Digitális Technika

Dr. Oniga István
Debreceni Egyetem, Informatikai Kar

8. Laboratóriumi gyakorlat

Egyszerű sorrendi hálózatok

- D latch
- D FF
- T FF

Lab8_1a: D latch

- Hozunk létre egy új projektet (Lab8_1)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab8_1.v).
- Specifikáljuk egy kapuzott D-latch működését.
- Funkcionális kód ellenőrzése szimulációval.

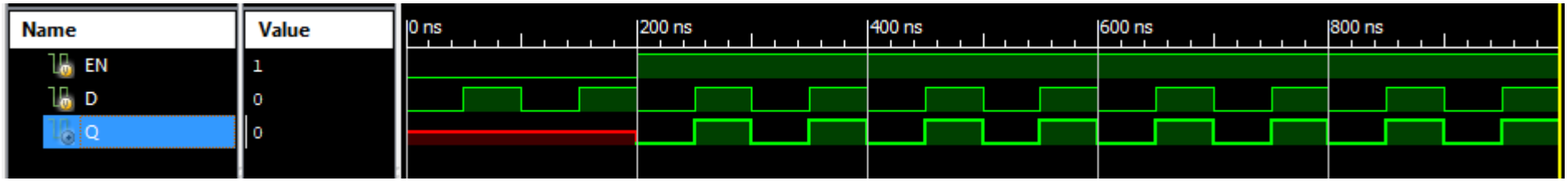
D-latch Verilog kódja

```
module v_Dlatch_G (input EN, D, output reg Q);
    always @(EN or D)
        begin
                if (EN)
                        Q = D;
end endmodule
```

Gerjesztő jelek specifikálása

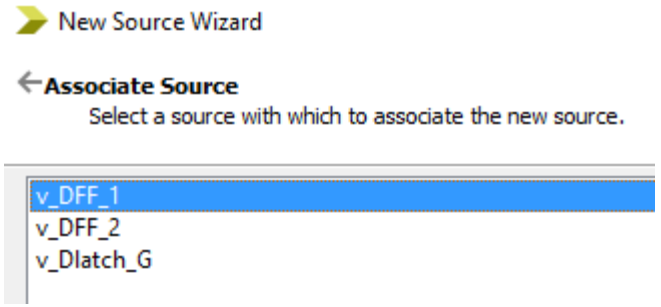
```
    // Add stimulus here
    #100;
    EN = 1;

end
always #50
    D <= ~ D;
```



Lab8_1b és c: D flip-flop

- Adjunk hozzá a Lab8_1.v VERILOG forrásfájlhoz a D-FF leíró Verilog kód részleteket.
- Funkcionális kód ellenőrzése szimulációval (két külön esett). Mind két esetben adjuk hozzá egy Verilog test fixture fájlt figyelve melyik UUT van kiválasztva



D-FF (órajel felfutó élére)

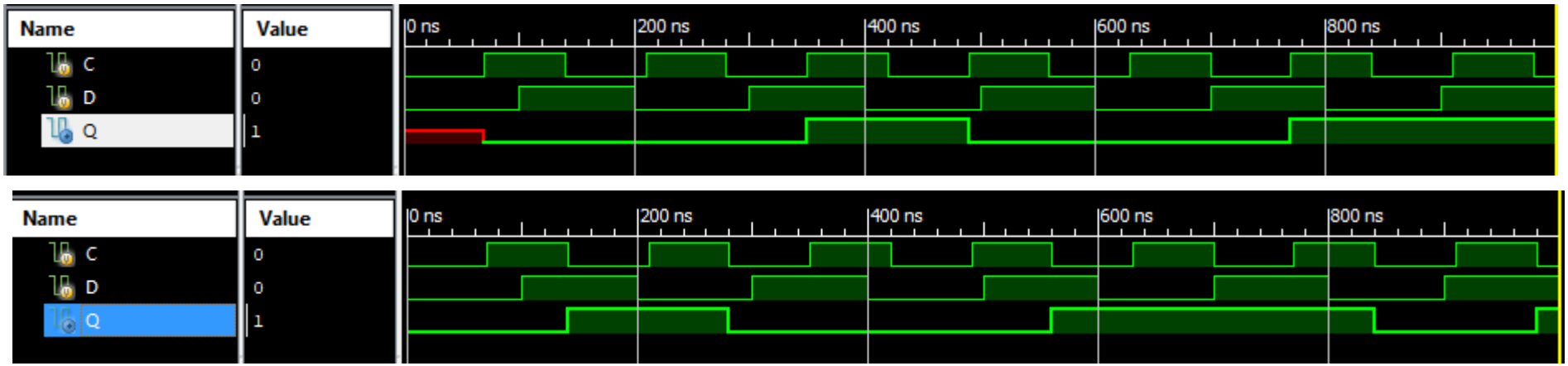
```
module v_DFF_1 (input C, D,
        output reg Q);
always @(posedge C)
        Q <= D;
endmodule
```

