Solution to Assignment 3

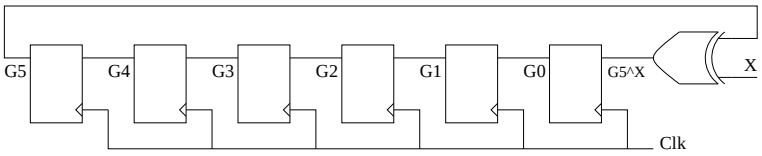
ELE-350: Digital Electronics

Winter 2001

---

1. Draw the circuit diagram that would be implemented by this code segment.

```
input x; reg [5:0] G;
always @(posedge clk)
   G <= {G[4:0], G[5]^x};
```



2. What function does this code segment represent?

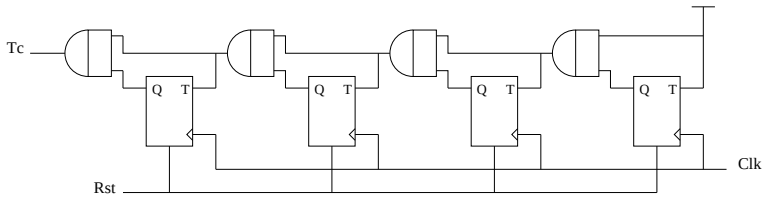
```
output out;
input i0, i1, i2, i3;
input s1, s0;
assign out = s1 ? (s0 ? i3 : i2) : (s0 ? i1 : i0);
```

Replace the *assign* statement with a more readable one (still one line).

This is a 4-to-1 Multiplexer. The inputs are i0, i1, i2, and i3. The output is out, and the selecting switches are s0 and s1. Considering the logic of the operation, the assignment can be changed to:

```
assign out=(~s1 & ~s0 & i0) | (~s1 & s0 & i1) | (s1 & ~s0 & i2) | (s1 & s0 & i3)
```

3. Write the full Verilog code for a 4-bit ripple counter using toggle flip-flops. The counter has an asynchronous reset and a terminal-count output. The cell library doesn't have toggle flip-flops.



Since the cell library doesn't have a toggle flip-flop, one should create it as a module. This can be done using structural code or procedural code. However, the toggle flip-flops must be connected to each other using structural code to produce the counter.

```
module ripple_counter(count, tc, clk, rst);

   output [3:0] count;
   output tc; \\ terminal count
   input clk, rst;

   reg [3:0] count;
   wire tc;
   wire [3:0] tin;  \\ inputs to the toggle flip-flops.

   assign tin[0]=1'b1; \\ very first carry_in connected to VDD

   toggle_flipflop tff0(count[0], tin[0], clk, rst);
   and n0(tin[1], count[0], tin[0]);
   toggle_flipflop tff1(count[1], tin[1], clk, rst);
   and n1(tin[2], count[1], tin[1]);
   toggle_flipflop tff2(count[2], tin[2], clk, rst);
   and n2(tin[3], count[2], tin[2]);
   toggle_flipflop tff3(count[3], tin[3], clk, rst);
   and n3(tc, count[3], tin[3]);

endmodule

module toggle_flipflop(q, t, clk, rst);

   output q;
   input t, clk, rst;
   reg q;

   always @(posedge clk or posedge rst)
      begin
       if (rst) q<=0; \\ if reset clear the output.
       else if (t) q<=~q; \\ flip the output only if t=1.
      end

endmodule
```