

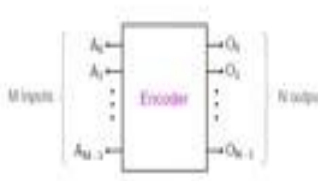
Basic Electronics
Prof. Mahesh Patil
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Lecture - 61
Introduction to sequential circuits

Welcome back to Basic Electronics. In this lecture we will look at another logical block the encoder. After that we will start with a different class of digital circuits namely sequential circuits. You will see in what way they are different from the circuits we have considered so far. We will take up the most basic block of sequential circuits that is the latch. Let us begin.

(Refer Slide Time: 00:47)

Encoders



- Only one input line is assumed to be active. The binary number corresponding to the active input line appears at the output pins.
- The N output lines can represent 2^N binary numbers, each corresponding to one of the M input lines, i.e., we can have $M = 2^N$. Some encoders have $M < 2^N$.
- As an example, for $N = 3$, we can have a maximum of $2^3 = 8$ input lines.

M. S. Patil IIT Bombay

Let us discuss encoders now. We have M inputs $A_0 A_1$ up to A_{M-1} , and N outputs $O_0 O_1$ up to O_{N-1} . And this is how it works. Only one input line is assumed to be active; that means, only one of these is active all others are inactive. And then the binary number corresponding to that active input line appears at the output pins over here.

Now, let us try to figure out the relationship between M and N here. And let us take an example say N is equal to 3; that means, we have 3 output pins say $O_0 O_1$ and O_2 . And how many different binary numbers can we make up with these 3 variables? The

answer is 2 raised to 3 that is 8 different binary numbers. And each of those binary numbers can correspond to 1 of these input lines.

So, therefore, we can have a maximum of 2 raised to 3 or 8 input lines. Only then we can have A 1 to 1 correspondence between the inputs and the outputs.

(Refer Slide Time: 02:24)

Encoders

8-to-3 encoder example

A7	A6	A5	A4	A3	A2	A1	A0	O2	O1	O0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

- Note that only one of the input lines is assumed to be active
- What if two input lines become simultaneously active?
 - There are "priority encoders" which assign a priority to each of the input lines.

M. S. Park @ SNU

We will take some examples and then this will become clear. So here is an 8-to-3 encoder example. We have 8 input lines and 3 output lines. And here is the truth table. And we will of course, assume that only one of these input lines is active. And these are all active high pins so therefore, active means 1. In this row for example, the active pin is A0.

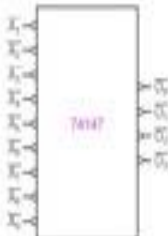
Now, the output that corresponds to this particular combination; that means, A0 being active and all others inactive is 0 0 0. For this combination where A1 is active and all others are inactive is 0 0 1 and so on. In the last row we have A7 as the active input line and corresponding to that we have an output of O2 equal to 1 O1 equal to 1 and O0 equal to 1 so this is an example of an 8-to-3 encoder.

In this example note that only one of the input lines is assumed to be active at a given time. And therefore, we have only one high pin here in any of the rows; that means, we will see only one 1 in any of the rows. The question that arises is what if 2 input lines become simultaneously active.

Now, to handle that situation we have something called priority encoders, in which a priority is assigned to each of the input lines. And then that is used to decide the output if 2 pins or more than 2 pins become simultaneously active.

(Refer Slide Time: 04:30)

74147 decimal-to-BCD priority encoder



\bar{A}_1	\bar{A}_2	\bar{A}_3	\bar{A}_4	\bar{A}_5	\bar{A}_6	\bar{A}_7	\bar{A}_8	\bar{A}_9	O_0	O_1	O_2	O_3
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	0	1	1	1	1	0	0	0
X	X	X	X	0	1	1	1	1	1	0	0	1
X	X	X	0	1	1	1	1	1	1	0	1	0
X	X	0	1	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

- Note that the higher input lines get priority over the lower ones.
For example, \bar{A}_9 gets priority over $\bar{A}_1, \bar{A}_2, \bar{A}_3, \bar{A}_4, \bar{A}_5, \bar{A}_6$. If \bar{A}_9 is active (low), the binary output is 1000 (i.e., 0111 inverted bit-by-bit) which corresponds to decimal 7, irrespective of $\bar{A}_1, \bar{A}_2, \bar{A}_3, \bar{A}_4, \bar{A}_5, \bar{A}_6$.
- The lower input lines are therefore shown as "don't care" (X) conditions.

