1. Registers

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
- A register is a generalization of a flip-flop. Where a flip-flop stores one bit, a register stores several bits
- The main operations on a register are the same as for any storage devices, namely
  - Load or Store: Put new data into the register
  - Read: Retrieve the data stored in the register (usually without changing the stored data)
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

Examples
- On the following three registers, which control signals are asynchronous and which are synchronous? How are the control signals asserted?

```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)
        if (CLR == 0) Q <= 0;
        else Q <= D;
endmodule
```
Verilog description of previous two registers

```verilog
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
```

2. Counters

A counter is a register capable of incrementing and/or decrementing its contents

\[ Q \leftarrow Q \text{ plus } n \]
\[ Q \leftarrow Q \text{ minus } n \]

- The definition of "plus" and "minus" depend on the way the register contents encode the integers
- Binary Counters: Encode the integers with the binary number code
Example: 3-bit binary counter:

<table>
<thead>
<tr>
<th>Count Sequence</th>
<th>Transition Table</th>
<th>State Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 0</td>
<td>1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
<td>2</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 0</td>
<td>3</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 0 1</td>
<td>4</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 0</td>
<td>5</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 1 1</td>
<td>6</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>7</td>
</tr>
</tbody>
</table>

What does the counter count?
- The output signals are just the state variables

Example: 3-bit binary up/down counter

<table>
<thead>
<tr>
<th>Transition Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1 1 1 0 1</td>
</tr>
<tr>
<td>0 0 1 0 0 0 1 0</td>
</tr>
<tr>
<td>0 1 0 0 1 0 1 1</td>
</tr>
<tr>
<td>0 1 1 1 0 1 0 0</td>
</tr>
<tr>
<td>1 0 0 0 1 1 1 1</td>
</tr>
<tr>
<td>1 0 1 1 0 0 1 1</td>
</tr>
<tr>
<td>1 1 0 1 1 1 0 0</td>
</tr>
</tbody>
</table>

Example: Binary mod 6 counter

<table>
<thead>
<tr>
<th>Transition Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0 0 1 0 1</td>
</tr>
<tr>
<td>0 1 0 1 1 0 1 0</td>
</tr>
<tr>
<td>0 1 1 1 0 0 1 0</td>
</tr>
<tr>
<td>1 0 0 0 1 0 0 1</td>
</tr>
<tr>
<td>1 0 1 0 0 1 0 0</td>
</tr>
<tr>
<td>1 1 0 1 0 0 1 0</td>
</tr>
</tbody>
</table>

State Diagram
Design of a Binary Up Counter

<table>
<thead>
<tr>
<th>Q2</th>
<th>Q1</th>
<th>Q0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Q0 toggles every clock cycle
- Q1 toggles on those clock cycles where Q0=1
- Q2 toggles on those clock cycles where Q0=Q1=1
- ...  

Count Sequence

- Qi toggles on every clock cycle where Qj = 1, for i > j ≥ 0
Design of a Binary Down Counter

- Qi toggles on every clock cycle where Qj = 0, for i > j ≥ 0

<table>
<thead>
<tr>
<th>Q2</th>
<th>Q1</th>
<th>Q0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Q0 Toggles every clock cycle
Q1 toggles on those clock cycles where Q0=0
Q2 toggles on those clock cycles where Q0=Q1=0

Binary Down Counter

CK
Synchronous, Series-Carry Binary Counter

\[ T_W \geq t_{PFF} + (n-2)t_{PG} + t_{su} \quad (\text{for } n \geq 2) \]

Synchronous, Parallel-Carry Binary Counter

\[ T_W \geq t_{PFF} + t_{PG} + t_{su} \quad (\text{for } n \geq 3) \]
- Asynchronous Counters

![Asynchronous Counter Diagram]

- Typical MSI counter chip

![74LS163 Diagram]

- LD and CLR are synchronous
- LD asserted during the rising edge of the clock loads the register from ABCD.
- CLR asserted during the rising edge of the clock clears the counter
- CLR overrides LD
- LD overrides EN
- \( R_{CO} = Q_D \cdot Q_C \cdot Q_B \cdot Q_A \cdot EN \), used for cascading chips

---

- Verilog description of the 74x163

```verilog
module V74x163 (CLK, CLR_L, LD_L, ENP, ENT, D, Q, RCO);
    input CLK, CLR_L, LD_L, ENP, ENT;
    input [3:0] D;
    output RCO;
    output [3:0] Q;
    reg [3:0] Q;
    reg RCO;

    always @(posedge CLK)
        if (CLR_L == 0) Q <= 4'b0000;
        else if (LD_L == 0) Q <= D;
        else if (ENT & ENP) Q <= Q + 1;

    always @(Q or ENT)
        if (Q == 15 && ENT == 1) RCO = 1;
        else RCO = 0;

endmodule
```
Verilog description of an up/down counter

```verilog
dmodule updowncount (R, Clock, L, E, up_down, Q);
parameter n = 8;
input [n-1:0] R;
input Clock, L, E, up_down;
output [n-1:0] Q;
reg [n-1:0] Q;
integer direction;

always @(posedge Clock)
begin
    if (up_down) direction = 1;
    else direction = -1;
    if (L) Q <= R;
    else if (E) Q <= Q + direction;
end
endmodule
```

Verilog description of mod-n counters

```verilog
dmodule upmodn (Ck, Q);
parameter n = 6;
input Ck;
output [3:0] Q;
reg [3:0] Q;

always @(posedge Ck)
if (Q == n) Q <= 0;
else Q <= Q + 1;
endmodule
```

```verilog
dmodule dwnmodn (Ck, Q);
parameter n = 5;
input Ck;
output [3:0] Q;
reg [3:0] Q;

always @(posedge Ck)
if (Q == 0) Q <= n;
else Q <= Q - 1;
endmodule
```
Design of Mod $n$ Counters

- Mod 6 Up Counter

- Mod 5 Down Counter

Decoding Binary Counter States

- The decoding spikes are hazards that cannot be designed out.
- The following circuit will mask the decoding spikes, at the cost of delaying the outputs one clock cycle.
3. Shift Registers

- How would you add a control signal to control when the shift register shifted?
- How would you add parallel input capability and why would you want to?
  - What kind of control signals are needed?
- Is the shift register drawn above a left shifter or a right shifter?
- How would you make a shift register that could shift either left or right and what control signals would you need?

Example: 74LS194

<table>
<thead>
<tr>
<th>S1</th>
<th>S0</th>
<th>Action</th>
<th>QA*</th>
<th>QB*</th>
<th>QC*</th>
<th>QD*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>hold</td>
<td>QA</td>
<td>QB</td>
<td>QC</td>
<td>QD</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>shift right</td>
<td>RIN</td>
<td>QA</td>
<td>QB</td>
<td>QC</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>shift left</td>
<td>QB</td>
<td>QC</td>
<td>QD</td>
<td>LIN</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>load</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

- Shift left is from A to D
- Shift right is from D to A
- CLR is asynchronous
Verilog Description Of A Shift Register

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

Ring Counters
Self-Correcting Ring Counter

Johnson counter, switch-tail counter, moebius counter
4. Review

☐ Register control signals and assertions.
☐ Binary counters and their operations.
  ■ Reset, Load, Output Enable.
  ■ Counter timing; maximum clock frequency.
☐ Mod-n counters
  ■ Synchronous vs. asynchronous load and reset signals.
☐ Shift registers and shift register counters.
  ■ Ring counters, Johnson counters, etc
  ■ Self-correcting counters
☐ Counter realization of sequential circuits