CSE 140 Homework Four

November 24, 2009

Only Problem Set Part B will be graded. Turn in only Problem Set Part B which will be due on December 4, 2009 (Friday) at 4:00pm.

1 Problem Set Part A

- textbook 5.13(c)
- textbook 5.16
- textbook 5.17(a)(c)
- textbook 6.2
- textbook 6.6
- textbook 6.10
- textbook 6.11
- textbook 6.13
- textbook 6.14
- textbook 6.19
- textbook 6.20
2 Problem Set Part B

1 (Priority Encoders)

The priority encoder that has been discussed in class implements an increasing priority; when both $D_i$ and $D_j$ are on, where $i > j$, it returns $i$. In this question, you are asked to implement a new priority scheme where the inputs with odd indices have priority over the ones with even indices. When there is a tie, the larger index gets the higher priority. For example, if only $D_3$ and $D_6$ are on, the input with odd index, $D_3$, has the higher priority and 3 is returned as the encoder output. If only $D_6$ and $D_4$ are on, the input with the larger index, $D_6$, has the higher priority and 6 is returned as the encoder output.

(PART A) Please fill in the missing entries in the following table. You can (and should) use don’t cares for some of the $D_i$’s in order to represent the requested case fully.

<table>
<thead>
<tr>
<th>$D_7$</th>
<th>$D_6$</th>
<th>$D_5$</th>
<th>$D_4$</th>
<th>$D_3$</th>
<th>$D_2$</th>
<th>$D_1$</th>
<th>$D_0$</th>
<th>$A_2$</th>
<th>$A_1$</th>
<th>$A_0$</th>
<th>ANY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(PART B) An implementation of an odd-even priority encoder is given in the following graph by using the 4-to-2 and 2-to-1 increasing priority encoders and 2-to-1 selectors. In the implementation given below, the input and output signals are missing. Please label the inputs and outputs by using $D_7,D_6,D_5,D_4,D_3,D_2,D_1,D_0$ and $A_0,A_1,A_0$, ANY, respectively.
2 (FSM transformations)

(Part A) The following FSM has been implemented using two T flip-flops and a set of gates for the next state logic.

A simple modification is made to the sequential circuit that implements this FSM; both T flip-flops are replaced with D flip-flops, while the next state logic is kept intact. Please identify the FSM that is implemented by the modified circuit through drawing transitions on the following FSM skeleton.
(Part B) The following FSM has been implemented using a D flip-flop, a T flip-flop, and a set of gates for the next state logic. However, we do not know whether the D flip-flop is used to control the most significant bit or the least significant bit.

A simple modification is made to the sequential circuit that implements this FSM; the T and the D flip-flops are swapped (so that the one that used to control the most significant bit now controls the least significant bit and vice versa), while the next state logic is kept intact. Please identify the FSM that is implemented by the modified circuit through drawing transitions on the following FSM skeleton.
3 (Sequential logic flip-flops)

Recall that an SR flip-flop has two inputs, $S$(set) and $R$(reset) that set or reset the flip-flop when asserted. In other words, when $S = 1$ and $R = 0$, the flip-flop output $Q$ is set to 1, while when $S = 0$ and $R = 1$ it is reset to 0. When $S = 0$ and $R = 0$, the flip-flop retains the current state. The excitation table of the SR flip-flop is shown at the end of the test.

A company developed a new type of flip-flop denoted as the CR (Crazy) flip-flop, which is very similar to an SR flip-flop: $C$ is used for set, $R$ for reset, and $C$ and $R$ are never asserted at the same time. The only difference is that when $C = 0$ and $R = 0$, the CR flip-flop toggles, instead of retaining the current state as in the case of an SR flip-flop.

(Part A) Fill in the excitation table below for the CR flip-flop.

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$Q$ (next)</th>
<th>$C$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(Part B) Analyze the sequential logic circuit implemented with the CR flip-flops given below. Specifically, derive the following:

1) excitation equations:

\[ C_1 = \]

\[ R_1 = \]

\[ C_0 = \]

\[ R_0 = \]

output \( Y = \)

2) the next-state equations:

\[ Q_0(next) = \]

\[ Q_1(next) = \]
3) the next-state table

4) the state diagram of the circuit.

(Part C) Based on the CR flip-flop, an engineer is to perform state encoding. Some colleagues suggested the use of the minimum-bit-change strategy. Basically, a minimum-bit-change strategy assigns Boolean values to the states in such a way that the total number of bit changes for all state transitions is minimized. An error prone engineer seems to have bungled things up and instead of utilizing a well-established heuristic such as the minimum-bit-change instead ended up using a maximum-bit-change approach, wherein the total number of bit changes over all state transitions is maximized.

Please let us know whether the engineer should be fired, since his designs would be unnecessarily costly, whether he can quietly keep his job, since it does not seem to make a difference whether one uses the maximum or minimum bit change heuristic for CR flip-flops, or whether he should be given a bonus and promotion since the maximum bit flip heuristic he inadvertently ended up using seems to be utilizing on the average less hardware for CR flip-flops. Please provide a reasoning for your answer.