Gate Level and Word Level Designs

0.1 Sequential circuits

The sequential circuits contributes a delay in the signal to pass through these circuits, called, *propagation delay*, we denote it by (Δ). The figure 1 shows an AND gate with propagation delay.

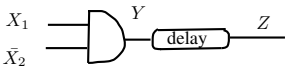

Figure 1: AND gate shown with propagation delay.

The expression for delayed output can be represented as:

$$z(t+\Delta) = y(t) = x_1(t)\bar{x_2}(t) \tag{1}$$

The symbols y is internal state variable, not accessible at the output of the gate. We note that, though the AND gate is a combinational circuit, but due to the finite propagation delay, it behaves like a sequential.

0.1.1 S-R Flipflop

Consider the flip-flop shown in figure 2. The circuit shows two NOR gates in a typical fashion; the arrangement is called S-R flip-flop (for Set -Reset).

The working of S-R is as follows: for input *SR=00*, the output is, $y_1 = 0$, $y_2 = 1$, provided that it was previously in the state $y_1 = 0$, $y_2 = 1$. Alternatively, it would now become, $y_1 = 1$, $y_2 = 0$, if it was earlier at $y_1 = 1$, $y_2 = 0$. Therefore, for *SR=00* input, the output y_1y_2 remains unchanged.

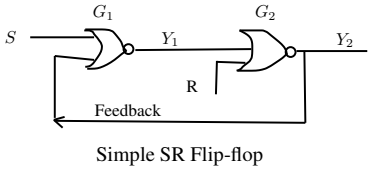

Figure 2: S-R Flip-flop.

For *SR=10*, there is $y_1 = 0 = \bar{y_2}$. For *SR = 01*, there is $y_2 = 0 = \bar{y_1}$. We note that, except for *SR = 00*, the outputs y_1 and y_2 are well defined. For *SR=00*, the output of the NOR gates are defined by their previous states.

Race Condition

Let us assume that $SR = 11$. Thus, after Δ time (i.e., propagation delay), $y_1 = y_2 = 0$. After this has happened, let us assume that we apply an input, *SR* as 00 and retain it indefinitely. We note that, after delay of Δ, the input propagates to output and y_1, y_2 again becomes 11, respectively. These outputs have been connected back to input of gates G_1 and G_2. Hence, after Δ time the input propagates to output, and y_1, y_2 again becomes 00. Thus, while the input is at 00, output *Oscillates* between 00 and 11. For this to happens, it is necessary that propagation delays of the two NOR gates, i.e., Δ_1, Δ_2, are equal. This is a serious condition for SR flip-flop, since it does not depend on inputs, neither it is in a stable state. Hence, there should not be an input, which causes this oscillation. Hence, $SR = 11$ input is called *forbidden state* of the SR flip-flop, as this input is disallowed.

Example 0.1.1 *Explain the behavior of SR for* $\Delta_1 \neq \Delta_2$ *?*

Hint: It will generate a square wave-form of unequal size ON and OFF periods. □

Race conditions are eliminated by timing signals (clock). The triggering / state transition in digital circuits are aligned with the clock transition, hence it does not take place at any other time during the clock. Triggering takes place on rising or trailing edge of clock pulse. Before this, the signals are supposed to be stabilized.

When *Sequential circuits* are combined with clock signal, these circuits are called *Synchronous circuits*. Circuits not timed by clock are *asynchronous circuits*; as seen earlier they are Prone to race conditions.

 Copyrighted Material.

Figure 3 shows a combinational circuit, along with clock and memory element. The previous outputs fed to the input, but not immediately, as they would otherwise again change the output. They are stored (saved) in memory and applied at the input in the next clock cycle. Assuming that combinational circuit **C** is computing a function f_1, the memory unit is equal to function f_2, and only part of the output z, i.e., z' is input to memory. Hence, we can represent the following functional relationship:

$$z(t + \Delta_1) = f_1(x(t), y(t)) \tag{2}$$

and

$$y(t + 2\Delta) = f_2(z'(t)) \tag{3}$$

Here, Δ_1 is propagation delay through the Combinational circuit, and Δ_2 is this delay through the memory element. Since memory is sequence by clock, $\Delta_2 = 1$ clock cycle.

