

CSE 140 Homework Three

July 20, 2010

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on July 30, 2010 (Friday) at 4:00pm.*

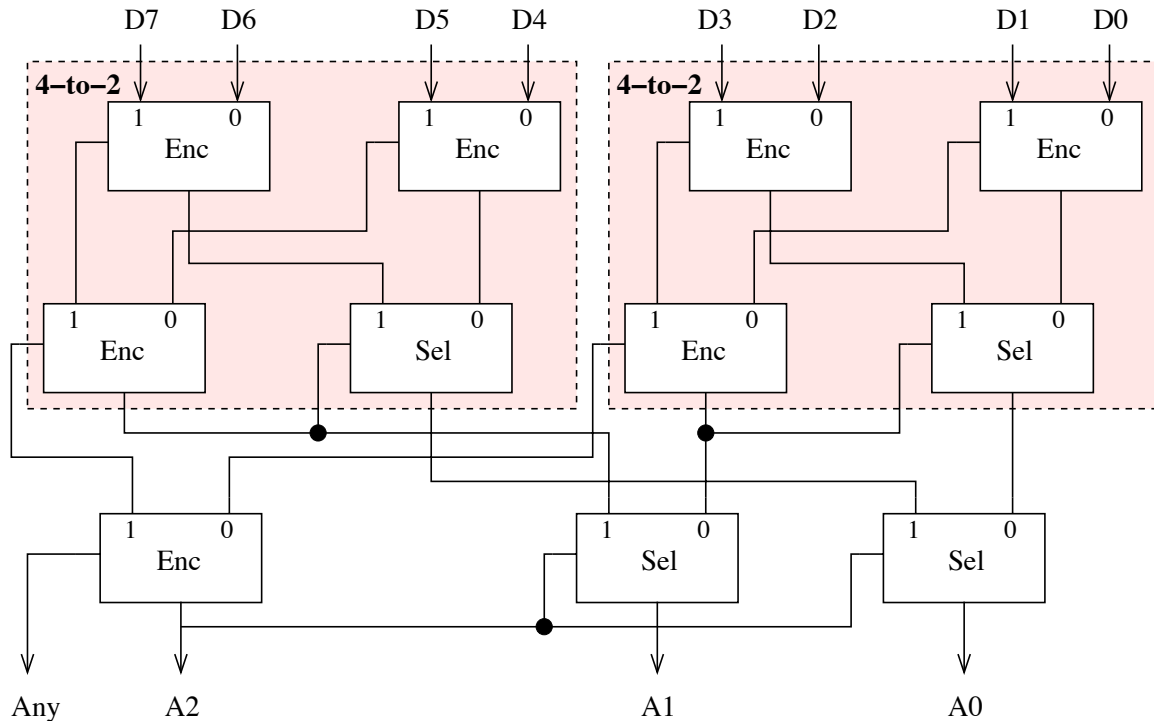
1 Problem Set Part A

- textbook 5.3
- textbook 5.9
- textbook 5.10
- 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)

Priority encoders produce as output an encoded representation of the active input line with the highest priority. We have studied in class the design of 8-to-3 priority encoders using 2-to-1 priority encoders and selectors, as shown in the following figure. The figure also shows that the first two levels of this 8-to-3 encoders can be furthermore broken down into two 4-to-2 priority encoders.



In this question, we will extend this 8-to-3 encoder design to 2^k -to- k encoders for larger k 's. Basically, a 2^k -to- k priority encoder is composed of k levels of 2-to-1 encoders and selectors.

	# of encoders	# of selectors
1st level	8	0
2nd level	4	4
3rd level		
4th level		

	# of encoders	# of selectors
i^{th} level		
total		

2 (Programmable Logic Arrays)

