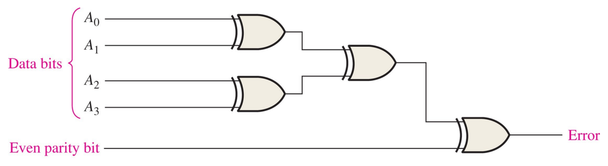
Study Guide for Midterm 2

Dr. Jordan Hanson - Whittier College Dept. of Physics and Astronomy

April 28, 2020

1 Chapter 5 - Combinatorial Logic Analysis



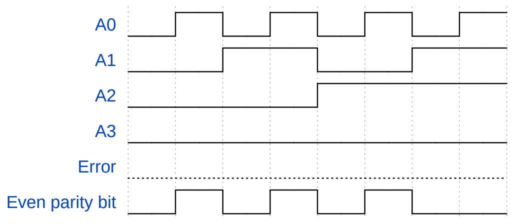


Figure 1: (Left) A circuit involving several XOR gates. (Right) An example timing diagram for the circuit at left.

1. Consider Fig. 1. (a) Generate the truth table for the output of the first three XOR gates, for all possible values of the 4-bit binar y input. What does this output represent numerically about the input $\vec{A}$? (b) Fill in the Error bit stream in Fig. 1.

A_3	A_2	A_1	A_0	input to parity XOR
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

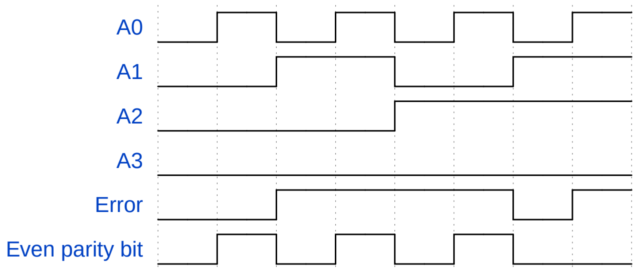
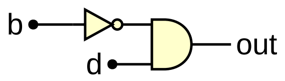


Table 1: (Left) The output of the first three XOR gates yields the *parity*: true if the input contains an odd number of 1's. (Right) Fig. 1 with the error line completed.

AB \ CD	00	01	11	10
00	0	1	1	0
01	0	0	0	0
11	0	0	0	0
10	0	1	1	0



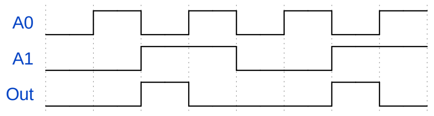


Table 2: The Karnaugh map for the circuit in Fig. 2.

2. In Fig. 2, (a) simplify the circuit via the Karnaugh map. (b) For the resulting domain-2 logic function, connect $A0$ and $A1$ from Fig. 1 (right) as inputs and produce the timing diagram assuming the same even parity bit stream.

   The domain-4 Karnaugh map is shown in Tab. 2 (left), and the simplified circuit in Tab. 2 (right). The gate structure only has two inputs because there are 4 adjacent cells in Tab. 2 (left). The parity bit stream is irrelevant because there are only two inputs. The correct output for the simplified gate is shown in Tab. 2 (bottom).

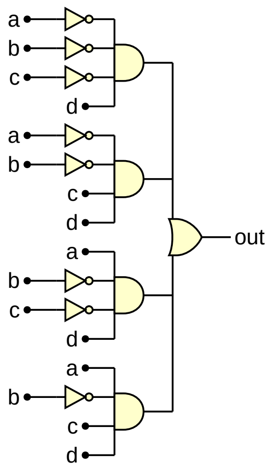


Figure 2: A domain-4 logic function.

2 Chapter 6 - Functions of Combinational Logic

1. Consider Fig. 3. (a) Design a circuit below that adds two numbers in binary corresponding to outputs from two separate encoders. (b) Demonstrate how your design would add 1 and 9. (c) Add a separate input line with XNOR gates that switches the system to subtraction in 2's complement form.

   (a) See Fig. 4. (b) If the 1 key is pressed on the upper pad, then 1 is encoded to 0001 in the encoder. If the 9 key is pressed on the lower pad, the encoder converts to 1001. If the subtract key is not pressed, then it remains HIGH and the outputs of the XNOR gates follow the outputs of the bottom encoder. The outputs of the two encoders are fed to the four pairs of inputs to the 8-bit adder with carry-in tied to ground. The result is the output of the 8-bit addition of 1001 and 0001, or 1010. (c) If the subtract key is pressed, the active-LOW signal pulls down one input on each of the XNOR gates, making the XNOR outputs the complements of the bottom encoder. Instead of connecting C_{in} to ground, we would connect it to the complement of the subtract signal (via an inverter).

