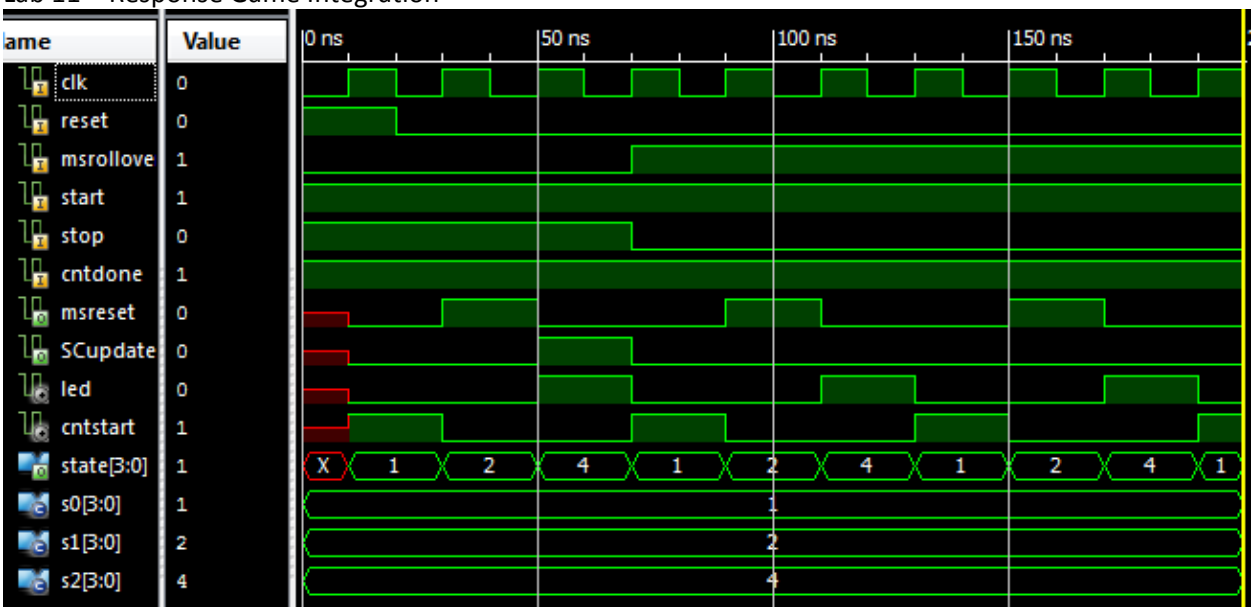
David DeGraw
3/31/2015
Lab 11 – Response Game Integration
Control FSM:

```
module stateMachine(clk, reset, stop, start, cntdone, SCupdate, msrollover, msreset, state);
   input clk, reset, msrollover, start, stop, cntdone;
   output msreset;
   output SCupdate;

   output reg [3:0] state;

   parameter s0 = 4'b001;
   parameter s1 = 4'b010;
   parameter s2 = 4'b100;

   always @ (posedge clk)
   begin
   if (reset)
      state <= s0;
   else
      case (state)

      s0 : if (start)
         state <= s1;

      s1 : if (cntdone)
         state <= s2;

      s2 : if (stop | msrollover)
         state <= s0;

      endcase
   end

   assign update = (state == s2 & stop) ? 1'b1 : 1'b0;

   assign led = (state == s2) ? 1'b1 : 1'b0;

   assign msreset = (state == s1) ? 1'b1 : 1'b0;

   assign cntstart = (state == s0 & start) ? 1'b1 : 1'b0;


endmodule
```

```
wave add / -radix hex

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns
isim force add start 1
isim force add reset 1 -time 0 -value 0 -time 20ns
isim force add cntdone 1
isim force add stop 1 -time 0 -value 0 -time 70ns
isim force add msrollover 0 -time 0 -value 1 -time 70ns

run 200ns
```

