Solution for HW- 4

Problem 1
B – 2-bit input \{B[1] B[0]\}
\(F(A,B) = \text{NOT}(A) + B\)

\textit{a. Draw the truth table for the function } F(A,B).

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 \\
\end{array}
\]
b. Implement $Y[0]$ using AND/ OR/ NOT/ XOR gates.

\[ Y[0] \]

\[ A[0] \quad B[0] \quad Y[0] \]

\[ Y[2] \]

\[ A[1] \quad A[0] \quad B[1] \quad B[0] \]

\[ Y[1] \]

\[ A[1] \quad A[0] \quad B[1] \quad B[0] \]

\[ Y[0] \]

\[ A[1] \quad A[0] \quad B[1] \quad B[0] \]

\[ \text{Output (Checks for 6)} \]

c. Given the blocks for implementation of $Y[2]$, $Y[1]$, $Y[0]$, implement the logic that will indicate if $F(A,B)$ is equal to 6.

Problem 2

You are assigned the task of designing a circuit with some sequential logic elements and logic gates. The circuit should turn on a light when the input remains 1 (High) for 3 consecutive clock cycles. Once the light is turned on, it should be in ON position irrespective of the changes in the input.

a. Draw the state diagram as in figure 3.31 on page 78 ItCS. How many memory elements do we need?

The states are represented by the following bit patterns for one hot encoding.

State A → 0001
State B → 0010
State C → 0100
State D → 1000
b. Get the truth table for the circuit.

<table>
<thead>
<tr>
<th>Input</th>
<th>Q₃</th>
<th>Q₂</th>
<th>Q₁</th>
<th>Q₀</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
c. Show the block diagram of the circuit and the combinational logic used as in figures 3.32a and 3.32b on page 79 of ItCS.

\[ D_0 = [\text{NOT}(\text{Input}) \ \text{AND} \ Q_0] \ \text{OR} \ [\text{NOT}(\text{Input}) \ \text{AND} \ Q_1] \ \text{OR} \ [\text{NOT}(\text{Input}) \ \text{AND} \ Q_2] \]

\[ D_1 = (\text{Input}) \ \text{AND} \ (Q_0) \]

\[ D_2 = (\text{Input}) \ \text{AND} \ Q_1 \]

\[ D_3 = [\text{NOT}(\text{Input}) \ \text{AND} \ Q_3] \ \text{OR} \ [(\text{Input}) \ \text{AND} \ Q_3] \ \text{OR} \ [(\text{Input}) \ \text{AND} \ Q_2] \]

d. How many fewer flip-flops would it have taken using binary state encoding?

Binary state encoding uses \( 2 \) flip-flops \([\text{ceil}(\log_2 N) \ \text{flip-flops where} \ N \ \text{is the number of states. One hot encoding uses} \ N \ \text{flip-flops}\) Here, \( 4 \) flip-flops.

Hence, \( 4 - 2 = 2 \) fewer flip-flops would have taken for the above circuit using binary state encoding.

Problem 3

a) Design a 2 input OR gate using only a 2:1 MUX
b) Design a 2 input AND gate using only a 2:1 MUX
(Note: No additional logic elements are to be used)

Problem 4

If you see the figure 4.3 in page 102 of ItCS, there is one register called PC (Program Counter). It is a 16 bit register and has an input which loads the program counter. Your job is to design the program counter using master-slave flip-flops (You can use AND/ OR/ NOT/ MUX). A single master-slave flip-flop can store 1 bit. Hence, you can go for a hierarchical design – show a 4 bit register using 4 master-slave flip-flops. Combine 4 of this type to get the 16 bit register (PC).

Constraint: Input to the clock terminal of the flip-flop should only be clock signal (i.e. you are not allowed to combine the clock signal with any other signal and input that at the clock terminal of the flip-flop)

Singe Bit flip flop with enable pin:
4-bit register:

(The same clock is fed to all the flip-flops)
Below figure is the 16 bit register.
Problem 5

*State the phases of instruction cycle and briefly describe what operations occur in each phase of the instruction cycle.*

- Fetch
- Decode
- Evaluate Address
- Fetch Operands
- Execute
- Store Address

For description of each of the stages, refer ItCS page105.

Problem 6

*A 32 bit instruction has the following format.

<table>
<thead>
<tr>
<th>OPCODE</th>
<th>DR</th>
<th>SR1</th>
<th>SR2</th>
<th>Unused bits (if any)</th>
</tr>
</thead>
</table>

If there are 255 opcodes and 64 registers,

\[ a) \text{ What is the minimum number of bits required to represent the OPCODE?} \]

There are 255 OPCODES and each requires a unique pattern. Therefore, we need \( \text{ceil}(\log_2{255}) = 8 \text{ bits} \).

\[ b) \text{ What can be the maximum number of bits that can represent the OPCODE?} \]

To find this, we have to find the number of bits that are used for the registers. Since there are 64 registers \( \Rightarrow \) 6 bits are needed \( \text{ceil}(\log_2{64}) = 6 \text{ bits} \) for each register.

It implies, DR, SR1 and SR2 all require 18 bits in all.
No of bits available for OPCODES = \( 32 - 18 \) = 14 bits.

\[ c) \text{ What can be the maximum number of the unused bits?} \]

Total number of bits used = bits used for registers + bits used for OPCODE
\[ = 18 + 8 \]
\[ = 26 \]
Total No. of Bits we have = 32
Hence, No. of Unused bits = \( 32 - 26 \)
\[ = 6 \text{ bits} \]
d) What could be the potential advantage of using more than the minimum required number of bits for representing the OPCODE?

Assume that we represent the OPCODE with 10 bits instead of 8 bits. This lets us to represent 1024 OPCODES instead of 256. Thus, increasing the No. of bits for OPCODE gives us the flexibility of expanding the structure of the instruction. (This is called Scalability)

Problem 7

a) If the machine cycle is 4 nanoseconds (i.e. $4 \times 10^{-9}$ seconds), how many machine cycles occur each second?

Duration of 1 machine cycle = $4 \times 10^{-9}$ seconds

$\Rightarrow$ In $4 \times 10^{-9}$ seconds $\rightarrow$ 1 machine cycle

Hence, in 1 second, $1 / (4 \times 10^{-9})$ machine cycles

$= 0.25 \times 10^9$ machine cycles

b) If the computer requires on the average 5 cycles to process each instruction, and the computer processes instructions one at a time from beginning to end, how many instructions can the computer process in 1 second?

Instructions/ second = $(1/5) \times 0.25 \times 10^9$

$= 0.05 \times 10^9$ Instructions/second.