

Design of Mechatronic Systems
Professor Prasanna S. Gandhi
Department of Mechanical Engineering
Indian Institute of Technology, Mumbai
Lecture 15
Microprocessor Memory and Addressing

So, what we have seen in the last class, we will continue now, take forward that discussion for further building blocks of the microprocessors. So, we saw combinational logic, we saw sequential logic, and now we will see how to build memory using now some of those fundamentals. And further we are going to kind of see the construction of like some primitive microcontroller or microprocessor, so, that we understand some basic stuff, and then we will go ahead with the modern microcontrollers or microprocessors discussing that.

(Refer Slide Time: 01:03)

The slide, titled "Building Memory", features a green background with a white gear icon in the top left corner. In the top right, there is a small grey box with the text "CALENDAR NPTEL Des Mechtr Today at 3:30 PM". The main content consists of three bullet points and a diagram. The diagram shows three D flip-flops connected in a row, each with a 'D' input, a 'Q' output, and a 'CK' (clock) input. A single clock signal line is connected to the 'CK' input of all three flip-flops. The entire assembly is labeled "Register". A note at the bottom right of the diagram states: "Notice clock signal for all D Flipflops is shared". The NPTEL logo is visible in the bottom left corner of the slide.

- Suppose we need to now construct memory using our D flip flops, how do we go about? What all we need.
- Lets start with putting 1 flip flop for storing one bit. Lets say for simplicity our one 'cell' or 'register' consist of 3 bit number
- Thus 3 flip flops for each cell would be needed and can be represented as the following

Register

Notice clock signal for all D Flipflops is shared

So, let me switch to the screen. So, what we are going to see is building, how do we build memory now, out of the D-flipflops that we have seen before. So, if you recall the D-flipflops hold their state as long as the clock input is given to them. So, some or say this is, this clock input if it is going, it is changed or it is going as a pulse here then whatever input is available here will be stored in the value, and that value will be available as a Q.

So, one kind of a D-flipflop is going to hold one kind of a bit memory. So, we put together such three flip flops and all all of them are having the same this clock signal, then the form a 3-bit kind of register. So, this is how like you can construct now 3-bit register. So, we have some three storage elements or one bit storage elements and they are all can be controlled by using single kind of a clock input to store some value.

(Refer Slide Time: 02:11)

Building Memory

- Notice the same clock/enable signal
- Ok fine we have basic blocks in place but there are following questions?
 - Q: How do we save the number say 101?
Have it ready on D lines as shown below and have signal pulse given to clock line named as write enable
- Q: Once saved how do we retrieve it back?
It will be always available on output Q lines as long as WR is low!
- Q: Will the number be lost if power is turned off? **YES**

The diagram shows three D-flipflops connected to a common clock line labeled 'WR (write enable)'. The D inputs are set to 1, 0, and 1 respectively. The Q outputs are shown as 1, 0, and 1. A waveform for the WR signal shows a pulse that coincides with the clock inputs of the flipflops.

Now, let us see how this value can be stored and retrieved from this register. So, we have this basic building block in place, but the question is if you want to save this number 101, how do we save that? That number should be available as high voltage, low voltage and high voltage again on these three pins for one flip flop here, for this first flip flop 1 and second flip flops 0, and third D-flipflop it should be 1.

If it is available, then when I give this write enable now this clock signal, we are calling it as WR or write enable kind of a signal. So, this signal like, is this kind of a pulse here then that pulse is going to get this input inside into and store the value inside this D-flipflop and now that value will be available at this terminal Q also. So, so, this is how one can save this value in the D-flipflop.

So, we are now thinking like creating this 3 bit register and from here we want to retrieve the value, retrieve value means like the value will be already available at these Q terminals. But now say we want to use the same data lines to transmit, to transfer the value into the register or take the value out of the register. So how do we do that is what is our next question.

And let us see if the, if the number if you, if you turn off the power that is given to these D flip flops then like the memory will be lost, we will not have the same number available when the power comes back. So, there is a, this is a different kind of like a memory where it is not able to hold the value when the power goes off.

There are some other memories where this feature can be possible, they will have some kind of a capacitive elements to hold the value or some other kind of technologies, we will not get into the details of that. So, you can have a look at and get that information, what we are looking at is like the fundamental principles of construction of memory and how do we address the memory here now.