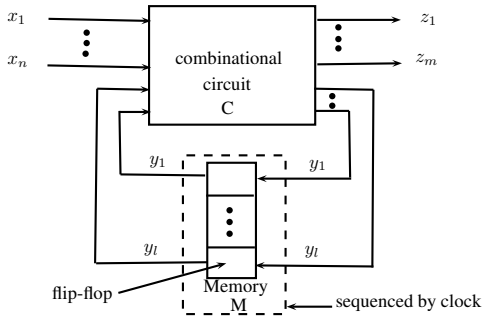

Figure 3: Combinational Circuit with memory element.

The clock signals are essential for timing of events, for example, transitions to occur through the gates. The figure 4 shows a typical clock.

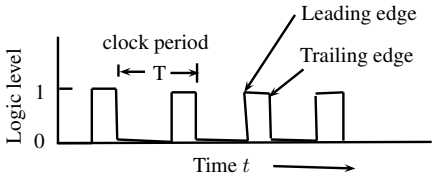

Figure 4: A typical clock signal.

The Clock active and inactive states are 1, 0. They also correspond to transition from 0-to-1 or 1-to-0. The period, during the time clock is inactive, must be at least as long as the worst case signal propagation delay through combinational circuit **C** (fig. 3), and active period must be long enough to allow the memory element M to make one complete transition. The behavior of sequential circuit is described by *State table*, to be discussed later.

Ponder over the following situation: What happens if clock period is long and signal passed through forward path feeds back to input during the current True period of clock?

0.1.2 Unclocked and Clocked SR flip-flop

The figure 5 shows the S-R flip-flop and its symbol.

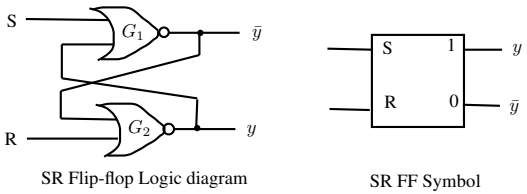

Figure 5: S-R Flip-Flop and its symbol.

For SR=00, there is no change in output $y, \bar{y}$. The combination, SR=11 is forbidden. The S(set)=1, R(reset)=0 *Sets* true output y to 1. The *SR=01 Resets* true output y to 0.

Let Δ is propagation delay of a NOR Gate, and τ, the propagation delay through two NOR gates is $\approx 2\Delta$. Let us assume that $S(t)$ is value of S at time t, hence output of gate G_1 appears at time $t+\Delta$, and that also reaches at the input of gate G_2 instantly. So we need to consider the value of R at time $t+\Delta$, i.e $R(t+\Delta)$, and output of gate G_2 is available at time $t+2\Delta$. Accordingly, we have y at $t+2\Delta$ as,

$$\begin{aligned} y(t+2\Delta) &= \overline{R(t+\Delta) + \overline{[S(t)+y(t+\Delta)]}} \\ &= \bar{R}(t+\Delta)[S(t)+y(t+\Delta)] \\ y(t+\tau) &\approx \bar{R}(t)[S(t)+y(t)] \end{aligned} \tag{4}$$

The above is after considering that the propagation time is much smaller compared to the clock time-period.

 Copyrighted Material.

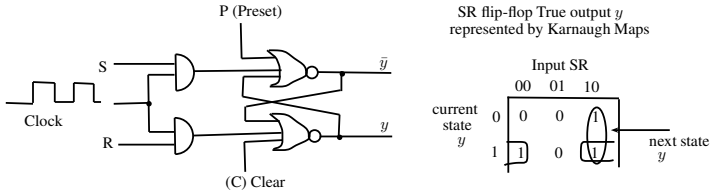

Figure 6: Clocked SR Flip-flop and Transition table.