David DeGraw
3/31/2015
Lab 11 – Response Game Integration

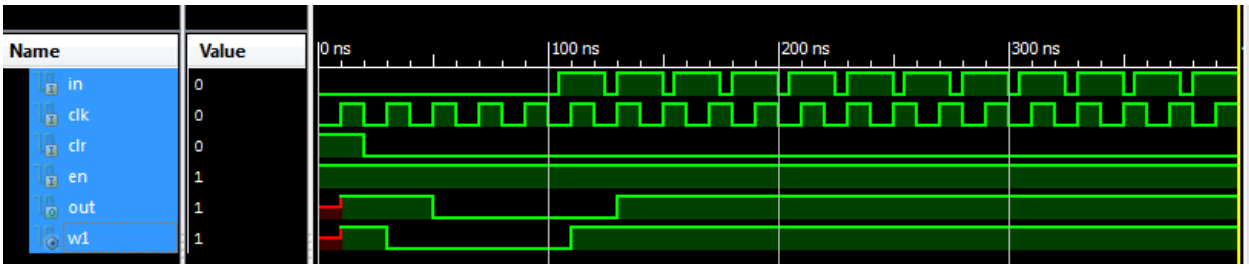


David DeGraw
3/31/2015
Lab 11 – Response Game Integration

Synchronizer:

```
module syncro(out, in, clk, clr, en);
   input in, clk, clr, en;
   output out;

   wire w1;

   FF_DCE ff1(w1, clk, in, clr, en);
   FF_DCE ff2(out, clk, w1, clr, en);

endmodule
```

```
wave add / -radix hex

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns
isim force add clr 1 -time 0 -value 0 -time 20ns
isim force add en 1
isim force add in 0 -time 0 -value 1 -time 105ns -repeat 25ns

run 400ns
```

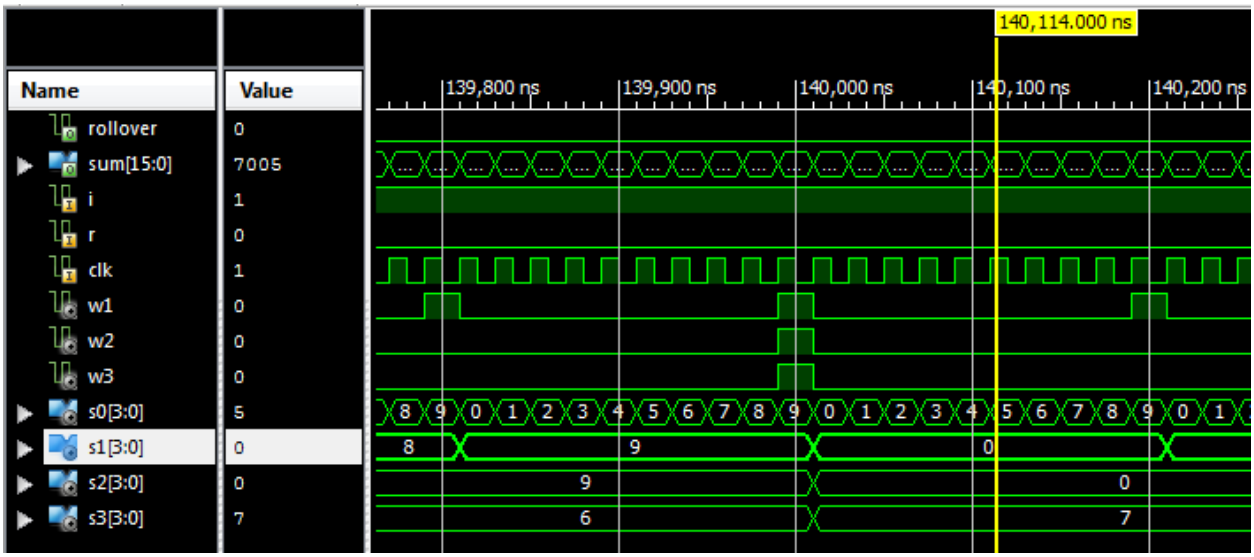


David DeGraw
3/31/2015
Lab 11 – Response Game Integration

MS Counter:

```
module mscounter(rollover, sum, i, r, clk);
   output rollover;
   output [15:0] sum;
   input i, r, clk;

   wire w1, w2, w3;
   wire [3:0] s0, s1, s2, s3;

   mod10counter m1(s0, w1, i, clk, r);
   mod10counter m2(s1, w2, w1, clk, r);
   mod10counter m3(s2, w3, w2, clk, r);
   mod10counter m4(s3, rollover, w3, clk, r);

   assign sum = {s3, s2, s1, s0};

endmodule
```

