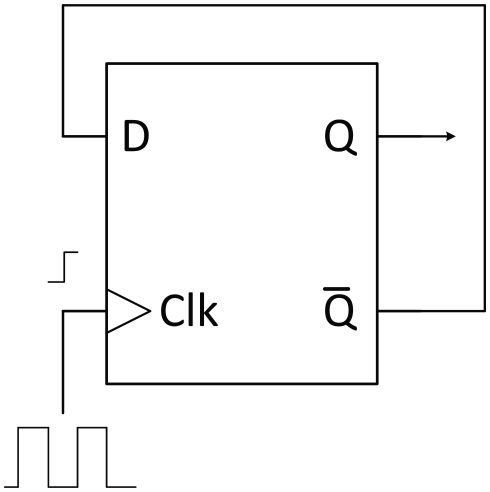

Q1.

Consider the edge-triggered D-type flip flop connected as shown.



(a) What is the output from Q like over several pulses of the clock signal?

> **Solution:** Since $D = \overline{Q}$, each time the clock signal changes $0 \rightarrow 1$, Q changes to the inverse of the value it contained prior to the clock edge. Thus Q will look like the sequence 0 1 0 1 0 1 0 . . .

(b) What would happen if the Q output is fed back to the D input, instead of the $\overline{Q}$ output?

> **Solution:** Since $D = Q$, in this case the output will never change.

(c) If the frequency of the clock signal is 1 Hz, what is the period of the output from Q?

> **Solution:** Since Q changes once for each rising $(0 \rightarrow 1)$ edge, it takes two cycles of the clock for Q to repeat. Thus the frequency of Q is 0.5 Hz and the period is 2 seconds.

Q2.

An encoder produces a binary code depending on which of its inputs is high, as shown in the following truth table:

D_0	D_1	D_2	D_3	A_1	A_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

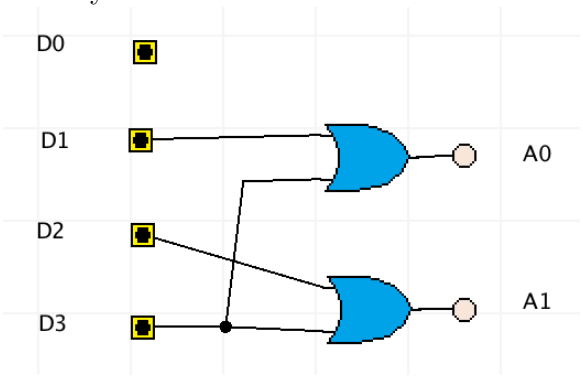
(a) Design a logic circuit that implements this truth table.

Solution: By inspecting the truth table we see:

$$A_1 = D_2 + D_3$$

$$A_0 = D_1 + D_3$$

These expressions are already simplified, so we can construct logic circuits directly from them:



(b) Modify your circuit to produce a *priority* encoder, that is, one that ignores higher D_N if a lower D_N is true. For this example you can safely ignore the case where no inputs are true.

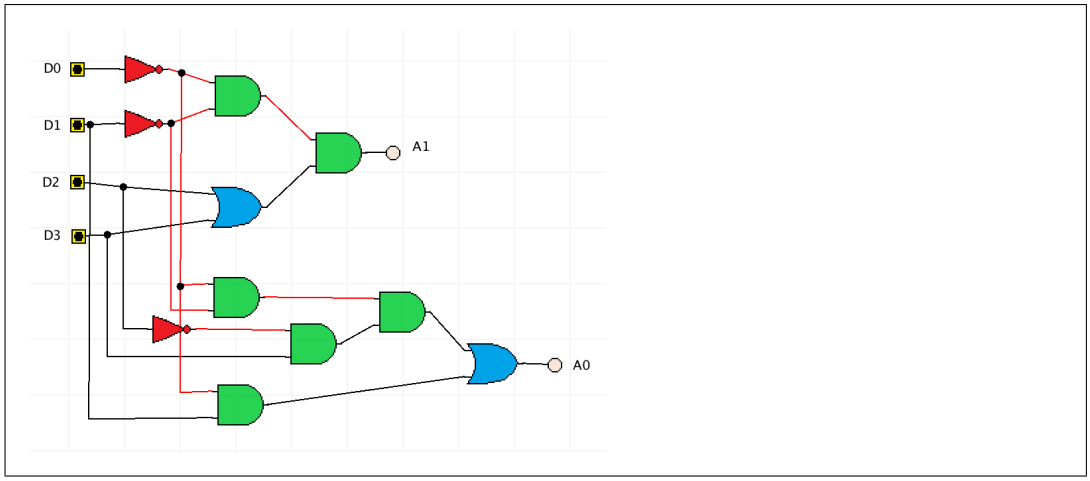
Solution: Currently A_1 goes high if D_2 or D_3 are true; in a priority encoder this will only happen if D_0 and D_1 are not true. So our expression for A_1 becomes

