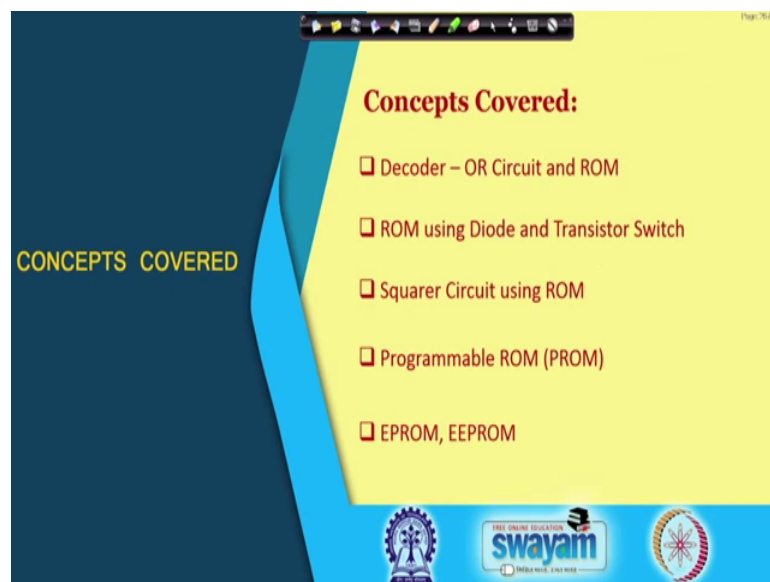


**Digital Electronic Circuits**  
**Prof. Goutam Saha**  
**Department of E & E C Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 59**  
**Read Only Memory (ROM)**

Hello, everybody. In earlier classes, we had seen random access memory which is also read write memory. So, the user can write as well as read information from that memory block. In today's class we shall look at Read Only Memory ok. So, from the user point of view reading will be done multiple times, but writing is to be done by the developer by certain mechanism or it will be, writing will be done in the factory end.

(Refer Slide Time: 00:55)



And these are the concepts that we will cover. We shall see the similarity between decoder or circuit the paradigm that we had use before and read only memory, then we shall see read only memory using developed using diode or transistor switch ok. And then as I said it has the similarity between simulate with decoder or circuit and essentially it is a combinatorial circuit or this ROM.

So, we can have a several combinatorial circuit design very easily using ROM. So, examples will be given for squarer circuit, but one can develop many other circuit following the same design philosophy and at the end we shall discuss programmable ROM it is different verities.

(Refer Slide Time: 01:50)

**Decoder-OR Circuit and ROM**

A Read Only Memory (ROM) is memory device where binary information is stored in certain interconnection pattern that is non-volatile.

$2^m \times n$  ROM

$m$  address inputs →  $n$  data outputs

$A_{2:0}$			$D_{2:0}$		
A	B	C	$F_2$	$F_1$	$F_0$
0	0	0	1	1	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	0	1

3 to 8 Decoder

Decoder - OR circuit that is equivalent to  $8 \times 3$  ROM

ROM is essentially a combinatorial circuit.

So, Read Only Memory in this the binary information is stored in certain interconnection pattern. So, what is that interconnection pattern that we shall see shortly and this pattern is non-volatile unlike ram the kind of things that we had discussed before. So, when if the power is OFF once it is return by the developer or the programmer in programmable ROM or standard ROM at the factory end. So, it will be remain it will remain available for reading because the pattern is ingrained in the memory block ok.

And so the basic block diagram of a ROM will be something like this. So,  $m$  address input ok. So, this we will generate  $2^m$  locations memory locations right and in each location if  $n$  bit information is stored so,  $n$  data outputs you can be generated by invoking a particular setup address bits ok. And there can be a additional control input like you know enable etcetera ok. So, but this is the basic block diagram right. So, there is no write option that you can see over here. So, it is mostly it is reading option because write is done writing is a done in a special manner.

Of course, somebody is writing otherwise how can we you know read it ok, but that is done separately ok. So, let us see the similarity with decoder or you know paradigm circuit. So, we consider a 3-bit address  $A_{2:0}$  and 3-bit data ok. So, this example this circuit that we have see over here I have taken it from our earlier discussion with decoder circuit in the combinatorial circuit discussion ok, somewhere in week 4 or so ok. So, you

can see this circuit over there ok, but just to compare and see how it is similar we have taken it again in this particular discussion right.