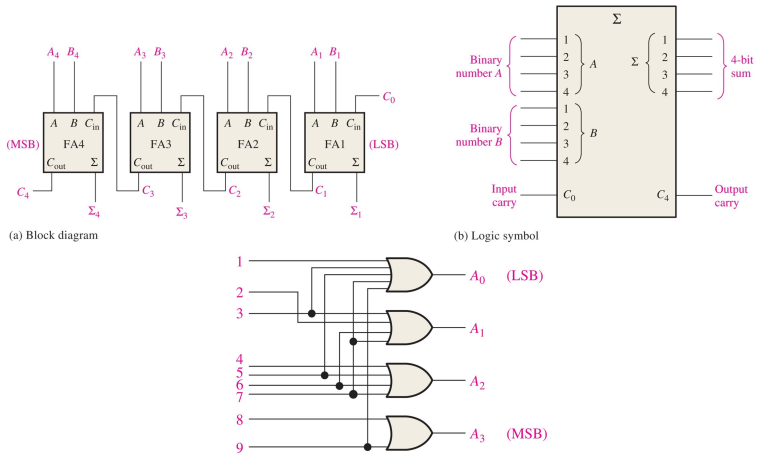


Figure 3: A 4-bit ripple carry adder (with logic symbol), and a 9-digit (zero ommitted) encoder.

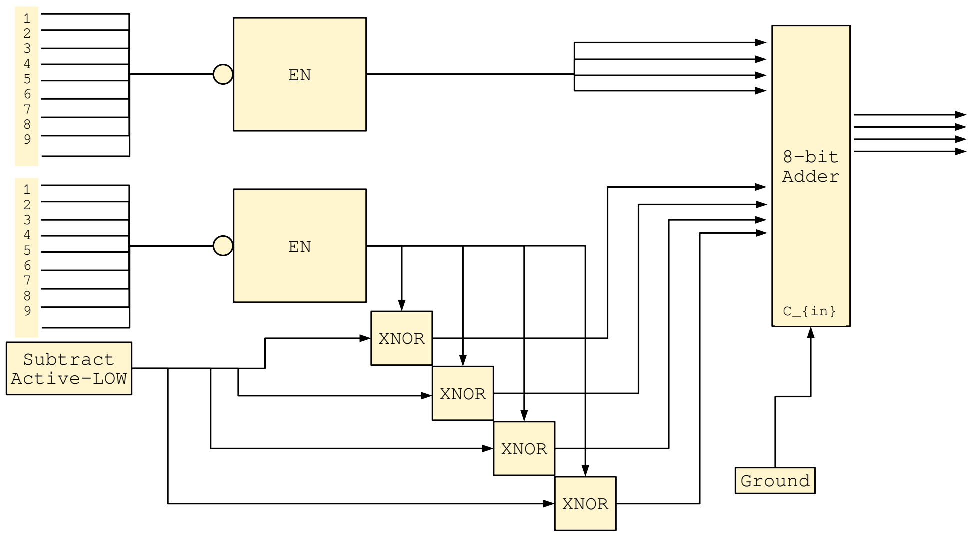


Figure 4: The system adds two active-LOW key presses 1-9 (0 is default) by passing them to two BCD encoders. The active-LOW negative sign switch converts the second digit to the 1's complement via the XNOR gate array.

2. Using the basic logic in Fig. 6, plus inverters, (a) create a circuit that compares the *magnitude* of two 2-bit binary numbers. Assume that one or both of the numbers is a negative one, and that if a number is negative it is in *1's complement form.*

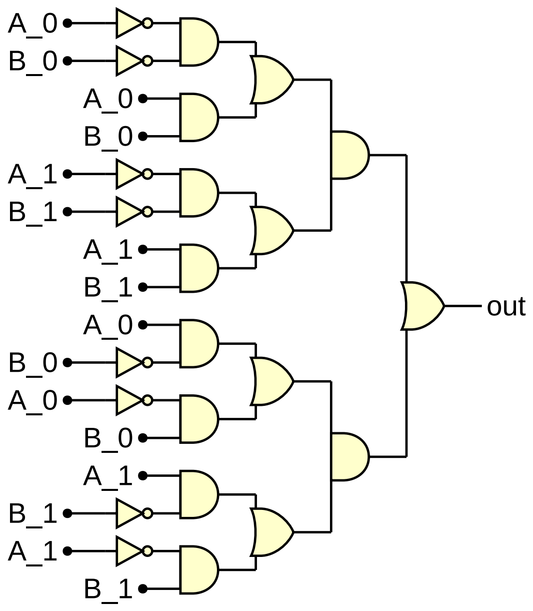


Figure 5: A comparator that compares the magnitude of a pair of 2-bit binary numbers. There are two top-level AND gates. The upper AND gate has two XNOR gates comparing A and B. The lower AND gate has two XNOR gates comparing $\bar{A}$ and B.

3. For the first four binary numbers (0, 1, 2, 3) show that the logic in Fig. 6 (right) converts binary to gray code.

See Tab. 3.

Input	Output
000	000
001	001
010	011
011	010

Table 3: Verification that Fig. 6 (right) contains the gray-code converter from binary.