M. S. Park of Soong

Let us look at an example of this. Here is the 74147 decimal-to-BCD priority encoder. We have 9 input pins \bar{A}_1 through \bar{A}_9 . And these are active low as indicated by these circles. We have 4 output pins O_0 through O_3 and these are also active low.

Here is the truth table and let us now look at these rows one by one. In the first row all input pins are 1; that means, inactive. And all output pins are also 1; that means, inactive. So this output number corresponds to the binary number 0 0 0 0. All right let us come to the second row. In the second row \bar{A}_9 is low; that means, active and there are these do not care in all other columns; that means, it does not really matter what these are and \bar{A}_9 will be given a priority; so this input pin is active and the corresponding output pins are 0 1 1 0. And since the output pins are active low this number actually corresponds to the binary number 1 0 0 1 and that is decimal 9.

So, there is a correspondence between which pin is active here and the corresponding binary number here. Let us note that in all subsequent rows here, \bar{A}_9 has been shown as inactive. And that is because it has the highest priority. And if this was 0 anywhere then it would not matter what all these other entries are and the output would then be this

one. Now let us take this row here A8 bar is active and the corresponding output is 0 1 1 1 which is the binary number 1 0 0 0 or decimal 8.

Similarly, when A7 bar is active, we will find that the output corresponds to the binary number 0 1 1 1 that is decimal 7 and so on. So let us summarize, note that the higher input lines get priority over the lower ones; that means A9 bar gets priority over all these others A8 bar gets priority over A7 bar A6 bar up to A1 bar and so on. For example, A7 bar gets priority over A1 bar A2 bar A3 bar A4 bar A5 bar and A6 bar. If A7 bar is active that is low the binary output is 1 0 0 0 that is 0 1 1 1 inverted bit by bit. And that corresponds to decimal 7 and that is irrespective of all of these other input pins.

So, the lower to input lines therefore, are shown as do not care conditions. Digital circuits can be broadly classified into 2 types: 1 combinatorial circuit and 2 sequential circuits.

(Refer Slide Time: 08:05)



Sequential circuits

- The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are combinational in nature, i.e., the output(s) depends only on the present values of the inputs and not on their past values.
- In sequential circuits, the "state" of the circuit is crucial in determining the output values. For a given input combination, a sequential circuit may produce different output values, depending on its previous state.
- In other words, a sequential circuit has a memory (of its past state) whereas a combinational circuit has no memory.
- Sequential circuits (together with combinational circuits) make it possible to build several useful applications, such as counters, registers, arithmetic/logic unit (ALU), all the way to microprocessors.

M. G. Park of IIT Bombay

We now want to discuss sequential circuits, but first let us make a few important points. The circuits we have seen so far gates multiplexer demultiplexer encoders decoders are all combinational in nature that is the output depends only on the present values of the inputs and not on the past values. For example, let us consider multiplexer.

Now, the present value of the output for a multiplexer will only depend on the current or present values of the input lines and the present or current values of the select lines. And


it really does not matter what happened in the past, what values these input lines or select lines had in the past that simply does not matter. So that is a feature of all combinatorial circuits.

On the other hand, in sequential circuits the state of the circuit is crucial in determining the output values. For a given input combination, a sequential circuit may produce different output values depending on its previous state, and we will see several examples of that. So for sequential circuit 2 things are important, 1 the present values of all its inputs and 2 the state of the circuit. Now what is the meaning of state that we will see as we go along?

In other words, sequential circuit has a memory of its past state, whereas a combinatorial circuit has no memory. And that is the big difference between the 2. And finally, sequential circuits together with combinatorial circuits make it possible to build several useful applications such as counters registers arithmetic logic unit or ALU moving all the way to microprocessors.