So, this 3-bit means 8 address location. So, this is the decoder output you can see the decoder output 0 0 0 0 1 1 0 right. So, there will be a 3 to 8 decoder as we see as we have seen in a memory block alright. After that the information that you want to store for example, forget about this functions F 1, F 2, F 3. So, you can consider this is D 2 this is D 1 and D naught and in each these of this location you want to store some information, some binary information right. So, here is an example say that you are storing 1 1 0 here 0 0 1 again 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 0 1 you can take some other value as well. Is it fine? Right.

So, if you do that; if you do that ok, then what you can see this is what you require. This is what you are looking for in you from your memory right and if you look at this particular circuit which is generating this you know this is the decoder outputs A prime, B prime, C prime to ABC all 8 you know this lines and this is the OR outputs which is taking specific values right and this specific values are in this particular truth table.

If you put it in the form of a truth table so, D 2 becomes your F 1 D 1 becomes your F 2 and D naught becomes your F 3 this is the way we have, you know name has been given we can give it in a different manner also change the order etcetera. So, then say D 2 this is there. So, 1 is over here, 1 is over here and 1 is over here right. So, you can write these as a summation of this is the mean terms 0 4 6 0 4 6 for D 1 which corresponds to your F 2 ok.

So, 1 is there here 0 and 5. So, this is these two mean terms we can add using a OR gate and F 3 1, 2, 3 and this is the 7. So, that is what you can see and after that what happens if you are just say invoking 0 0 0, so, 0 0 0 over here right. So, then this line will be high and rest of the lines will be 0 all of them will be 0 right. So, this line high means this is OR gate.

So, this is having a input 1 0 0. So, this output will be 1, this will be 1 and this is 0 because all these are 0 from the because of the decoder. So, these output will be 1 and here all these inputs are 0 because coming from here. So, all the inputs to the OR gate will be 0 to 0 right. So, these output will be 0. So, will be reading if you put 0 0 0 at its output 1 1 0 as D 2, D 1 and D naught ok.

Similarly, for every other location you can see that is happening. So, if it is 0 0 1, what will happen? So, A prime B prime C these particular will be output will be 1 and rest all 0 these are all 0. So, what happens to these input of this OR gate? So, this is 0 0 0 because of this that this decoder outputs are 0 for this respective cases. Similarly, about this is 0, this is 0 here this particular 1 is 1 and rest all are 0 ok. So, this is output will be 0 0 1, is it fine? Is it understood? Right.

So, you place the location address bit and at the output you will get the way we have configured it D 1, D 2, D 1 and D naught by this arrangement, is it fine? So, ROM is essentially a combinatorial circuit that is what we have noted.

(Refer Slide Time: 08:28)

**Diode Switch and ROM**

$A_{2:0}$			$D_{3:0}$			
A	B	C	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	0	0	0	1	1	1
0	0	1	1	0	0	0
0	1	0	1	0	1	1
0	1	1	1	1	0	0
1	0	0	0	1	1	0
1	0	1	1	0	0	1
1	1	0	0	0	1	1
1	1	1	1	1	1	0

From the list of data to be stored, IC manufacturer produces a mask (photographic template) of the circuit which is used in the production of ROM.

$$Y_3 = \sum m(1,2,3,5,7)$$

$$Y_2 = \sum m(0,3,4,7)$$

$$Y_1 = \sum m(0,2,4,6,7)$$

$$Y_0 = \sum m(0,2,5,6)$$

Now, let us move forward we look at another such you know thing where this is a 8 cross 4 ROM that we are interested in ok. So, A 2 to 0 is my address bits. So, it is generating all 8 you know possible locations in the memory right. So, this is the decoder block over here right. So, the first one is a bar B bar C then A bar B bar C. So, this is first one is A bar B bar C bar second 1 is A bar B bar C and this is the way that we have seen it before alright.