$$A_1 = \overline{D_0} \cdot \overline{D_1}(D_2 + D_3)$$

Similarly, A_0 currently goes high when D_1 or D_3 are asserted; in a priority encoder this should only happen when no other inputs with higher priority (i.e. D_0 for D_1, D_0, D_1 and D_2 for D_3) are true.

$$A_0 = \overline{D_0} \cdot D_1 + \overline{D_0} \cdot \overline{D_1} \cdot \overline{D_2} \cdot D_3$$

Creating logic circuits from these expressions, we get:


D0
D1
D2
D3
A1
A0

Q3.

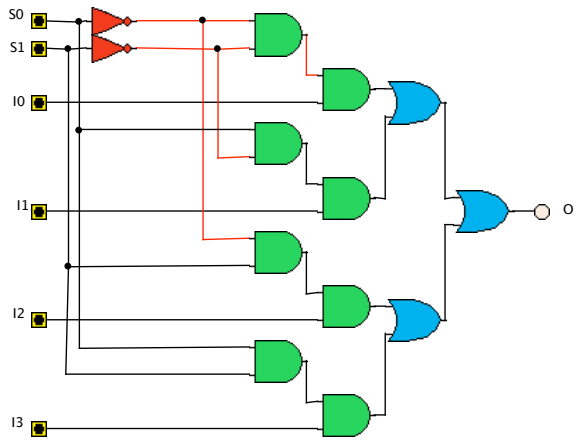
Recall that a multiplexer is a combinational circuit that provides a path from one of a selection of inputs to a single output (e.g., `2:1`). A *demultiplexer* provides a path from a single input to one of a selection of outputs (e.g., `1:8`).

(a) A 4:1 multiplexer requires two select lines. Write down the truth table for a 4:1 multiplexer in terms of the inputs and select lines. Based on your truth table, derive a Boolean expression for the multiplexer and draw a circuit diagram that implements this expression.

Solution: Assuming that the select lines are S_1 and S_0, and the inputs are I_0, I_1, I_2, and I_3, the output would be:

S_0	S_1	O
0	0	I_0
1	0	I_1
0	1	I_2
1	1	1_3

$$O = \overline{S}_1\overline{S}_0 I_0 + \overline{S}_1 S_0 I_1 + S_1\overline{S}_0 I_2 + S_1 S_0 I_3$$



(b) Draw a truth table for a 1:2 *demultiplexer* (note: a single select line is required).

Solution: The demultiplexer has a single input, I. Since it's a 1:2 demux there is one select line, S, and two output lines, O_0 and O_1. The truth table could be written as

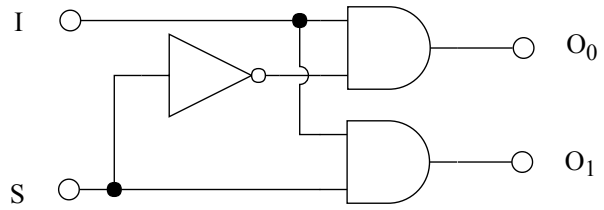
Inputs		Outputs	
I	S	O_0	O_1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

(c) With the help of the truth table, design a logic circuit to implement a 1:2 demultiplexer.

Solution: From the truth table, the Boolean expressions for the two outputs are:

$$\begin{aligned} O_0 &= I\overline{S} \\ O_1 &= IS \end{aligned}$$

Therefore the logic circuit looks like the following:



Q4.

The battery charge controller in an electric car outputs the current charge level as a 3-bit binary number (Full charge = 7, No charge = 0). Design a combinational logic circuit that will turn on a "low charge" indicator when the charge level number is less than 40%.

Solution: The low charge indicator will have a 3-bit input: B_0, B_1, and B_2.

