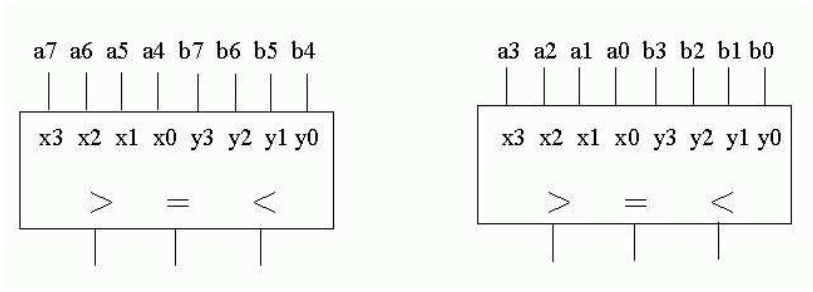
(b) Assume that an 8x4-bit EPROM is available, provide a programmable implementation of the FSM in Part (a):

i. (4 marks) Fill in the contents of the EPROM in the table below by setting up the addresses of the EPROM as follow: $a_2 = w$, $a_1a_0 = y_1y_0$ (present state), and explain what the contents of the EPROM represents.

| Address | Contents |
|---|---|
| $a_2a_1a_0$ | $d_3d_2d_1d_0$ |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

ii. (3 marks) Draw a schematic diagram for the implementation.

5. (10 marks) Two comparators for 4-bit unsigned numbers are available. Each comparator has two 4-bit inputs: $X = x_3x_2x_1x_0$ and $Y = y_3y_2y_1y_0$, and 3 logical outputs: $X$ greater than $Y$ ($>$), $X$ equal to $Y$ ($=$), and $X$ less than $Y$ ($<$). Construct a comparator for two 8-bit unsigned numbers $A = a_7a_6a_5a_4a_3a_2a_1a_0$ and $B = b_7b_6b_5b_4b_3b_2b_1b_0$ using the two 4-bit comparators and any additional logic gates, and draw the resulting circuit.



6. This question deals with the programmable processor module used in Lab 7. The function table of the 74181 ALU, and the instruction set of the processor are given in the Appendix.

   (a) (5 marks) Determine the contents of the Program Counter (PC), Accumulator (ACCA), and the Carry Bit (C) *after* the execution of each of the following instructions using the given initial values of the Input Switches (Sw), Carry Bit (C), Program Counter (PC), Accumulator (ACCA) and content of the Memory Location 2 (M(2)).

Each of the following instructions has initial conditions as given:

Sw = 1011 C = 0 PC = 0010 ACCA=0110 M(2) = 1010

| | PC | ACCA | C |
|---|---|---|---|
| (i) ADDA S | | | |

Sw = 1100 C = 1 PC = 1100 ACCA=0101 M(2) = 0101

| | PC | ACCA | C |
|---|---|---|---|
| (ii) ROLA | | | |

Sw = 0110 C = 1 PC = 1010 ACCA=0110 M(2) = 1010

| | PC | ACCA | C |
|---|---|---|---|
| (iii) JEQ 0011 | | | |

Sw = 1011 C = 0 PC = 0010 ACCA=1111 M(2) = 1010

| | PC | ACCA | C |
|---|---|---|---|
| (iv) INCA | | | |

Sw = 1000 C = 0 PC = 0010 ACCA=0110 M(2) = 0111

| | PC | ACCA | C |
|---|---|---|---|
| (v) SUBA 2 | | | |

(b) (2 marks) Fill in the microcode table for the following instructions:

