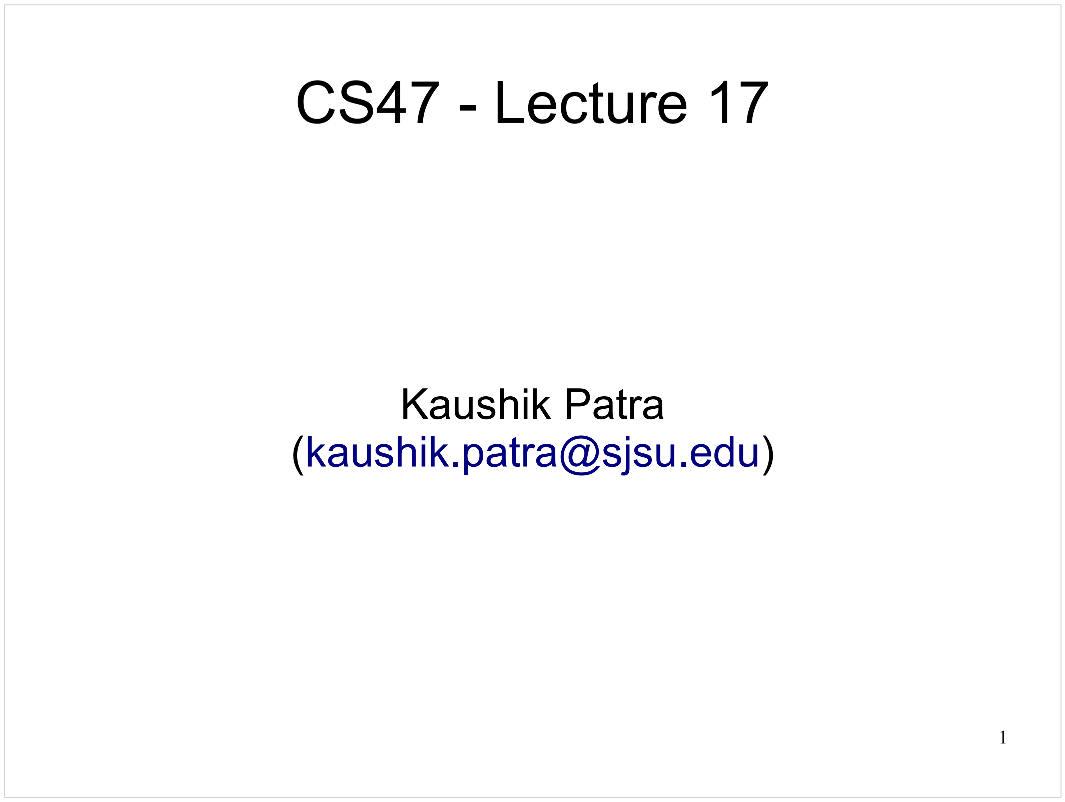
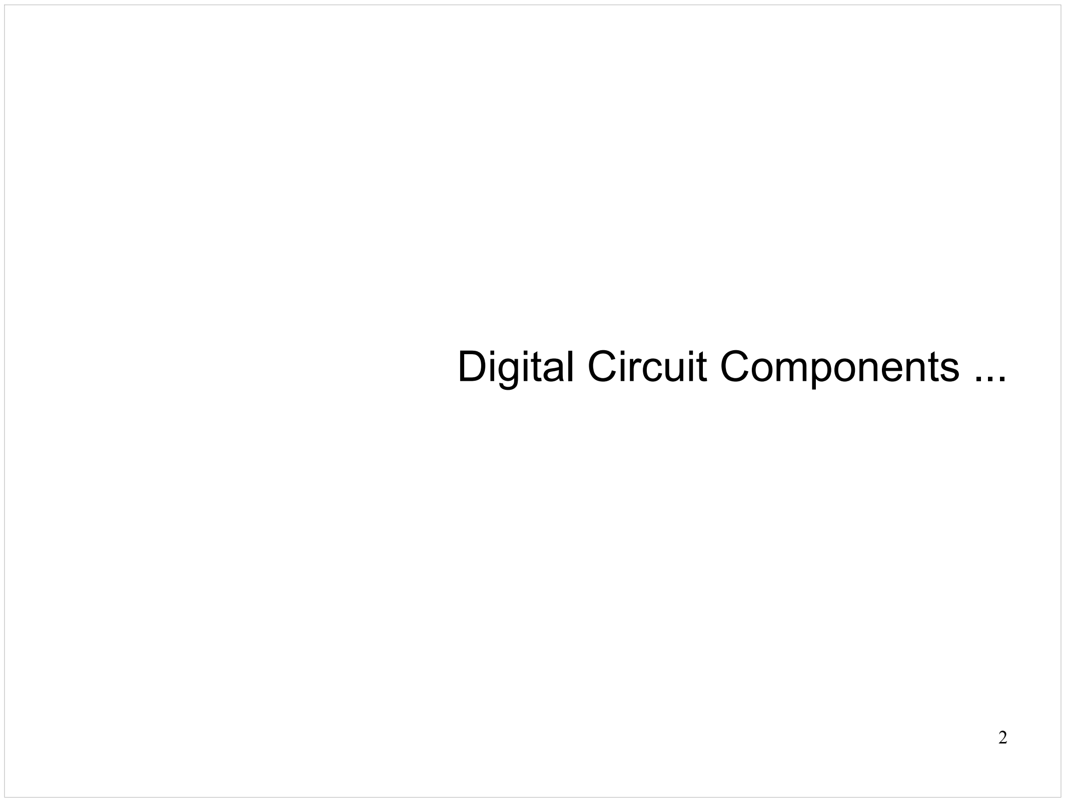


CS47 - Lecture 17

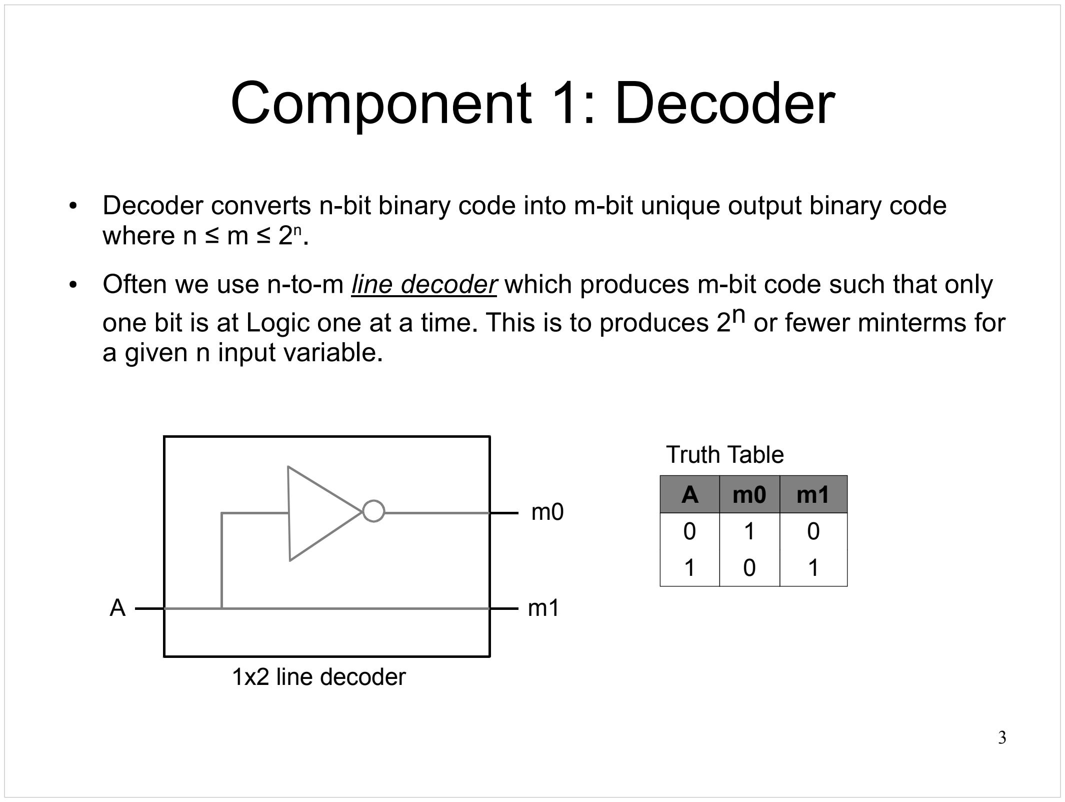
Kaushik Patra
(kaushik.patra@sjsu.edu)

- Digital Circuit Components

[Chapter 3, 5, 7, 9-4 of Logic & Computer Design Fundamentals, 4th Edition, M. Morris Mano, Charles R. Kime]

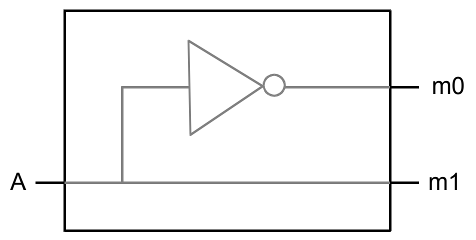


Digital Circuit Components ...



Component 1: Decoder

- Decoder converts n-bit binary code into m-bit unique output binary code where $n \leq m \leq 2^n$.
- Often we use n-to-m *line decoder* which produces m-bit code such that only one bit is at Logic one at a time. This is to produces 2^n or fewer minterms for a given n input variable.



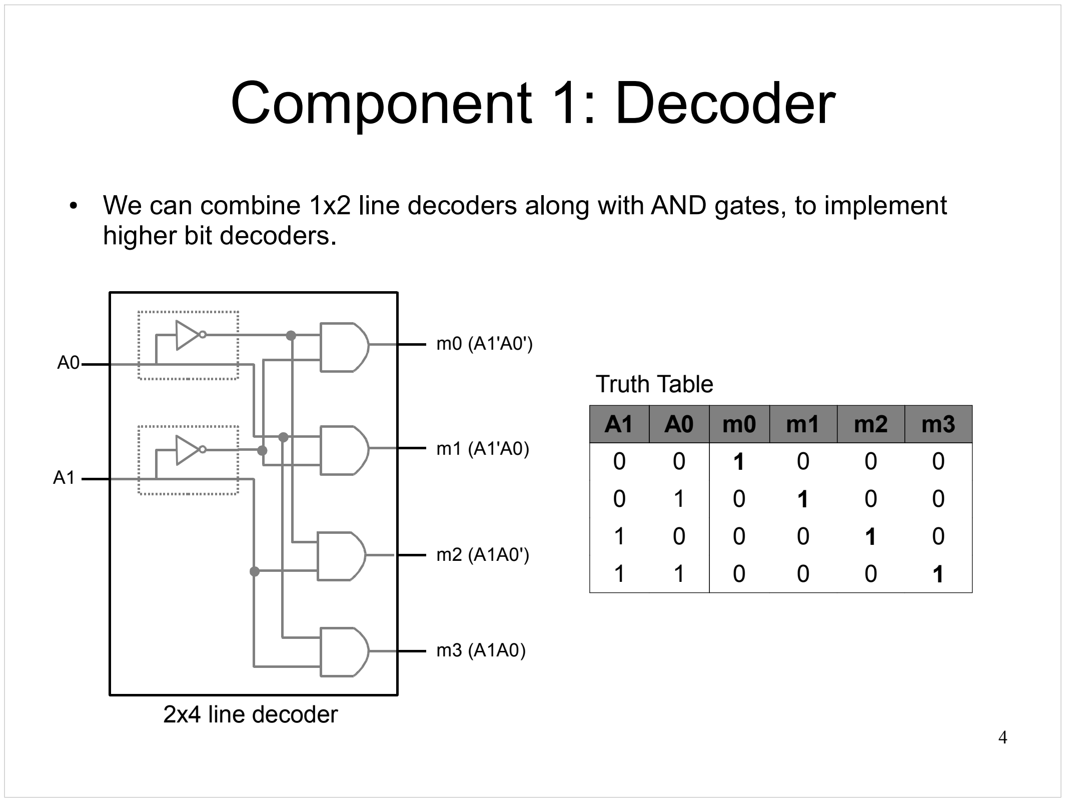
1x2 line decoder

Truth Table

A	m0	m1
0	1	0
1	0	1

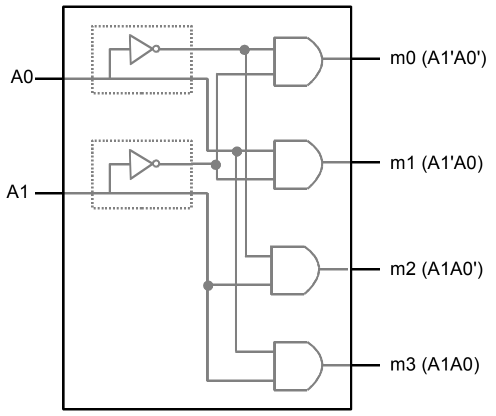
3

- For a 1x2 line decoder one inverter is sufficient to derive two minterms – A and A' in this example. If A=0, it is the m0 minterm and the decoder will give m0 a 1 and m1 as 0. On the other hand if A=1, it is the m1 minterm and the decoder will give m0 as 0 and m1 as 1.



Component 1: Decoder

- We can combine 1x2 line decoders along with AND gates, to implement higher bit decoders.



2x4 line decoder

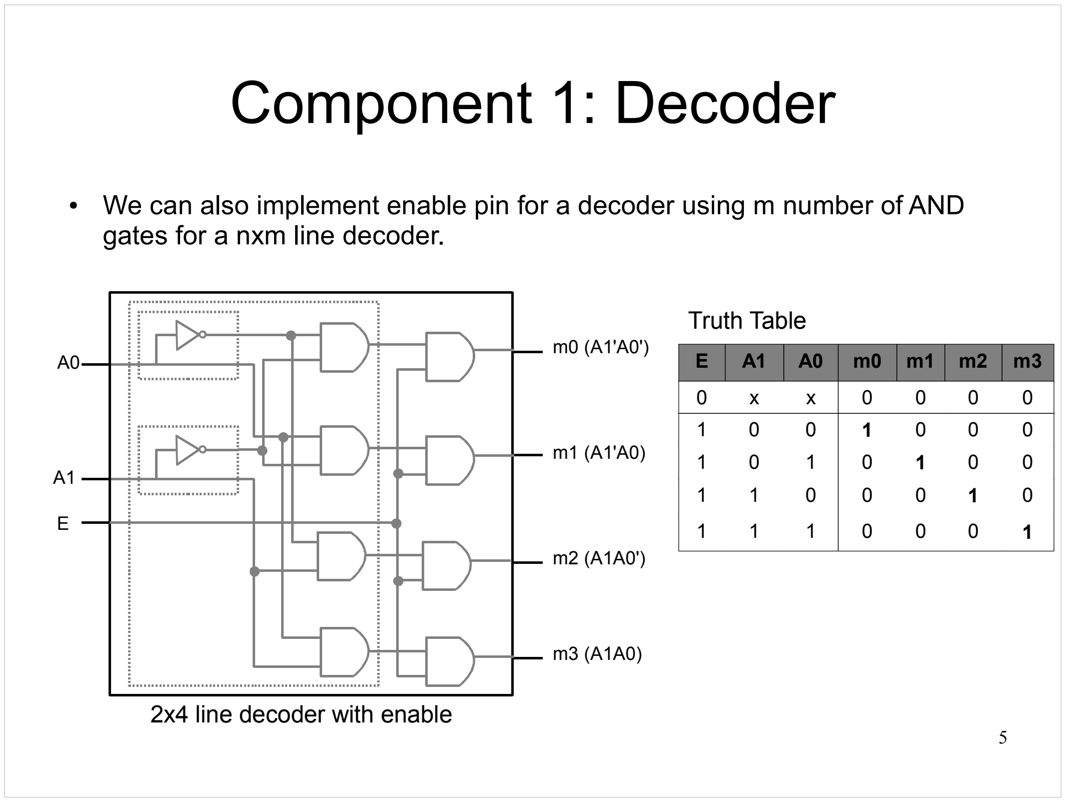
Truth Table

A1	A0	m0	m1	m2	m3
0	0	**1**	0	0	0
0	1	0	**1**	0	0
1	0	0	0	**1**	0
1	1	0	0	0	**1**

4

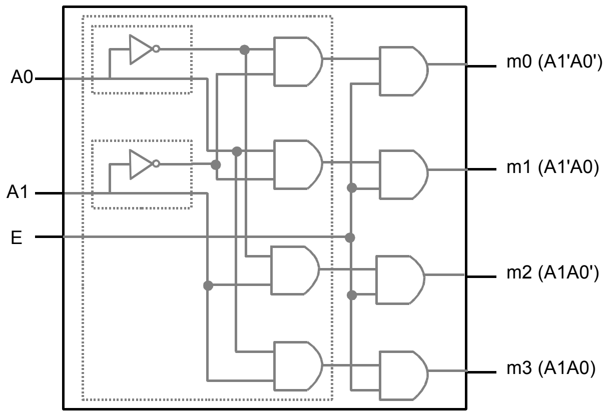
- The output of the 2x4 line decode is one-hot. This means out of 4 output only single output will be at logic 1 and all other will be at logic 0 for a given input. The output pattern is captured in the truth table.

- 2x4 line decoder is implemented using two 1x2 decoder. Each decoder will give the minterm for single variable connected to it. Hence two decoder will produce the term A0, A0' , A1, A1'. These four terms are then combined by four AND gates with get the terms A1'.A0', A1'.A0, A1.A0' and A1.A0. These are nothing but the minterms m0, m1, m2 and m3 of the combined Boolean variable A1 and A0.



Component 1: Decoder

- We can also implement enable pin for a decoder using m number of AND gates for a nxm line decoder.



2x4 line decoder with enable

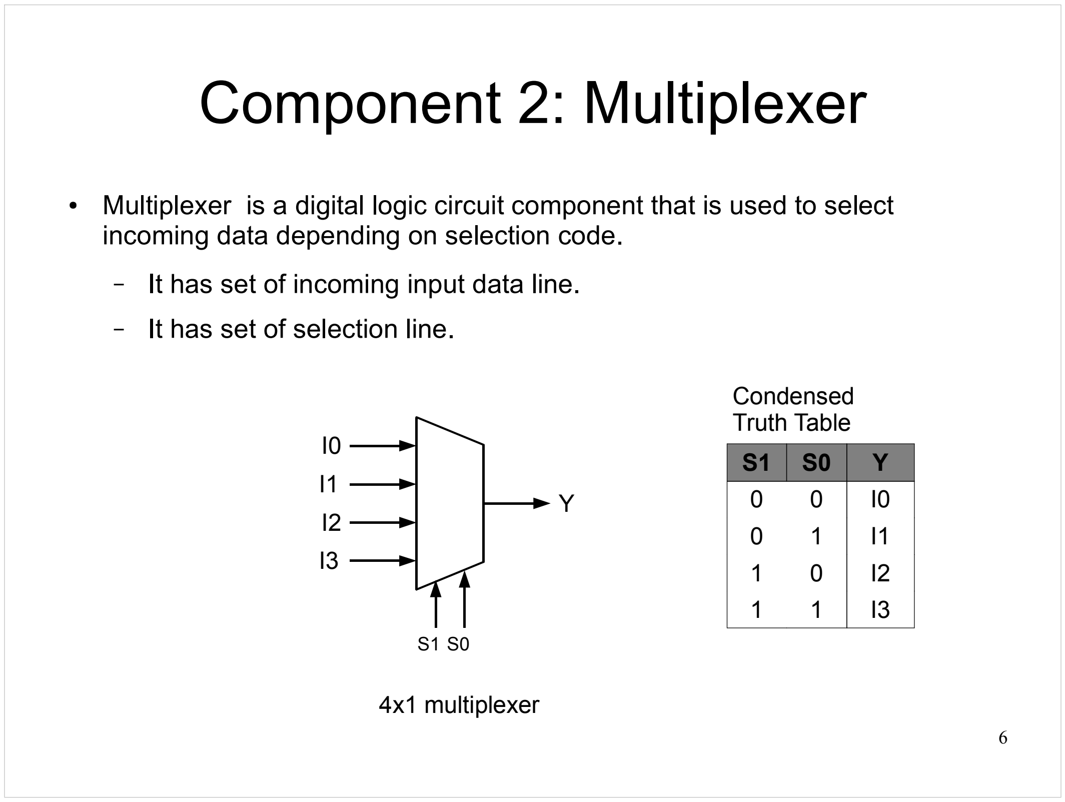
Truth Table

E	A1	A0	m0	m1	m2	m3
0	x	x	0	0	0	0
1	0	0	**1**	0	0	0
1	0	1	0	**1**	0	0
1	1	0	0	0	**1**	0
1	1	1	0	0	0	**1**

5

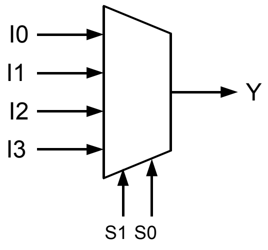
- We can cascade another four AND gates to implement enable signal E for the decoder. If E is 0, the output of the decoder should be all 0 no matter what the other inputs are. Once E is set to 1, decoder starts functioning and produce correct result at the outputs.

- The output of the decoder of the slide 13 is gated by signal E using AND gate. Once the E is one, the value of the decoding will pass to the final output. Final output will remain 0 otherwise.



Component 2: Multiplexer

- Multiplexer is a digital logic circuit component that is used to select incoming data depending on selection code.
  - It has set of incoming input data line.
  - It has set of selection line.



4x1 multiplexer

Condensed Truth Table

S1	S0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

6

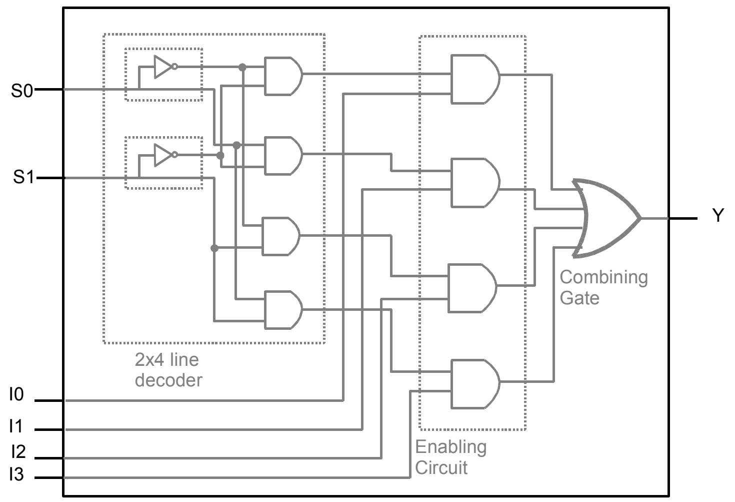
- Conceptually multiplexer implements decision algorithm. The example mux (short form of multiplexer) in this slide, implements simple case statement like following.

```
Case (S1 S0)
Begin
   00: Y = I0;
   01: Y = I1;
   10: Y = I2;
   11: Y = I3;
End
```

- This is one of the most important component in computer architecture. This gate enable controlling of right control flow and data flow through a computer.

Component 2: Multiplexer

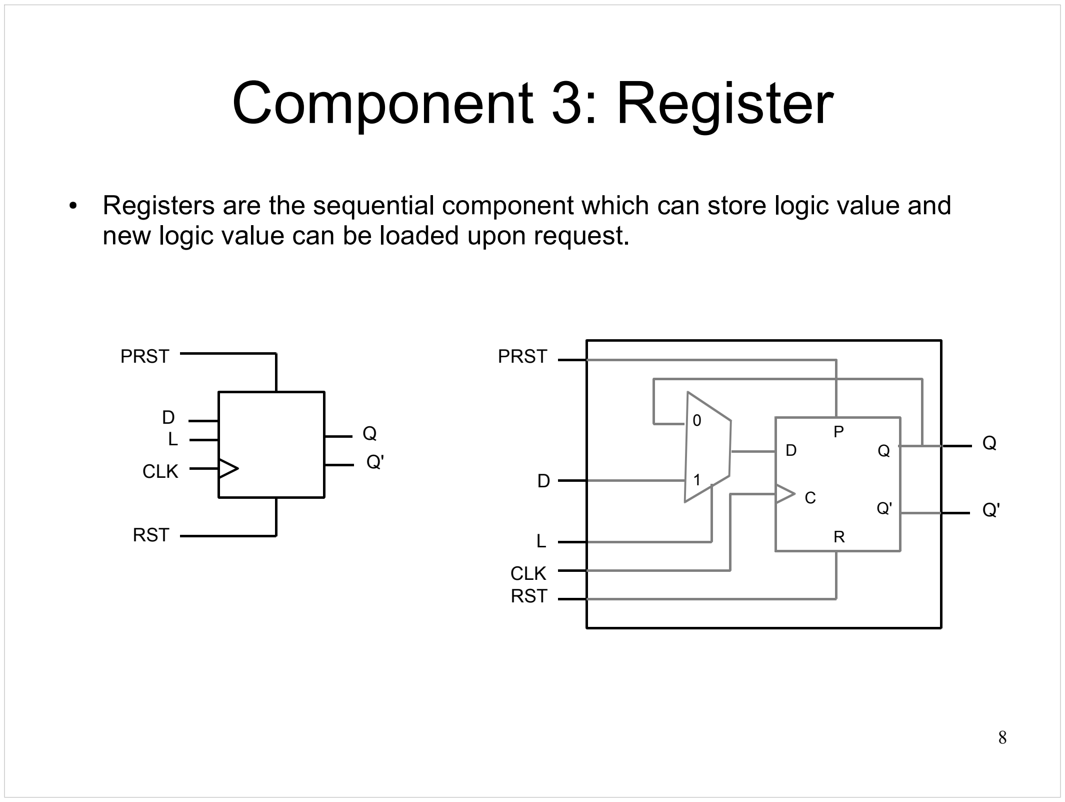
- We can user decoder and Enabling circuit to implement multiplexer



4x1 1-bit multiplexer

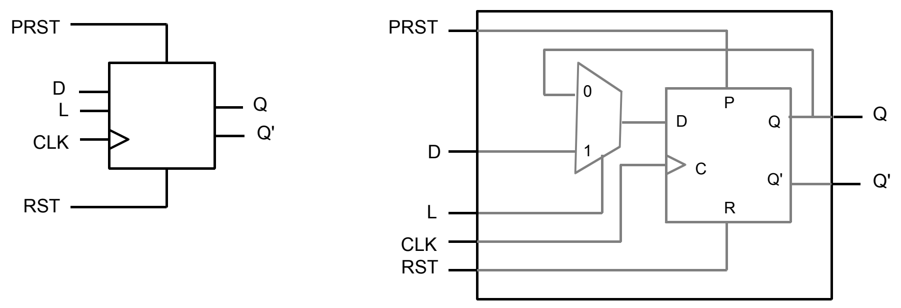
7

- To implement mux, decoder and enabling circuit can be re-used with minor modification. Since the line decoder produces one hot output corresponding the input binary pattern, the decoder output can be used to select one of the input data to send for final output. The outputs from the enabling circuit is connected to OR gate in next stage. Since, except for the enabled data, all the other outputs from the enable circuit will be logic 0, the OR gate will pass Boolean logic value of the selected signal.



Component 3: Register

- Registers are the sequential component which can store logic value and new logic value can be loaded upon request.

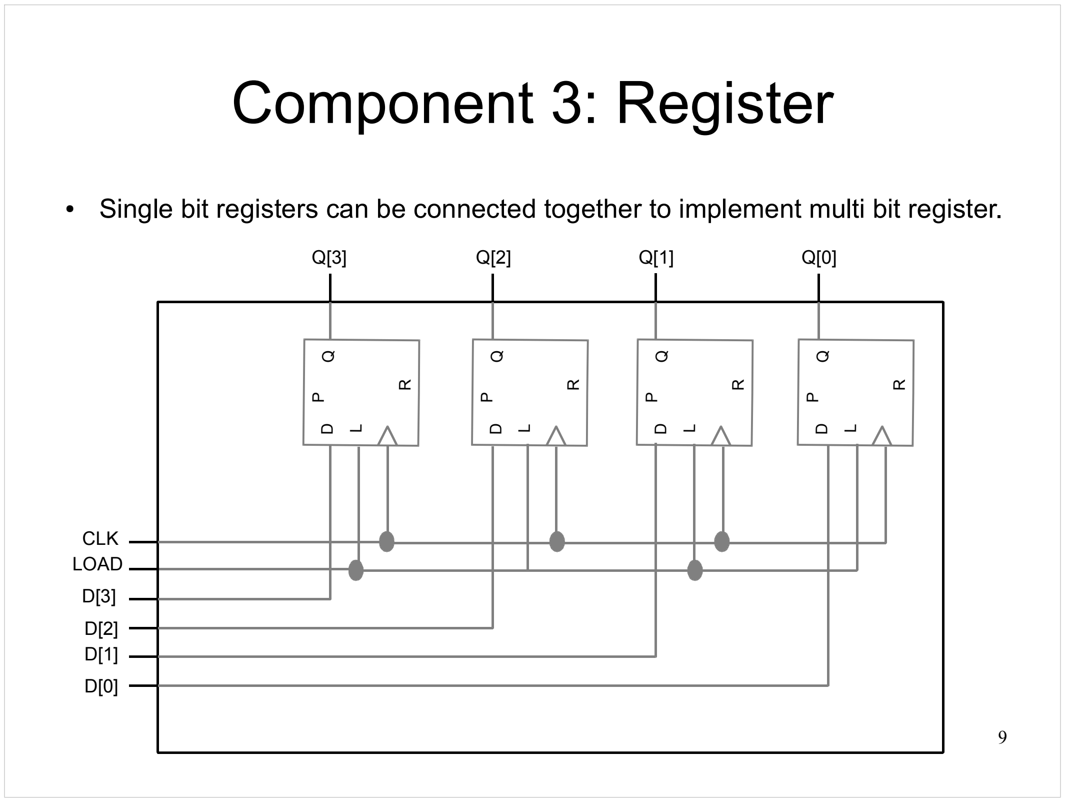


8

- In a computer operation, it is not always desired to load whatever value comes into the data input of a flipflop. Register provides one extra control 'L' (or sometime it is explicitly called 'load') for selectively loading data into a flipflop. Storage element with selectable load is called a register.

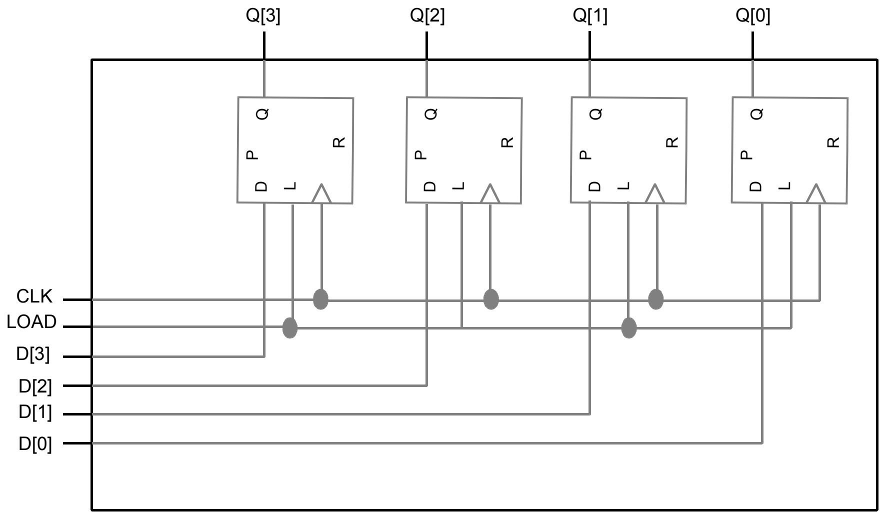
- The selective load feature is implemented using a combination of flipflop and a 2x1 mux. The output of the mux is connected to the data input of the flop. One of the input for the mux is the output of the flop and the other input is the incoming data input. The mux selection control is the load input value.

- If the load control logic 'L' is 0, the output of the flop is feedback into its data input. As a result, with L=0, output in next clock cycle is same as the value at previous clock cycle (data is held). With L=1, the incoming data value is placed on the data input of the flop and thus the flop is loaded with new value that is placed on the register's data input.



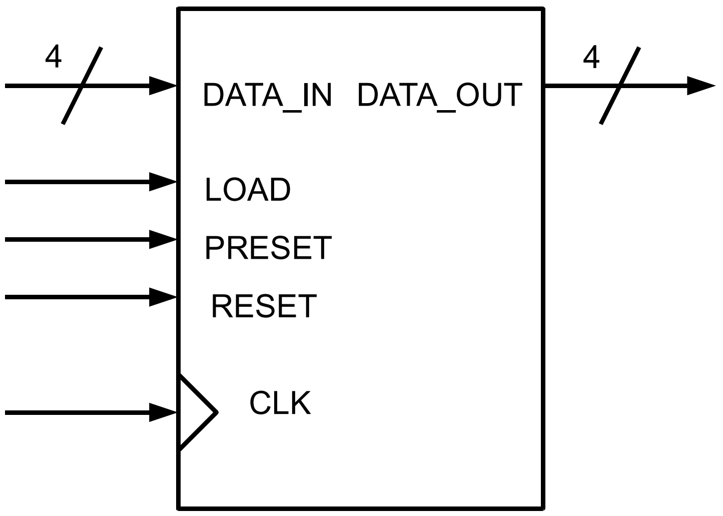
Component 3: Register

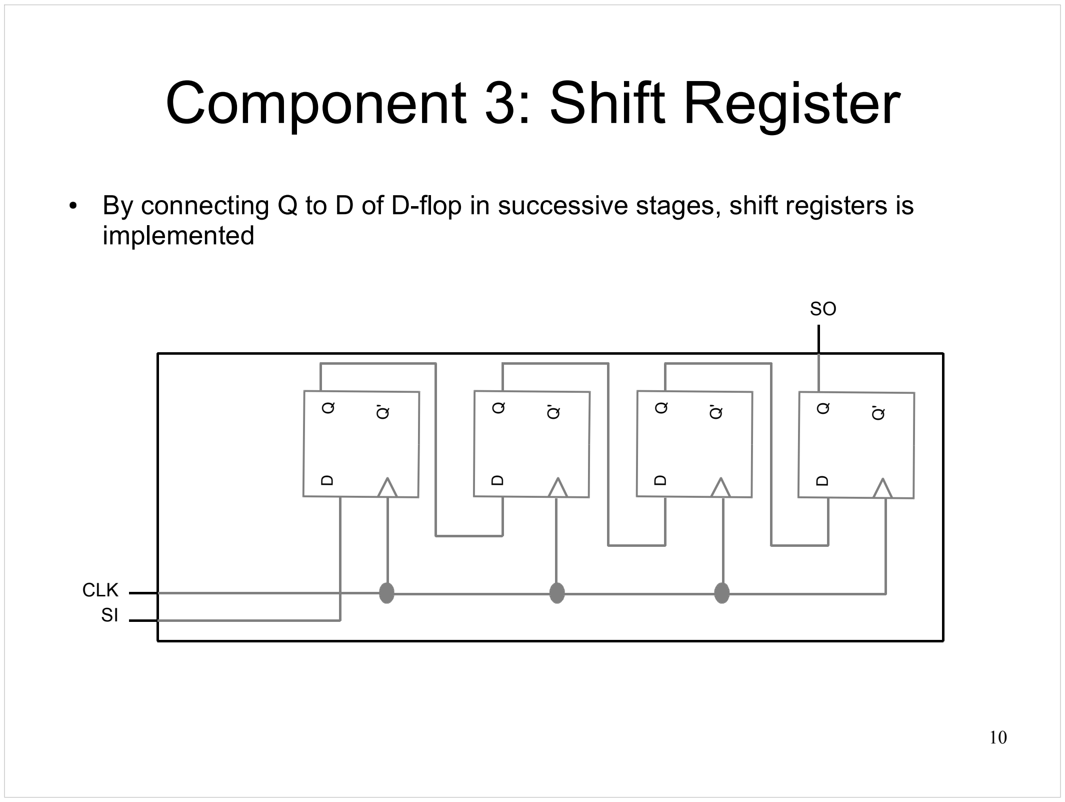
- Single bit registers can be connected together to implement multi bit register.



9

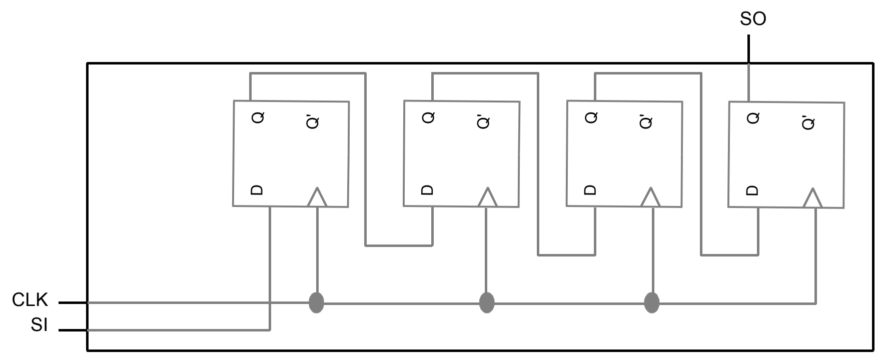
- Multiple registers can be grouped together to form a multi-bit register. The clock and load signal is shared among the flops in the group. The following is a schematic diagram for register. The PRESET and RESET signals are asynchronous. PRESET=1 will set the output to 0xF and RESET=1 will set the output to 0x0. Both the input and output are 4-bit wide.





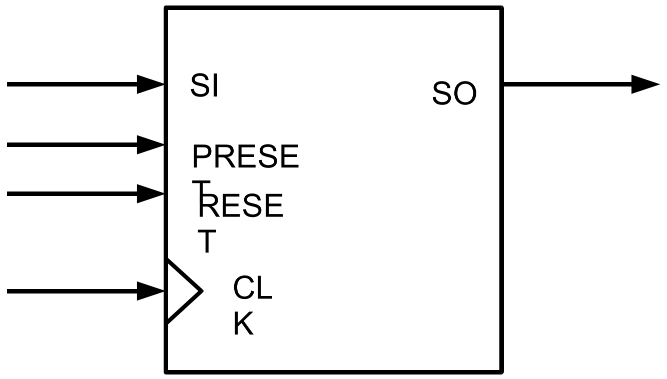
Component 3: Shift Register

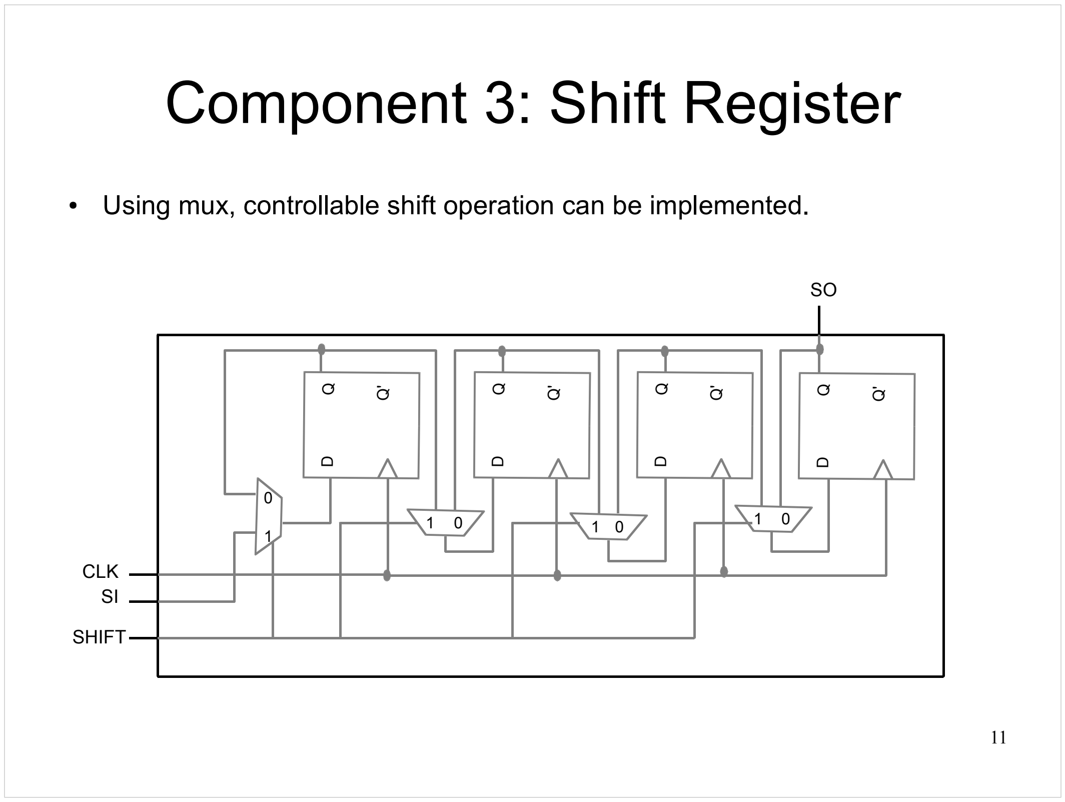
- By connecting Q to D of D-flop in successive stages, shift registers is implemented



10

- If the output of 1-bit register is connected to input of the register in the next bit position, we get a shift register. There is one serial input SI and serial output SO. A continuous bit stream comes onto SI and after 4 clock cycle (in this example) a continuous bit stream occurs in FIFO (First In First Out) manner. At each clock cycle data from one flop is transferred to the next flop in the stage. For a n-bit shift register, it takes n clock cycle for the first bit of SI to be appeared at the SO. However, with this arrangement there is no control of when to start the shift and when to stop it.

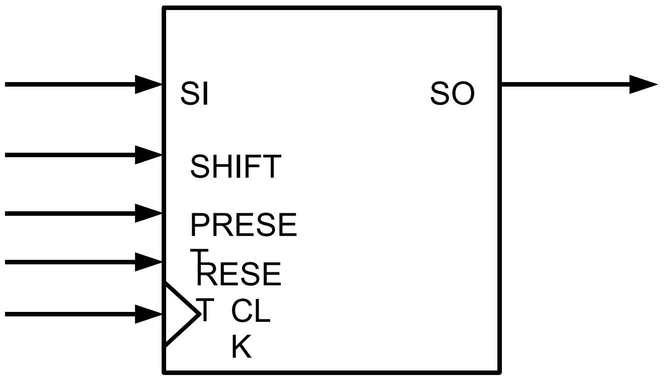




Component 3: Shift Register

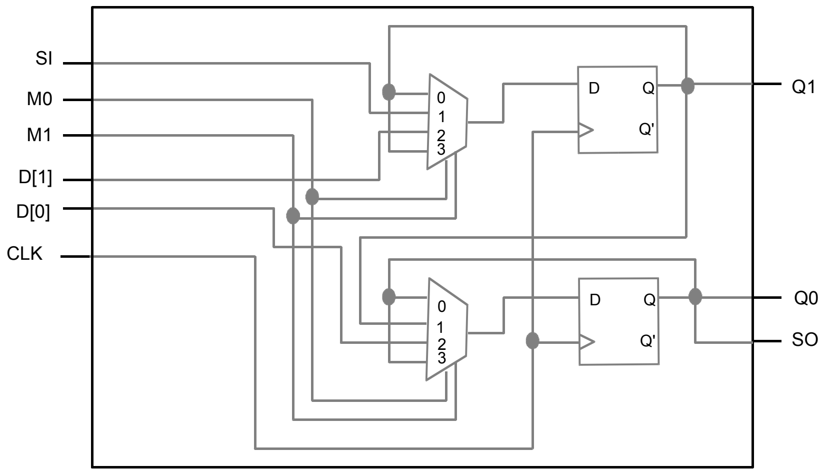
- Using mux, controllable shift operation can be implemented.

- Mux based shift control can be implemented. In this case, an external control signal 'SHIFT' is added. If SHIFT=0, the data holds and not shift operation is performed. If SHIFT=1, shift operation resumes. Incoming data on a register is controlled by a mux for each register, so that data source can either be itself (feedback from its own output) or from previous flop or SI (in case of the the first register). If the SHIFT is 1, data on a register comes from the previous stage register or SI and hence the shift operation resumes. If SHIFT is 0, data comes from itself for a register and thus hold is performed. However, for this circuit, there is no way to load a predefined bit pattern in one clock cycle.



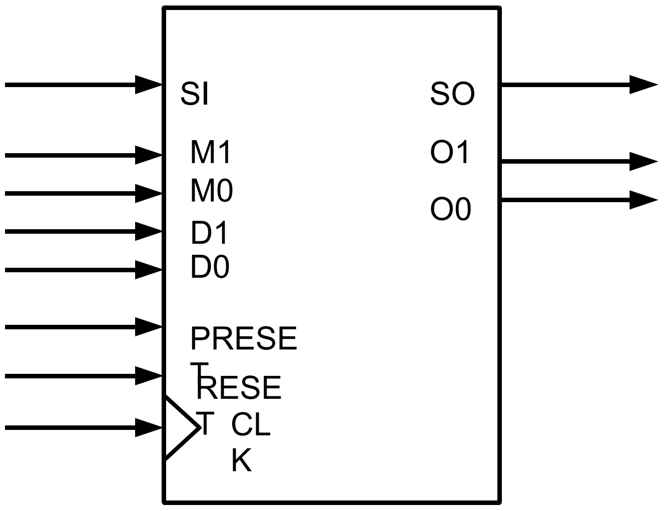
Component 3: Shift Register

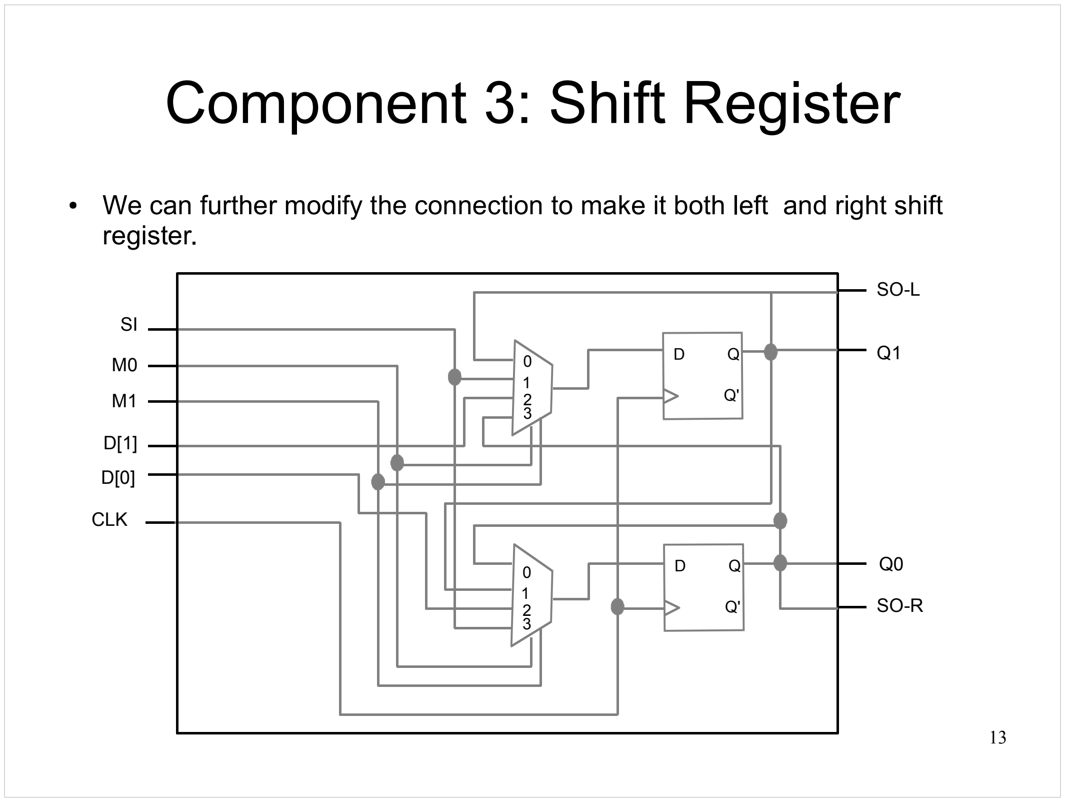
- We can further add parallel load to this sift register



12

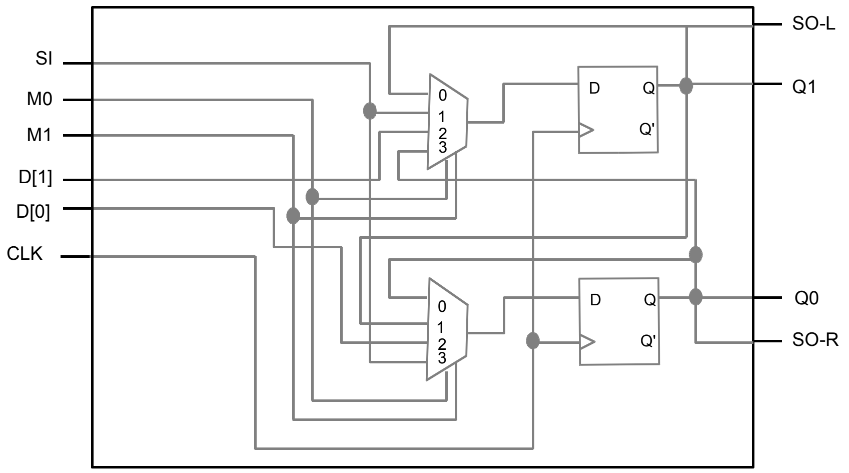
- Using a 4x1 mux, a parallel loadable shift register can be implemented. In this case, it has two control signal (or mode signal) bit M1 and M0. The data sources for a single bit register is either it self (hold operation), or serial in / data from previous register (for shift operation) or independent input assigned for that register (parallel load). In the above circuit, M1M0 value '00' implements hold operation, where '01' resumes shift operation and '10' will load independent data from input. The feedback data is also connected to the 4th selected point on the mux to make sure '11' control pattern does not do any unexpected operation. Since it is connected to the feedback of the register, '11' will perform hold operation.



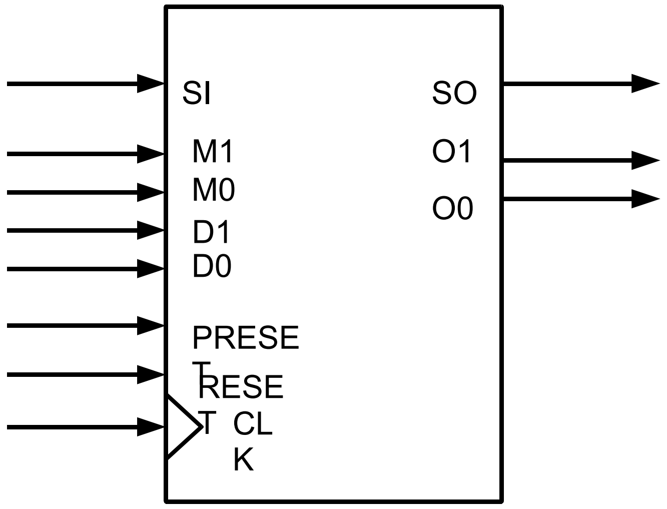


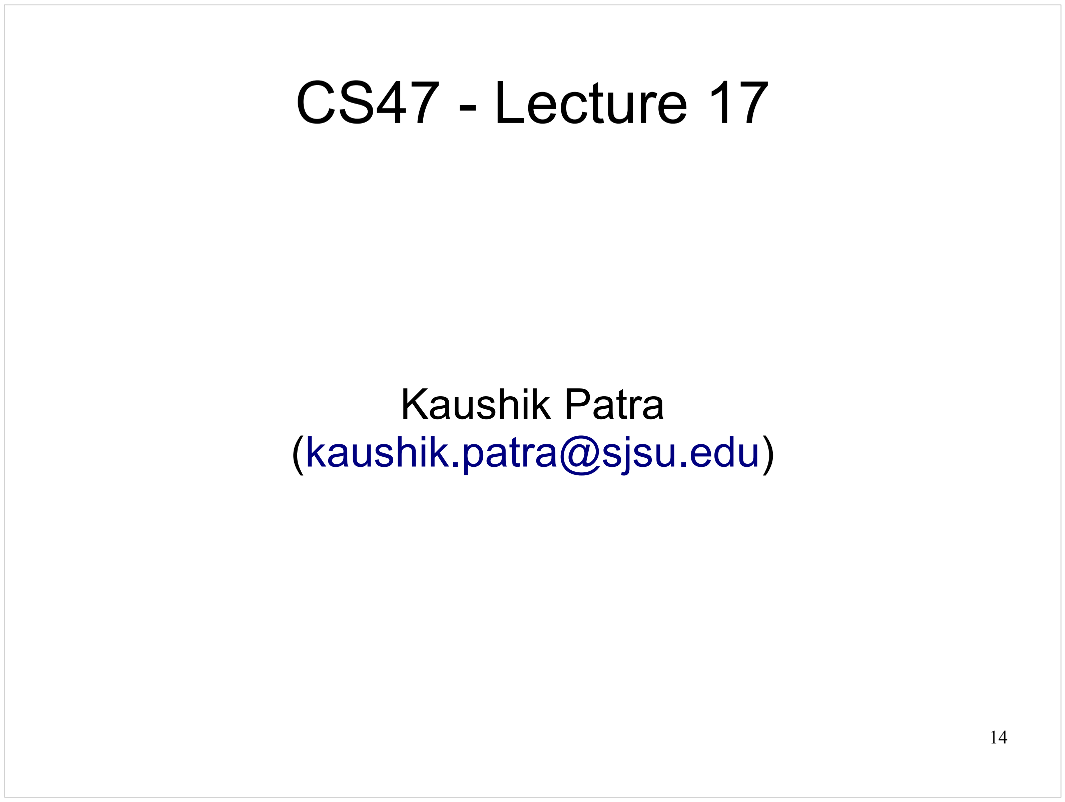
Component 3: Shift Register

- We can further modify the connection to make it both left and right shift register.



- If we connect the register data input to the selection mux from next stage to previous stage we can implement both left shift and right shift. In this case M1M0 as '11' with perform left shift where '00' will perform the right shift. The interface does not have any change with respect to the previous register.





CS47 - Lecture 17

Kaushik Patra
(kaushik.patra@sjsu.edu)

- Digital Circuit Components

[Chapter 5 of Logic & Computer Design Fundamentals, 4th Edition, M. Morris Mano, Charles R. Kime]