To synchronize the transition of flip-flop, a clock pulse is gated to flip-flop along with the inputs S, R, such that the transition occurs in synchronization with the clock pulse. The figure 6 shows the clocked SR Flip-flop, and the transition-table for Clocked *SR* flip-flop. The P, C inputs are *Preset/Clear* to Set/ Clear the output, without the clock.

Let the current value of outputs at time t are: $y(t)$ and complemented output as $\bar{y}(t)$. We are interested to find out the outputs $y(t+1)$ and $\overline{y}(t+1)$, both after time $t+1$, i.e., after one clock period. The output y at time $t+1$, i.e $y(t+1)$ is computed using minimization using Karnaugh maps. Accordingly, we obtain from figure 6.

$$\begin{aligned} y(t+1) &= \bar{R}(t)S(t) + \bar{R}(t)y(t) \\ &= \bar{R}(t)[S(t) + y(t)] \end{aligned} \tag{5}$$

Here $y(t+1)$ is new state and $y(t)$ is previous state. We note that minimized expression for clocked SR flip-flop is same as that was obtained in equation 4. Note that propagation delay (Δ)is much smaller than the clock period, hence it is ignored. Further note that, we are still not allowing the SR to be 11 in the SR flip-flop.

The other types of FFs are derived from SR flip-flops. In these, no state change occurs without clock pulse. The clock pulse should be sufficiently long, to allow the inputs to stabilize.

0.2 JK and D Flip-flop

The figure 7 shows the $J-K$ and D flip-flops, along with the transition-table of JK flip-flop.

The JK flip-flop has been derived from *SR FF*. Using Karnaugh map on table 2 the minimized expression for true output is

$$y(t+1) = J(t)\bar{y}(t) + \bar{K}(t)y(t)) \tag{6}$$

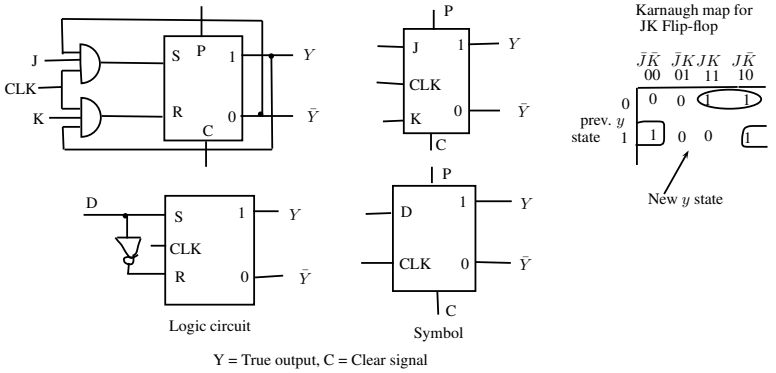

Figure 7: J-K and D Flop-flops.

For $JK = 00$, there is no change in output, 10 and 01 causes set and reset of fflip-flop, and $JK = 11$ toggles the Flip-flop (i.e., input 11 is not forbidden now). This is real advantage of JK flip-flop when compared with the SR flip-flop.

The D (*Delay*) flip-flop has a single input D. With $D = 1$, it sets, and with $D = 0$ it resets the flip-flop, i.e., $y(t + 1) = D(t)$.

0.2.1 D-Latch

Most commonly used flip-flop are D flip-flop (fig. 8), because they are useful for temporary storage of data. Latch means the previous data are latched in it. After that, if input changes, latched data is not effected. The latched value acts as a memory.

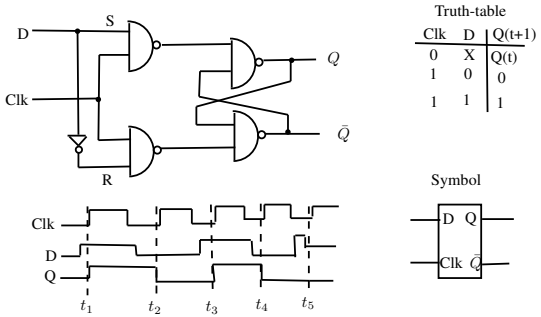

Figure 8: D flip-flop as a latch.