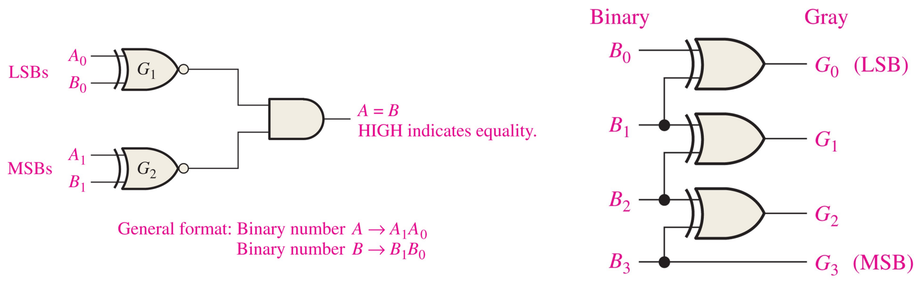
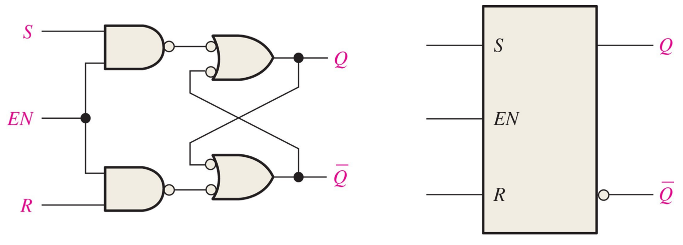


Figure 6: (Left) This is how a 2-bit comparator works. (Right) A binary to gray code converter.

3 Chapter 7 - Latches, Flip-flops, and Timers



(a) Logic diagram (b) Logic symbol

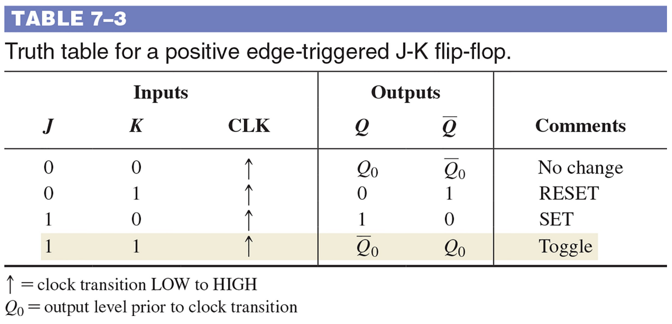


TABLE 7–3

Truth table for a positive edge-triggered J-K flip-flop.

Inputs			Outputs		
J	***K***	**CLK**	***Q***	$\overline{Q}$	**Comments**
0	0	↑	Q_0	$\overline{Q}_0$	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q}_0$	Q_0	Toggle

↑ = clock transition LOW to HIGH

Q_0 = output level prior to clock transition

Figure 7: (Top) The gate-enabled SR latch. (Bottom) The truth table for the JK flip-flop.

1. Using the logic for the gate-enabled SR latch in Fig. 7 (top), develop a system that (a) activates a green LED when the circuit is enabled and in SET mode, (b) a yellow LED switch when the circuit is enabled and in RESET mode, and (c) a red LED when the circuit is not enabled.

   See Fig. 8 (left). The red LED is pulled low with a LOW on the enable input. When Q is SET, the LED is on, and when the complement of Q is RESET, the yellow LED is on. Q and its complement are never high simultaneously.

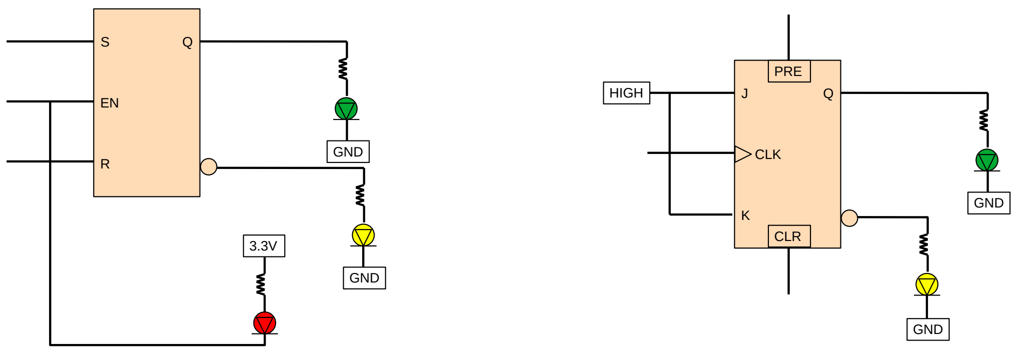


Figure 8: (Left) Remember that the gate-enabled SR latch shown in Fig. 7 is an active-HIGH design. (Right) The posedge-triggered JK flip-flop with preset and clear functions toggling two LEDs.

2. The truth table for a positive clock-edge JK flip-flop is shown in Fig. 7 (bottom). Use such a device to design a circuit below that (a) powers two LEDs, and toggles between two states: one in which one LED is activated, and another in which the opposite LED is activated. The lights should switch on positive edges of the clock. (b) Add logic that holds LED1 on, and LED2 off, regardless of the outputs of the JK flip-flop (the equivalent of *preset* and *clear* functions of JK flip-flops).

   See Fig. 8 (right). The key is that the J and K inputs are tied together so that the LEDs are always toggled on posedge. Preset and clear active-LOW would hold one or the other LED on regardless of J and K.