(Refer Slide Time: 04:15)

Building Memory

- Q: Once saved how do we retrieve it back?
It will be always available on output Q lines as long as WR is low!
- Input (new data) is to be given to D output (stored data) is available on Q
- Q: We want same data bus to be used for retrieving the data from register and putting the data into register. How can it be achieved?
- We need some control before we connect Q and D to data bus

The diagram shows three D flip-flops connected to a common data bus. Each flip-flop has a 'D' input and a 'Q' output. The outputs of the flip-flops are connected to a shared data bus. A 'WR (write enable)' signal is shown as a pulse that goes low to enable writing. The flip-flops are labeled with '1' and '0' on their D inputs, and '1' and '0' on their Q outputs. The slide also includes the NPTEL logo and a calendar widget showing 'NPTEL Des Mechtr Today at 3:30 PM'.

So let me turn off this. So let us say if you have this value to be retrieved from the same data bus, value to be retrieved to the same database and from that same data bus I want to save say some other value to store. So, I want to use same symbol, single data bus or data wires, which are kind of giving the value to the memory and which are taking the value out of memory.

And so, for this, we need to have this Q and D terminals connected to the data bus. But now if they are both connected directly to the data bus that not going to be there, they are shorted, if they are shorted together then they will not function well, the value will be messed up with the feedback or some some kind of other circuit will happen there.

So, so, we want to have that control over whether the value is to be stored into the, into the register or value is taken off the register and given to the data or data bus. So, that control we can achieve by using some kind of a small element called tri-state kit.

(Refer Slide Time: 05:51)

Fundamentals of

Tri-state gates

Z state is similar to case of open switch

OE: Output Enable

It is different from input connected to either 1 (3v) Or 0 (0v)

Construct truth table:

OE	I	F
0	1	Z
0	0	Z
1	0	1
1	1	0

Inverting Tristate gate
See that output F is inverted Version of input I

CMOS implementation of a tri-state gate

So, let us look at what is this tri-state kit. It has three states, instead of only two states, it has third state which is see here which is high impedance kind of a state. So, we will, we will not worry about what is that state for now, it is just a state where the input is, the output is such that it is into high impedance kind of a level and that helps in in not draining the current.

So, if the impedance is low the current typically will get drained, but now, if it is a high impedance state, it will not drain the current. So, it is, that is a kind of a state that we have here. So, the truth table for this is now we have this kind of a special gate which is called tri state gate, which has a input here and it is controlled by using this output enable kind of a pin and then it is getting reflected to output F.

So, in this case, this is a circuit we do not worry about how the circuit is constructed and why like it is like that, but one can get into details of this, this is kind of a CMOS circuit which is realize this tri-state gate. Which is inverted, inverted kind of a tri-state gate, because output enable is 1, input is 0, output is 1. So, one can use some kind of a NOT gate here to make sure that input and output has the same value. So, you can imagine that, this this gate is such that this input value will be on on the output only when output enable is enabled, or it is 1.

(Refer Slide Time: 07:31)

The slide features a green background with a white gear icon in the top left corner. The title "3-bit Register" is written in a large, bold, white font. Below the title, there are three bullet points in white text. At the bottom of the slide, there is a circuit diagram showing two rows of three D flip-flops each, labeled 'A' and 'B'. Each flip-flop has a 'D' input, a 'Q' output, and a 'CK' (clock) input. The 'Q' outputs of all flip-flops are connected to a vertical bus labeled 'DATABUS'. The 'D' inputs of the flip-flops are also connected to this bus. The 'CK' inputs of all flip-flops are connected to a common horizontal line. The NPTEL logo is visible in the bottom left corner.

- We connect Q through tri-state gate to the data bus as shown. Output of Q will go on databus only on enabling the tristate gate
- So think if all registers are connected in similar way to databus can it work and how?
- Where D should get connected?

So, this is a gate, we will make use of in this construction now. So, this is constructed in this kind of fashion, with this we connect this Q to the data bus through this tri-state gate. So, whenever we want this value to be available on the data bus, we enable these pins of the tri-state gates all pins connected together, the way you have these clock inputs connected together like that, we will connect all these inputs together and then that, when that is enabled, all these data values will be available on respective lines of the data bus.

