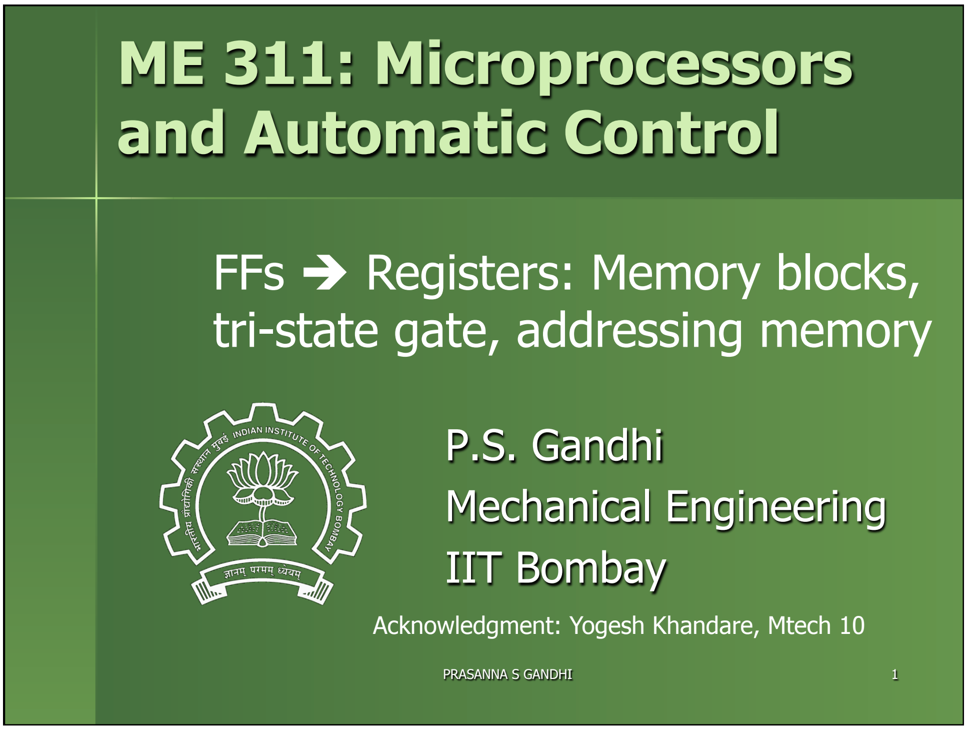
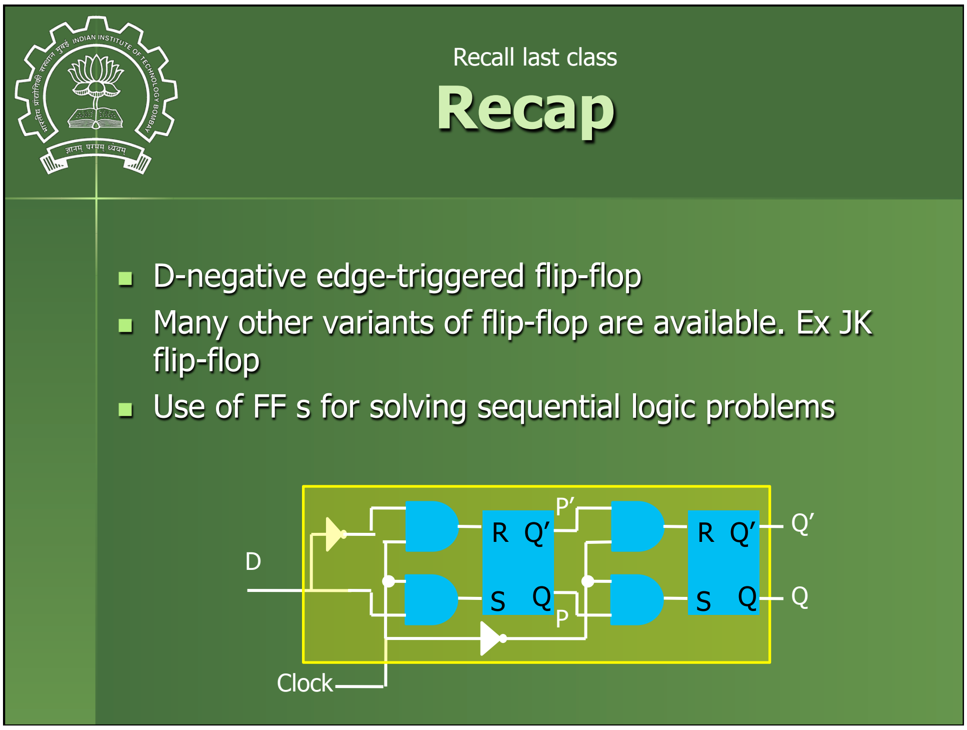


ME 311: Microprocessors and Automatic Control

FFs → Registers: Memory blocks, tri-state gate, addressing memory

P.S. Gandhi
Mechanical Engineering
IIT Bombay

Acknowledgment: Yogesh Khandare, Mtech 10



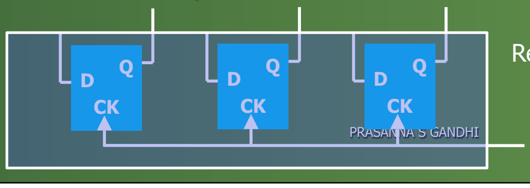
Recall last class

Recap

- D-negative edge-triggered flip-flop
- Many other variants of flip-flop are available. Ex JK flip-flop
- Use of FF s for solving sequential logic problems

Building Memory

- 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



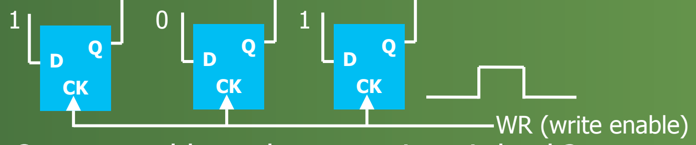
Register

Notice clock signal for all D Flipflops is shorted but Not connected to clock

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?

