Seung Ki Lee

CSE 3381 <Digital Logic Design N14 1172>

Post-Lab 08

1. Write Up

   After the reading, the lab itself was rather not difficult. While I struggled a bit in designing the top level, reference to my past materials greatly aided me. Adder_top.v seems to be a great reference when I need something to start me off for top level.

   For Extra Credit,

   The numbers generated in using PRNG is not completely random and repeats itself in a specific cycle. This is due to the initial condition that is fed to the algorithm, or seeds. The periodic output only provides stream of numbers that looks random yet are actually deterministic.

2. Source Code (Dff.v, Dffsc.v, prng.v, prng_top.v)

Dff.v

```
module Dff (
      input wire clk,
      input wire clr,
      input wire D,
      output reg q
      );
      always @(posedge clk or posedge clr)
            if (clr == 1)
                  q <= 0;
            else
                  q <= D;
endmodule
```

Dffsc.v

```
module Dffsc (
      input wire clk,
```

```
      input wire clr,
      input wire D,
      output reg q
      );
      always @(posedge clk or posedge clr)
            if (clr == 1)
                  q <= 1;
            else
                  q <= D;
endmodule
```

Prng.v

```
module prng (
input wire clk,
input wire clr,
output reg [3:0]q
);
always @(posedge clk or posedge clr)
      if (clr == 1)
            q <= 1;
      else
            begin
                  q[3] <= q[3] ^ q[0];
                  q[2:0] <= q[3:1];
            end
endmodule
```

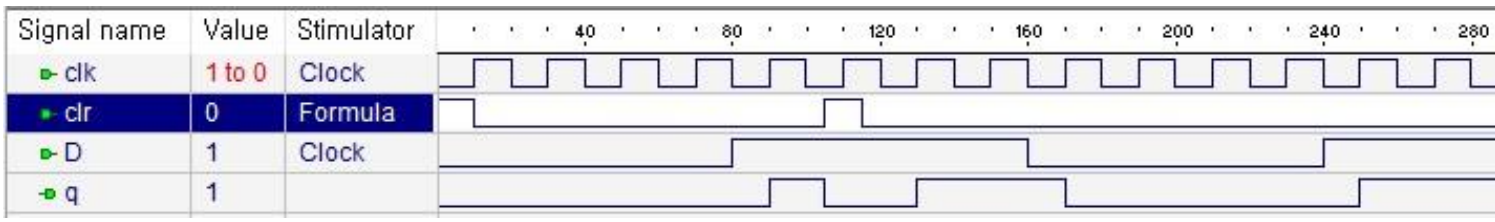
prng_top.v

```
module prng_top (
input wire [0:0] btn,
input wire mclk,
output wire[3:0] ld
```

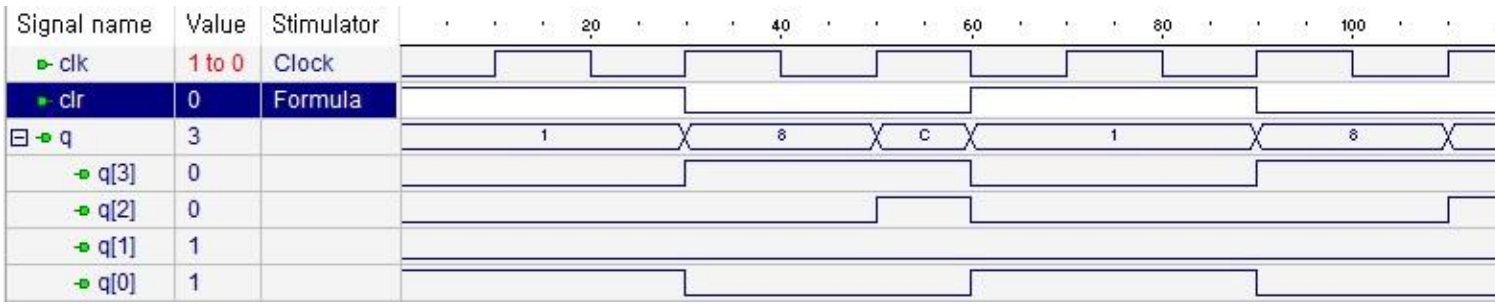
```
        );
    prng U1(
    .clr(btn[0]),
    .clk(mclk),
    .q(ld[3:0])
    );
    endmodule
```

3. Simulation (dff.v, prng.v)

Dff.v



Prng.v



4. Demo Code

Demo-Code: 9195