ENGN3213
Digital Systems and Microprocessors

Writing Period: 2 hours
Study Period: 15 minutes duration

Permitted Materials: Pens

Total 50 Marks (20% of Subject)

You must attempt all 5 questions
Q1. Fixed point arithmetic (10 marks; 20% Total)

Evaluate the following (1 mark each),

(a) $130_{10} = X_{2}$, => $X = \text{10000010}$
(b) $123_{16} = Y_{5}$, => $Y = 123_{16} = 291_{10} = 213_{5}$
(c) $0111 + 011 = \text{1110}$
(d) $0110 - 0001 = \text{101}$
(e) $0101 \times 0011 = \text{1111}$
(f) $1100111100_{2} = X_{16}$ => $X = (0011)(0011)(1100) = \text{33C}$
(g) $1100111100_{2} = Y_{8}$ => $Y = ((001)(100)(111)(100) = \text{1474}$
(h) What is the integer value of the C variable A defined by,

```
signed char A = 133;
```

-123

(i) Compute $4_{10} + (-6)_{10}$ using two’s complement arithmetic. $0100 + (-0110) = 0100 + 1010 = 1110$ is 2’s comp. Thus $1110 - 1 = 1101 => 0010 = -2_{10}$

(j) If unsigned short $Y = X*X$, where $X = 257$ what does $Y$ evaluate to? You may assume that an unsigned short has a width of 16 bits. **513**
Q2. General Knowledge (20 marks; 40% Total)

(a) Describe how a PIC microcontroller executes a branch instruction such as a subroutine call. (3 marks)

The PIC pushes the current value of the program counter onto the stack register, replaces the PC with the new address of the branch instruction, executes the branch and then returns by popping the stack back into the PC.

(b) Explain using diagrams how a Schmitt Trigger can debounce a switch. (3 marks)

See lecture 11 from the 2009 lecture notes.

(c) (1) Draw the truth table of a T flip-flop. (2 marks)

<table>
<thead>
<tr>
<th>d</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Q</td>
<td>Q'</td>
</tr>
<tr>
<td>0</td>
<td>Q</td>
<td>Q'</td>
</tr>
</tbody>
</table>

(2) Sketch a circuit implementation of the T flip-flop using one D-flip-flop. (3 marks)

![Circuit Diagram](a)

![Truth Table](b)

ENGN3213 Digital Systems and Microprocessors
(d) The D flip-flop shown in the following figure is applied the input signals shown in the traces.

(1) Define in words the meaning of setup and hold time (2 marks)
   Setup time is the time that the input to a sequential device should be stable before the active clock edge for the output to be valid.
   Hold time is the time that the input to a sequential device should be stable after the active clock edge for the output to be valid.

(2) Redraw the traces in the figure in your script books and label the setup and hold times of the flip-flop (2 marks).
(e) In terms of the functional blocks shown in the following figure of a simplified microcontroller, describe in detail the steps involved when the microcontroller adds two numbers stored in RAM and outputs them to an IO port. (5 marks).

The first data value is fetched from RAM and stored in the working register. The second value is fetched from RAM and added to the contents of the working register. The result is stored in the working register. The contents of the working register are transferred to the IO port.

\[ PC = \text{Program Counter (register)} \]
\[ W = \text{Working Register} \]
\[ SR = \text{Status Register} \]
Q3. Logic Synthesis and Karnaugh Maps (6 marks; 12% Total)

Given the following expression representing a detector for prime numbers lower than 16,

\[ F = \Sigma_{A,B,C,D} (1, 2, 3, 5, 7, 11, 13) \]  \hspace{1cm} (1)

(1) Write out the SOP representation of this expression. \textbf{(1 mark)}

\[ F = ABCD + A\overline{B}C\overline{D} + \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}CD \]

(2) Draw the truth table. \textbf{(1 mark)}

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(3) Draw the Karnaugh map from the truth table. \textbf{(1 mark)}