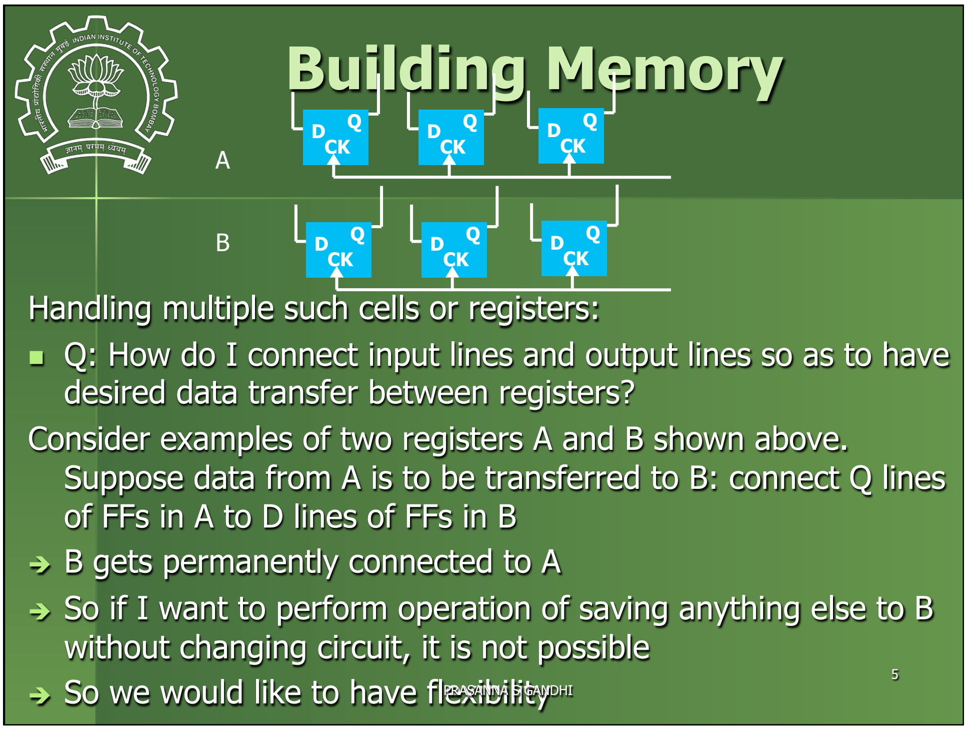
Have it ready on D lines as shown below and have signal pulse given to clock line named as write enable



- 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



Building Memory

Handling multiple such cells or registers:

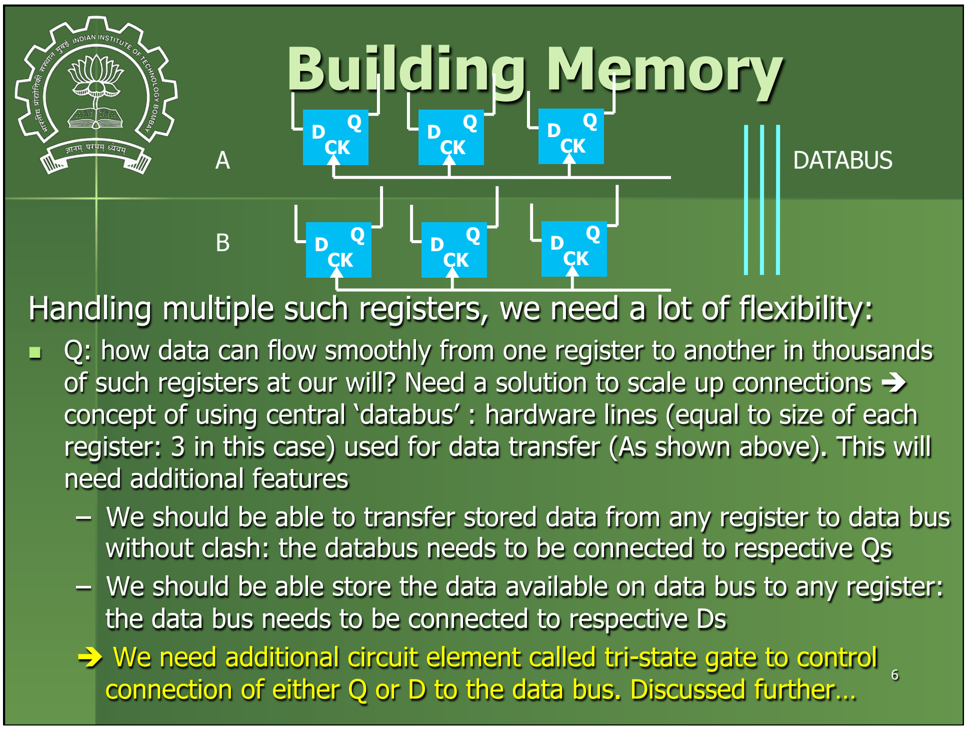
- Q: How do I connect input lines and output lines so as to have desired data transfer between registers?

Consider examples of two registers A and B shown above. Suppose data from A is to be transferred to B: connect Q lines of FFs in A to D lines of FFs in B

➔ B gets permanently connected to A

➔ So if I want to perform operation of saving anything else to B without changing circuit, it is not possible

➔ So we would like to have flexibility

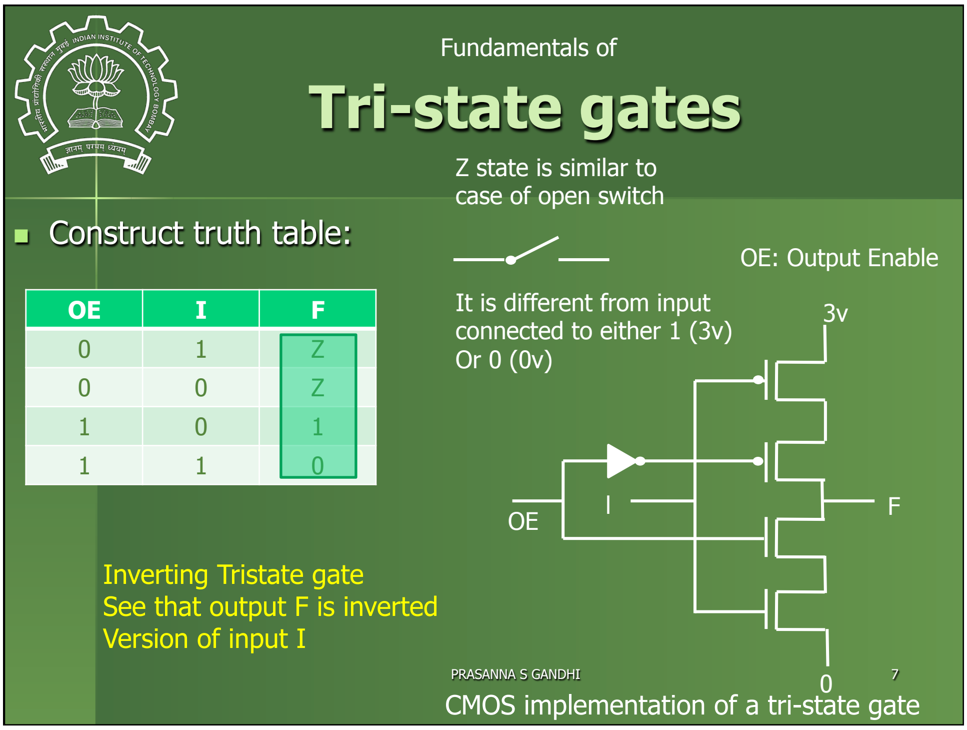


Building Memory

Handling multiple such registers, we need a lot of flexibility:

- Q: how data can flow smoothly from one register to another in thousands of such registers at our will? Need a solution to scale up connections ➔ concept of using central 'databus' : hardware lines (equal to size of each register: 3 in this case) used for data transfer (As shown above). This will need additional features
  - We should be able to transfer stored data from any register to data bus without clash: the databus needs to be connected to respective Qs
  - We should be able store the data available on data bus to any register: the data bus needs to be connected to respective Ds

  ➔ We need additional circuit element called tri-state gate to control connection of either Q or D to the data bus. Discussed further…



Fundamentals of

Tri-state gates

- Construct truth table:

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

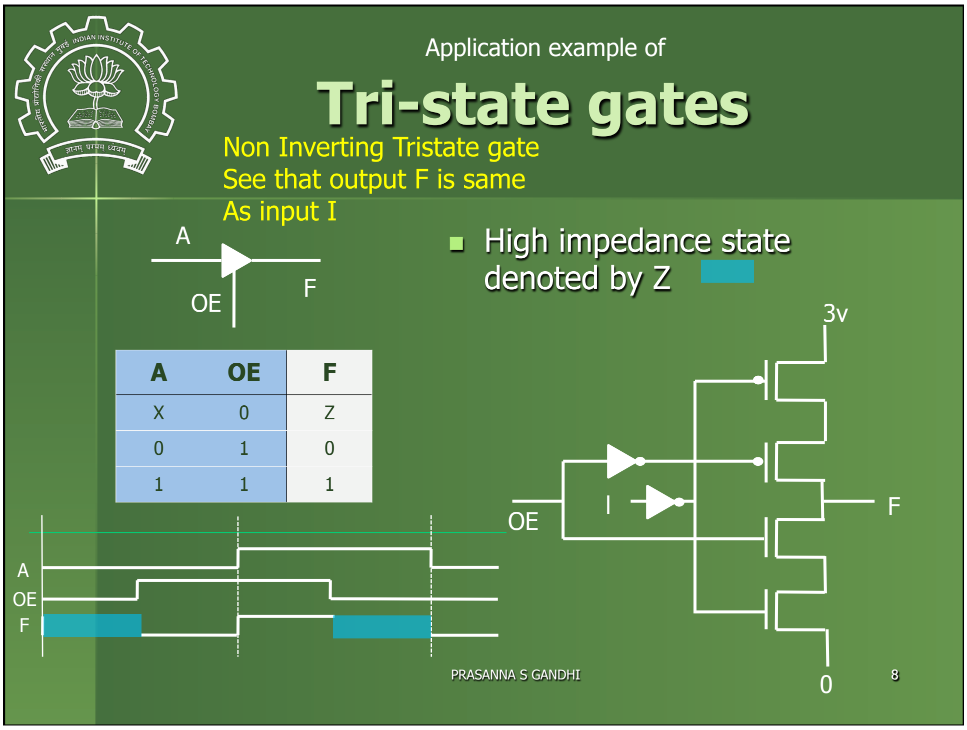
Z state is similar to
case of open switch

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

OE: Output Enable

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

CMOS implementation of a tri-state gate



Application example of

Tri-state gates

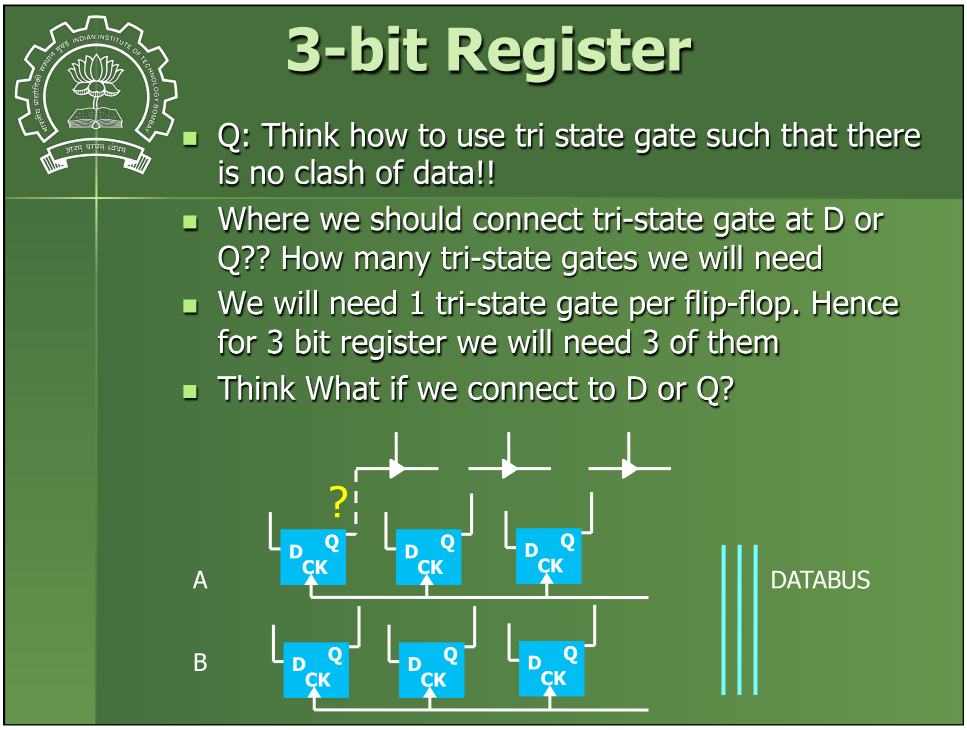
Non Inverting Tristate gate
See that output F is same
As input I