Now, in each of this locations this is D 3 to 0. So, this is D 3, D 2, D 1 and D naught for again forget about this Y 3 etcetera which we shall take up later right and what is the information that we want to store? So, you can take something which as I said this is fixed by the developer or the factory, you are not changing it again and again. So, what

you want to fix you have to tell it in advance ok. So, it has to be told to the IC manufacturer which we will prepare a specific mask a photographic template of the circuit and which will be used in the production of the ROM ok.

So, say you have said I want in the first location 0 1 1 0 triple 1, second location 1 triple 0, third location 1 0 1 1 this need to be stored this is the way the rest of the cases ok. So, this is what is over there. Now, one way to achieve this which is equivalent to decoder or that part I shall discuss later is by using diode switch by using diode switch ok. How what does it do? So, we can see this is the decoder output and these are the 4 lines right they signify the D 3, D 2, D 1 and D naught right.

And wherever 1 is present you connect a diode, connect a diode, connect a diode and wherever 0 is there at that cross point no diode is connected. So, that means, here is 1 wire another wire is just going over it there is no connection between them. So, there is no connection between these two ok. So, this column over here this vertical line is not getting any input from A prime B prime C prime like. So, they are not connected with each other. Here it is connected Y 2, Y 1, Y naught are connected; is it alright? So, when 0 0 0 is placed what will happen? 0 0 0 is placed over here so, only this one will be high and rest all are 0 because of the decoder action. And at that time so, this is there is no connection over here.

So, at the input of over this line only 0 is present I mean no connect no current is flowing through this, no i r drops Y 3 will be 0 and here there is a I voltage 1 is present. So, some 0.7 voltage drop of these diode or so and then that is something voltage drop is occurring across this so, which will be considered as logic high. So, this Y 2 will be considered as high Y 1 will be considered as high, Y naught will be considered as high alright.

So, your D 3 will be 0 and D 2, D 1, D naught will be 1 1 1, is it fine? So, that way continuing for each of these cases wherever a 1 is present in the memory for example, 1 triple 0. So, here there is a in the cross point diode is there right and rest of the cases no diode is present. So, this is the way when you will generate a pattern right by masking process right and come up with this ROM right. This is the storage that we will see in the memory block right; this 8 cross 4 ROM block right.

Now, as we have noted so, this memory is also is a combinatorial circuit right. So, now, if we represent D 3 as Y 3 a function a Boolean function getting generated and D 2 as Y

2, D 1 as Y 1 and D naught as Y naught right then this Y 3 is what? If you look at the whole of it earlier we are looking at each individual location wise. Now, if you look at all the columns for put together for this particular Y 3, the way we had seen in our earlier decoder or arrangement, but specifically a OR gate is put. Here we are not putting any OR gate just resistance is there, but it is giving a equivalent representation and that is what that is you will find that this is 1 2 3 5 7. So, these are the min terms that is getting summed up.

So, Y 3 D 3 are that is also a Boolean function generator in the place of D 3 which is generating this function. Similarly Y 2 is generating this if you look at 0 0 3 4 and 7, Y 1 0 2 4 6 7 and Y naught is 0 2 5 6, is it fine. So, here we do not see a you know OR gate as such specially put only diode and resistance, but if this is also within your decoder OR equivalent relationship fine.

(Refer Slide Time: 14:45)

### A Squarer Circuit using ROM

Consider a squarer circuit with 3-bit input in this example.

Input			Output							
$A_2$	$A_1$	$A_0$	Dec.	$D_5$	$D_4$	$D_3$	$D_2$	$D_1$	$D_0$	Dec.
0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	1	1
0	1	0	2	0	0	0	1	0	0	4
0	1	1	3	0	0	1	0	0	1	9
1	0	0	4	0	1	0	0	0	0	16
1	0	1	5	0	1	1	0	0	1	25
1	1	0	6	1	0	0	1	0	0	36
1	1	1	7	1	1	0	0	0	1	49

$D_0 = A_0$   
 $D_1 = 0$

$D_2 = \sum m(2,6)$   
 $D_3 = \sum m(3,5)$   
 $D_4 = \sum m(4,5,7)$   
 $D_5 = \sum m(6,7)$

Less than 8 x 6 ROM does.

So, this we can utilize in generating various important you know combinatorial circuit block ok. So, where is an example of a squarer circuit using ROM, because essentially it is a decoder or architecture is there right, even if you use diode or you know resistance and all at the cross point and then output taken across the resistance right.

