Digitális Technika

Dr. Oniga István
Debreceni Egyetem, Informatikai Kar

10. Laboratóriumi gyakorlat

- Regiszterek
  - 4 bites regiszter felfutó órajellelre működő, aszinkron PRESET és órajel engedélyezővel
  - 8 bites balra léptető regiszter
  - 8 bites balra léptető regiszter, aszinkron párhuzamos töltéssel
  - 8 bites két irányba léptető regiszter, soros be- és párhuzamos ki- menettel
  - Gyűrűs számláló
  - Johnson számláló

Lab10_1: 4 bites regiszter felfutó órajellel, aszinkron PRESET és órajel engedélyezővel

- Hozunk létre egy új projektet (Lab10_1)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab10_1.v).
- Specifikáljuk regiszter működését.
- Adjunk hozzá egy Verilog test fixture fájlt.
- Gerjesztő jelek specifikálása. Funkcionális kód ellenőrzése szimulációval.

```
//
// 4-bit Register with Positive-Edge Clock, Asynchronous Set and Clock Enable
//
module v_registers_5 (input C, CE, PRE, input [3:0 ] D,
      output reg [3:0] Q);

      always @(posedge C or posedge PRE)
      begin
            if (PRE)
            Q <= 4'b1111;
            else if (CE)
            Q <= D;
      end
endmodule
```

Lab10_2: 8 bites balra léptető regiszter

- Hozunk létre egy új projektet (Lab10_2)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab10_2.v).
- Specifikáljuk regiszter működését.
- Funkcionális kód ellenőrzése szimulációval. (Nem kötelező)
- Adjunk hozza egy órajel osztót, és ezzel generáljuk a regiszter órajelet (kb. 1 Hz)
- Kössük össze a top modulban a két module-t az alábbi ábra szerint
- UCF fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

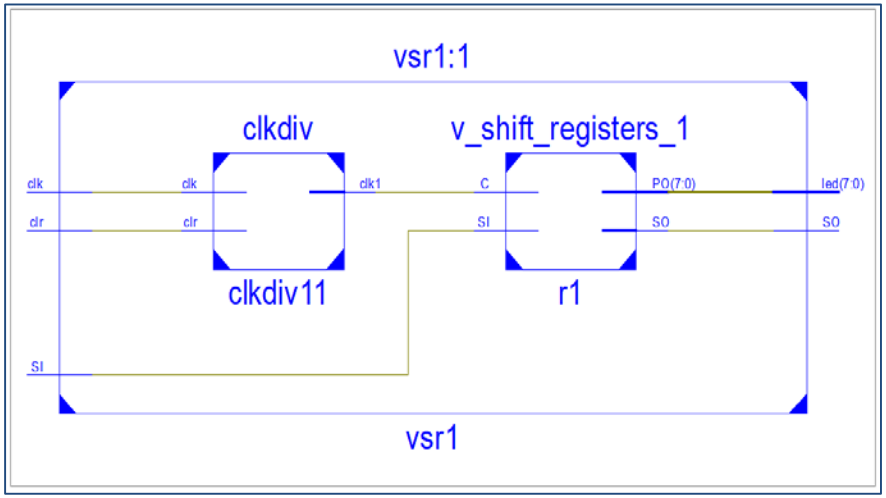
```
// 8-bit Shift-Left Register with Positive-Edge
// Clock, Serial In, and Serial Out

module v_shift_registers_1 (input C, SI,
output reg [7:0] PO, output SO);

    always @(posedge C)
    begin
          PO <= PO << 1;
          PO[0] <= SI;
    end

    assign SO = PO[7];

endmodule
```



Lab10_3: 8 bites balra léptető regiszter, aszinkron párhuzamos töltéssel

- Hozunk létre egy új projektet (Lab10_3)
- Az előző feladathoz hasonlóan implementáljuk és teszteljük a működését.

```
// 8-bit Shift-Left Register with Positive-Edge Clock,
// Asynchronous Parallel Load, Serial In, and Serial Out

module v_shift_registers_6 (C, ALOAD, SI, D, SO);
    input C,SI,ALOAD;
    input [7:0] D;
    output SO;
    reg [7:0] tmp;

    always @(posedge C or posedge ALOAD)
    begin
        if (ALOAD)
            tmp <= D;
        else
            tmp <= {tmp[6:0], SI};
    end

    assign SO = tmp[7];

endmodule
```

Lab10_4: 8 bites két irányba léptető regiszter, soros be- és párhuzamos ki- menettel

- Hozunk létre egy új projektet (Lab10_4)
- Az előző feladathoz hasonlóan implementáljuk és teszteljük a működését.

```
// 8-bit Shift-Left/Shift-Right Register with Positive-Edge Clock,
// Serial In, and Parallel Out

module v_shift_registers_8 (C, SI, LEFT_RIGHT, PO);
    input C,SI,LEFT_RIGHT;
    output PO;
    reg [7:0] Q;

    always @(posedge C)
    begin
        if (LEFT_RIGHT==1'b0)
          Q <= {Q[6:0], SI};
        else
          Q <= {SI, Q[7:1]};
    end

    assign PO = Q;

endmodule
```

Lab10_5: Gyűrűs számláló

- Hozunk létre egy új projektet (Lab10_5)
- Az előző feladathoz hasonlóan implementáljuk és teszteljük a működését.

```
module ring_count(input clk, load, output reg [7:0]q);

  always @(posedge clk)
       if(load==1)
            q<=8'b10000000;
        else
            begin
                q <= {q[0], q[7:1]};
            end
endmodule
```

Lab10_6: Johnson számláló

- Hozunk létre egy új projektet (Lab10_6)
- Az előző feladathoz hasonlóan implementáljuk és teszteljük a működését.

```
module johnson_count(input clk, clr, output reg [7:0]q);

  always @(posedge clk)
       if(clr==1)
            q<=8'b00000000;
        else
            begin
                q <= {~q[0], q[7:1]};
            end
endmodule
```