EE-421: Digital System Design

Sequential-Circuit Building Blocks: Register and Counter

Dr. Rehan Ahmed [rehan.ahmed@seecs.edu.pk]

NUST

SCHOOL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE (SEECS)

Register(s)

D flip-flop with Synchronous reset

```
module flipflop (D, Clock, Resetn, Q);
    input D, Clock, Resetn;
    output reg Q;

    always @(posedge Clock)
        if (!Resetn)
            Q <= 0;
        else
            Q <= D;

endmodule
```

An N-BIT Register

- Since registers of different sizes are often needed in logic circuits, it is advantageous to define a register module for which the number of flip-flops can be easily changed.

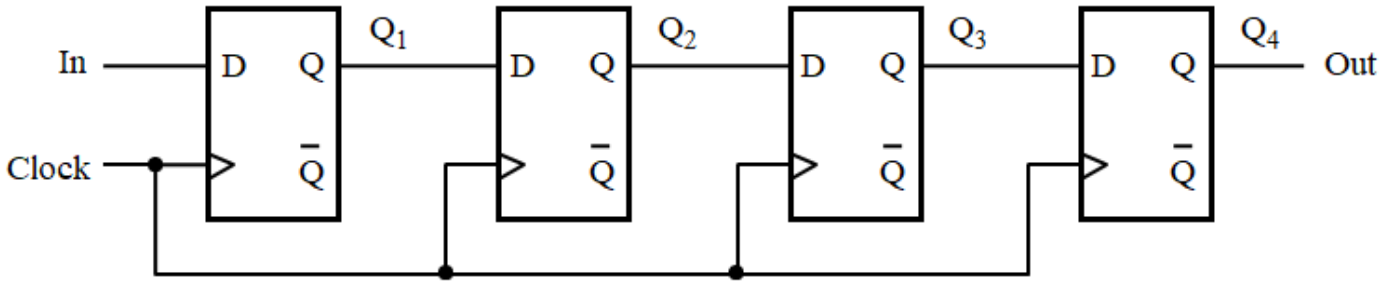
An N-BIT Register

```
module regn (D, Clock, Resetn, Q);
    parameter n = 16;
    input [n-1:0] D;
    input Clock, Resetn;
    output reg [n-1:0] Q;

    always @(negedge Resetn, posedge Clock)
        if (!Resetn)
            Q <= 0;
        else
            Q <= D;

endmodule
```