So, this is how I can transmit my data, from a register to a data bus. Now, I want to get the data from the data bus to the resistor, how do I connect? So, where this D should be connected? So, you might have guessed it right, pause and think about before you listen to the answer.

So, yes, the D lines can be directly connected to data bus, because there is no conflict because unless, so lot of, lot of these data buses will have a lot of these data which is continuously changing, but that data will not disturb this register, because unless a clock goes high or this input goes high, this D value cannot be stored into the, into the register.

(Refer Slide Time: 08:56)

8-bit Register

- Now we can create an 8 bit register with tri-state gate in similar way
- OE (output enable) and WR (write enable) are additional inputs
- OE and WR both Cannot be 1 for the same register at the same time: Think what will happen if so??
- Feedback and data clash

8-bit register

OE

WR

D7-D0

D7 D6 D5 D4 D3 D2 D1 D0

Q

D

CK

WR

OE

NPTEL

82

So, now we have we are slowly kind of building up now to see how the data can be controlled in and out of the registers. So, this is the same principle now we can extend for getting the 8-bit register created. So, in the very similar way, we can put 8 kind of a D flip flops together and create a 8-bit kind of register here.

So, so, this has all these clock inputs connected together to this right pin and all these input to the tri-state gates connected together to the output enable and you see this data versus directly connected to the D pin and and the Q is connected through the tri-state gate to the data bus. This is how a circuit for the 8-bit register would look like.

So, so now, henceforth will just like look at this. So, you can imagine that see, how each of these, each of these parts can be seen at different scales. So, for example, if 8 bit register just in and just kind of a block diagram like that, it has inside these kind of a circuit. Inside the each of these D-flipflops there are these lot of these RS flip flop, AND gates and that kind of a logic blocks, each of those logic blocks will have CMOS transistors working at the base.

So, like that one can imagine like this is getting now scaled up further. So, so, this is how, our actually our computer gets built like it has all these elements at PC at the at the user level if you kind of start digging in more details what is at a hardware will have you will find all these kinds of features to be there at a hardware level.

So, now this variable referred henceforth this register just as this small little block with output enable pin and WR pin, this output enable when it goes high the data from this 8 bit register

will be available to the data bus here and when right pin is high, then the data from the data bus will be written to this register that is what is the, they both cannot be high at the same time because then it can, if you see if they both are high then these data clash will be there, it is like shorting the Q to the D, which is like having some kind of a data clash.

(Refer Slide Time: 11:19)

8-bit Register

- If we want to store data available on databus WR needs to be enabled for finite time with OE being zero
- If we need to transfer data from register to databus OE needs to be given a pulse (with WR=0) → READ operation
- With this data can be exchanged between registers through databus think how?

D7-D0

8-Bit register

OE

WR

D7 D6 D5 D4 D3 D2 D1 D0

Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

CK

WR

OE

NPTEL

83

So, from here we move on to how do we kind of store the data and retrieve the data. So, if you want to store the data again as I said, WR pin needs to get enabled for some finite time with output enable being 0. So, now you intend to make sure the output enable is 0 and WR is made high for some small pulse kind of is given on the, on the WR pin and then your data will be stored.

Now, if you want to transfer the data from the register, now, this register to the data bus, we need to enable output enable and we need to have of course WR to be low. And then the read operation will happen, that we are reading the value from this register and that is available on the data bus.

(Refer Slide Time: 12:20)

8-bit Register

- Transfer of data from register A to B
 - OE of A should be high so that the data is available on data bus
 - During the same time (with OE_A high) WR of B should see a pulse (recall edge-triggered) at completion of which WR is again disabled
 - OE of A should go low
- Also when OE of one register is enabled OE of no Other register should be enabled.
- WR of several registers can be enabled at a time (storing same data in many registers)

Diagram showing two 8-bit registers connected to a data bus (D7-D0). Register B is above Register A. Both have OE and WR inputs. Bidirectional arrows connect each register to the data bus.

NPTEL 84

Now, this data can be exchanged between the registers through this data bus. So, now we can think how this can happen. So, you have these two registers here, and this register A and register B and from here I want to transfer the value to this register. From register A, I want to transfer the data to register B. So, from the, from the A I should first transfer this data to the data data bus and from the data bus I should transfer the data to the register B.

So, so, I will transfer the data from from register A to the data bus by using output enable for the A register to be high. Now this this data is available here. Now at the same time I need to do this WR to be high for a short duration then that data will be stored in this register B and then I can make WR to be low. And now A's data is also there in the register B.