(4) On the Karnaugh map loop out the logical adjacencies. \textbf{(1 mark)}
(5) Write down the minimised Boolean logic expression arising from the Karnaugh map. (1 mark)

\[ F = D\overline{A} + D(\overline{B}C + \overline{C}B) + C\overline{A}\overline{B} \]

(6) Draw the circuit implementation. (1 mark)
Q4. State machines and VERILOG  (10 marks; 20% Total)

A synchronous system has an input X and an output Y where the output will only be '1' when the number of 1’s received is even and the number of 0’s received is odd respectively. Assume initially that no 0’s or 1’s have been received.

(1) Design a synchronous state machine and input logic to perform this task. Use any convenient flip flop arrangement for its implementation.  (6 marks)

(2) Write VERILOG code that implements this system in HDL.  (4 marks)
We must receive a finite number of 1’s and 0’s before the FSM starts operating. The only input is X. The state diagram below shows the even/odd detector. The detector is a Moore machine because the output is only a function of the state. All states have output 0 except for $S_6$ where it is 1 because we have received an even number of 1’s and an odd number of 0’s. From the State diagram we have the following next state table.

<table>
<thead>
<tr>
<th>State</th>
<th>X</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>$S_1$</td>
<td>$S_4$</td>
<td></td>
</tr>
<tr>
<td>$S_1$</td>
<td>$S_0$</td>
<td>$S_2$</td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>$S_4$</td>
<td>$S_6$</td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>$S_4$</td>
<td>$S_6$</td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>$S_3$</td>
<td>$S_5$</td>
<td></td>
</tr>
<tr>
<td>$S_5$</td>
<td>$S_6$</td>
<td>$S_4$</td>
<td></td>
</tr>
<tr>
<td>$S_6$</td>
<td>$S_0$</td>
<td>$S_0$</td>
<td></td>
</tr>
</tbody>
</table>

ENGN3213 Digital Systems and Microprocessors
Notice that after $S_6$ we go back to $S_0$, ignoring all past inputs and starting over. We need at least 3 bits to describe the states so define the following.

<table>
<thead>
<tr>
<th>State</th>
<th>ABC</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>ABC</td>
<td>000</td>
</tr>
<tr>
<td>$S_1$</td>
<td>ABC</td>
<td>001</td>
</tr>
<tr>
<td>$S_2$</td>
<td>ABC</td>
<td>010</td>
</tr>
<tr>
<td>$S_3$</td>
<td>ABC</td>
<td>011</td>
</tr>
<tr>
<td>$S_4$</td>
<td>ABC</td>
<td>100</td>
</tr>
<tr>
<td>$S_5$</td>
<td>ABC</td>
<td>101</td>
</tr>
<tr>
<td>$S_6$</td>
<td>ABC</td>
<td>110</td>
</tr>
</tbody>
</table>

where A, B and C represent the state bits.

From the next state table we have the following Karnaugh maps. We have introduced a fake state 111 which returns to $S_0$ (it will be handled by the default case in VERILOG). Finally don’t forget to arrange the inputs in Gray order. The three Karnaugh maps are shown on the following page.

The Karnaugh maps lead to the following relations

$$A' = (\overline{B} + \overline{A})\overline{C}X + \overline{A}B + \overline{B}A(X + C)$$

$$B' = (C + B)X\overline{A} + A\overline{B}X$$

$$C' = \overline{ABC} + \overline{XBC}$$

Where the prime denotes the next state.
ENGN3213 Digital Systems and Microprocessors
The output logic table is.