A 4-bit Shift Register



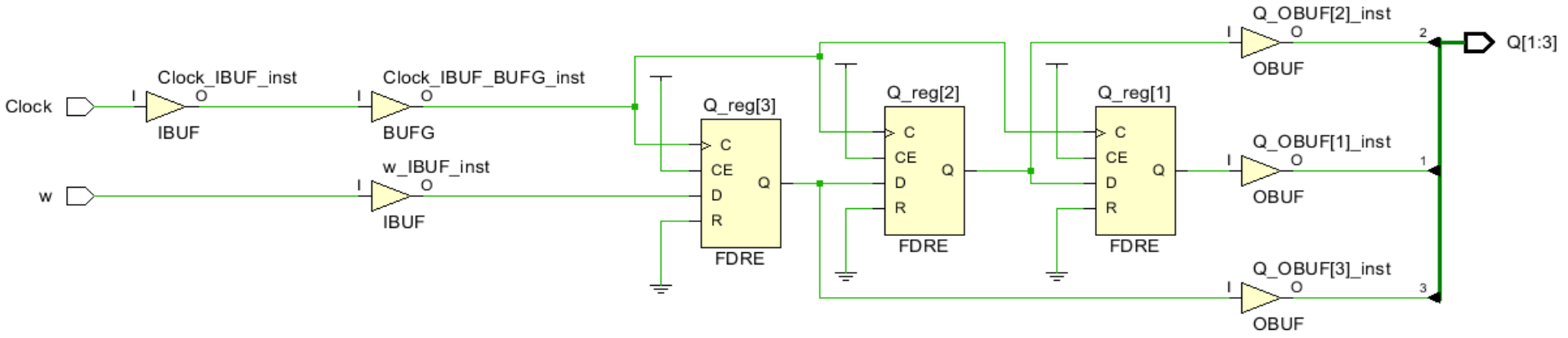
(a) Circuit

	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

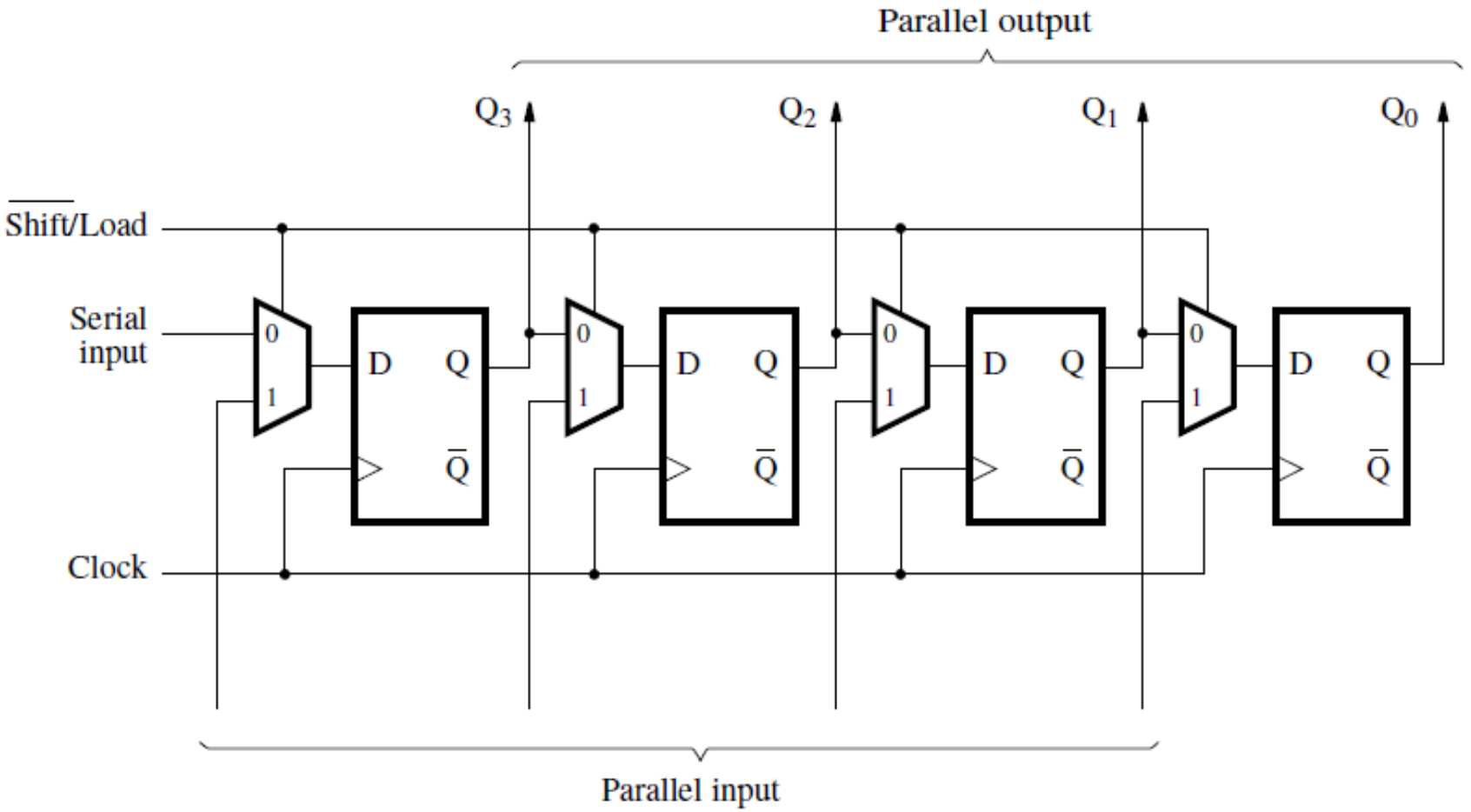
(b) A sample sequence

Expressing a 3-bit Shift Register

```
module shift3 (w, Clock, Q);
  input w, Clock;
  output reg [1:3] Q;
  always @(posedge Clock)
  begin
    Q[3] <= w;
    Q[2] <= Q[3];
    Q[1] <= Q[2];
  end
endmodule
```

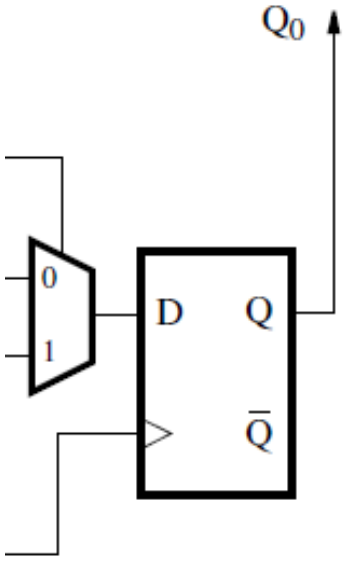


A 4-bit Parallel Access Shift Register



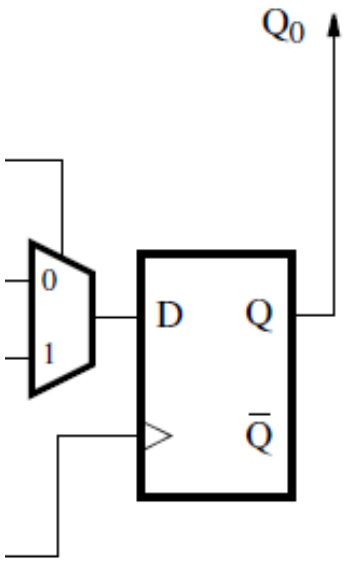
A 4-bit Parallel Access Shift Register

- Approach-1:
  - Hierarchical approach that uses sub-circuits
    - Each subcircuit consists of a D flip-flop with a 2-to-1 multiplexer connected to the *D* input.