So is a A, data in the A register lost? No, we have not written anything to A register that data is still intact there, so like that this operation is going to happen. So, this is about likeness, so we have constructed memory elements. Now we will need to address this memory. So, we, how do we know like that, we are at this register or this register? There will be like thousands and millions of registers like these available.

(Refer Slide Time: 13:57)



The slide features a green background with a white gear logo in the top left corner. The title 'Addressing Memory Registers' is centered at the top in a bold, white font. Below the title, there is a list of four bullet points in white text. In the bottom left corner, there is a small NPTEL logo. In the bottom right corner, there is a small number '85'.

- Recall memory construction using registers
- Q: How do we select a particular designated register for read or write operations?
- Think of scaling memory!!!
- Its like you have many rooms available and need to have address to specify something to be fetched from some specific room or stored in another specific room.

Now we need to address this register, we need to know which register I want to move the value from, I cannot give this like just some some arbitrary ABCD kind of a thing. We need to have some numbers or addresses given to the, to the registers. So, how do we address, created that address kind of a feature in the register, is what we need to see.

So, so, this was, this is like we have all these rooms constructed for the hostel. And students are to live in this hostel. But now, we do not know that room numbers are not there yet, that is a kind of scenario. So, you have this many rooms available but you do not have address available for those rooms. So, we cannot designate which student or which student is allotted which room for for stay.

So, to do that, now why, why this will be required? See if there are two, three rooms it is very easy, but if there are millions of rooms like if you want to scale this up to millions of registers, where you have say some megabytes of memory then this becomes a issue. So, you want to give this address, these addresses are given as a binary numbers, some binary number is given as the address then I should designate that this goes to this register.

So, that means like I need to have some binary number representing when that number happens on some some some other lines which are called address bus then I am particularly referring to that particular kind of a register. So, how that happens? So, let us see.

(Refer Slide Time: 15:19)

n 8-bit Registers

■ A huge memory can be formed by putting together the elemental registers (this is RAM, why??)

■ Q: How to address it? Meaning how to access data (read or write) in one of the locations?

■ Ans: Assign address to each of the locations Then what?

8 bit Data bus

D7-D0

1

2

3

n

OE WR

OE WR

OE WR

OE WR

NPTEL

PRASANN S GANDHI

So, let us say we have this n 8-bit registers, a huge memory can be formed like that, this is random access memory that we have here. So, so, now to address it, so, so, these are like so many kinds of registers connected together and then this is a 8-bit data line which is available. Why this 8-bit data line or why there are 8 lines here? Because this is a 8-bit register.

So, this register has 8-bit, one register has 8-bit memory locations, like that you can have 16-bit data also, we can have 32-bit data also like that you have that kind of scaling possibility. Now, to address give give address to each of these registers, to address that we need to create some more kind of logic here.

(Refer Slide Time: 16:16)

The slide features a green background with a white gear icon in the top left corner. The title 'Addressing memory location' is written in a bold, white font. Below the title, there is a diagram of a 2-to-4 decoder. The decoder is represented as a blue box with two input lines labeled 'A1' and 'A0' at the bottom. Four output lines labeled 'I0', 'I1', 'I2', and 'I3' extend from the top of the box. To the right of the diagram, there are four bullet points in white text. The first bullet point states: 'Say we have 4 registers so the address can be from 00, 01, 10, and 11'. The second bullet point asks: 'Now we need to select one of them at a time either for reading data (OE=1) onto data bus or writing data (WR=1) from data bus. How?'. The third bullet point explains: 'Use decoder logic: given the address, the specified output line goes high'. The fourth bullet point poses a question: 'Q: how do we connect outputs of decoder and respective OE and WR pins of registers??'. In the bottom left corner, there is a small NPTEL logo. In the bottom right corner, the text 'PRASAWNA S GANDHI' and the number '87' are visible.

Addressing memory location

- Say we have 4 registers so the address can be from 00, 01, 10, and 11
- Now we need to select one of them at a time either for reading data (OE=1) onto data bus or writing data (WR=1) from data bus. How?
- Use decoder logic: given the address, the specified output line goes high
- Q: how do we connect outputs of decoder and respective OE and WR pins of registers??