 Copyrighted Material.

The SR input will always be 01/10 only to reset or set the flip-flop, respectively. Where as for the D-flip-flop, during the positive half of clock, D is supposed to remain stable, else transition would occur at the output Q.

0.2.2 T-Flip-flop

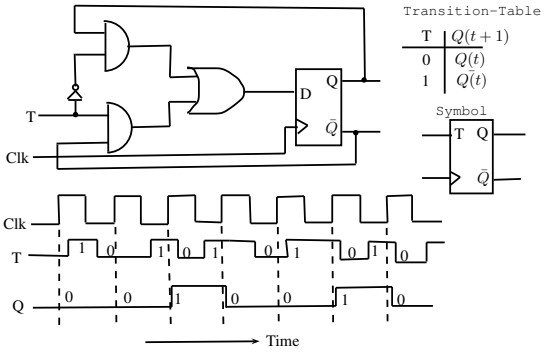

Figure 9: T-Flip-flop.

A T-FF changes its state every clock cycle(toggles) if its input T is equal to 1. The triggering (level change at the output) takes place at positive edge of clock. The figure 9 shows a T flip-flop, its toggling, which produces the output waveform of half the frequency of the clock frequency, if T = 1. The transition table of the T FF is also shown along with waveform. The toggling will not take place if T = 0. Because of this nature of the T FF, they are used as counters.

0.2.3 Serial addition using JK

The figure 10 shows the serial addition.

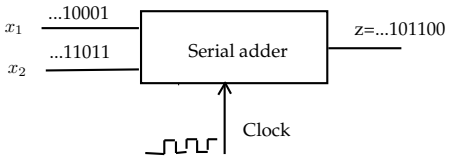

Figure 10: Serial Adder.

Two-bit addition takes place for each clock pulse. $z(t) = x(t) + y(t)$. It may produce a carry bit $c(t)$. Thus two states (S_0, S_1) are possible, for $c(t-1) = 0$, and $c(t-1) = 1$. Thus, $z(t) = x(t) + y(t) + c(t-1)$. The table1 demonstrates the transitions in serial adder. For example, with input as $x_1x_2 = 11$, and previous state s_1 (carry), the outputs are $s_1, 1$, i.e., sum is 1, and state s_1 indicates carry state as the next state.

Table 1: Serial adder Transition Table.

Previous state	input x_1x_2			
	00	01	10	11
s_0	$s_0, 0$	$s_0, 1$	$s_0, 1$	$s_1, 0$
s_1	$s_0, 1$	$s_1, 0$	$s_1, 0$	$s_1, 1$

Each entry has form $s(t+1), z(t)$. y is FF's state variable, $y = 0$ for s_0, and $y = 1$ for s_1. The logic circuit for Serial addition is shown in figure 11.

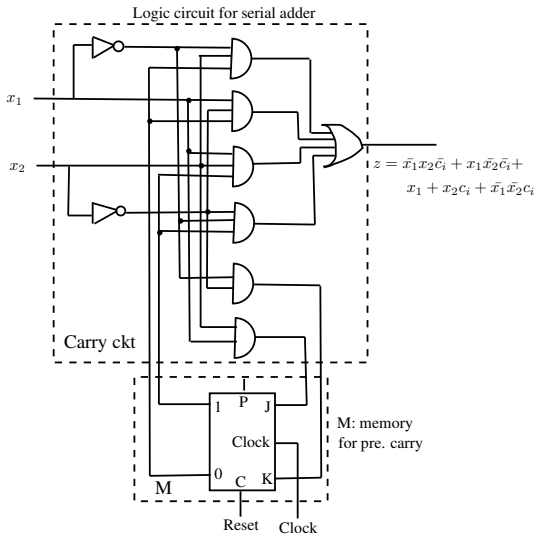

Figure 11: Serial Adder.