A
F
OE

A	OE	F
X	0	Z
0	1	0
1	1	1

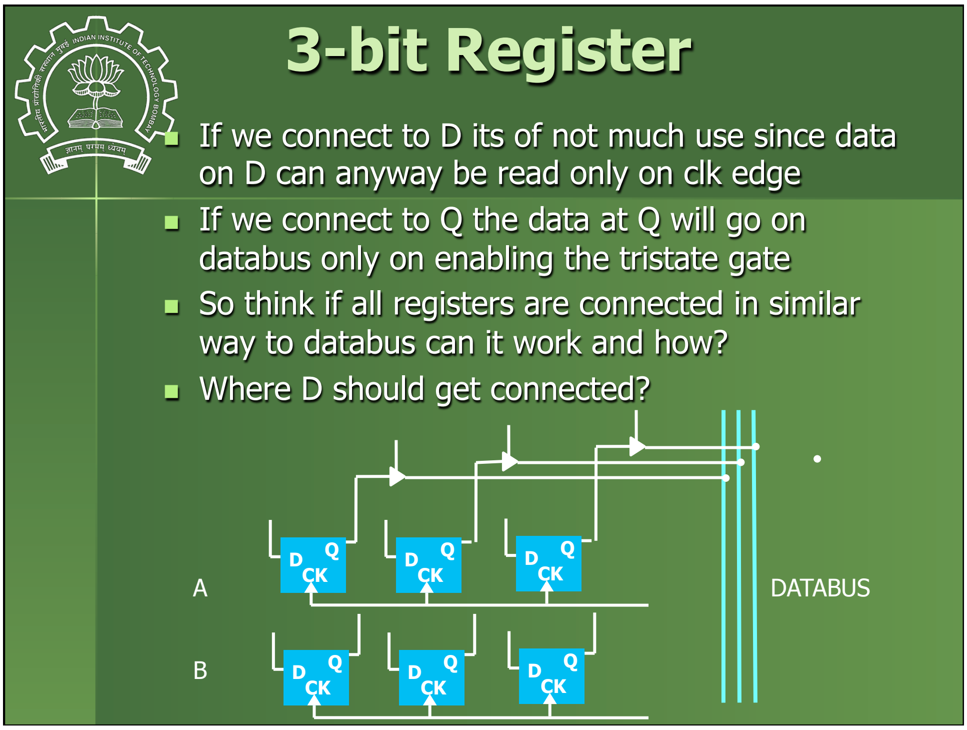
- High impedance state denoted by Z 🟦

3v
OE
I
F
0
A
OE
F



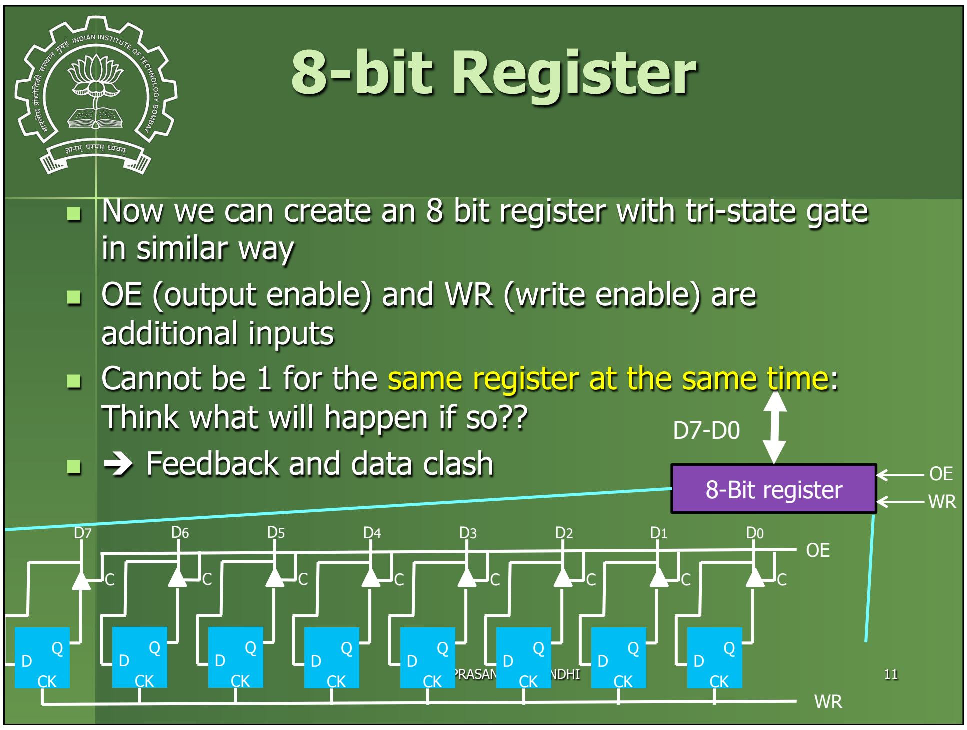
3-bit Register

- Q: Think how to use tri state gate such that there is no clash of data!!
- Where we should connect tri-state gate at D or Q?? How many tri-state gates we will need
- We will need 1 tri-state gate per flip-flop. Hence for 3 bit register we will need 3 of them
- Think What if we connect to D or Q?



3-bit Register

- If we connect to D its of not much use since data on D can anyway be read only on clk edge
- If we connect to Q the data at 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?

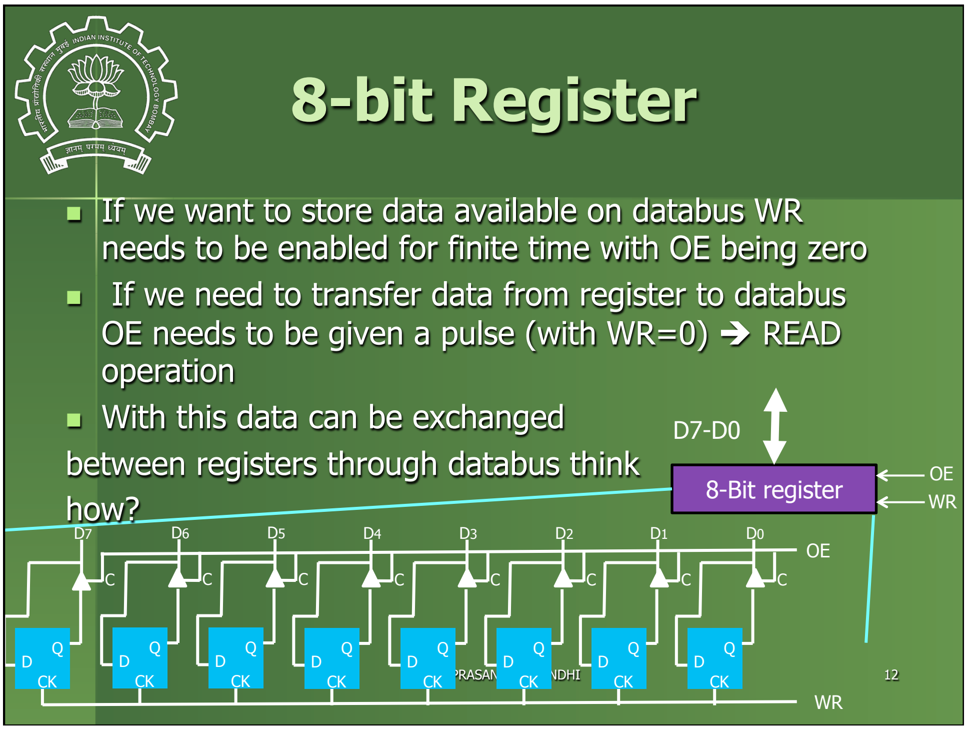


8-bit Register

- 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
- Cannot be 1 for the same register at the same time: Think what will happen if so??
- ➔ Feedback and data clash

PRASAN NDHI

11



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

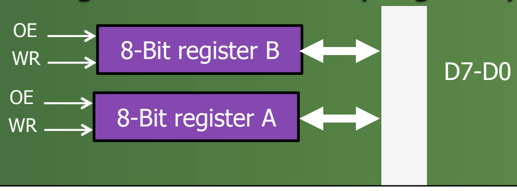
between registers through databus think how?

PRASAN NDHI

12

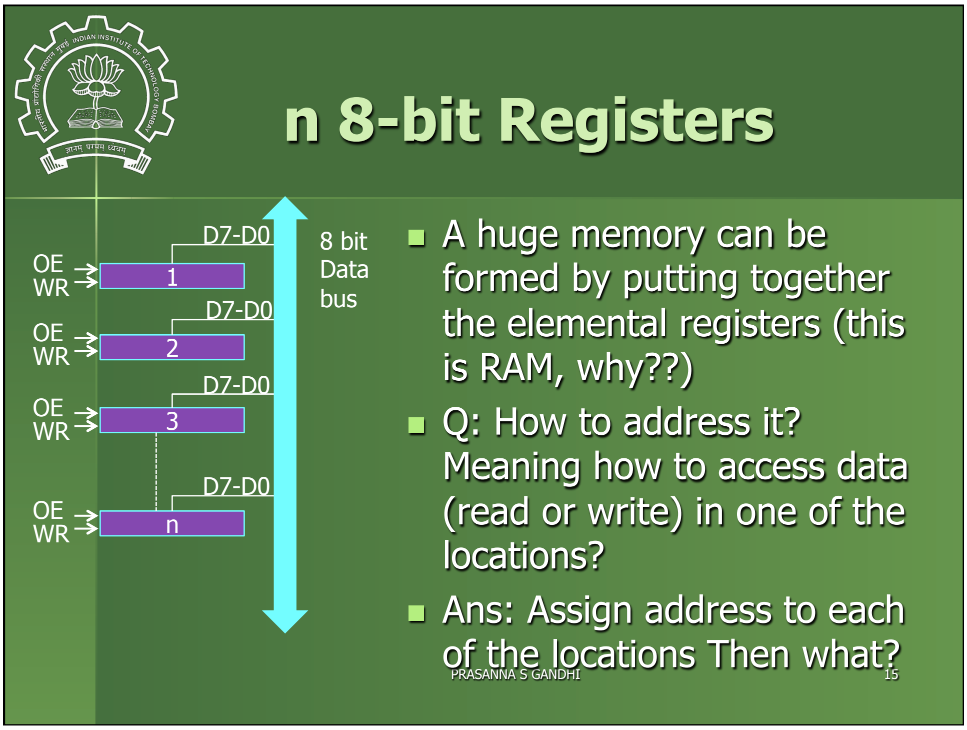
8-bit Register

- 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)



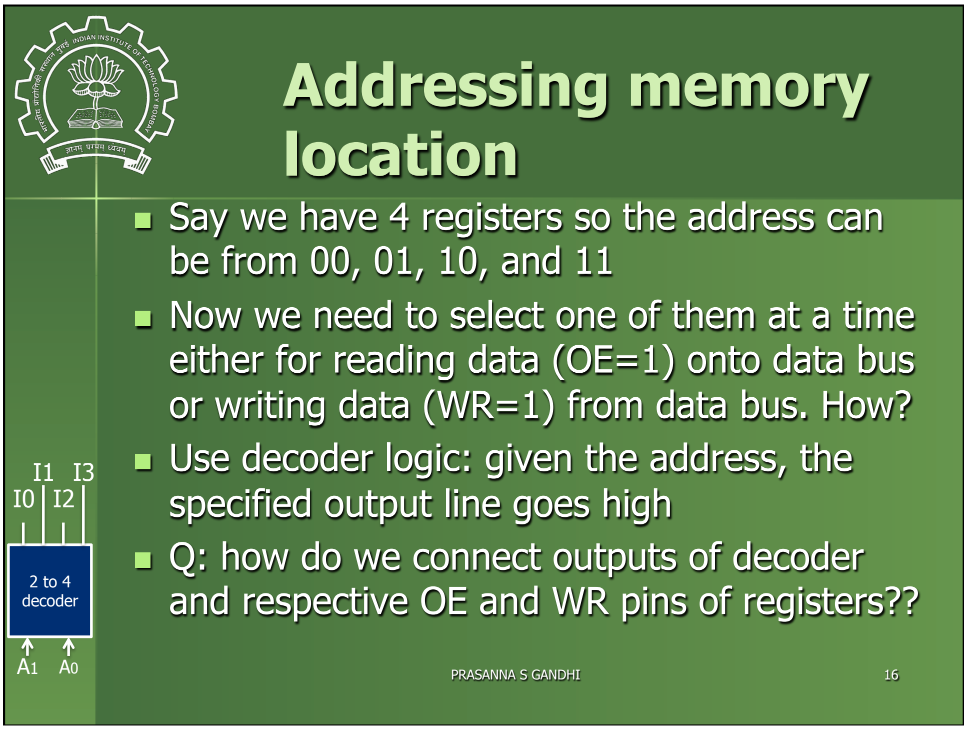
Addressing Memory Registers

- 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.



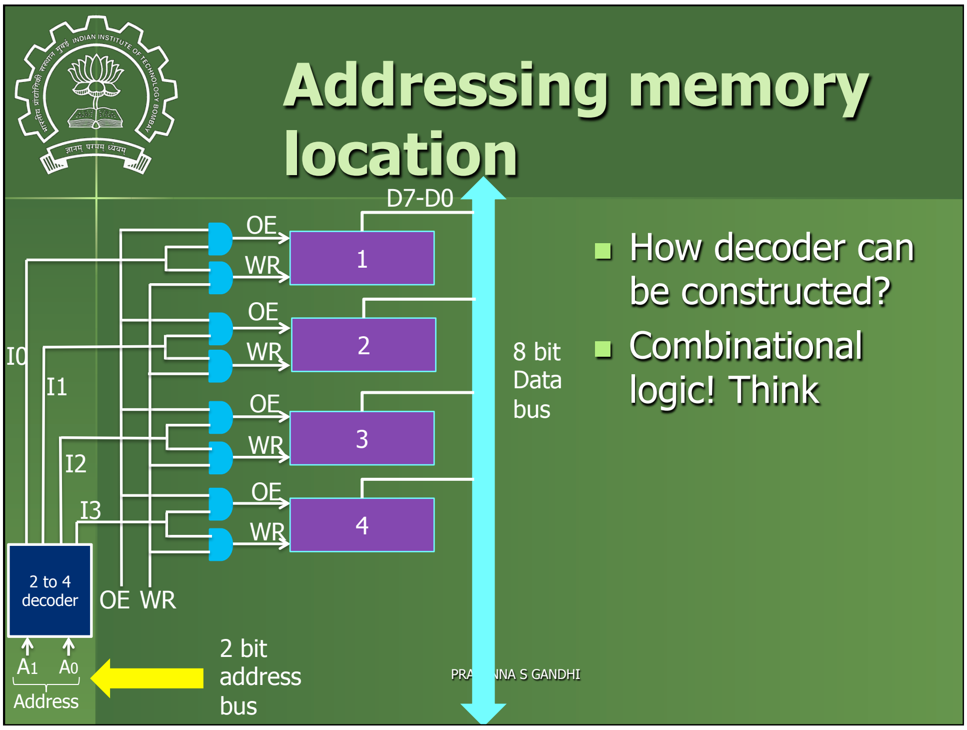
n 8-bit Registers

- 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?



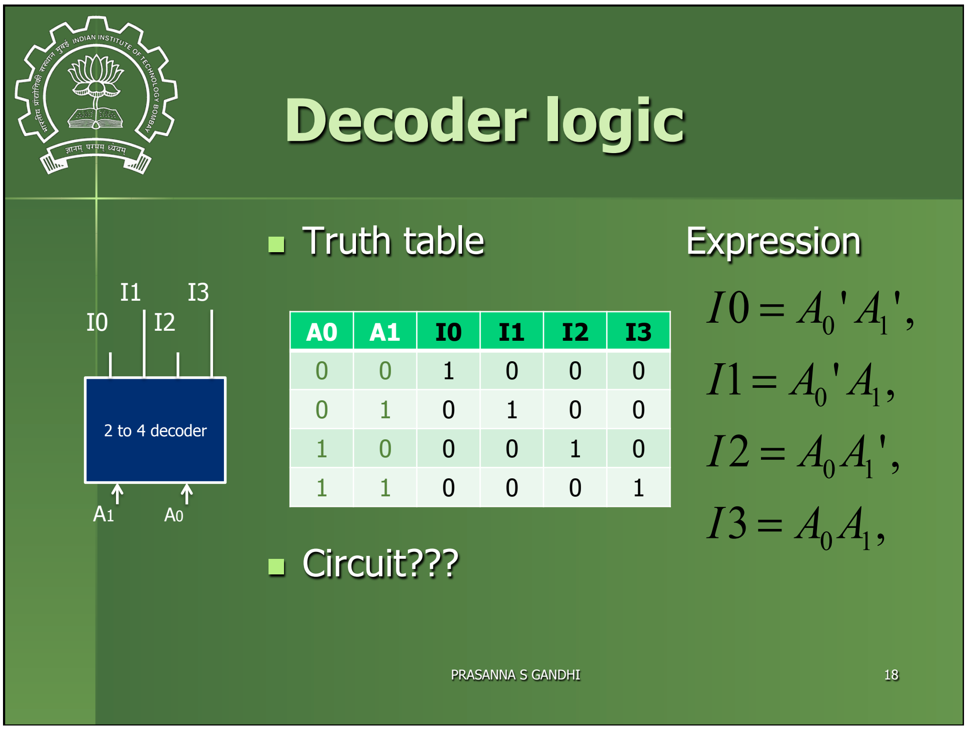
Addressing memory location

- 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??



Addressing memory location

- How decoder can be constructed?
- Combinational logic! Think



Decoder logic

I1 I3
I0 I2
2 to 4 decoder
A1 A0

- 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

- Circuit???

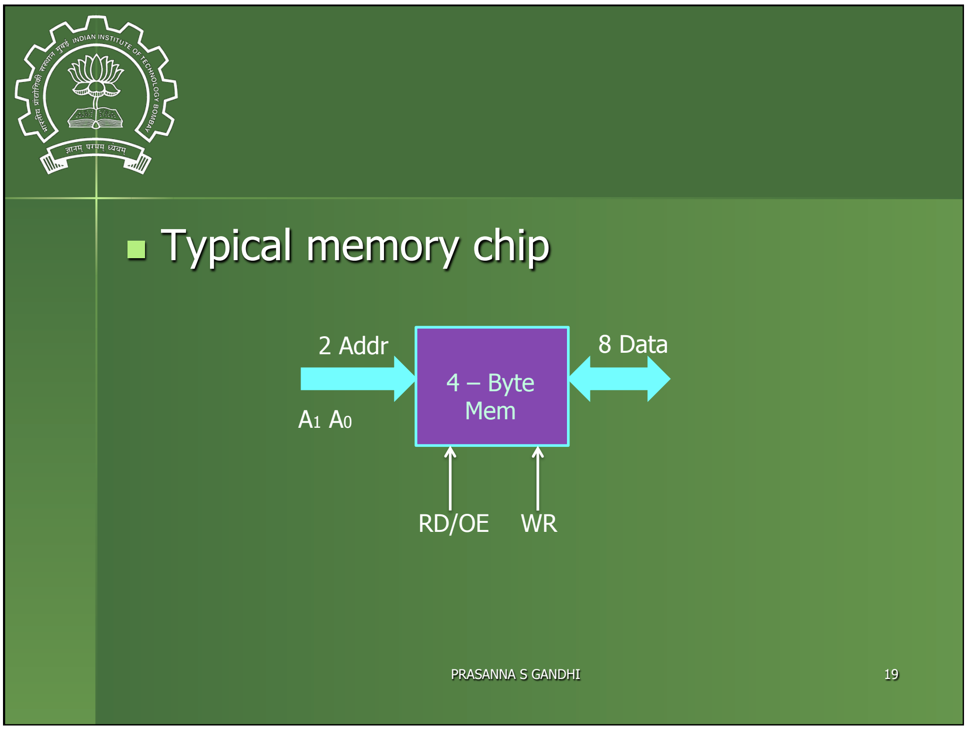
Expression

$$I0 = A_0' A_1',$$

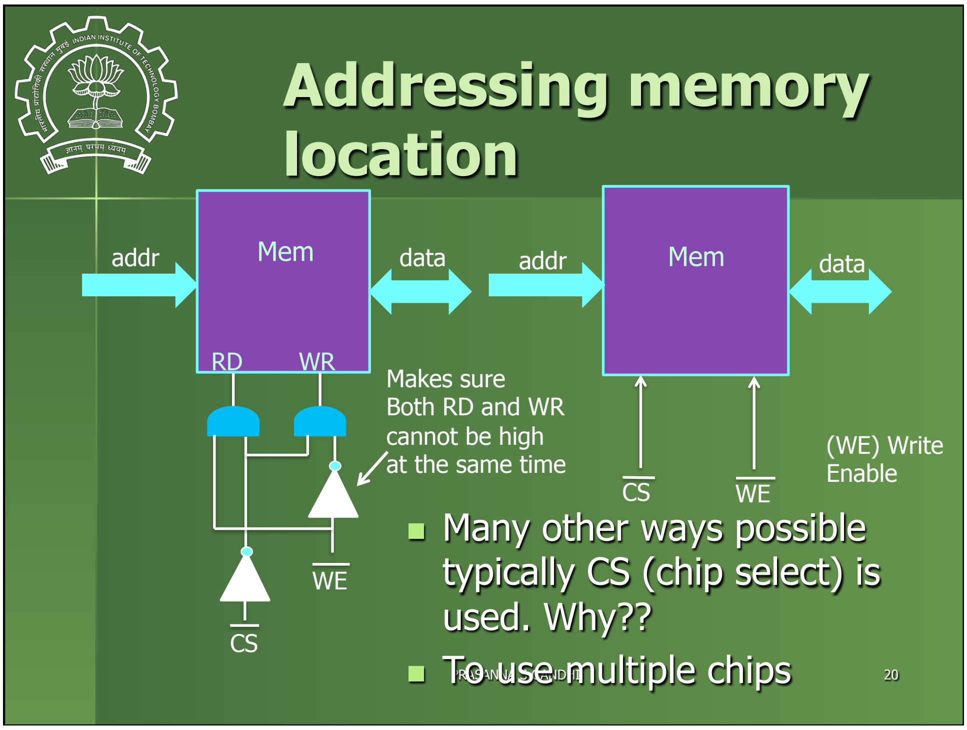
$$I1 = A_0' A_1,$$

$$I2 = A_0 A_1',$$

$$I3 = A_0 A_1,$$



- Typical memory chip

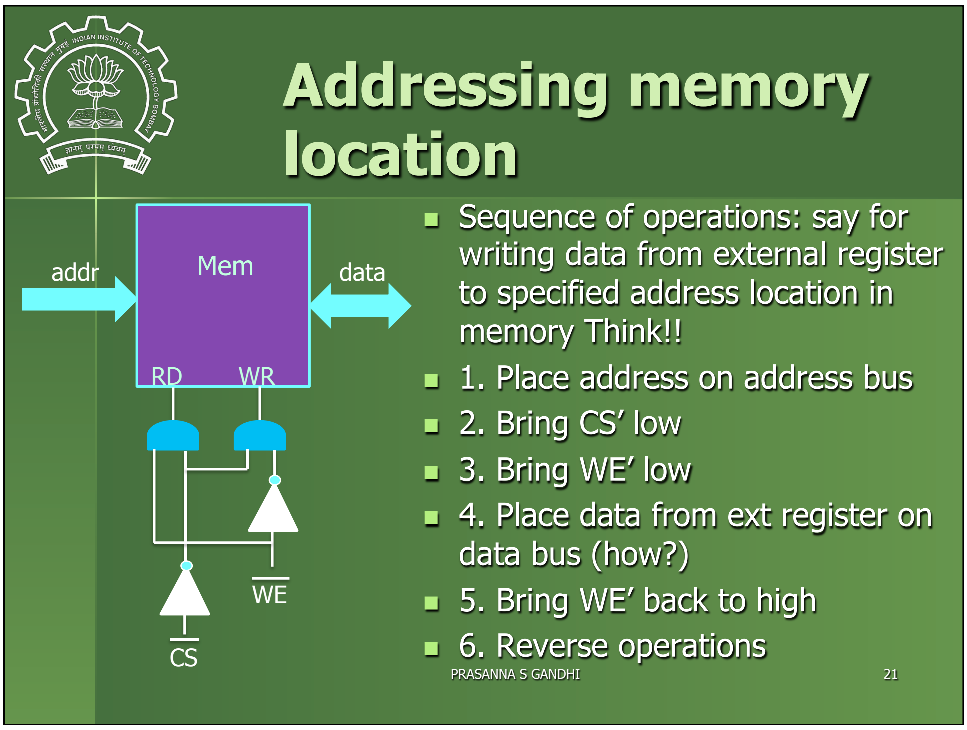


Addressing memory location

Makes sure Both RD and WR cannot be high at the same time

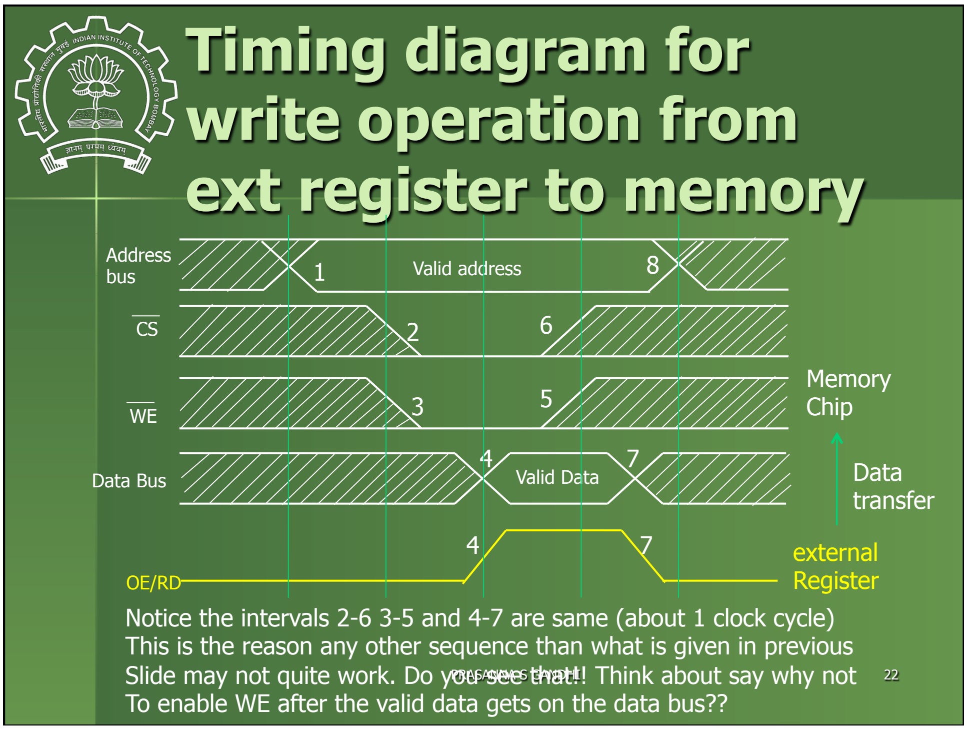
(WE) Write Enable

- Many other ways possible typically CS (chip select) is used. Why??
- To use multiple chips



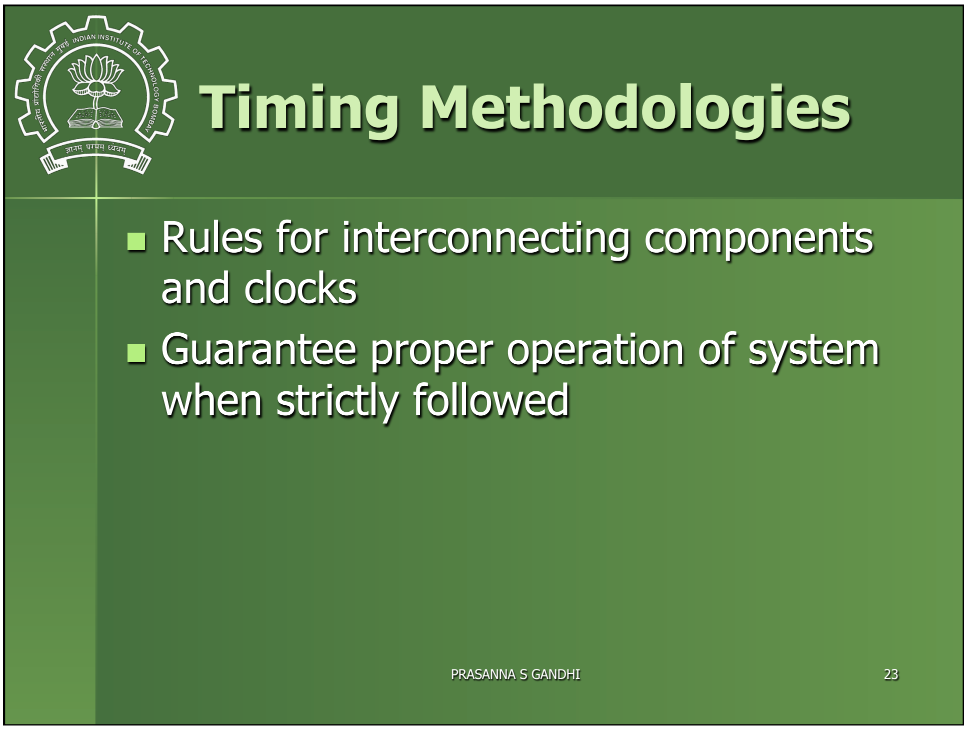
Addressing memory location

- 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



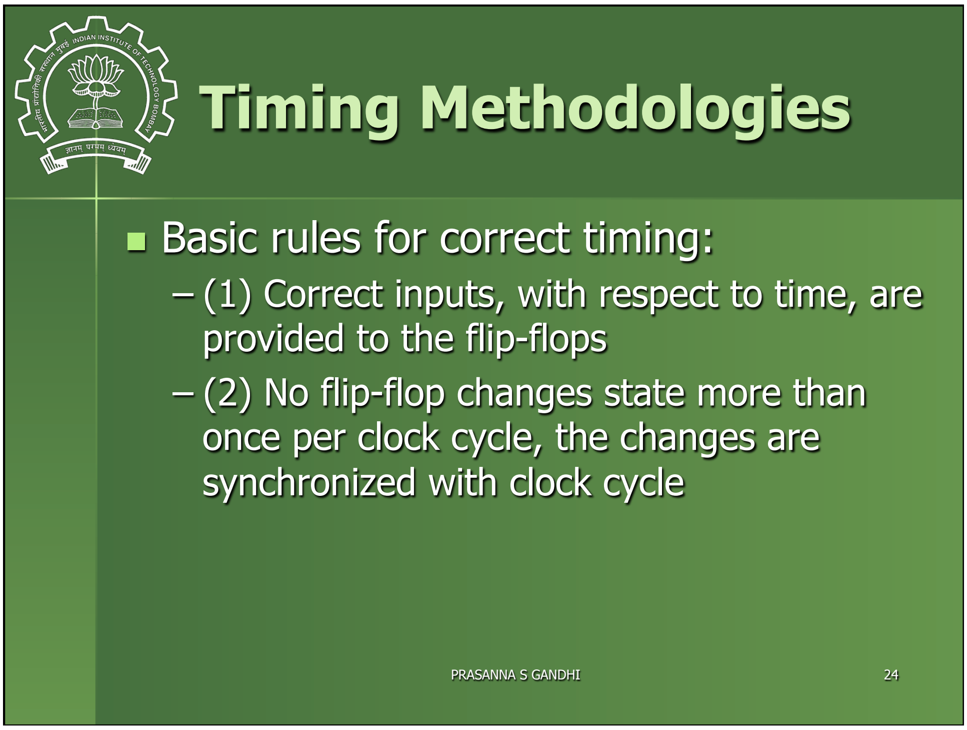
Timing diagram for write operation from ext register to memory

Notice the intervals 2-6 3-5 and 4-7 are same (about 1 clock cycle)
This is the reason any other sequence than what is given in previous
Slide may not quite work. Do you see that!! Think about say why not
To enable WE after the valid data gets on the data bus??



Timing Methodologies

- Rules for interconnecting components and clocks
- Guarantee proper operation of system when strictly followed



Timing Methodologies

- Basic rules for correct timing:
  - (1) Correct inputs, with respect to time, are provided to the flip-flops
  - (2) No flip-flop changes state more than once per clock cycle, the changes are synchronized with clock cycle

THANK YOU

PRASANNA S GANDHI gandhi@me.iitb.ac.in