

CSE 140 Homework Five

November 21, 2016

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on December 2, 2016 (Friday) at 3:00pm.*

*Submit homework via gradescope <gradescope.com> as a **PDF**. Other formats may or may not be viewed and evaluated accurately.*

1 Problem Set Part A

All questions in this part are from Roth&Kinney, 7th Edition.

- 14.12, 14.13, 14.21, 14.22, 14.23
- 15.10, 15.11, 15.13, 15.14, 15.15, 15.17, 15.18, 15.19, 15.24, 15.26, 15.27, 15.28, 15.33
- 9.13, 9.14(a), 9.16, 9.18, 9.19, 9.20, 9.21, 9.27, 9.29, 9.31, 9.36

2 Problem Set Part B

1: FSM Analysis and Unused States

Usually, the number of states of an FSM is not a perfect power of 2, thus introducing don't-care states in an implementation. When one analyzes a circuit, it may be difficult to distinguish a spurious state from a real one. This question probes your understanding of how this puzzle can be resolved.

For this problem, assume that the FSM has been minimized, and assume that the next state logic is in the minimum 2-level form. You should assume state encodings were assigned randomly – no heuristics were applied to generate state encodings.

(Part A) The first technique we discussed in class was to guess that a certain encoding was indeed unused, and update the corresponding positions in the next-state logic K-Maps. Comparing the resulting logic with the initially given logic allows one to make inferences about whether or not the state was used or not.

Below, we have given the state transition diagram for FSM *A* that was implemented using T Flip Flops, which had input *I* and output *O*. **Fill in the K-Maps for the T inputs to the T flip flops and for the output function. Then, apply the technique discussed above to the 101 encoding, and tell us whether it must have been a used encoding; whether it must have been an unused encoding; or if it could be either used or unused, based on the comparison of the resulting logic and the original logic. If you were able to say definitively whether the encoding is used or not, explain why the results indicate this. If you cannot make this determination, explain why not.**

State Name	Current State			Next State/Output	
	Q_2	Q_1	Q_0	$I = 0$	$I = 1$
A_0	0	0	0	111/1	110/0
A_1	0	0	1	111/0	111/0
A_2	0	1	0	111/0	110/1
A_3	0	1	1	111/0	110/0
A_4	1	0	0	111/1	110/0
A_5	1	0	1	111/0	011/0
A_6	1	1	0	111/0	110/1
A_7	1	1	1	111/0	000/0

T_2				
$Q_2Q_1 \setminus Q_0I$	00	01	11	10
00				
01				
11				
10				

T_1				
$Q_2Q_1 \setminus Q_0I$	00	01	11	10
00				
01				
11				
10				

T_0				
$Q_2Q_1 \setminus Q_0I$	00	01	11	10
00				
01				
11				
10				

O				
$Q_2Q_1 \setminus Q_0I$	00	01	11	10
00				
01				
11				
10				

(Part B) Repeat this technique for encoding 011. Again, **tell us whether it must have been a used encoding; if it must have been an unused encoding; or if it could be either, and explain why the results justify your decision.**

State Name	Current State			Next State/Output	
	Q_2	Q_1	Q_0	$I = 0$	$I = 1$
A_0	0	0	0	111/1	110/0
A_1	0	0	1	111/0	111/0
A_2	0	1	0	111/0	110/1
A_3	0	1	1	111/0	110/0
A_4	1	0	0	111/1	110/0
A_5	1	0	1	111/0	011/0
A_6	1	1	0	111/0	110/1
A_7	1	1	1	111/0	000/0

T_2							T_1						
$Q_2Q_1 \setminus Q_0I$		00	01	11	10		$Q_2Q_1 \setminus Q_0I$		00	01	11	10	
00							00						
01							01						
11							11						
10							10						

T_0							O						
$Q_2Q_1 \setminus Q_0I$		00	01	11	10		$Q_2Q_1 \setminus Q_0I$		00	01	11	10	
00							00						
01							01						
11							11						
10							10						

(Part C) Another technique we discussed in class was to compare two implementations of the same FSM functionality via FSM equivalence testing.

Below, we give the state transition table for an FSM B with equivalent functionality to FSM A from (Part A). **Perform FSM equivalence testing on FSMs A and B . Then, tell us what the state matchings tell us about used and unused encodings in each FSM.** (*Hint: if any states are equivalent, there is one pair of states that had better be equivalent right off the bat. You can use this insight to speed up the implication table filling.*)

State Name	Current State			Next State/Output	
	Q_2	Q_1	Q_0	$I = 0$	$I = 1$
A_0	0	0	0	111/1	110/0
A_1	0	0	1	111/0	111/0
A_2	0	1	0	111/0	110/1
A_3	0	1	1	111/0	110/0
A_4	1	0	0	111/1	110/0
A_5	1	0	1	111/0	011/0
A_6	1	1	0	111/0	110/1
A_7	1	1	1	111/0	000/0

State Name	Current State			Next State/Output	
	Q_2	Q_1	Q_0	$I = 0$	$I = 1$
B_0	0	0	0	111/0	110/0
B_1	0	0	1	111/0	010/0
B_2	0	1	0	111/0	100/0
B_3	0	1	1	111/0	000/0
B_4	1	0	0	111/0	100/1
B_5	1	0	1	111/1	100/0
B_6	1	1	0	111/0	101/1
B_7	1	1	1	111/0	101/0

	A0	A1	A2	A3	A4	A5	A6	A7
B0								
B1								
B2								
B3								
B4								
B5								
B6								
B7								

(Part D) Of the two techniques discussed previously, one technique indicates that certain encodings are definitely used if a difference arises between the old implementation and the new one; the other indicates that certain encodings are definitely unused. Let's call the former technique a *state confirming technique*, and the latter technique a *state discarding technique*.

- (i) Suppose you know encoding S is definitely a used encoding. With only this knowledge, it is nearly always possible to determine that other states are definitely used as well, without having to go to the trouble of editing and re-synthesizing the entire FSM. **Identify a relationship between an encoding E and the definitely-used encoding S which allows you to say that E is also definitely used.**

- (ii) Next, you may have noticed that certain pairs of encodings in one of the FSMs in (Part C) were equivalent. However, since the designers minimized both FSMs, only one encoding could have been the intended encoding. **Use the relationship you found in (Part Di) to determine which of the encodings in each pair of equivalent encodings was actually used by the designer.**

- (iii) Similarly, suppose S is definitely NOT used. **Identify a relationship between a state E and the definitely-unused state S which allows you to say that E is also unused. Then, identify the pairs of states in the FSMs from (Part C) that exhibit this relationship.**

2: State Minimization

As we have studied in class, state minimization can be performed through utilizing an *implication table*. A n -state FSM typically needs a $(n - 1) \times (n - 1)$ table. To fill in the table, we start by entering an “ $\times$ ” in the entry corresponding to state pair (s_i, s_j) , if s_i and s_j have distinct outputs. On the other hand, if s_i and s_j have identical outputs, we enter the next state pair(s) of s_i, s_j , if any, that need to be equivalent in order for s_i and s_j to be equivalent.

(Part A) State minimization is to be performed on a **12-state** FSM. The FSM produces a 2-bit output. According to the output values, the twelve states can be first partitioned into **four** classes of states with identical outputs. The sizes of these classes are 5, 4, 2, 1, respectively.

If a single implication table is to be used to perform state minimization for this 12-state FSM, how many “ $\times$ ”s will be placed in the table initially? Please give a brief explanation for your answer.

(Part B) The results of state minimization show that the **12-state** FSM in (Part A) has 6 groups of equivalent states. The sizes of these 6 groups are 4, 3, 2, 1, 1, 1, respectively. At the end of the state minimization process, how many entries in the implication table are left blank (i.e., have not been crossed out during the state minimization process)? Please give a brief explanation for your answer.

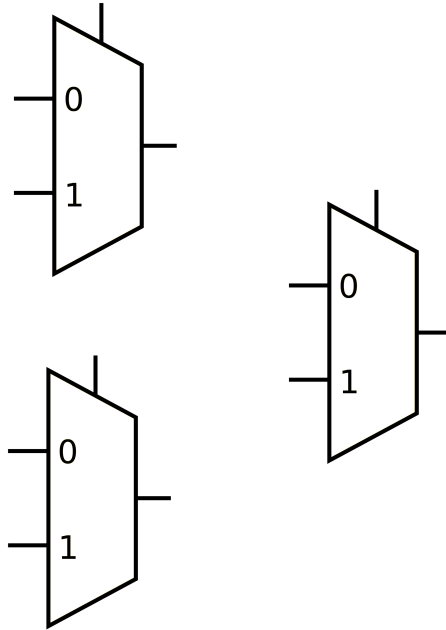
(Part C) A group of 140 students from a previous year had been working on the following state minimization problem for an FSM with 7 states. After filling in the implication table, they took a break and went to Price Center for lunch. When they came back, they found that someone had poured some coffee on their working sheet, and some entries of the implication table had become illegible.

S_1	$\langle 3,4 \rangle$					
S_2		$\langle 4,5 \rangle$				
S_3		$\langle 0,5 \rangle$				
S_4				$\langle 4,6 \rangle$		
S_5			$\langle 3,6 \rangle$		$\langle 3,5 \rangle$	
S_6						
	S_0	S_1	S_2	S_3	S_4	S_5

Rather than starting from the beginning (or chasing the coffee spiller across Price Center, for that matter), they turned instead to one of their 140 colleagues for help who took pity on them and told them that the states S_0 and S_1 are **non-equivalent**. It was later discovered that the 140 colleague turned out to be the culprit who actually spilled coffee on their papers in a brazen attempt to curve-fix. The gruesome developments are perhaps best left to police annals. With the benefit of hindsight and having read the police report we now know S_0 and S_1 , are actually **equivalent**. Can you help these students identify all the equivalent state classes of the final answer?

The equivalent state classes are:

(Part C) Using the results of the transformation that you just performed, the function $g(a, b, c) = a'b'c + bc'$ can now be implemented using only multiplexers (and possibly inverters). Connect the three multiplexers provided and label their inputs to correctly implement the function $g(a, b, c)$.



(Part D) In the previous parts of this question the function $g(a, b, c)$ was processed along its variables in the following order: $a \rightarrow b \rightarrow c$. Consider the other possible orderings. Does the order affect the number of multiplexers used to implement the function? If so, please give an ordering that facilitates an implementation using the fewest multiplexers. If no better ordering exists, please give a brief explanation as to why.