 Copyrighted Material.

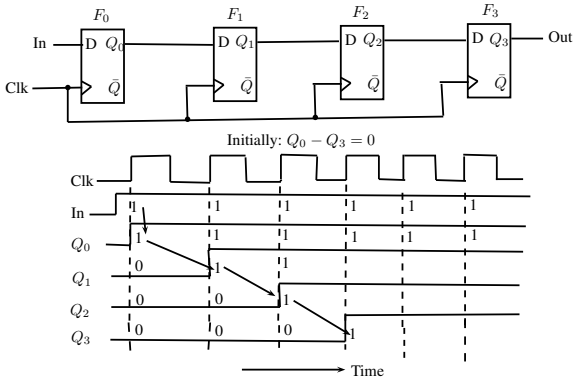

Figure 12: Shift-right register.

0.3 Registers and Shift Registers

Registers store machine words, they also Shift and rotate data one bit at a time. Figure 12 shows a serial shift-right register. The shift registers can be used to handle one-bit at a time for input, output, and processing, for example for testing if carry is generated or there exists a odd or even parity, etc.

The bits can be shifted in parallel also. The figure 13 shows the circuit of a Parallel Shift Register.

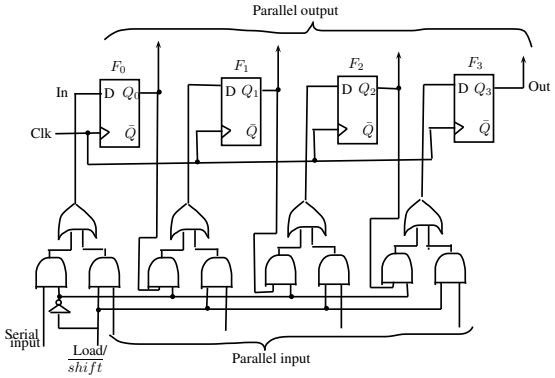

Figure 13: Parallel Shift and load register.

The registers can be used for building counters. At the machine lan-

guage, they are useful to count the number of clock pulses, the number of bits input or output, or number of bytes to be transfered from source to destination in memory or IO device. They can also be used to up-count from 0 to some higher value or down count from a set value to zero.

Counters also generate control and timing signals in figure 13. Q_0 frequency= $clk/2$, Q_1 frequency = $clk/4$, ..., called *ripple counter*. If all FF change state together it is called *synchronous counter*.

The figure 14 shows a 4-bit up-counter.

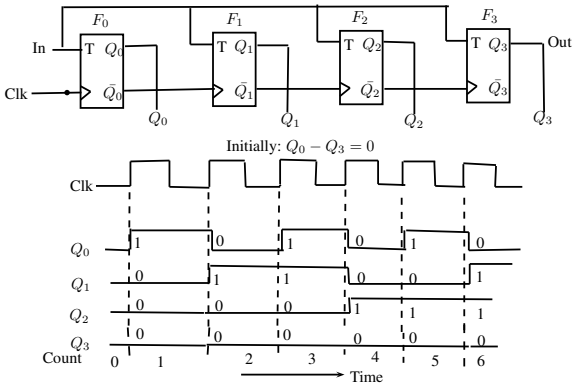

Figure 14: 4-bit Asynchronous up-counter.

0.4 Word Level Design

0.4.1 Word Gates

The usual gates have number of binary inputs as bits. The word-gates can be any time of the gates discussed earlier, but their inputs are words. A word is usually a group of bits, in order, e.g., 4-bits, 8-bits, or 16-bits are common. The operation performed is on the words together, e.g., one word ANDed with other or ORed with other. Figure 15 shows a NAND gate which performs AND operation of m-bit with other m-bits, and then inverts the result.

Let, X and Y are binary words of m-bits each.

$$X = (x_0, x_1, \ldots, x_{m-1}),$$
$$Y = (y_0, y_1, \ldots, y_{m-1})$$

 Copyrighted Material.