(Refer Slide Time: 10:30)

NAND latch (RS latch)



A	B	X ₁	X ₂
1	0	0	1
0	1	1	0
1	1	previous	previous
0	0	1	1

- A, B: inputs, X₁, X₂: outputs
- Consider A = 1, B = 0
 $B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A X_2} = \overline{1 \cdot 1} = 0$
 Overall, we have X₁ = 0, X₂ = 1.
- Consider A = 0, B = 1
 $\rightarrow X_2 = 1, X_1 = 0$
- Consider A = B = 1
 $X_1 = \overline{A X_2} = \overline{X_2}, X_2 = \overline{B X_1} = \overline{X_1} \Rightarrow X_1 = \overline{X_2}$
 If X₁ = 1, X₂ = 0 previously, the circuit continues to "hold" that state
 Similarly, if X₁ = 0, X₂ = 1 previously the circuit continues to "hold" that state
 The circuit has "latched in" the previous state.
- For A = B = 0, X₁ and X₂ are both 1. This combination of A and B is not allowed for reasons that will become clear later.

M. G. Park @ Boston

So, here is a simple sequential circuit, and it is called a NAND latch. Then because it uses NAND gates and latch we will see why it is called a latch. It is also called RS latch and the meaning of this terminology will also become clear very soon all right. So here are 2 NAND gates, gate 1 gate 2. 2 inputs A going to the first gate B going to the second gate.

And the second input of the first gate is coming from the output of the second gate like that. And the second input of the second gate is coming from the output of the first gate.

So, our inputs are A and B and the outputs are X 1 and X 2. And now we want to prepare a table of the inputs and the outputs. Let us start with A equal to 1 and B equal to 0. So we have 1 here and 0 here. And we do not know what the other input of this gate is yet, but we do know that if this is 0, then X 2 is equal to be 1 because any one input of a nand gate being 0 implies that the output must be 1.

So, we can say that X 2 is 1 and now we come back to this gate. Its first input is 1 and the second input is also 1. And we know therefore, that X 1 must be the nand of A and X 2 and that is 1 and 1 and bar of that that is of course, 0 like that. So when we are A equal to 1 and B equal to 0 we have X 1 equal to 0 X 2 equal to 1. So that entry is now completed.

Next let us take A equal to 0 and B equal to 1. So this is 0 and this is 1. And by the same logic X 1 would now be equal to 1 and X 2 would be nand of 1 and 1 that is 0. So that is the second entry A equal to 0 B equal to 1, X 1 is equal to 1 and X 2 is equal to 0.

Now, let us consider A equal to B equal to 1. So 1 here, 1 here, and now X 1 is A X 2 bar, A X 2 and of that and then the not of that so A X 2 bar, that is X 2 bar, because A is one similarly X 2 is B and X 1 and not of that. So that is B X 1 bar and because B is equal to 1 it is equal to X 1 bar. So what it implies is that X 1 and X 2 must be inverse of each other. X 1 is X 2 bar X 2 is X 1 bar. And this relationship is indicated here by writing X 2 bar here and X 1 bar here. So X 1 is the same as X 2 bar X 2 is the same as X 1 bar.

Now, whether X 1 is 0 or 1 we do not know. And let us take 2 cases for example, if X 1 was 1 earlier and X 2 was 0, then that situation will continue to hold because it does satisfy this relationship. Similarly, if X 1 was 0 and X 2 was 1 previously then the circuit will continue to hold that state. So both of these are possible we can have X 1 equal to 1 and X 2 equal to 0 or X 1 equal to 0 and X 2 equal to 1. And which one of these will actually happen depends on what was X 1 what was X 2 before. And in the table for the latch we write this observation as previous; that means, whatever was the previous condition either this one or that one will continue to hold. And that is why this circuit is

called a latch, because in a sense the circuit is latched in the previous state it is not changing.