D-FF (órajel lefutó élére)

```
module v_DFF_2 (input C, D,
        output reg Q);
always @(negedgeC)
        Q <= D;
endmodule
```

Gerjesztő jelek specifikálása

```
    // Add stimulus here
    #100;
end
always #100
    D <= ~ D;
always #70
    C <= ~ C;
```



Lab8_2a-d: Flip-Flopok

Lab8_2a

- Hozunk létre egy új projektet (Lab8_2)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab8_2.v).
- Specifikáljuk a következő FF működését (a kódok letölthetők a labor weboldaláról).
- Funkcionális kód ellenőrzése szimulációval.

```
//
// Flip-Flop with Negative-Edge Clock and Asynchronous Clear
//

module v_registers_2 (input C, D, CLR, output reg Q);

    always @(negedge C or posedge CLR)
    begin
        if (CLR)
            Q <= 1'b0;
        else
            Q <= D;
    end

endmodule
```

Lab8_2a-d: Flip-Flopok

Lab8_2b

- Hozunk létre egy új projektet (Lab8_2)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab8_2.v).
- Specifikáljuk a következő FF működését (a kódok letölthetők a labor weboldaláról).
- Funkcionális kód ellenőrzése szimulációval.

```
//
// Flip-Flop with Positive-Edge Clock
//

module v_registers_1 (input C, D, output reg Q);

    always @(posedge C)
    begin
        Q <= D;
    end

endmodule
```

Lab8_2a-d: Flip-Flopok

Lab8_2c

- Hozunk létre egy új projektet (Lab8_2)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab8_2.v).
- Specifikáljuk a következő FF működését (a kódok letölthetők a labor weboldaláról).
- Funkcionális kód ellenőrzése szimulációval.

```
//
// Flip-Flop with Positive-Edge Clock and Clock Enable
//

module v_registers_4 (input C, D, CE, output reg Q);

    always @(posedge C)
    begin
        if (CE)
            Q <= D;
    end

endmodule
```

Lab8_2a-d: Flip-Flopok

Lab8_2d

- Hozunk létre egy új projektet (Lab8_2)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab8_2.v).
- Specifikáljuk a következő FF működését (a kódok letölthetők a labor weboldaláról).
- Funkcionális kód ellenőrzése szimulációval.

```
//
// Flip-Flop with Positive-Edge Clock and Synchronous Set
//

module v_registers_3 (input C, D, S, output reg Q);

    always @(posedge C)
    begin
        if (S)
            Q <= 1'b1;
        else
            Q <= D;
        end

endmodule
```

Lab8_3a: T-FF

- Hozunk létre egy új projektet (Lab8_3)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab8_3a.v).
- Specifikáljuk a következő aszinkron törlő jellel működő T-FF -ot .
- Funkcionális kód ellenőrzése szimulációval.

```
//-----------------------------------------------------
// T flip-flop async reset
//-----------------------------------------------------
module tff_async_reset ( input data, clk, reset, output reg q);

always @ ( posedge clk or negedge reset)
if (~reset) begin
  q <= 1'b0;
end else if (data) begin
  q <= !q;
end

endmodule
```

Lab8_3b: T-FF

- Hozunk létre egy új projektet (Lab8_3)
- Adjunk hozzá egy új VERILOG forrásfájlt (Lab8_3b.v).
- Specifikáljuk a következő szinkron törlő jellel működő T-FF -ot.
- Funkcionális kód ellenőrzése szimulációval.

```
//-------------------------------------------------------
// T flip-flop sync reset
//-------------------------------------------------------
module tff_sync_reset ( input data, clk, reset, output reg q);

always @ ( posedge clk)
if (~reset) begin
  q <= 1'b0;
end else if (data) begin
  q <= !q;
end

endmodule
```

Kérdések

1. Mi a különbség a D latch és a D FF között és melyik kód részben van ez leírva?
2. Mit jelent a #50 utasítás a testfixture fájlban ?
3. Milyen működést ir le a következő kód:
    1. Always @ #50
    2. D <= ~ D
4. A #50 utasítást lehet implementálni?
5. Miben különbözik egy aszinkron illetve egy szinkron törlő jellel működő T-FF leírása?