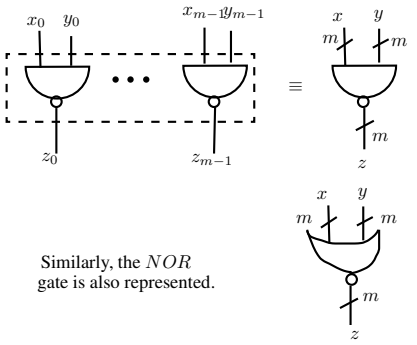

Figure 15: m-bit NAND and NOR word gates.

Assume that a word Z is a function of X, Y. Thus, $X = f(X, Y)$, if $z_i = f(x_i, y_i)$ for $i = 0, \ldots, m-1$. Therefore, $Z = \overline{XY} = (z_0, z_1, \ldots, z_{m-1})$. That is, $Z = (\overline{x_0 y_0}, \overline{x_1 y_1}, \ldots, \overline{x_{m-1} y_{m-1}})$.

0.4.2 Multiplexer

A K-input m-bit multiplexer has k input lines, each m-bit wide. The number of select lines p to select a unique input are given by expression $k = 2^p$, where total number of inputs are k.

Source selection is determined by an *encoded* pattern of p-bits. $z_i = \sum_{i=0}^{k-1} x_{ij} a_i e$, for $j = 0, 1, ..., m-1$. Figure 16 shows the components of m-bit multiplexer.

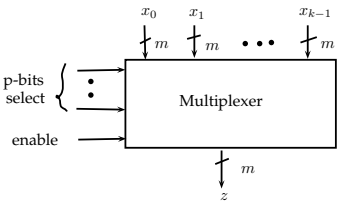

Figure 16: k-input m-bit multiplexer.

Figure 17 shows the operation of a 2-input 4-bit Multiplexer. It has two input words, each 4-bit long (x_0, x_1, x_2, x_3 and y_0, y_1, y_2, y_3). Select line selects either $x_0 x_1 x_2 x_3$ or $y_0 y_1 y_2 y_3$, and delivers at output as $z_0 z_1 z_2 z_3$.

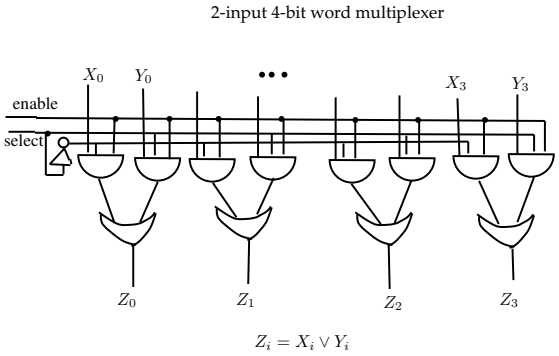

Figure 17: 2-1 multiplexer.

0.4.3 Decoders and Encoders

A decoder is a 1-out-of-2^n or a $1/2^n$ decoder is a combinational circuit with n input lines and 2^n output data lines. For input $x_0x_1 = 00, 01, 10, 11$ the output z_0z_1 z_2z_3 is $1000, 0100$, $0010, 0001$ respectively. Hence it is called 2-to-4 *decoder* (see fig. 18).

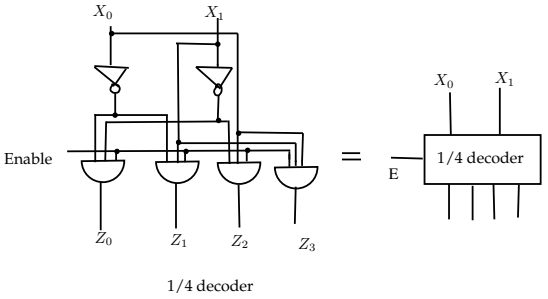

Figure 18: 2-4 decoder.

An encoder generates the address. It has inverse function of decoder. Has 2^k input lines and k output lines. Figure 19 and corresponding table 2 show a 4-to-2 *encoder*.

