CSE140L: Components and Design Techniques for Digital Systems

Registers, shift registers, counters

D Latch Truth Table

\[
\begin{array}{c|c|c|c|c|c|c}
CLK & D & \overline{D} & S & R & Q & \overline{Q} \\
\hline
0 & X & \overline{X} & 0 & 0 & Q_{prev} & \overline{Q}_{prev} \\
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]
D-Flip Flop

- Built using a **master D-latch** and a **servant D-latch**
- Stores 1 bit of information (either a 0 or a 1)
- Samples a new value on each rising edges of the clock (an alternative design can sample on the falling edge)

Characteristic Equation

\[ Q(t+1) = D(t) \]

<table>
<thead>
<tr>
<th>Id</th>
<th>D</th>
<th>Q(t)</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Building blocks with FFs: Basic Register

- Register: a sequential component that can store multiple bits
- A basic register can be built simply by using multiple D-FFs
module Register (D, Clk, Q);
input [3:0] D;
input Clk;
output reg [3:0] Q;

always @(posedge Clk)
  Q <= D;
endmodule
Register

- A register is a memory device that can be used to store more than one bit of information.
- A register is usually realized as several flip-flops with common control signals that control the movement of data to and from the register.
  - Common refers to the property that the control signals apply to all flip-flops in the same way.
  - **Load** or **Store**: put new data into the register.
  - **Read**: retrieve the data stored in the register (usually without changing the stored data).
  - **Clear**: writes a default value into the register (usually all zeros).
  - **(Read/Write)-Enable**: Enables respectively reading from and writing to the register.

Example: register with clear

```verilog
module reg1 (STO, CLR, D, Q);
  parameter n = 16;
  input STO, CLR;
  input [n-1:0] D;
  output [n-1:0] Q;
  reg [n-1:0] Q;

  always @(posedge STO or negedge CLR)
  begin
    Q <= 0;
    else Q <= D;
  end
endmodule
```
Synch/Asynch control signals

- Control Signals
  - When they are asserted, they initiate an action in the register
  - **Asynchronous Control Signals** cause the action to take place immediately
  - **Synchronous Control Signals** must be asserted during a clock assertion to have an effect

- “Traditional” control signals:
  - Load/Store/Write
  - Read
  - Clear
  - Enable

- Depending on the functionalities of your register, you might want to design other control signals
  - Dummy example: a control signal that writes a 0 on all even positions
Example 2: register with load, clear and output enable

module reg2 (CLK, CLR, LD, OE, D, Q);
parameter n = 4;
input CLK, CLR, LD, OE;
input [n-1:0] D;
output [n-1:0] Q;
reg [n-1:0] IQ, Q;
integer k;

always @ (posedge CLK)
    if (CLR) IQ <= 0;
    else if (LD) IQ <= D;

always @ (OE)
    if (OE) Q = IQ;
    else Q = 'bz;

endmodule

Sources: TSR, Katz, Boriello & Vahid
A shift register is a register capable of shifting its bits from one FF to the next one.

NOTE: do not confuse the shift register with the logic/arithmetic shifter.

Is the shift register drawn above a left shifter or a right shifter?
Example: Verilog shift register with load

```
module shift4 (D, LD, LI, Ck, Q);
  input [3:0] D;
  input LD, LI, Ck;
  output [3:0] Q;
  reg [3:0] Q;

  always @(posedge Ck)
    if (LD)
      Q <= D;
    else
      begin
        Q[0] <= Q[1];
        Q[1] <= Q[2];
        Q[2] <= Q[3];
        Q[3] <= LI;
      end

endmodule
```
Example 2: A multifunction shift register with control signals

- Shift left is from A to D
- Shift right is from D to A
- CLR is asynchronous
Counters

- A counter is a register capable of incrementing and/or decrementing its contents.
- More general definition: a register capable of changing its content between a set of possible predefined sequences (this definition accounts for more “fancy” counters).

- What other possible control signals you can apply to counters:
  - Counting up/down
  - Modulus
  - …
Example: counter

```verilog
module counter (C, CLR, Q);
input C, CLR;
output [3:0] Q;
reg [3:0] tmp;

always @(posedge C or posedge CLR)
begin
    if (CLR)
        tmp = 4'b0000;
    else
        tmp = tmp + 1'b1;
end
assign Q = tmp;
```

- What is this counter doing?
- The CLR signal is synchronous or asynchronous?
Example 2: counter

module counter (C, S, Q);
input C, S;
output [3:0] Q;
reg [3:0] tmp;

always @(posedge C)
begin
  if (S)
    tmp = 4'b1111;
  else
    tmp = tmp - 1'b1;
end
assign Q = tmp;
endmodule

• What is this counter doing?
• The set (S) signal is synchronous or asynchronous?