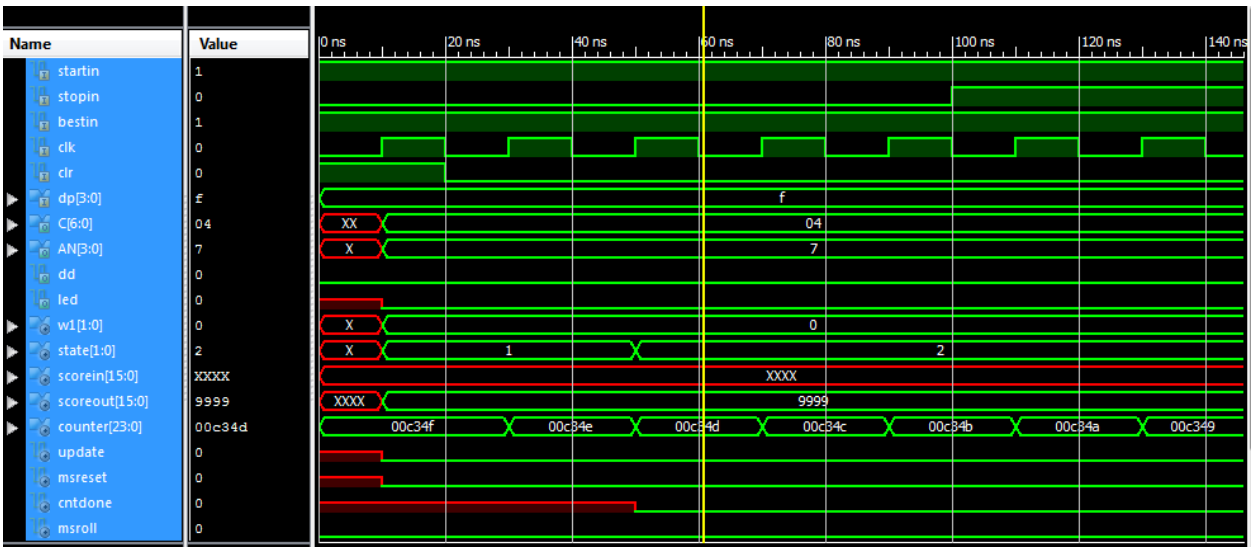
```
wave add / -radix hex

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns
isim force add r 1 -time 0 -value 0 -time 20ns
isim force add i 1

run 200000ns
```



David DeGraw
3/31/2015
Lab 11 – Response Game Integration

Response Game:

```
module game(C, dd, AN, led, startin, stopin, bestin, clr, dp, clk);
   input startin, stopin, bestin, clk, clr;
   input [3:0]dp;
   output [6:0] C;
   output [3:0] AN;
   output dd, led;

   wire [1:0] w1;
   wire [1:0] state;
   wire [15:0] scorein, scoreout;
   wire [23:0] counter;
   wire update, msreset, cntdone, msroll, cntstart, start, best, stop, zero, tp;

   syncro sb1(start, startin, clk, 1'b0, 1'b1);
   syncro sb2(stop, stopin, clk, 1'b0, 1'b1);
   syncro sb3(best, bestin, clk, 1'b0, 1'b1);

   stateMachine fsm(clk, clr, stop, start, cntdone, update, msroll, msreset, sta

   mscounter mscnt(msroll, scorein, zero, msreset, clk);

   delay d_timer(clr, zero, clk, cntstart, cntdone);

   scoringUnit scoring(scorein, update, scoreout, clk, best, clr);

   prog_timer p_timer(clk, clr, 1'b1, 24'd50000, counter, zero, tp);

   controller47 seg_con(dp, AN, scoreout[15:12], scoreout[11:8], scoreout[7:4],

endmodule
```



David DeGraw
3/31/2015
Lab 11 – Response Game Integration

```
wave add / -radix hex

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns
isim force add clr 1 -time 0 -value 0 -time 20ns
isim force add startin 1
isim force add stopin 0 - time 50ns -value 1 -time 100ns
isim force add bestin 1
isim force add dp 1111

run 200000ns
```

Anomalies:

The bit widths on the pdf instructions were wrong. It would say like [16:0] (for a 16 bit number), but it should say [15:0].