So, say we say we have a 4 registers, just for an example. And the addresses are 00, 01 they are binary addresses, four binary addresses addressing these 4 registers. So, we need to select one of them at a time to address or to take a value from that register, to write a value from that register, to write a value to that register.

For that we need to select one at a time. How do we do that? So so, say we first create a bus called address bus. So, on address bus, we will get these numbers available, 00, if 00 is available, I need to address only the first register. So, how to address that, how do I address first register? By using this kind of logic called decoder logic. So, given address it gives particular line to go high.

So, this question is how do we connect now the output of this, so this is a next question. First how do we kind of like, what is this logic here? So, this logic will be something of this sort, where you have this address that is available here, say this is the address line here on which the address is available, when this address is available, then only one of these four pins will go high and each of these pins will be then used for either right or, right or output enable for that particular kind of register.

So how do we use that is a next question. But this is a base kind of requirement where we have the address given then like no one of the pins is going high. So, if we have four registers, then we need two addresses, address lines. If we have say 10 registers, think about

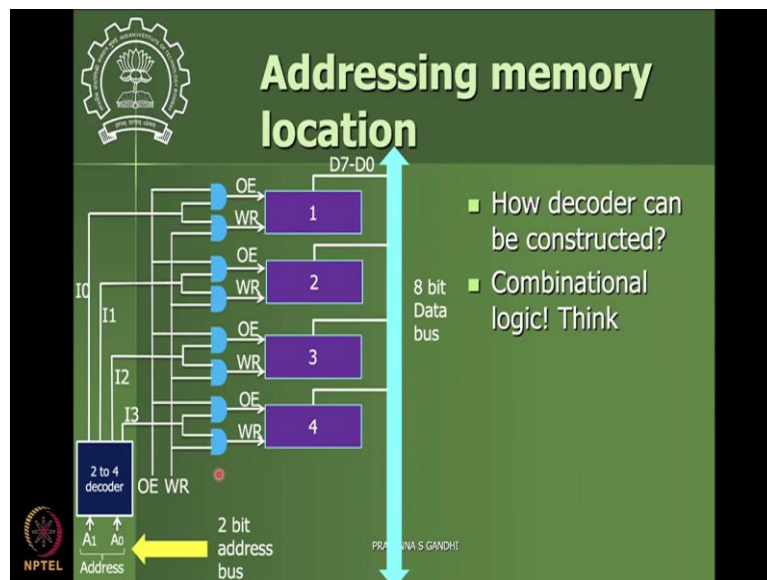
how many number of these address lines will be required. So, so, say 4 binary numbers will give us 16-bit kind of a representation.

So, so, 4 binary, so we can go up to 16 registers, for 16 registers to be addressed we need 4 data, address bits. So, irrespective of address and the data bits are irrespective of each other say I can have 3-bit data or I can have 8-bit data, but number of registers is important for address. If I have 10 registers, then like no that determines like what is the address.

But if I have say 4 registers now, of only 2-bit kind of a memory, I still need this for, the two lines for addressing like that you need to think about. So, so, the data capacity or data bits in one register is independent of number of registers that I am using and number of address bits that will be required. So, so, now, how do we connect these, this connection is is shown here.

So, you think about how do we connect this output of the decoder to respective OE and WR pins of registers, so that we can address them, so that OE and WR cannot be 1 at the same time. And we are able to address whenever we want to write output, like when you have to write I do not have to kind of, I can do it in a very in a simple manner.

(Refer Slide Time: 20:07)



So, so, to do that, this is a way it can be connected, you can see that we have put some kind of AND gates here. So, this AND gate is connected to these each of these one of the pins of AND gates is, first AND gates are connected to both output enable and WR as the output of the AND gate is connected there, and then inputs are shorted to connect to this decoder.

So, that now I can address this particular memory location and depending upon what I write, what I put here this is now, my output enable one single pin, for all the registers, is one single pin will be there. So, I do not have now see if you remember like previously, we had multiple pins for this multiple output enable and WR pins.

So now I am creating some kind of additional feature to address them independently and write and read operations are kind of controlled independently. So, so, you address this register and then read and write will happen only on this register. If you address some other registers for this, by the same pins output enable and WR, read and write will happen on different register now.

So, this is how one can think about, creating these kind of features that we would like to have in the system, so that we can address now the data in each of these registers and address that particular register and then get the data out of that register or write the data into that register. And and how do we exchange the data, by using this data bus.