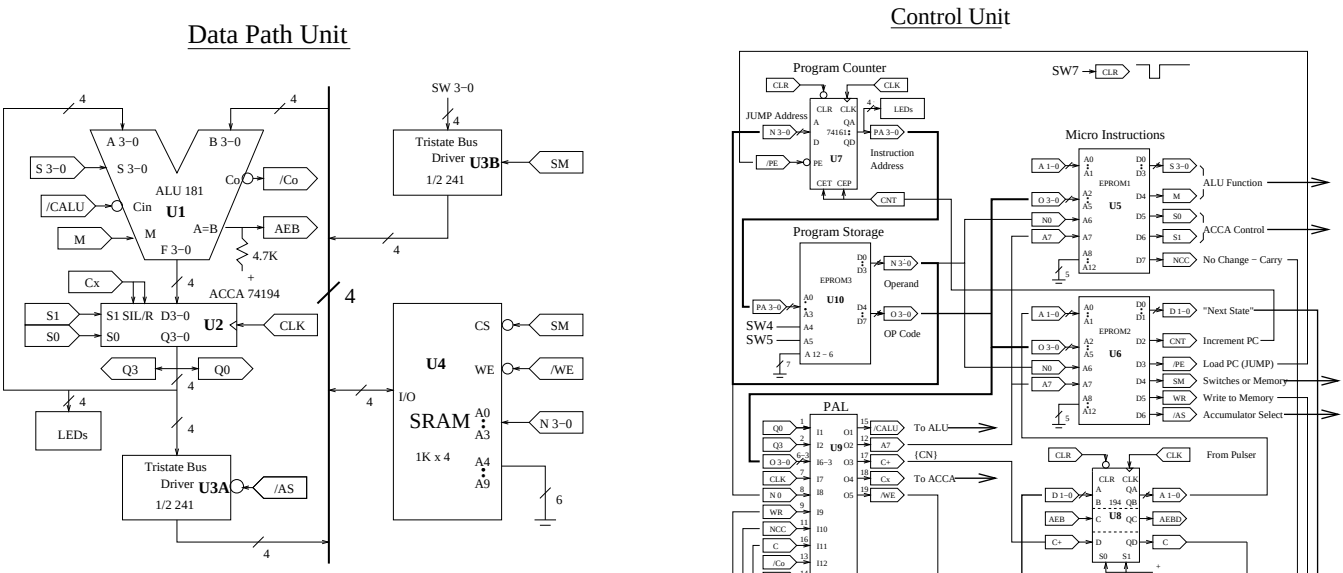
| | EPROMs 1&2 Address Lines | | | | | | | EPROM1 | | | | | | | | EPROM2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | EPROM3 | | | | | | PAL | ACCA | | ALU 181 | | | | | | DATA PATH | | | 161 | | EPROMs | |
| | JP | N0 | OP Code | | | | Mic Code | NCC | S1 | S0 | M | S3 | S2 | S1 | S0 | | /AS | WR | SM | /PE | CNT | A1+ | A0+ |
| | A7 | A6 | A5 | A4 | A3 | A2 | A1 A0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SUBA S | | | | | | | 0 0 | | | | | | | | | | | | | | | | |
| | | | | | | | 0 1 | | | | | | | | | | | | | | | | |
| | | | | | | | 1 0 | | | | | | | | | | | | | | | | |
| | | | | | | | 1 1 | | | | | | | | | | | | | | | | |
| STAA N | | | | | | | 0 0 | | | | | | | | | | | | | | | | |
| | | | | | | | 0 1 | | | | | | | | | | | | | | | | |
| | | | | | | | 1 0 | | | | | | | | | | | | | | | | |
| | | | | | | | 1 1 | | | | | | | | | | | | | | | | |

(c) (4 marks) Suppose that the STSW N instruction is required to be changed to a new instruction STEZ N. The new instruction STEZ N is used to compare the contents of the accumulator (ACCA) to the data from the Input Switches (Sw). If they are equal, a value of zero will be stored in the memory location N; otherwise, the next program instruction following the STEZ N instruction will be executed. Fill in the microcode for the STEZ N instruction in the following table:

| | EPROMs 1&2 Address Lines | | | | | | | EPROM1 | | | | | | | | EPROM2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | EPROM3 | | | | | | PAL | ACCA | | ALU 181 | | | | | | DATA PATH | | | 161 | | EPROMs | |
| | JP | N0 | OP Code | | | | Mic Code | NCC | S1 | S0 | M | S3 | S2 | S1 | S0 | | /AS | WR | SM | /PE | CNT | A1+ | A0+ |
| | A7 | A6 | A5 | A4 | A3 | A2 | A1 A0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| STEZ N | 0 | 0 | | | | | 0 0 | | | | | | | | | | | | | | | | |
| | 0 | 0 | | | | | 0 1 | | | | | | | | | | | | | | | | |
| | 0 | 0 | | | | | 1 0 | | | | | | | | | | | | | | | | |
| | 0 | 0 | | | | | 1 1 | | | | | | | | | | | | | | | | |
| STEZ N | 1 | 0 | | | | | 0 0 | | | | | | | | | | | | | | | | |
| | 1 | 0 | | | | | 0 1 | | | | | | | | | | | | | | | | |
| | 1 | 0 | | | | | 1 0 | | | | | | | | | | | | | | | | |
| | 1 | 0 | | | | | 1 1 | | | | | | | | | | | | | | | | |

(d) (3 marks) Write a program (not to exceed 16 instructions) to swap the data that are already stored at memory locations 2 and 3. You may use other memory locations as temporary storage.

# Appendix




CN = O3*/O2*/O1*/O0*N0 + O3*/O2*O1*/O0*(N0*Q3+/N0*Q0)/O3*/O2*/O1*C0 + /O3*O2*O1*/O0*C0 + NCC*C

CALU = O3*/O2*/O1*O0*N0 + /O3*/O2*/O1*C + /O3*O2*O1*/O0*C

A7 = O3*O2*/O1*/O0*C + O3*O2*/O1*O0*AEBD + O3*O2*O1*/O0*Q3

74194 (ACCA) Mode Selection

| S1 | S0 | Mode |
|---|---|---|
| 0 | 0 | Hold |
| 0 | 1 | Shift Right |
| 1 | 0 | Shift Left |
| 1 | 1 | Load |