Simple Encoder: We discuss a restricted encoder. For the required outputs, the input should be exactly as shown. If input deviates from the given, the output is undecidable.

 Copyrighted Material.

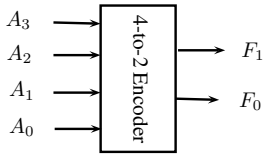

Figure 19: 4-2 Encoder.

Table 2: Truth-table for 4-2 encoder.

Inputs				Outputs	
A_3	A_2	A_1	A_0	F_1	F_0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Priority Encoder: It generates the output corresponding to the bit it does for simple encoder. X indicates *do not* care bits. Figure 20 shows a priority encoder, and table 3 is corresponding truth-table.

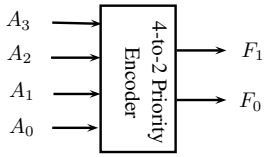

Figure 20: Priority-Encoder.

Exercises

1. Construct a SR Flip-flop using NAND gates only.
2. Construct a D Flip-flop using NOR gates only.
3. Construct a 4-bit decade counter using appropriate components.
4. Find a minimum cost implementation of the function $f(x_1, x_2, x_3, x_4)$,

Table 3: Priority encoder

Inputs				Outputs	
A_3	A_2	A_1	A_0	F_1	F_0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

where $f = 1$ if either one or two of the input variables have the logic value 1. Otherwise, $f = 0$.

5. Prove that associative rule does not apply to the $NAND$ operator.

6. How you will construct a down counter, i.e, from 1111 to 0000 ?

7. The address lines $A_0 \ldots A_{15}$, use encoder to access the corresponding physical address (T/F).

8. What is the Boolean expression for the output f for multiplexer shown in figure 21?

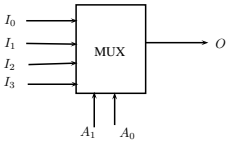

Figure 21: 4-Input multiplexer.

Ans. The expression for f for the 4-input multiplexer (fig. 21) is:

$$f = R(\bar{P}\bar{Q} + PQ) + \bar{R}(\bar{P}Q + P\bar{Q})$$

9. Consider the circuit shown in figure 22 involving positive edge trigged D-FF, as well the corresponding timing diagram. Let A_i represents the logic level on the line A in the i^{th} clock period.

   Let A' represents the complement of A. The correct output sequence Y over the clock period 1 through 5 is:
   A) $A_0A_1A_1'A_3A_4$ B) $A_1A_2A_2'A_3A_4$
   C) $A_0A_1A_2\prime A_3A_4$ D) $A_1A_2'A_3A_4A_5$

 Copyrighted Material.

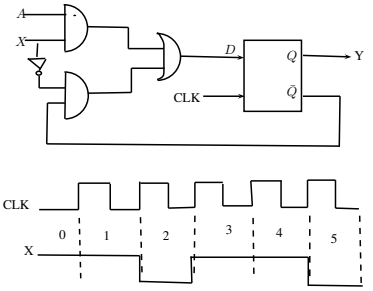

Figure 22: D-FF circuit

Ans. Correct Answer is (A). (note that the value A_i is effective in the clock period, that means - not before the start of the clock period).

10. Consider the circuit composed of XOR gates (figure 23) and non-inverting buffers. The non-inverting buffers have delays of $\delta_1 = 2ns$ and $\delta_2 = 4ns$. Both XOR and wires have zero delays. If the above waveform's 7-cycles are applied at input A, how many transition(s) (change of levels) occur at B during the interval $0 - 7ns$?

Ans. Total change of levels=6

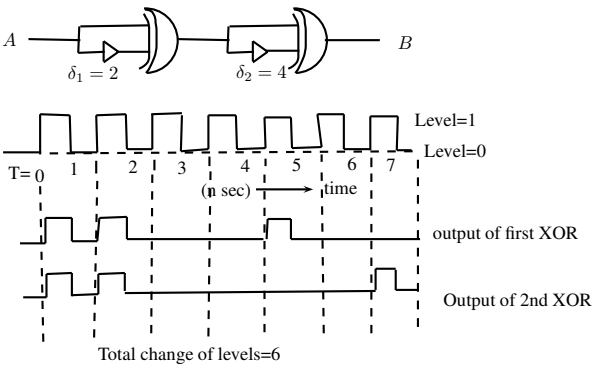

Figure 23: XOR Circuit and waveform.