(Refer Slide Time: 21:55)

Decoder logic

- Truth table

A0	A1	I0	I1	I2	I3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- Expression

$$I0 = A_0' A_1'$$

$$I1 = A_0' A_1$$

$$I2 = A_0 A_1'$$

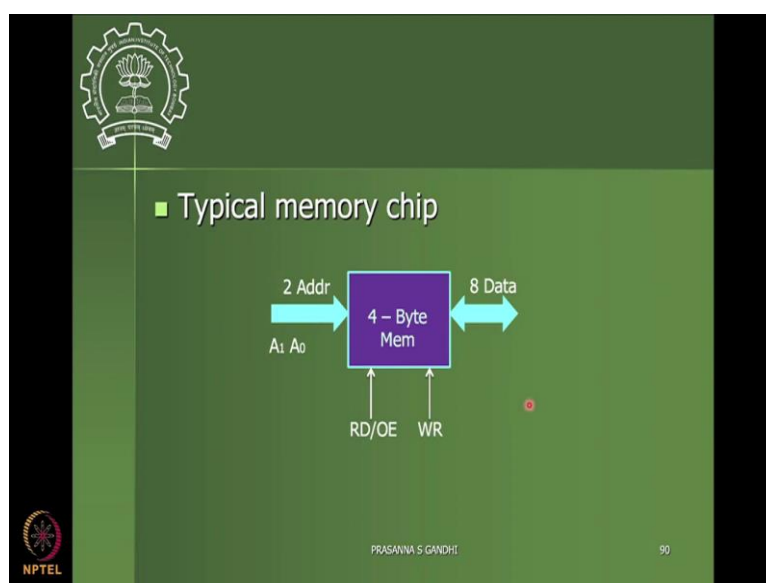
$$I3 = A_0 A_1$$

- Circuit???

The slide also features a diagram of a 2-to-4 decoder with inputs A1 and A0, and outputs I0, I1, I2, and I3. The NPTEL logo is visible in the bottom left corner, and the text 'PRASAWNA S GANDHI' and '89' are in the bottom right corner.

So, we will not worry about what is a decoder logic that is constructed, you can think about that, I am skipping these slides for that. So, decoder logic will have some kind of a truth table and expressions are given here.

(Refer Slide Time: 22:04)



So, typical memory chip will look something of this sort in terms of some kind of a architecture kind of diagram, it will have say address bus and it will have a data bus and it will have read or read or output enable kind of a pin and it will have a write kind of a pin. So, now we have a 4-byte memory. And this has 8-bit data registers.

So, then it will have eight lines for the data and then two address lines. So, because it is a 4-bit, 4-byte kind of a memory. Like that if you have like larger and larger memory size, you can see how much like how much, how long the data per second need or if you have say eight lines of the address bus, eight lines of the address bus then how many maximum number of registers can be there in the memory.

So, eight, eight lines, means you can have 256 registers in that memory. So, eight-line address bus and whatever number of data, data bits we do not know, we can have any number of data bits does not matter. Once you fix the address, number of address bus lines, then number of registers get fixed in the memory. So, so, this is a main understanding for these from the memory perspective.

(Refer Slide Time: 23:38)

The slide features a green background with a white gear logo in the top left corner. The title 'Addressing memory location' is written in a bold, white font. Below the title is a diagram of a memory chip labeled 'Mem'. The chip has an 'addr' input on the left and a 'data' output on the right. Below the chip are two blue AND gates and two white NOT gates. The first AND gate has inputs from the 'addr' bus and a NOT gate connected to 'CS'. The second AND gate has inputs from the 'addr' bus and a NOT gate connected to 'WE'. The outputs of these AND gates are connected to the 'RD' and 'WR' pins of the memory chip. The 'RD' pin is connected to the first AND gate, and the 'WR' pin is connected to the second AND gate. The 'CS' and 'WE' pins are connected to the NOT gates. The text 'PRASAVNA S GANDHI' and 'NPTEL' are visible at the bottom of the slide.