One last point for A equal to B equal to 0 X 1 and X 2 are both 1 if you have 0 here; that means, X 1 is 1 if we have 0 here; that means, X 2 is 1 and that is irrespective of this other input. So for A equal to B equal to 0 X 1 and X 2 are both 1, but this combination of A and B is not allowed and we will see the reason for that little later. All right so we have written 1 1 here, but we should bear in mind that this combination is not really allowed.

(Refer Slide Time: 16:23)

NAND latch (RS latch)

A	B	X ₁	X ₂
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	

R	S	Q	\bar{Q}
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	

- The combination $A = 1, B = 0$ serves to reset X_1 to 0 (irrespective of the previous state of the latch).
- The combination $A = 0, B = 1$ serves to set X_1 to 1 (irrespective of the previous state of the latch).
- In other words,
 - $A = 1, B = 0 \rightarrow$ latch gets reset to 0.
 - $A = 0, B = 1 \rightarrow$ latch gets set to 1.
- The A input is therefore called the RESET (R) input, and B is called the SET (S) input of the latch.
- X_1 is denoted by Q, and X_2 (which is \bar{X}_1 in all cases except for $A = B = 0$) is denoted by \bar{Q} .

M. S. Park of Samsung

Now, these names A B X 1 X 2 are not really used in practice. What we use in practice is some more meaningful names and let us see what those are. And we start with some observations first. The combination A equal to 1 B equal to 0 serves to reset X 1 equal to 0 and why are we saying that look at this table. A equal to 1 B equal to 0 X 1 is 0. And this happens irrespective of the previous state of the latch. So if you go back one slide you will realize that this is derived with no reference to the previous state of the latch.

Similarly, the combination A equal to 0 B equal to 1 the second one, serves to set X 1 to 1 you have X 1 equal to 1 here, and that again is irrespective of the previous state of the latch. In other words, with A equal to 1 B equal to 0 the latch gates reset to 0 this entry here A equal to 1 B equal to 0 the output of the latch which we will consider as X 1 is

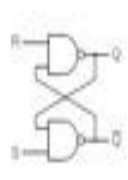
getting reset to 0. And with A equal to 0 B equal to 1 the latch gets set to 1 this entry here A equal to 0 B equal to 1 and the latch output which is X 1 is now getting set to 1.

So, the A input therefore, is called the reset or R input because when this A is active or 1 then the latch is getting reset, the latch output is getting reset to 0. And B is called the set or S input of the latch. When B is 1 or active the latch output is getting set to 1. The latch output X 1 is denoted by Q and X 2 is denoted by Q bar. And the reason for that is X 2 is the inverse of X 1, here as well as here .and if you recall in the last slide we saw that in this third row when A and B are both one X 1 and X 2 were inverse of each other. Only in the fourth row X 1 and X 2 are not inverse of each other, but that row is anyway invalid.


So, this is our circuit diagram with the new input and output names are here for reset S here for set Q that is a latch output and Q bar is the inverse of this. And here is the table that describes of the latch works when R is 1 S is 0 Q is 0, and it is easy to remember this. R equal to 1 means something is getting reset and that something is the latch output Q so Q is becoming 0. S equal to 1 means something is getting set and Q is getting set to 1 over here. When R and S are both 1 as we saw in the last slide nothing really changes and the previous state Q as well as Q bar continues to hold and the last entry as we have seen is invalid and later we will explain why this entry is considered invalid.

(Refer Slide Time: 20:19)

NAND latch (RS latch)



R	S	Q	Q̄
1	0	0	1
0	1	1	0
1	1	previous	previous
0	0	invalid	invalid



- Up to $t = t_1$, $R = 0$, $S = 1 \rightarrow Q = 1$.
- At $t = t_1$, R goes high $\rightarrow R = S = 1$, and the latch holds its previous state \rightarrow no change at the output.
- At $t = t_2$, S goes low $\rightarrow R = 1$, $S = 0 \rightarrow Q = 0$.
- At $t = t_3$, S goes high $\rightarrow R = S = 1$, and the latch holds its previous state \rightarrow no change at the output.

