Verilog Specification of a Bit Serial Adder. You already know about parallel adders. Your task is to specify in behavioral and structural Verilog, a bit serial adder. Such a system, built around shifters and a single bit full adder, works by shifting in the bits to be added, one at a time, forming the sum of those bits, one at a time. The basic structure of the system is shown below implementing $S[3:0] = A[3:0] + B[3:0]$.

Control signals are in *italics*, and data flows are arrows. *Reset* and *Clk* are control signals. The inputs to the adder are $A$, $B$, and $C_{in}$. Its outputs are $Sum$ and $C_{out}$. The shifters have parallel load inputs $I[3:0]$, serial shift right input $ShI$, parallel outputs $Q[3:0]$, and serial shift right output $ShO$. They read the parallel inputs on the positive edge of the clock when $Ld$ is true, and shift right on the positive edge when $ShR$ is true. They shift in $ShI$ and shift out $ShO$. $Ld$ has priority over $ShR$ if both are asserted at the same time. Also on the positive edge of $Clk$, the Carry Flip Flop $C$ is reset to 0 when $Rst$ is true. It loads $C_{out}$ otherwise.

Write behavioral Verilog for the following components:

a. Carry Flip Flop
module Carry_FF(Q, D, Rst, Clk);
output Q;
input D, Rst, Clk;
reg Q;

always @(posedge Clk) begin
  if (Rst) Q <= 0;
  else Q <= D;
end
endmodule

b. Right Shift Register

module r_shift (Q, ShO, I, ShI, Ld, ShR, Clk);
output [3:0] Q;
output ShO;
input [3:0] I;
input ShI, Ld, ShR, Clk;
reg [3:0] Q;
assign ShO = Q[0];
always @(posedge Clk) begin
  if (Ld) Q <= I;
  else if (ShR) Q <= {ShI, Q[3:1]};
end
endmodule

c. Full Adder

module adder(Sum, Cout, A, B, Cin);
output Sum, Cout;
input A, B, Cin;
assign {Cout, Sum} = A + B + Cin;
endmodule

The D shifter is initialized to 4'b1000 when LdD is true. When the 1 shifts through to the output Cnt, it is a signal to the Control that four cycles of single bit addition have been executed.

The system works as follows. The Reset signal is asserted for exactly one clock period. On the first rising clock edge with Reset true, the Control enters the Initialize state and asserts the control signals Ld and Rst. On the next rising edge, with Ld and Rst true, the system loads A[3:0] into the A shifter, B[3:0] into the B shifter, 4'b1000 into the D shifter, and 0 into C.

At the same time, the system enters the Add state. Shortly after entering the Add state, A[0] and B[0] become visible to the adder’s inputs, with Cin = 0. After a further propagation delay, a single Sum bit S[0] and a new Cout are formed. With the control signal ShR set to true, the shifters are set to shift to the right on the next clock edge.

On this next edge, the shifters actually shift to the right. S shifts in the 0th sum bit and C receives the previous Cout. After a short delay, A[1] and B[1] appear at the shift right outputs of A and B, and the previous carry is now at Cin. After a further propagation delay, S[1] is formed with a new Cout-.

Complete a sketch of the system’s timing using the diagram below as a start:

![Timing Diagram]

### d. Write the Behavioral Verilog for the Control

```verilog
module control(Ld, ShR, Rst, Reset, Cnt, Clk);
output Ld, ShR, Rst;
input Reset, Cnt, Clk;
reg [1:0] NS;
reg [1:0] CS;
assign Ld = (CS == 0)? 1: 0;
assign Rst = (CS==0)? 1: 0;
assign ShR = (CS == 1)? 1:0;
always @(Reset or Cnt)
    Case (CS)
        0: NS = (Reset) ? 0 : 1;
        1: NS = (!Cnt)? 1: 2;
        2: NS = (!Reset) ? 2:0;
    end case;
always @(posedge Clk)
    CS <= NS;
endmodule
```