Addressing memory location

- Sequence of operations: say for writing data from external register to specified address location in memory Think!!
- 1. Place address on address bus
- 2. Bring CS' low
- 3. Bring WE' low
- 4. Place data from ext register on data bus (how?)
- 5. Bring WE' back to high
- 6. Reverse operations

So, then there are these different different kinds of variants that will be available. So, now, the question is, how do we sequence, some sequence of operation if used to be carried out, how do we carry out that? So, this is one of the examples. So, for writing data from some external register to the specified address location in the memory, this is the operation that is given.

So, what do you think about this, writing is the data from the external register to a specified address location in the memory? So, you need to place some address on the address bus. What does it mean, is that address bus lines are like now 01 according to the address that is needed there. So, so, physically that is, that is the meaning like address bus has some lines are at 0 volts and some lines are at a higher volt, 5 volts or 3 volts depend upon what is the voltage level used.

So, that will create some kind of address on the address bus. Now, that particular register can be addressed. Then you want to kind of have data on the, on the data bus coming from the external register. So, so this so, you select these chips. So, there are some some additional features that will be there, where you, there are multiple chips that are there and you want to select one of the chips then you can select this chip for the, for the operation to be done.

Then this this write enable is there. So, it is where you place the, bring this also to to low, that means like you enable the write operation. So, you can have these three and four can be

interchanged also, you can place the data from the external register on the data bus and then this bring it to low or and then bring it to high again, that is one possibility.

Or you can have this basically three and four can exchange with each other. So, and then this write enable is again closed or disabled that means the data is now in the memory. So, from external register it is now locked into the that particular address location in the memory. So, these are, this is kind of one of the sequences of operation that can happen.

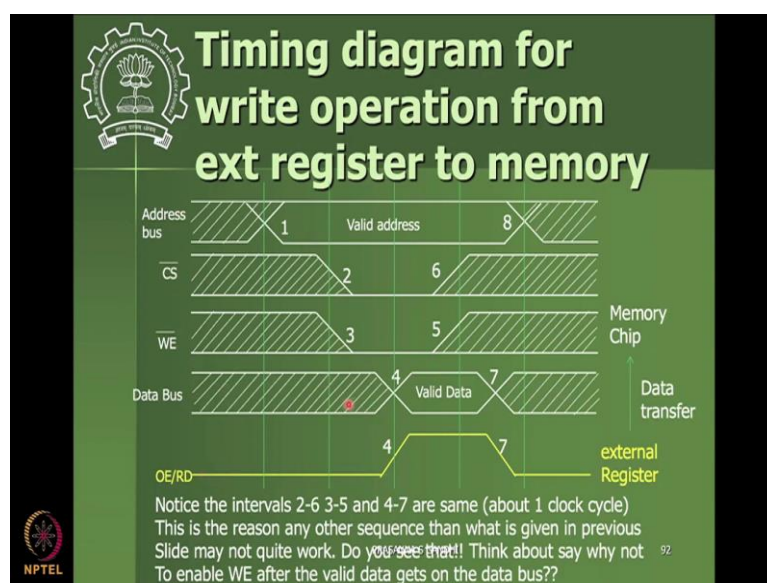
Like that one can think of many different kinds of possibilities for like doing some kind of a, say we want to move the data from one of the memory locations to other memory location, what is the operation, like that one can start thinking about different different sequences of these operations that can be possible.

And, when you start writing your program, or that program gets executed, this is what happens at the, at the hardware level. How does it happen? We do not worry about that right now. We will see some part, we see some glimpses of it in the, in the, in the next class maybe. So, but right now, you see that, these kinds of operations can be performed with different kinds of like now operation sequence of operations can be performed by using this, this steps for executing that into, into the, into your system.

And the basic thinking philosophy here is this, this helps us to develop our understanding about how things are, things could be happening at a base hardware level. So, again, I am giving you some kind of a thing that, this is not, this understanding is not really a must kind of a thing for you to start programming microcontroller.

Anyway, we are programming our microprocessors, computer is programming, this is not really necessary, but if you have this understanding, this will help you think better in terms of your programming. So, that is why we are going to this this little bit more details about how things happen at the hardware level.

(Refer Slide Time: 28:11)



So, so, now, let us see this is a, this entire thing can be represented in terms of some kind of a timing diagram. So, this timing diagram is is given like this. So, this address bus is given as as like, we are now, using each, each line what is a value and thing like that, we are just representing entire address bus as a one kind of a signal where from this point in time to this point in time 1 to 8 kind of numbers are like steps of the time that are given, all the time this valid address is there on the address bus.