A 4-bit Parallel Access Shift Register

- Approach-1:
  - Hierarchical approach that uses sub-circuits
    - Each subcircuit consists of a D flip-flop with a 2-to-1 multiplexer connected to the *D* input.



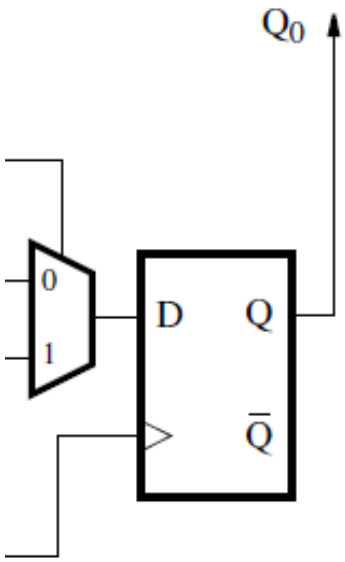

```
module muxdff (D0, D1, Sel, Clock, Q);
    input D0, D1, Sel, Clock;
    output reg Q;

    always @(posedge Clock)
        if (!Sel)
            Q <= D0;
        else
            Q <= D1;

endmodule
```

A 4-bit Parallel Access Shift Register

- Approach-1:
  - Hierarchical approach that uses sub-circuits
    - Each subcircuit consists of a D flip-flop with a 2-to-1 multiplexer connected to the *D* input.



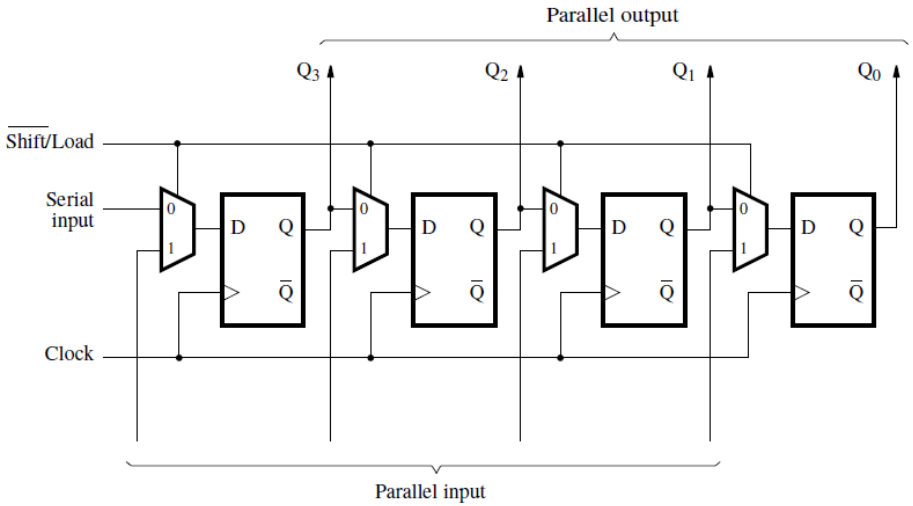

```
module muxdff (D0, D1, Sel, Clock, Q);
    input D0, D1, Sel, Clock;
    output reg Q;

    wire D;
    assign D = Sel ? D1 : D0;

    always @(posedge Clock)
        Q <= D;

endmodule
```

A 4-bit Parallel Access Shift Register




```
module shift4 (R, L, w, Clock, Q);
    input [3:0] R;
    input L, w, Clock;
    output [3:0] Q;
    wire [3:0] Q;

    muxdff Stage3 (w, R[3], L, Clock, Q[3]);
    muxdff Stage2 (Q[3], R[2], L, Clock, Q[2]);
    muxdff Stage1 (Q[2], R[1], L, Clock, Q[1]);
    muxdff Stage0 (Q[1], R[0], L, Clock, Q[0]);

endmodule
```

A 4-bit Parallel Access Shift Register

- Approach-2:

```
module shift4 (R, L, w, Clock, Q);
    input [3:0] R;
    input L, w, Clock;
    output reg [3:0] Q;

    always @(posedge Clock)
        if (L)
            Q <= R;
        else
        begin
            Q[0] <= Q[1];
            Q[1] <= Q[2];
            Q[2] <= Q[3];
            Q[3] <= w;
        end

endmodule
```