11. Investigate the operation of following circuit. Assume an initial state of 0000(=output of four flip-flops). Trace the output (the Qs) as the clock ticks and determine the purpose of the circuit. (See fig. 24

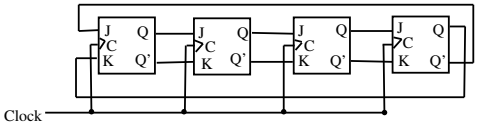

Figure 24: JK Flip-Flops.

12. Design a priority encoder for 4 interrupts, $I_0 \prec I_1 \prec I_2 \prec I_3$, where, $I_i \prec I_j$ indicates that I_i has lower priority that I_j. The lines O_0O_1 are to be set as per the following table. ('X' indicates don't care interrupts). (see table 4).

Table 4: Priority Encoder.

Input				Output	
I_3	I_2	I_1	I_0	O_1	O_0
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

13. Design a 4-to-1 bit multiplexer using minimum gate count. Use fundamental gates only.

14. Construct a minimized comparator to compare 2-bit words $w_1 = a_1a_2$ and $w_2 = b_1b_2$ producing 1-bit output, which is 1 when $w_1 > w_2$ else 0. Use any type of gates.

15. Compute the truth table (5) for the sequential circuit shown in figure 25:

 Copyrighted Material.

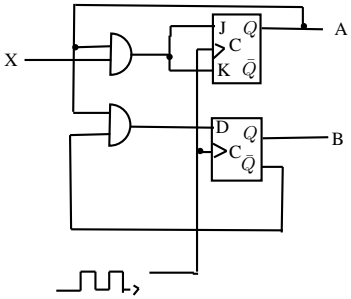

Figure 25: Sequential circuit.

Table 5: Truth table for sequential circuit

A	B	X	Next A	next B
0	0	0	?	?
.	.	.	.	.
1	1	1	?	?

16. Carry out the gate-level design of a synchronous modulo-4 down counter. It is to have four input lines: a count enable line COUNT; an up-down select line DOWN; a clock line CLOCK; and a RESET line CLEAR. There are two output lines $Z = (z_0, z_1)$. Use only NAND gates and D flip-flops in your circuit, and attempt to minimize the total number of logic gates.

17. Design a 8-bit priority encoder using two copies of 4-bit priority encoder. Additional gates may also be used if needed.

18. Design a comparator for two 10-bit numbers using only 4-bit comparators.

19. If the operands are in 2's complement representation, what operations out of A+B, A-B, A++, B++ (increment by 1) can be performed using the ALU diagram shown in figure 26?

20. Find the maximum clock frequency at which the counter shown in figure 27 can be operated.

Copyrighted Material.

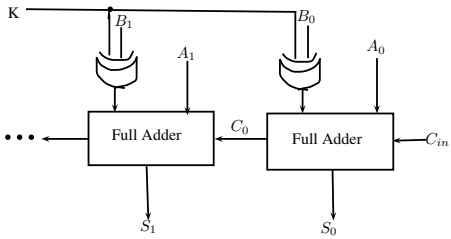

Figure 26: ALU

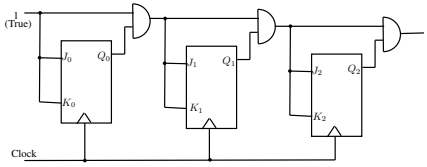

Figure 27: Counter

Assume that the propagation delay through each flip-flop and AND gate combined is 10 nsec. Also assume that setup time (time to stabilize signal) for the JK flip-flop inputs is negligible.

Copyrighted Material.