So, here the in these example; so, this is a 3-bit squarer input is there for which a squaring is being is to be done and 3-bit. So, these are the different possible values. So, 0 to 7 these are the decimal equivalent right and corresponding output is 0, 1 is 1, 2 is 4, 3

is 9, 4 is 16, 5 is 25, 6 is 36 and 7 is 49 and you can write them in binary right. So, 0 is all 0, 1 is 5 0 and then 1, 4 is 0 0 0 1 0 0 right, 9 is this is 1 0 0 1, 16 is 1 all 0 over here 25 means 16, 8, 24 and there is 1 36. So, 32 and 4 this 2 you can see and 49, 32, 16 and this is 1 ok.

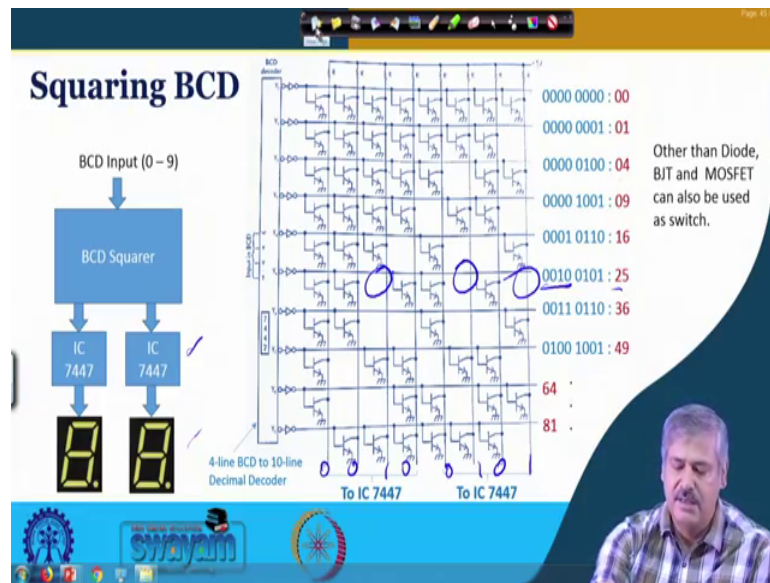
So, this becomes a kind of you know look up table that if this is the input corresponding output should be like this right. So, if you just look at it and you know try to realize using a ROM then you may think of that I need a 8 cross 6 because there are 8 inputs and 6 outputs, 8 cross 6 ROM ok. But, if you look at it look at it closely then perhaps you can see that you can do with less lesser size ROM as well ok. How?

You can see that D 1 is always 0 ok. D 1 is always 0 you can permanently can connect it to ground over here and D naught you can see whenever A naught is 0 D naught is 0, A naught is 1 D naught is 1, just following you know. So, D naught can be rarely connected to A naught and for rest of the cases or rest of the cases you can see that say D 2 is min terms; this is 2 and 6 right.

These two put together in a sop realization that decoder or the min term getting added by OR ok. So, for D 3 we can see this is D 3 – 3 and 5, for D 4 – 4, 5 and 7 and 5 for D 5 – 6 and 7. So, all this four are required. So, we can have a 8 cross 4 RAM instead of 8 cross 6 right because these two are already you know realized by a simpler means ok.

So, we can connect it in this manner and within this 8 cross 4 this is the way the a pattern need to be you know ingrate then this particular block will be a squarer circuit for a 3-bit input, isn't it?

(Refer Slide Time: 18:24)



So, similar thing you can extend for a more complex circuit and you know for example, squaring of a BCD number right. So, squaring of a BCD number; so, BCD input is 0 to 9 right and the output could be in this case 0 to 1 81, 9 is 81, 8 is 64, 7 is 49 ok. So, this is the way the number will be there and here what is ask is that the output will be shown in the BCD format unlike the previous or previous it was a binary format. So, this is the requirement that also is possible. How? We shall see.

So, this is BCD squarer. So, if you look at it that the number is up to 81, so, 2 digit number. So, 1 digit will be fed from one particular fed through one IC 7447 to a display and another digit load significant you know the one this binary coded decimal ok. So, that will be fed to the 7447 and the corresponding decimal number will be fed, is it fine? So, this is the overall these thing and then we will look at having a particular pattern ingrate in a ROM.