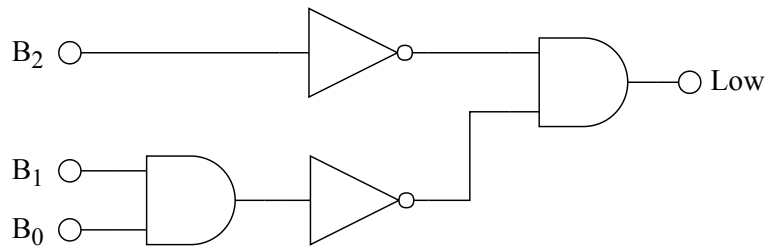
If full charge is equal to 7, then 40% charge is equivalent to 2.8. Although 2.8 isn't directly representable in the 3-bit code, codes for 0 to 2 represent levels less than 40% (codes 3 to 7 are above 40%). So the truth table for the indicator circuit should show a 1 for all input codes 0 to 2:

B_2	B_1	B_0	*Low*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

From the truth table, the Boolean expression for *Low* is (first in sum-of-minterms form, then simplified by Boolean algebra):

$$\begin{aligned} Low &= \overline{B_2}\overline{B_1}\overline{B_0} + \overline{B_2}\overline{B_1}B_0 + \overline{B_2}B_1\overline{B_0} \\ &= \overline{B_2}\,\overline{(B_1B_0)} \end{aligned}$$

Therefore the logic circuit looks like the following:



Q5.

You're working on a digital signal processing system that includes the 4-bit shift register shown in Fig. 1.

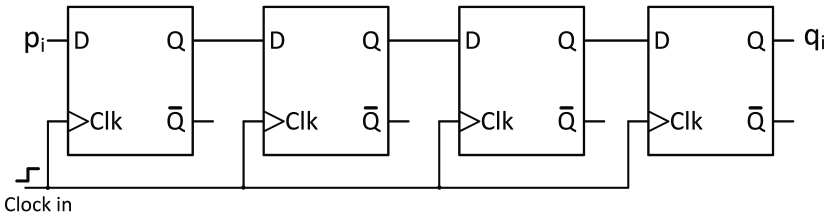


Figure 1: 4-bit shift register

(a) Is this shift register circuit *synchronous*?

> **Solution:** Yes, the register is synchronous. All of the flip-flops are clocked from a common clock signal.

(b) If the clock signal is a square wave of frequency 1 kHz, what is the delay (in milliseconds) created by the shift register between the register input, p_i, and the register output, q_i?

> **Solution:** If the clock signal has a 1 kHz frequency, then the period of the clock (time from one rising edge to the next) is
>
> $$\begin{aligned} Period &= \frac{1}{frequency} \\ &= \frac{1}{1000} \\ &= 10^{-3}\ seconds, \end{aligned}$$
>
> or 1 millisecond. The input passes through 4 flip-flops before it will appear on the output, so the register must be clocked 4 times to move an input value to the output. Therefore the delay between the register input and output is $Delay = 4 \times Period = 4$ milliseconds.

Q6.

A *Pseudo Random Binary Sequence* (PRBS) generator is a special form of shift register, useful for applications such as encryption. For this circuit, a combination of the flip-flop outputs is used to generate the input to the first stage. A 3-bit example is shown in Fig. 2.

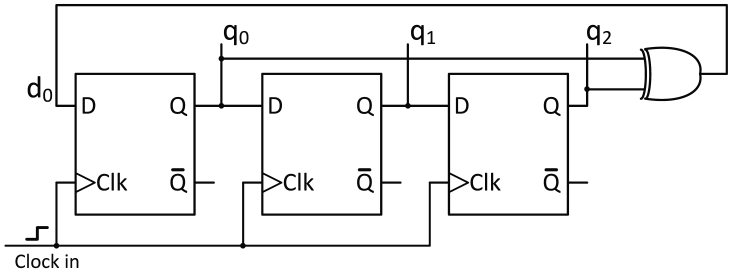


Figure 2: PRBS generator

(a) Write a Boolean expression for the input d_0.

Solution: Starting from d_0, and tracing back along the line connected to that input, we can see that d_0 is generated by the output of an XOR gate. The inputs of the XOR gate are q_0 and q_2. Therefore the expression for d_0 is

$$d_0 = q_0 \oplus q_2$$

(Note: could also be written as $d_0 = q_0$ xor q_2)

(b) Draw up a table with columns for the current output of the PRBS (q_2 q_1 q_0), the current input (d_0), and the next output that will be generated (q_2' q_1' q_0') when the clock has a rising edge. Begin with q_2 q_1 q_0 = 0 0 1 in the first row, and generate 10 rows that show the sequence of outputs from the generator.