A n-bit Parallel Access Shift Register

```
module shiftn (R, L, w, Clock, Q);
    parameter n = 16;
    input [n-1:0] R;
    input L, w, Clock;
    output reg [n-1:0] Q;
    integer k;

    always @(posedge Clock)
        if (L)
          Q <= R;
        else
        begin
          for (k = 0; k < n-1; k = k+1)
              Q[k] <= Q[k+1];
              Q[n-1] <= w;
        end

endmodule
```

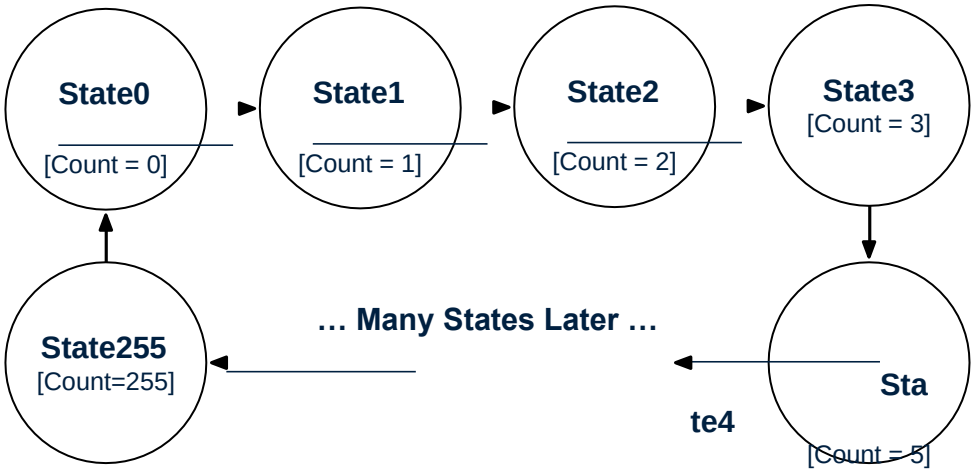
Counters

Synchronous Up-Counter

Clock cycle	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Counters Implemented as State Machine?

- Consider an 8-bit counter




```
always @ (p_state,w)
begin
   case (p_state)
   State0: n_state = State1;
   State1: n_state = State2;
   State2: n_state = State3;
    ……..……
   State255: n_state = State0;
   endcase

end
```

Up-Counter

```
module upcount (Resetn, Clock, E, Q);
    input Resetn, Clock, E;
    output reg [3:0] Q;

    always @(negedge Resetn, posedge Clock)
        if (!Resetn)
            Q <= 0;
        else if (E)
            Q <= Q + 1;

endmodule
```

Up-Counter with Parallel Load

```
module upcount (R, Resetn, Clock, E, L, Q);
    input [3:0] R;
    input Resetn, Clock, E, L;
    output reg [3:0] Q;

    always @(negedge Resetn, posedge Clock)
        if (!Resetn)
            Q <= 0;
        else if (L)
            Q <= R;
        else if (E)
            Q <= Q + 1;

endmodule
```

A down-counter with a Parallel Load

```
module downcount (R, Clock, E, L, Q);
    parameter n = 8;
    input [n-1:0] R;
    input Clock, L, E;
    output reg [n-1:0] Q;

    always @(posedge Clock)
        if (L)
            Q <= R;
        else if (E)
            Q <= Q - 1;

endmodule
```

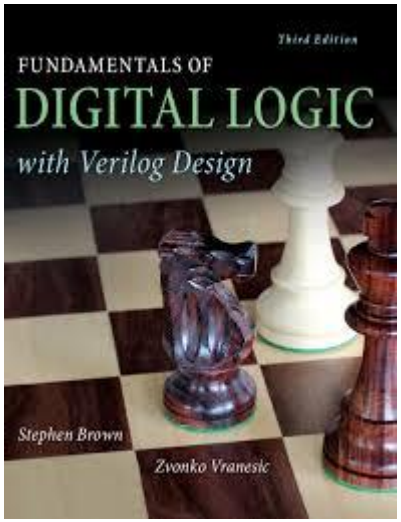
Up/Down Counter

```
module updowncount (R, Clock, L, E, up_down, Q);
    parameter n = 8;
    input [n-1:0] R;
    input Clock, L, E, up_down;
    output reg [n-1:0] Q;
    always @(posedge Clock)
        if (L)
            Q <= R;
        else if (E)
            Q <= Q + (up_down ? 1 : -1);

endmodule
```

Recommended Reading

- Digital System Design with Verilog HDL, 3/e, b **S**tephen Brown and **Z**vonko Vranesic. [**S&Z**]
  - S&Z,
    - Chapter-5
      - 5.1 – 5.11 for general review
      - 5.13 for Verilog



THANK YOU