So, here in this example we show that the connection at the cost point is made through transistor a BJT instead of diode that is also possible. It can be done through MOSFET also right. So, wherever this is connected like this you can understand that you know this what you can see if in this particular case right, this is 5 volt ok. So, if this is on; please understand the difference between what we had seen before and this one. So, if this is on, this is high then this transistor will be off. So, this will be put to saturation very low



voltage 0.1 or 0.2 volt. So, this voltage here will be low. So, whatever you are reading here will be low, is it alright? Clear.

So, voltage at the input of this the transistor next the output low right at the cross point and if at the cross if at the cross point no such thing is there, then these output will be you can see over here these output over here, there is no connection in this cross point. So, this 5 volt will be coming over here and depending on the load impedance or so, some voltage you know some current drawn some current will be drop across these particular resistance for that will be sufficiently small for which the output will get recognized as logic high.

So, presence of transistor will give you a low value when this particular row is access. So, decoder output becomes high and absence of transistor will make it high; presence of transistor low, absence of transistor is high. Is it fine? Is part this part understood? How this circuit is working, how the current flow and the voltage level coming over here at the output these are the outputs ok.

Now, if you have understood that then the first 1 the BCD input it is a 0 0 0 0 right. So, this one will be this is a 4-line BCD to 10-line decimal decoder. So, this one will be activated right 0 0 0 0 ok. So, this for this particular 0 0 0 0 this is getting activated Y 0 right. So, at that time what we wanted? The output right, 0 0 both of them should 0 here, 0 here and 0 here right. So, this 4-bit should be 0 0 0 0 this 4-bit should be 0 0 0 0 ok.

So, accordingly all the transistors need to be present because absence of transistor will make a logic high, one value present here clear. So, next one is 1. So, it should be 0 0 0 over here and 0 0 1 over here. So, 0 0 0 0 and here it should be 0 0 0 1. So, here you can see that the condition is absent right. So, when this line is invoked 0 0 0 1 is present. So, this is decoded. So, all of this particular place is this transistors will be on right up to this. So, this will be output will be low as 0.2 volt, 0.1 volt this whatever the saturation voltage or so. Here it is no connection is there. So, 5 volt it is whatever you know because of the load current some drop over here it will be logic 5.

So, similarly 4, 9, we take up just say one example over here say 25 5 has been put; so, in this line. So, it should be because it is presented through 7-segment display. So, 0 0 1 0 should come here 0 0 1 0 and here 0 1 0 1 should come. So, what you can see? That 0 0 1 0; so, this point we do not have a transistor here 1 0 1. So, this point and this point we

do not have transistor, is it fine? And then it is connected to this line goes to this 7447 and through that to 7-segment display decoder.

So, this way many different kind of you know look up table kind of you know this pattern can be generated and corresponding circuit can be obtained.

(Refer Slide Time: 24:19)

**Programmable ROM (PROM)**

- PROM allows the user, instead of the manufacturer, to store the data.
- An instrument called a *PROM programmer* stores the words by "burning in."
- Originally, all diodes with a *fusible link* remain connected at the cross points.
- The PROM programmer sends destructively high currents through diodes that are to be removed.

To store 1001 ( $Y_{30}$ ) in the address location  $ABC = 000$ , fuses at the cross points of  $Y_2$  and  $Y_1$  in the  $\bar{A}\bar{B}\bar{C}$  row need to be burnt. Similarly for other cross points according to what is stored in each address.

So, far what we have discussed is ROM which is the pattern is ingrained pattern is prepared in the factory itself ok. But, if a developer wants to put his own pattern he cannot send it to the factory or so for which we have what is called programmable ROM ok.

So, in this at the cross point the connections that you are saying say diode is there, so, after that there will be a fuse ok. So, wherever you want the connection not to be there you send a high current by which the fuse is burnt and the connection is lost. So, to begin with you have all the fuses present like this is the original thing. All the fuses are present in the OR plane ok.

And this is the decoder, this is the fixed and array which is you know acting as a decoder alright. And then whenever you want to you know put something you just do accordingly to burn the fuses. So, this is the burning in process ok. So, this is destructively high destructively high currents are sent through diodes.