<table>
<thead>
<tr>
<th>State</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>000</td>
</tr>
<tr>
<td>S1</td>
<td>001</td>
</tr>
<tr>
<td>S2</td>
<td>010</td>
</tr>
<tr>
<td>S3</td>
<td>011</td>
</tr>
<tr>
<td>S4</td>
<td>100</td>
</tr>
<tr>
<td>S5</td>
<td>101</td>
</tr>
<tr>
<td>S6</td>
<td>110</td>
</tr>
</tbody>
</table>

The output logic is therefore simply

\[ Y = AB \]

One can combine the three above next state logic equations, the three flip flops and the output logic to obtain the following circuit diagram.
The Next state logic circuits are as follows.

ENGN3213 Digital Systems and Microprocessors
There are two ways to write the VERILOG. (i) Implement the above circuits directly. This would require that the D flip flops be modelled by a sequential (non-blocking) ALWAYS block and the next state and output logic as combinational (blocking) ALWAYS blocks. Alternatively (ii) we can use the coding style of Wakerly p. 651 directly from the next state and output logic tables. Following the latter we have:

```verilog
module Q4VERILOG(clk,X,Y);
    input clk;
    input X;
    output Y;
    reg Y;
    parameter S0 = 3'b000;
    parameter S1 = 3'b001;
    parameter S2 = 3'b010;
    parameter S3 = 3'b011;
    parameter S4 = 3'b100;
    parameter S5 = 3'b101;
    parameter S6 = 3'b110;
    reg [2:0] state;
    reg [2:0] nextstate;

    //state memory
    always @(posedge clk) state <= nextstate;

    //next state logic
    always @(*) begin
        case (state)
            S0: if(X) nextstate = S4; else nextstate = S1;
            S1: if(X) nextstate = S2; else nextstate = S0;
            S2: if(X) nextstate = S6; else nextstate = S4;
            S3: if(X) nextstate = S6; else nextstate = S4;
            S4: if(X) nextstate = S5; else nextstate = S3;
            S5: if(X) nextstate = S4; else nextstate = S6;
            S6: if(X) nextstate = S0; else nextstate = S0;
            default: nextstate = S0;
        endcase
    end

    //output logic
    always @(state) begin
        case (state)
            S0: Y = 0;
            S1: Y = 0;
            S2: Y = 0;
            S3: Y = 0;
            S4: Y = 0;
            S5: Y = 0;
            S6: Y = 1;
            default: Y = 0;
        endcase
    end
endmodule
```

ENGN3213 Digital Systems and Microprocessors
Q5. C Programming (4 marks; 8% Total)

The code below demonstrates the use of pointers in C. Answer the following questions.

(1) What are the values of the array “Dorian” after execution of the function “image” (2 marks).

Before image the values of Dorian are the numbers from 0 - 9. After the execution of the image the numbering of these is reversed. That is Dorian takes on the values:

9 8 7 6 5 4 3 2 1 0

(2) Describe how data is passed from the main routine to “image” (2 marks).
The variable Dorian is a pointer to an array of chars. Dorian is passed by reference to the new pointer Gray in the scope of image. The value len however is passed by value. Variables that are passed by reference (i.e. their memory address is passed) retain any modifications produced by the subroutine (image). Variables that are passed by value (such as len) are unaffected by the subroutine (image).
For example study the operation of the following programs. What are the values of X produced by the printf statements in each case?

```c
#include <stdio.h>

void sub( int Y ) {
    Y = 0;
}

int main() {
    int X = 1;
    printf("X = %d\n",X);
    sub( X );
    printf("X = %d\n",X);
    return 0;
}

#include <stdio.h>
#include <stdlib.h>

void sub( int *Y ) {
    *Y = 0;
}

int main() {
    int * X;
    X = malloc(sizeof(int));
    *X = 1;
    printf("*X = %d\n",*X);
    sub( X );
    printf("*X = %d\n",*X);
    return 0;
}

#include <stdio.h>

void sub( int *Y ) {
    *Y = 0;
}

int main() {
    int X = 1;
    printf("X = %d\n",X);
    sub( &X );
    printf("X = %d\n",X);
    return 0;
}
```

ENGN3213 Digital Systems and Microprocessors