Then this chip select will go low, chip select bar bar will go low, that means chip select will be happening here in this domain and say WR, this right write enable is happening here in this domain and and data becomes available, valid data will become available from this 4 to 7 or from this time duration here.

So, so, these numbers are written as let us say different different kinds of a step that is going to happen. So, first step you will have this in time happening, then second in time happening, third in time happening, then there will have valid data available when you, when you have this external register output enable is enabled on that and then that data will be taken off that register.

But before that you will have the WE going high that means the data is kind of locked. So otherwise, what will happen is if this is still low and some some some data is changed here or some valid, invalid data comes this is some kind of invalid data in this domain comes here

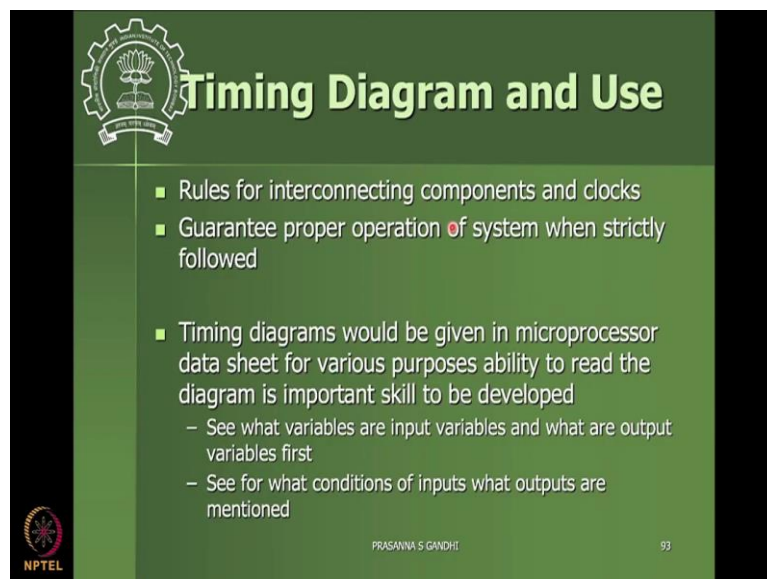
and still this is, this operation is not completed yet. Then that data may get stored in the, in that particular address, instead of whatever valid data that we want you store.

So, this is how typically the timing diagrams are given and our ability to read these timing diagrams is important. How do we think? We think our first, what are the inputs and what are the outputs. So, so, here like for example, this output enable is an input given to this external register and as a result of this what is happening on the data bus is this. So, like that this is this is input, this is output.

Then you need to see this chip select and this, this address bus, these all are inputs only. So, this is what we are giving as inputs and only like whatever is available in the data bus is the output because, like that we are assuming that data bus is connected to the external register and when external register is made output enable high then the valid data is available on the data bus.

So, like this, the these are through these operations, this data from the external register is getting transferred to this whatever address that you have, addressed put in on the address bus that particular address of the memory this data will get transferred to that particular register. So, these are the operations that are happening, as they are supposed to happen if you want to transfer the data.

(Refer Slide Time: 31:45)



Timing Diagram and Use

- Rules for interconnecting components and clocks
- Guarantee proper operation of system when strictly followed
- Timing diagrams would be given in microprocessor data sheet for various purposes ability to read the diagram is important skill to be developed
 - See what variables are input variables and what are output variables first
 - See for what conditions of inputs what outputs are mentioned

NPTEL PRASANNA S GANDHI 93

Now, how do we make this happen is, for this you need a timing circuit, or timing diagram, or timing, additional kind of a timing inputs. So how do, how do we do that? We will see it in

the next class along with some of the primitive kind of a microprocessor constructed so that you can get a glimpse of how this whole thing are, such a nice kind of a way they are interconnected and made to function to get us the functionality that that we may want as a user of microprocessor or microcontroller.

So, so we will see that in the next class and will not get into too many details about how things happen, but these are the base kind of a elements that we are putting together to just see, this is, based on these elements now, your entire microprocessor architecture and all the interfaces are built based on on this kind of a basic kind of a building blocks or components.

So, so, to just give you this glimpse, this is order this is not completely necessary for us to program or understand mechatronics, this, if this is understood, we are very well equipped to do the job in the, in the different scenarios. So, we will stop here and we will move on to the next class.