So, for example; in the first location you want to store 1 0 0 1. So, when this is 0 0 0 what you want to store in that corresponding location is 1 0 0 1. So, this decoder output will be accessed ok. So, at that time this point and this point if you want to store, so, these two corresponds the if you just will be burnt and similarly, other points depending on what you want to store the corresponding points we burnt ok. And then here we have got the resistance and connected to the ground which effectively gives you a OR gate realization.

(Refer Slide Time: 26:29)

**EPROM, EEPROM**

- The erasable PROM (EPROM) uses MOSFETs.
- Data is stored with an EPROM programmer.
- All stored data can be erased by shining ultraviolet light through a quartz window that releases all stored charges.
- There is one time programmable EPROM without window.

Electrically Erasable PROM (EEPROM) is similar to EPROM where data is erased from target cells by removing the charge for which a pulse of opposite polarity is sent. EEPROM is very slow.

Flash Memory is further advancement of EEPROM. It is much faster as data writing is in block (say, 512 bytes) instead of 1 byte at a time.

Electrical charge is forced on floating gate. When electron is present, threshold voltage is higher than normal which is usually considered as logic '0'; else, logic '1'.

Reading: 5V to gates of both  $Q_1$  and  $Q_2$ .  $Q_1$  OFF if charge in floating gate.

Row decoder output

Programme Line (P)

Floating Cell

Column

Now, in PROM once you engrain the pattern you cannot remove it. So, then comes what is called erasable PROM, a further advancement here where the data is written of course, EPROM programmer ok. And so, here the data writing is done through something called a floating gate ok. So, the MOSFET based you know realization these memory cell realization. In this there is a; this is the normal standard MOS gate MOSFET the gate side, but here there is an additional gate called floating gate that is put over here ok.

And by a particular phenomena when appropriate physics works out so, that we will not going to the details of that tunneling etcetera ok. These specific phenomena that happens over there from electrical or electronics you know circuit point of view what we understand is that in this particular floating gate electrical charge when it is stored the threshold voltage increases and when it is not stored so, threshold voltage is the normal threshold voltage that you see for the gate to work ok.

So, threshold voltage when it is higher than normal usually considered as logic 0 and otherwise it is considered as a logic 1 right. So, this is what is used in erasable PROM. So, this floating at whatever charge is getting stored or not stored that condition is altered when the erasing process is done alright. And in normal EPROM it is done through ultraviolet you know light impinging on this particular IC through a quartz window where all stored charges get released ok. It becomes all equal and then after that during programming you send a appropriate programming pulse of particular magnitude which will ensure you know storage of charge in specific location. So, that is the writing part of it.

And in Electrically Erasable PROM, so, that is normally EEPROM where ultraviolet you know light is used for using purpose. In Electrically Erasable PROM the data is erased by removing the charge using a electrical pulse of opposite polarity instead of UV ray now electric electrical pulse of opposite polarity is used for erasing and of course, this is very slow.

And after that came flash memory where which is faster than EEPROM or E square PROM often it is called often it is called E square PROM also ok. These Electrically Erasable PROM where the data writing is happens in blocks which is of the size of say 512 bytes instead of what happens for normal E square PROM which is 1 byte at a time, so, that makes it faster.

So, here is just one memory cell you know kind of example. So, during programming is sent the pulse is sent of high magnitude relatively higher magnitude and the floating gate the charge is stored whenever we require it. And during the reading process both of them get 5 volt and depending on this particular case this Q 2 will be off if charge is there in the floating gate because the threshold voltage becomes much higher and if it is on, then it will be the normal operations. So, this this will be also on and this voltage will be accordingly, is it fine?

(Refer Slide Time: 30:45)

### EPROM: IC 2716

- For reading, valid address,  $\overline{CE}$  and  $\overline{OE}$  are successively placed or made L to get valid output.
- Access time up to 450 ns.
- Erasing by exposing to light with wavelength shorter than 4000 Å. (2537 Å recommended.)
- Opaque label to prevent slow erasing by sun / room light.
- During programming,  $V_{pp}$  is at 25 V,  $\overline{OE}$  is at H, data to programmed is applied to output pins in parallel. Data and Address level is TTL.
- When Address and Data are stable a 50 ms TTL pulse to  $\overline{CE}$ .
- A verification is done after programming.