M. G. Park of Nirma


Let us now look at some waveforms so as to understand this table better. Here is the R input for the latch here is the S input that is 0 and that is 1; similarly, 0 and 1 and so on. This success processes the time axis. Now in the beginning up to t_1 that is we have R equal to 0 S equal to 1, and the output Q is therefore, determine without any ambiguity. So R is 0 S is 1 so the flip flop is set to 1 so we have Q equal to 1 and Q bar equal to 0. All right now at t_1 there is a change R changes from 0 to 1, is continues to be 1 and now let us look at this table and figure out what should happen.

So we have R equal to 1 S equal to 1, here and this is previous; that means, Q will stay at it is previous value Q bar will also stay at it is previous value. And that is what we see over here Q remains equal to 1 Q bar remains equal to 0. At t_2 there is a change S changes from 1 to 0, R continues to be 1 so we have R equal to 1 S equal to 0. So let us look at this entry R equal to 1 S equal to 0 so that resets the flip flop; that means, Q becomes equal to 0. And this happens irrespective of what happened in the past, so Q becomes equal to 0.

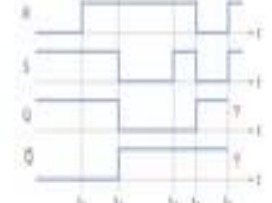
And Q bar becomes equal to 1 and that continues up to t_3 , when there is one more change that is S goes back to 1 R continues to be 1 so we have R equal to 1, S equal to 1 and therefore, Q and Q bar retain their previous values. So previously Q was 0 so it continues to be 0 Q bar was 1 it continues to be 1.

(Refer Slide Time: 22:49)

NAND latch (RS latch)



R	S	Q	Q̄
1	0	0	1
0	1	1	0
1	1	previous	previous
0	0	invalid	invalid



Why not allow $R = S = 0$?

- It makes $Q = \bar{Q} = 1$, i.e., Q and \bar{Q} are not inverse of each other any more.
- More importantly, when R and S both become 1 simultaneously (starting from $R = S = 0$), the final outputs Q and \bar{Q} cannot be uniquely determined. We could have $Q = 0, \bar{Q} = 1$ or $Q = 1, \bar{Q} = 0$, depending on the delays associated with the two NAND gates.

All of these observations are listed here and you can go through that once again. If you like the question we want to answer now is why not allow R equal to S equal to 0, in this table for R and S equal to 0, we say invalid here so the question is why is this invalid all right. So there are 2 parts to the answer one it makes Q and Q bar both equal to 1, as we have seen earlier that is Q and Q bar are not inverse of each other anymore, and it does not make sense to call this output Q and this Q bar. So that is one reason, but that is relatively trivial reason there is a much more important reason and let us look at that now.

Let us look at these waveforms which we saw in the previous slide. And if you recall we had come up to here in the last slide. We had R equal to 1 S equal to 1 Q equal to 0 and Q bar equal to 1. And now imagine that R has gone to 0, S has also gone to 0 so we have R equal to 0 S equal to 0. Now when R becomes equal to 0 the output of this nand gate becomes 1 no matter what the other input is, and similarly the output of this nand gate also becomes equal to 1 if S becomes equal to 0. So then we have Q equal to 1 and Q bar equal to 1, and that is what is shown over here. Q is equal to 1 Q bar is equal to 1. And that situation continues up to t_5 , and suppose that at t_5 R has gone back to 1, and S has also gone back to 1 so now, we have R equal to 1 S equal to 1, what is the table say for R equal to 1 S equal to 1 it says previous.

Now, the question is which previous value? Is it Q equal to 1 or is it Q bar equal to 1. And there is a question mark here. What happens in real life is that depending on the delays of these 2 gates the last will finally, settle down to either Q equal to 0 or Q equal to 1; that means, Q equal to 0 and Q bar equal to 1 or Q equal to 1 and Q bar equal to 0. So there is some uncertainty about what will happen in this case and that is why we consider this situation as invalid.

Let us summarize. We have started looking at sequential circuits we have seen how they are different than combinatorial circuits and we have looked at the latch which is made up with nand gates. That is all for now, see you in the next class.