74161 (PC) Mode Selection

| /PE | CNT | Mode |
|---|---|---|
| 1 | 0 | No Change |
| 1 | 1 | Count |
| 0 | x | Load |

Figure 1: Processor in Lab 7

*Arithmetic and logic functions performed by the 74181 ALU*

| $S_3$ | $S_2$ | $S_1$ | $S_0$ | M = 1 Logic functions | M = 0 Arithmetic operations /Cn = 1 (no carry) | M = 0 Arithmetic operations /Cn = 0 (with carry) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | F = /A | F = A | F = A plus 1 |
| 0 | 0 | 0 | 1 | F = /(A + B) | F = A + B | F = (A + B) plus 1 |
| 0 | 0 | 1 | 0 | F = /A*B | F = A + /B | F = (A + /B) plus 1 |
| 0 | 0 | 1 | 1 | F = 0 | F = minus 1 (2's comp) | F = 0 |
| 0 | 1 | 0 | 0 | F = /(A*B) | F = A plus A*/B | F = A plus A*/B plus 1 |
| 0 | 1 | 0 | 1 | F = /B | F = (A + B) plus A*/B | F = (A +B) plus A*/B plus 1 |
| 0 | 1 | 1 | 0 | F = A XOR B | F = A minus B minus 1 | F = A minus B |
| 0 | 1 | 1 | 1 | F = A*/B | F = A * /B minus 1 | F = A * /B |
| 1 | 0 | 0 | 0 | F = /A + B | F = A plus A*B | F = A plus A*B plus 1 |
| 1 | 0 | 0 | 1 | F = /(A XOR B) | F = A plus B | F = A plus B plus 1 |
| 1 | 0 | 1 | 0 | F = B | F = (A + /B) plus A*B | F = (A+/B) plus A*B plus 1 |
| 1 | 0 | 1 | 1 | F = A*B | F = A*B minus 1 | F = A*B |
| 1 | 1 | 0 | 0 | F = 1 | F = A plus A | F = A plus A plus 1 |
| 1 | 1 | 0 | 1 | F = A + /B | F = (A + B) plus A | F = (A + B) plus A plus 1 |
| 1 | 1 | 1 | 0 | F = A + B | F = (A + /B) plus A | F = (A + /B) plus A plus 1 |
| 1 | 1 | 1 | 1 | F = A | F = A minus 1 | F = A |

Figure 2: ALU Function Table

### Processor Instruction Set

- **0 0 0 0** $N_3N_2N_1N_0$, ADDA N : ACCA := ACCA + (N) + C.
- **0 0 0 1** $N_3N_2N_1N_0$, SUBA N : ACCA := ACCA - (N) - /C. **Note:** /C = Borrow Bit
- **0 0 1 0 X X X X**, INPA : load accumulator with input data from the switches, ACCA := Sw
- **0 0 1 1** $N_3N_2N_1N_0$, LDAA N : load accumulator with the contents of memory location N.
- **0 1 0 0** $N_3N_2N_1N_0$, STAA N : store ACCA into memory location N.
- **0 1 0 1** $N_3N_2N_1N_0$, JMP N : jump to program address N.
- **0 1 1 0 X X X 0**, ADDA S : ACCA := ACCA + Sw + C.
- **0 1 1 0 X X X 1**, SUBA S : ACCA := ACCA - Sw - /C.
- **0 1 1 1** $N_3N_2N_1N_0$, ANDA N : AND the contents of N with ACCA and store in ACCA.
- **1 0 0 0 X X X 0**, CLC : clear carry bit (C = 0).
- **1 0 0 0 X X X 1**, SEC : set carry bit (C = 1).
- **1 0 0 1 X X X 0**, DECA : decrement ACCA.
- **1 0 0 1 X X X 1**, INCA : increment ACCA.
- **1 0 1 0 X X X 0**, RORA : rotate ACCA right ($Q_3$ := C and C := $Q_0$). (Note: In your circuit $Q_3$ is on the left whereas in the 74194 specs, $Q_3$ is considered to be the rightmost bit).
- **1 0 1 0 X X X 1**, ROLA : rotate ACCA left ($Q_0$ := C and C := $Q_3$).
- **1 0 1 1** $N_3N_2N_1N_0$, STSW N : (N) := Sw.
- **1 1 0 0** $N_3N_2N_1N_0$, JCS N : jump to program address N if carry set.
- **1 1 0 1** $N_3N_2N_1N_0$, JEQ N: jump to program address N if A = B.

  Note: ALU must be in subtract mode and CALU clear (borrow = 1) to make F = 1111 (AEB HIGH) when A = B. For these instructions, A = ACCA and B = switches (input).
- **1 1 1 0** $N_3N_2N_1N_0$, JMI N : jump to program address N if ACCA -ve ($Q_3$ = 1).
- **1 1 1 1 X X X 0**, CLRA : ACCA := 0000.
- **1 1 1 1 X X X 1**, SETA : ACCA := 1111.