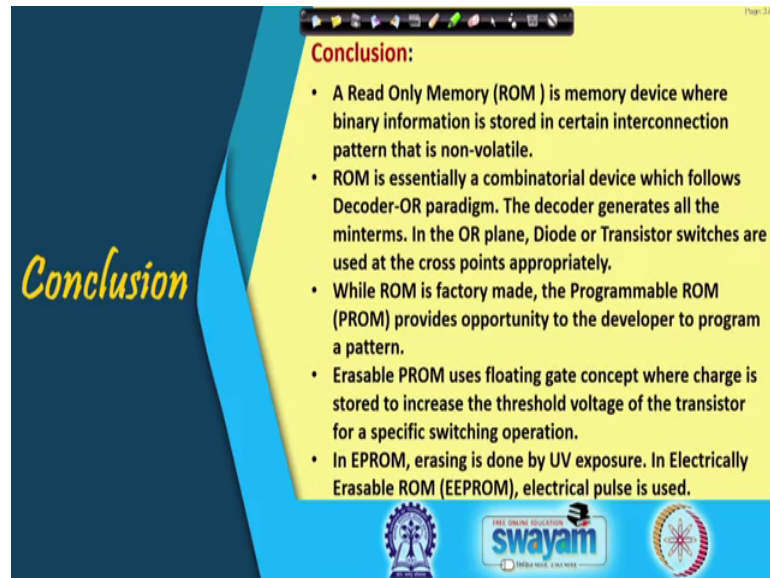
To end just we look at one particular EPROM IC which you might be using in the lab which is IC 2716. It is often used in the digital electronic circuit lab. So, this is a of course, you can see there is a quartz window over here. So, this is a UV ray it I mean this EPROM can be erased by using UV ray and when you want to read it first the a valid address is put after that chip enable and then output enable are successfully placed and then the valid output is obtained access time can be up to 550 nanosecond. So, in that sense it is a bit slower ok.

And erasing of course, by ultraviolet ray which is shorter than 4000 angstrom, recommended is 2537 angstrom. After writing a opaque level is put over here to prevent sunlight or room light coming in other because those light has components which is less than 4000 angstrom. So, by which these information that is stored there can get slowly erased and for writing of course, there is no data write-in inputs. So, these data outputs that are there ok, there only the data is placed for writing the TTL level the data and these are the address that will be there and at that time a programming pulse that is there of 25 volt is used out. These output enable is at high and data to be programmed is applied at the output pins in parallel over here and data and address level are all a TTL.

And when address and data are stable 50 millisecond TTL pulse is placed to chip enable alright and by that process writing is gets done. And after writing a verification is done with that the data written is what was intended and if verification passes then the devices

becomes ready for the use ok. This is the way the EPROM you know writing or programming takes place.

(Refer Slide Time: 33:18)



**Conclusion:**

- A Read Only Memory (ROM ) is memory device where binary information is stored in certain interconnection pattern that is non-volatile.
- ROM is essentially a combinatorial device which follows Decoder-OR paradigm. The decoder generates all the minterms. In the OR plane, Diode or Transistor switches are used at the cross points appropriately.
- While ROM is factory made, the Programmable ROM (PROM) provides opportunity to the developer to program a pattern.
- Erasable PROM uses floating gate concept where charge is stored to increase the threshold voltage of the transistor for a specific switching operation.
- In EPROM, erasing is done by UV exposure. In Electrically Erasable ROM (EEPROM), electrical pulse is used.

Quickly to conclude ROM is a device where information is stored in the form of interconnection pattern which is non-volatile. Essentially it is a combinatorial device following decoder or architecture and we have diode or transistor based switches placed in the cross points in the OR plane. ROM is factory made, but PROM programmable ROM can be programmed by a developer to inscribe a particular pattern. Erasable PROM uses floating gate concept where charge is in stored to increase the threshold voltage of the transistor for a specific switching operation. And in EPROM erasing is done by ultraviolet exposure and electrically erasable PROM electric pulse used for erasing purpose.

Thank you.