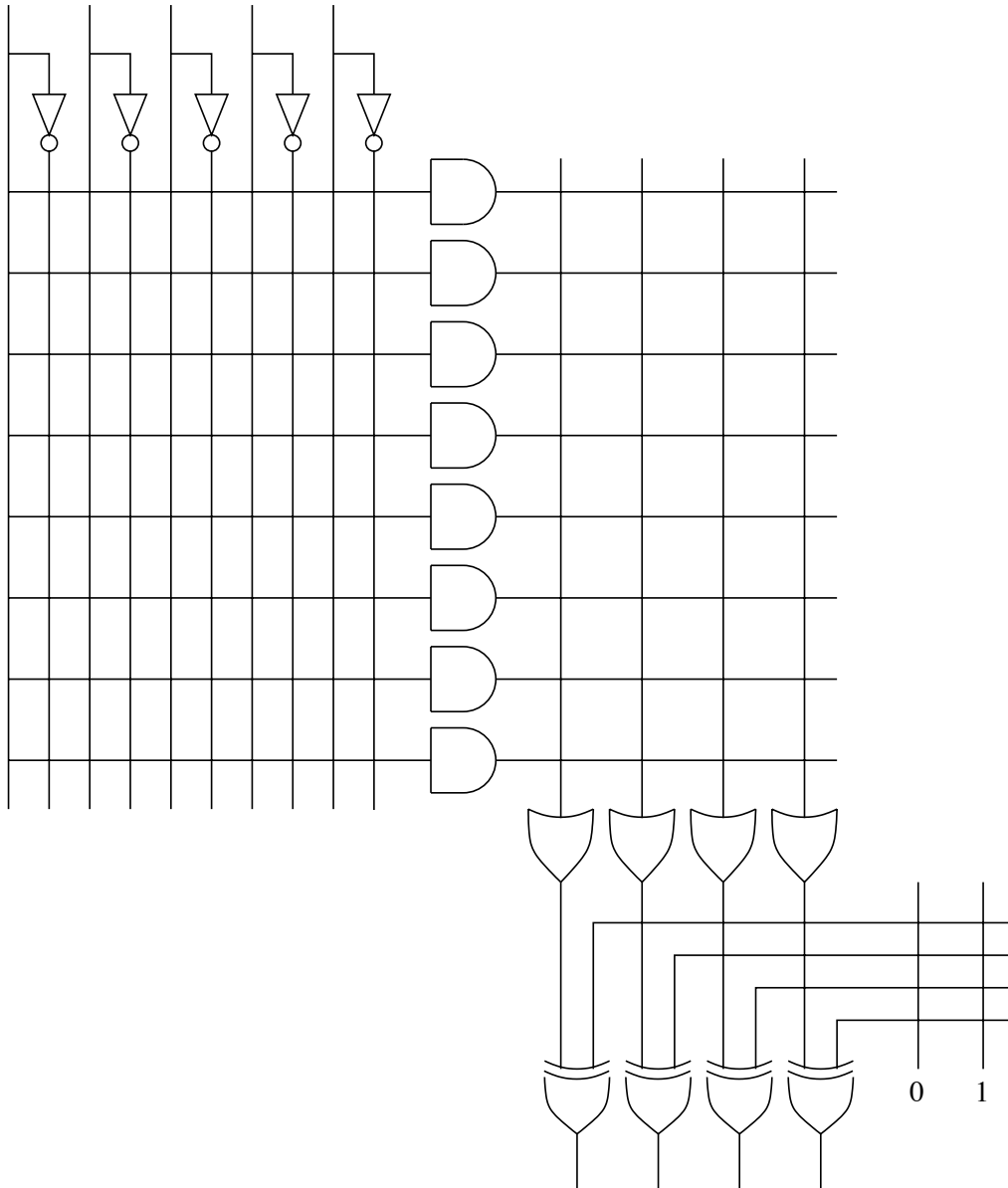
Programmable Logic Arrays (PLAs) are widely used to implement Boolean functions. Fundamentally, each PLA is characterized by its number of inputs, its number of rows (used to represent subcubes of inputs), and its number of outputs. The efficiency of a PLA implementation is usually determined by the number of **rows** (also known as *words* in PLAs) that need to be used.

Due to the characteristics of PLAs, two typical *optimization techniques* can be applied to improve the efficiency of an implementation. Firstly, because each PLA has an **XOR** array that allows us to conditionally complement the output values, one can program the AND and OR arrays to implement the complement of a function, if the function has only a few 0's, particularly if they are contiguously positioned. Secondly, if multiple functions are to be implemented simultaneously, one can exploit the possibility of sharing common row(s) among several functions to reduce the total number of words that need to be used.