Solution: We can find d_0 using the expression from the previous part. The next outputs, q_2' q_1' q_0', are the result of the contents of each flip-flop being shifted to the right and d_0 being shifted into the first flip-flop. The value of q_2' q_1' q_0' becomes the current output in the next row of the table:

Current			Input	Next		
q_2	q_1	q_0	d_0	q_2'	q_1'	q_0'
0	0	1	1	0	1	1
0	1	1	1	1	1	1
1	1	1	0	1	1	0
1	1	0	1	1	0	1
1	0	1	0	0	1	0
0	1	0	0	1	0	0
1	0	0	1	0	0	1
0	0	1	1	0	1	1
0	1	1	1	1	1	1
1	1	1	0	1	1	0

(c) How many different codes are in the sequence produced by the PRBS generator?

Solution: By looking at the table generated in the previous part, we can see that the PRBS generator produces 7 different codes before it begins to repeat its output.

Q7.

Your team lead asks you to analyse the 2-bit counter shown in Fig. 3, which produces a sequence of numbers that differs slightly from a normal binary counting sequence.

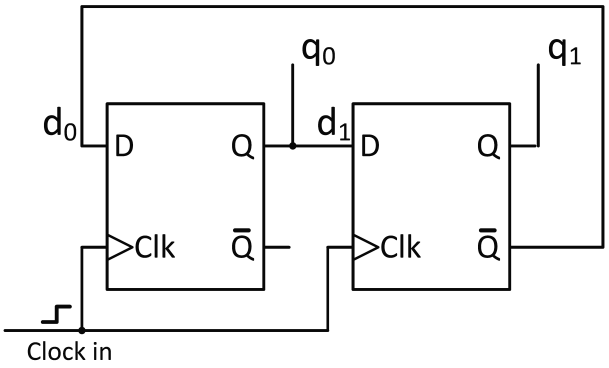


Figure 3: 2-bit Gray counter

(a) Write Boolean expressions for the flip flop inputs, d_0 and d_1.

Solution: We can read these directly off of the diagram by looking at what the input lines are connected to:

$$\begin{aligned} d_0 &= \overline{q_1} \\ d_1 &= q_0 \end{aligned}$$

(b) Draw up a table with columns for the current output of the counter (q_1 q_0), the current input (d_1 d_0), and the next output that will be generated (q_1' q_0') on a clock rising edge. Begin with q_1 $q_0 = 0$ 0 as the current output in the first row, and generate 6 rows that show the sequence of outputs from the counter.

Solution: We can find d_1 d_0 using the expressions from the previous part. The next outputs, q_1' q_0', come from the inputs. We can then take the value of q_1' q_0' and use it as the current output in the next row of the table:

Current		Inputs		Next	
q_1	q_0	d_1	d_0	q_1'	q_0'
0	0	0	1	0	1
0	1	1	1	1	1
1	1	1	0	1	0
1	0	0	0	0	0
0	0	0	1	0	1
0	1	1	1	1	1

(c) The output sequence is known as a *Gray code*. Looking at the sequence of outputs, what do you think distinguishes the Gray code from the normal binary counting sequence?

Solution: The Gray code is designed to have only a single bit change in value from one code to the next (i.e., only a single bit changes per clock event).

(d) Is this counter circuit *synchronous*?

Solution: Yes, the counter is synchronous. All of the flip-flops are clocked from a common clock signal.

Q8.

The shift register and counter, at least when they are designed to be synchronous, are really special forms of finite state machine. Recall that for the *Moore* FSM, the outputs at any time are entirely determined by the state; for the *Mealy* FSM, the outputs are determined by the state and the inputs.

(a) Are the register circuits in Fig. 1 and Fig. 2 Moore or Mealy?

> **Solution:** Both register circuits are *Moore* machines. Their outputs depend only on the current state of the flip-flops.

(b) Is the counter circuit in Fig. 3 Moore or Mealy?

> **Solution:** The counter circuit is a *Moore* machine. Its outputs depend only on the current state of the flip-flops.

END OF QUESTIONS