WEEK 9-2017

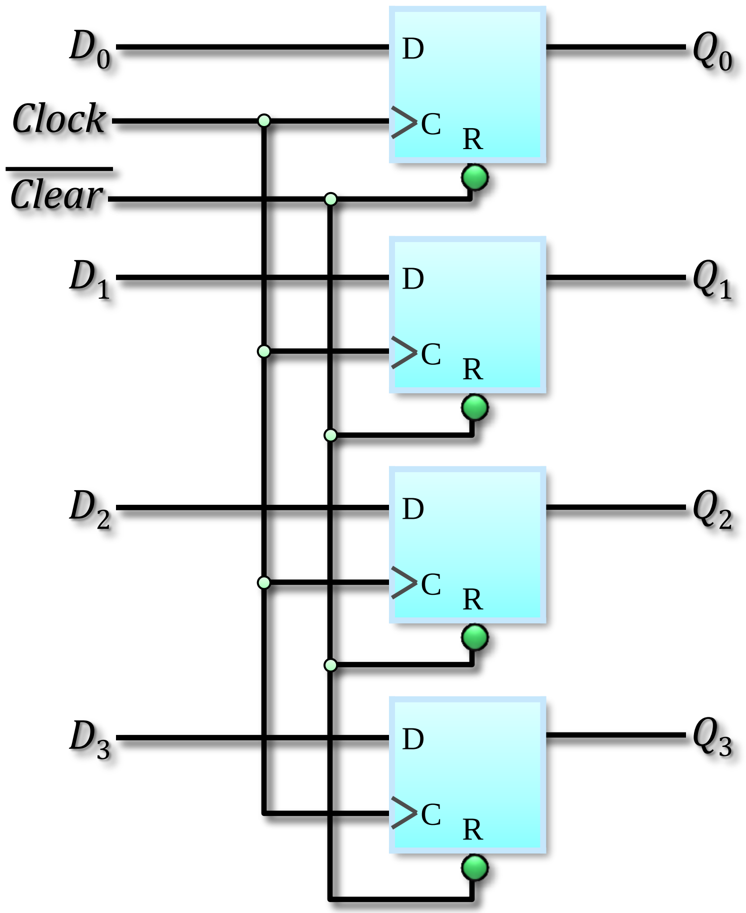
Verilog HDL-Shift Register

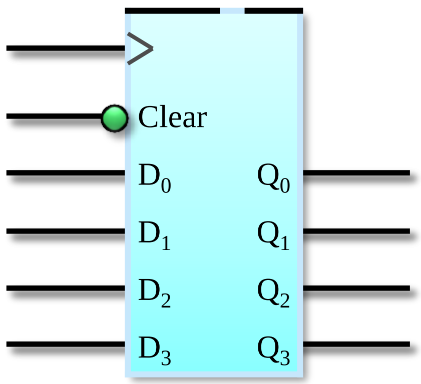
ELEC2141: Digital Circuit Design

Register

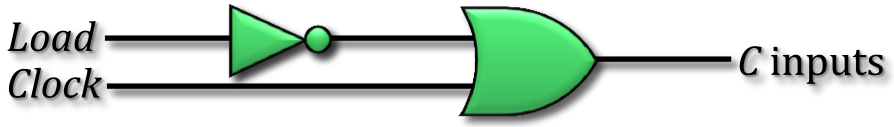
- An n- bit register consists of n flip-flops and is capable of storing n-bits of binary information.
- May also have combinational circuit that implements state transitions of the flip-flops.
- The flip-flops hold data and the combinational circuit determine the new or transformed data that will transfer to the flip-flops.
- A counter is special type of register that goes through a predetermined sequence of states.
- Useful for storing and manipulating information in digital computers.

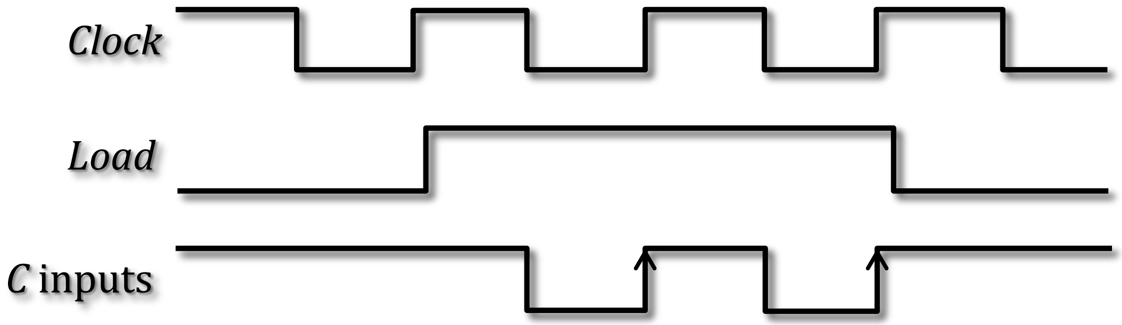
4- bit Register





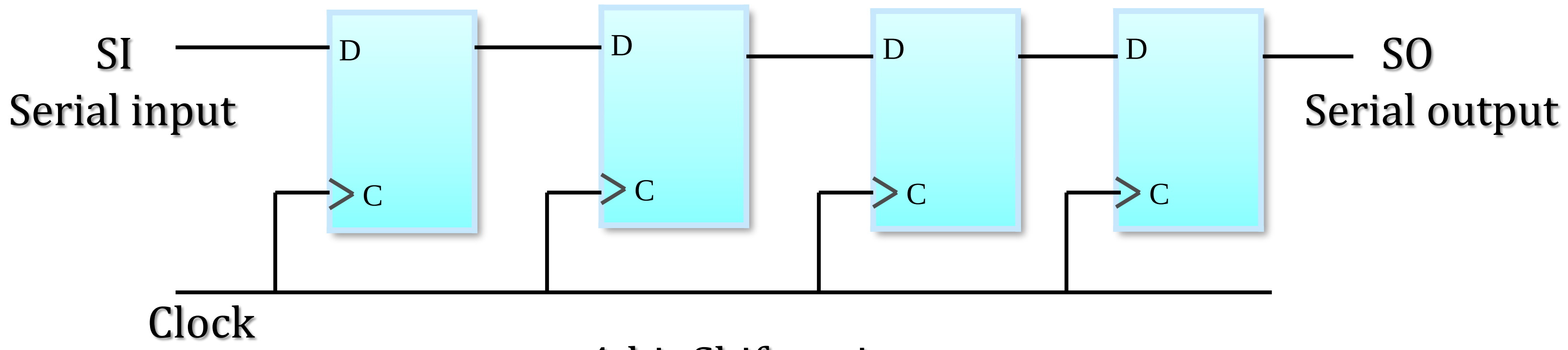
Clock gating



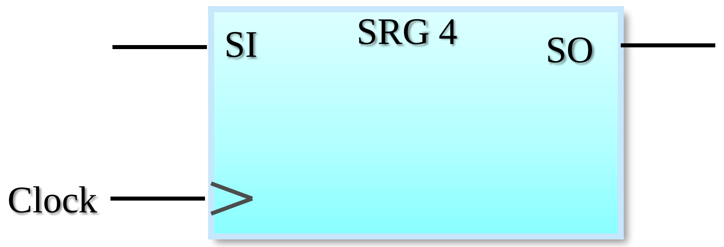


Shift-register

- **A register capable of shifting its stored bits laterally in one or both directions.**



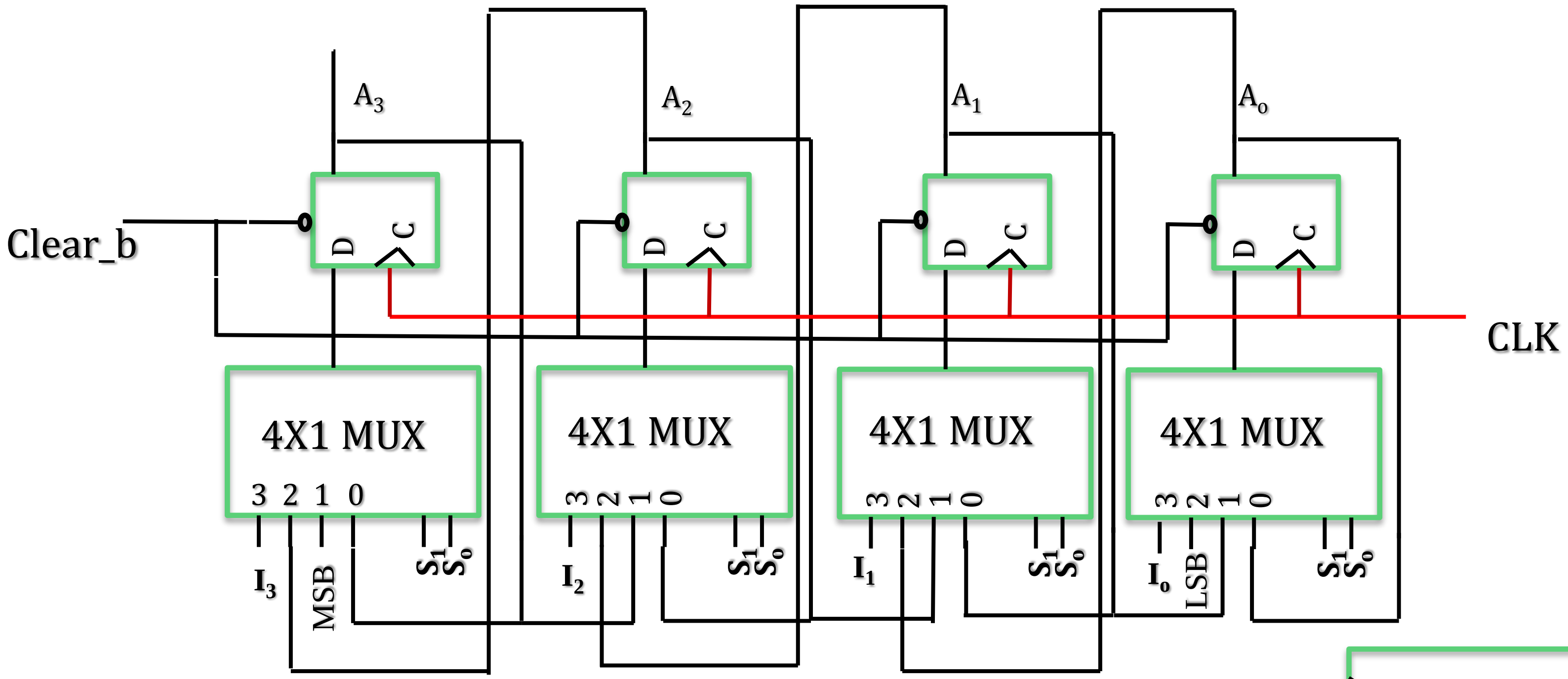
4-bit Shift register



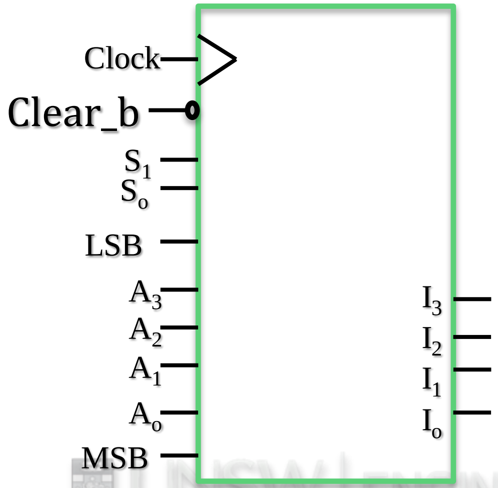
- **Bidirectional shift register with parallel loading provides versatile operation.**

Bidirectional Shift Register

- **Capable of: left, right serial shift, parallel loading, and holding data.**



Mode Control		**Register Operation**
S_1	S_o	
0	0	No change (Hold)
1	1	Parallel load
1	0	Shift left ($A_o \leftarrow$ LSB)
0	1	Shift right ($A_3 \leftarrow$MSB)



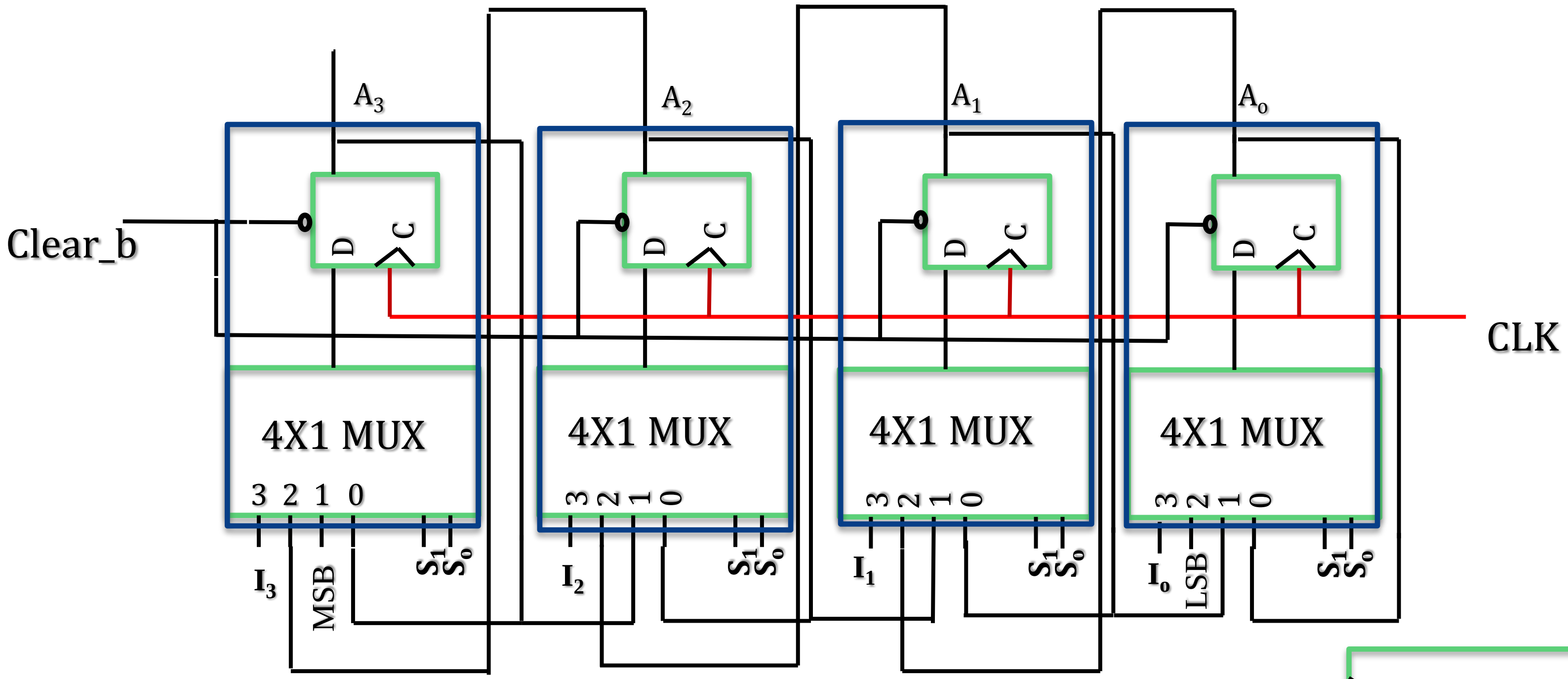
HDL for Shift Register

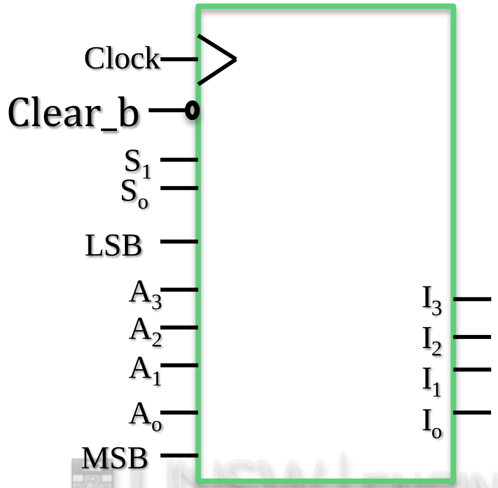
```
// Behavioral description of a 4-bit universal shift register

module  Shift_Register_4_beh(
 output reg    [3:0]  A;             // Register output
 input         [3:0]   I;           //Parallel output
 input         [1:0]  S;             //Select inputs
 input           MSB, LSB,        //Serial inputs
              CLK, Clear_b );      //Clock and Clear
always @ (posedge CLK, negedge Clear_b)
  if (Clear_b == 0)   A <= 4’b0000;
  else
   case (S)
     2’b00:  A <= A;                     // no change
     2’b01:  A <= {MSB, A[3:1]};   // Shift right
     2’b10:  A <= {A[2:0], LSB};    // Shift left
     2’b11:  A <= I;                      //Parallel load of input
   endcase

endmodule
```

Bidirectional Shift Register





HDL for the Top Level Module

```
// Structural description of a 4-bit universal shift register

module  Shift_Register_4_str(
 output    [3:0]  A;             // Register output
 input     [3:0]   I;            //Parallel output
 input     [1:0]  S;             //Select inputs
 input      MSB, LSB,       //Serial inputs
            CLK, Clear_b );      //Clock and Clear

// instantiate the four stages

stage ST0 (A[0], A[1], LSB, I[0], A[0], S, CLK, Clear_b);
stage ST1 (A[1], A[2], A[0], I[1], S, CLK, Clear_b);
stage ST2 (A[2], A[3], A[1], I[2], A[2], S, CLK, Clear_b);
stage ST3 (A[3], MSB, A[2], I[3], A[3], S, CLK, Clear_b);

endmodule
```

HDL for the Next Top Level Module

```
// Declaring one stage of shift register

module  stage(i0,i1,i2,i3,Q,select, CLK, Clr_b);

 input     i0, i1, i2, i3;            // Register output
 output       Q;             //Parallel output
 input [1:0]  select;              //Select inputs
 input       CLK,Clr_b;      //Serial inputs
 wire         mux_out;     //Clock and Clear

// instantiate mux and flip-flop

mux_4_x_1 M0 (mux_out, i0, i1, i2, i3, select);
DFF   M1 (Q,mux_out,CLK, Clr_b);

endmodule
```

Declaring the Low Level Modules

```
// Declaring  4x1 multiplexer

module  Mux_4_x_1(mux_out,i0,i1,i2,i3,select);

 input     i0, i1, i2, i3;
 output reg       mux_out;
 input [1:0]  select;

Always @ (select, i0,i1,i2,i3)
  case (select)
     2'b00: mux_out = i0;
     2'b01: mux_out = i1;
     2'b10: mux_out = i2;
    2'b11: mux_out = i3;
 endcase

endmodule
```

```
// Declaring  DFF

module  DFF(Q,D,CLK, Clr_b);

 output reg   Q;
 input     D, CLK, Clr_b;

  always @ (posedge CLK, negedge Clr_b)
       if (Clr_b == 0)  Q<= 1'b0;
        else
        Q <= D;
 endmodule
```