(Part A) You are given a Boolean function $F(w, x, y, z)$ shown in the following Karnaugh map:

$wx \backslash yz$	00	01	11	10
00	0	0	0	0
01	1	0	1	1
11	1	1	1	1
10	0	0	0	0

Please implement the function using an AND-OR PLA in the **most economical** way, i.e., using the minimal number of rows. Denote the variables and output and mark the connections in the figure provided below. Note that you might not need to use all the wires in the figure. *For partial credit, you may want to generate the minimal cover expressions of the function.*

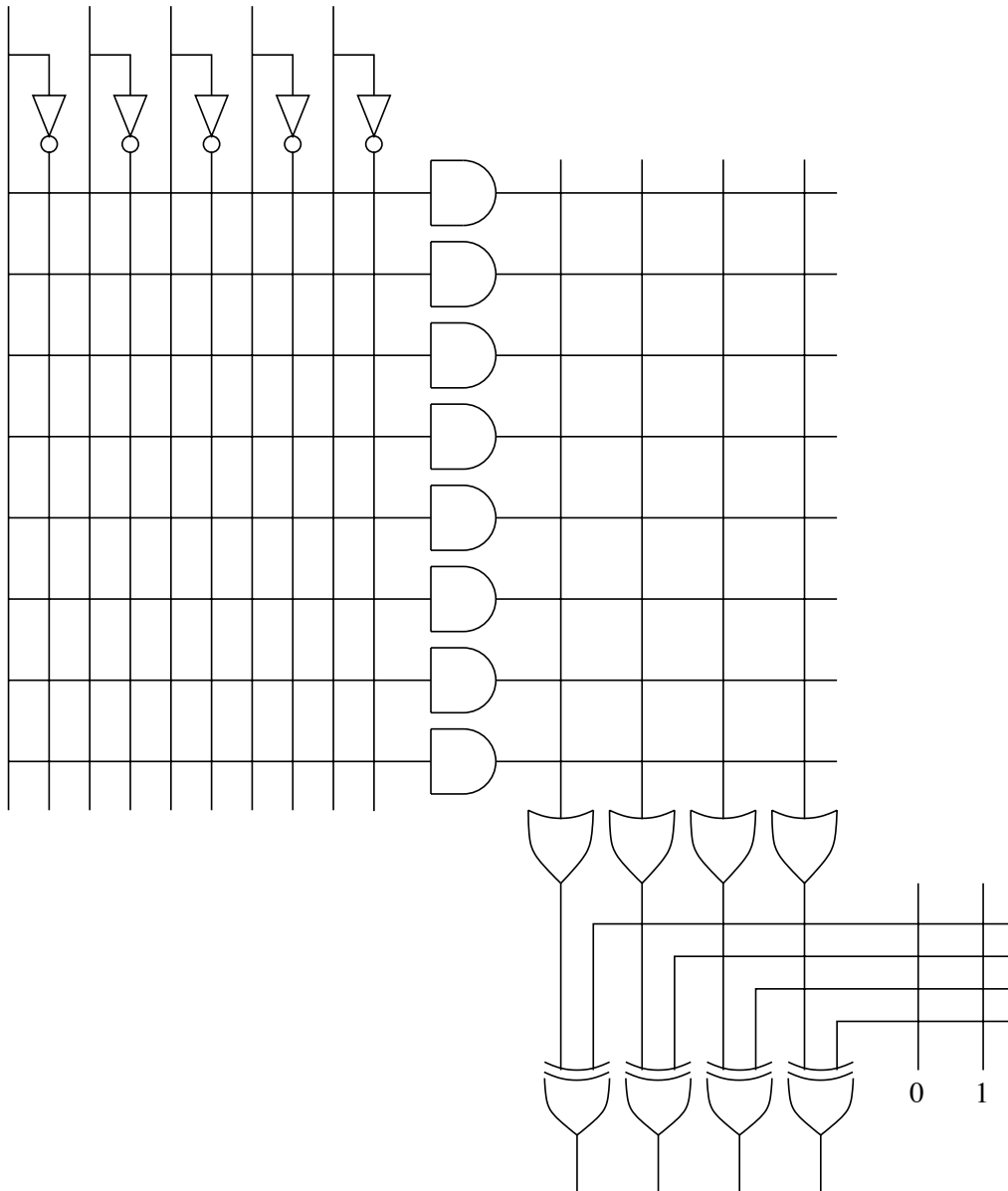


$$F(x, y, z, w) =$$

(Part B) In addition to the function F shown in (Part A), you are given an extra Boolean function $G(w, x, y, z)$ shown in the following Karnaugh map:

$wx \backslash yz$	00	01	11	10
00	0	1	1	0
01	0	0	1	0
11	1	1	1	0
10	0	1	1	0

Please implement **both** F **and** G using an AND-OR PLA in the **most economical** way, i.e., with minimal number of rows. Denote the variables and output and mark the connections in the figure provided below. Note that you might not need to use all the wires in the figure.



$$F(x, y, z, w) =$$

$$G(x, y, z, w) =$$

3 (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: