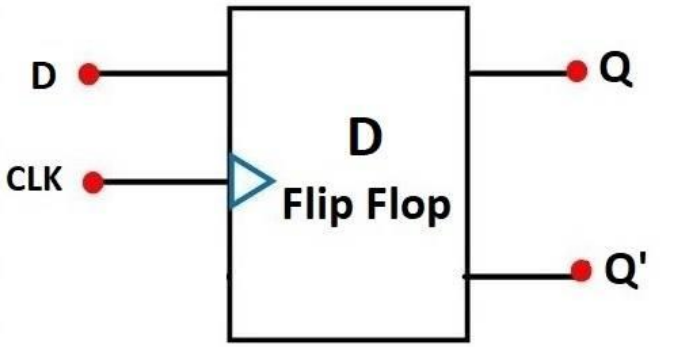
ÖRNEK 1: Sırasıyla “D, T, JK ve SR” flip flop tasarlayınız ve simüle ediniz.

D flip flop için:

Truth Table of DFlip Flop

D	CLK	Q	Q`
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0



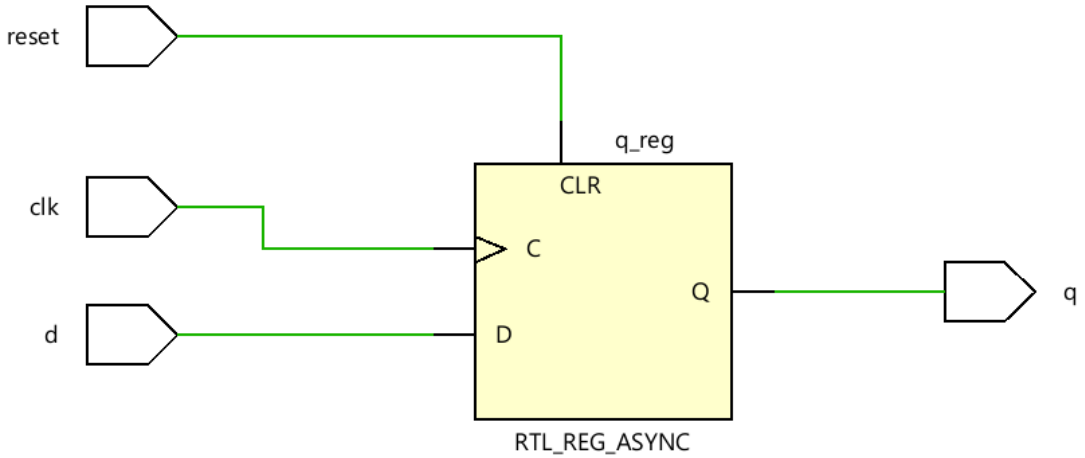
hackatronic.com

Design kodunu yazacak olursak:

```
module d_flipflop(
    input clk,
    input reset,
    input d,
    output reg q // reg ifadesi çıkış değerinin bir değer tutabileceğini
belirtir. sequential logic için ve always blokları için kullanılır.

    );

    // always@() yapısı sequrntial logic için kullanılır.
    // sequential yapılarda = değil <= kullanılır. Bunun sebebi tüm ifadeleri
eş zamanlı çalıştırmasıdır.
    always @(posedge clk or posedge reset) begin // yükselen kenarda çallışır.
        if(reset) begin
            q <= 1'b0;
        end
        else begin
            q <= d;
        end
    end

endmodule
```

(NOT: Input / Output tanımlamaları yaparken inputlar için herhangi bir tanımlama yapmazsak System Verilog otomatik olarak “wire” algılar. Bu kod özelinde örnek doğrudur ve çalışır.)

Şemamız şöyle olacaktır:

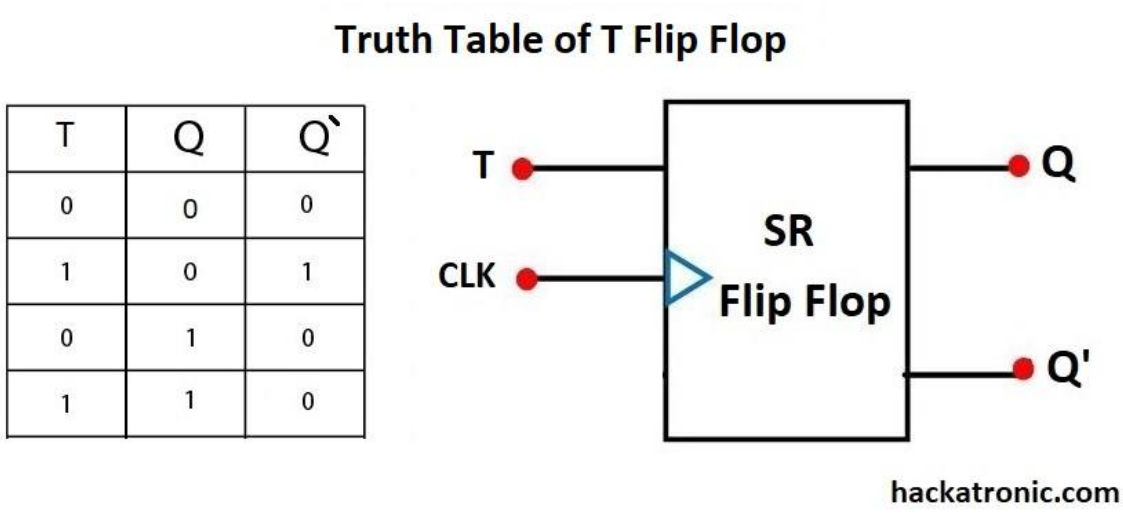


Testbanch kodu için:

```
module tb_d_flipflop();
logic clk, reset, d, q;

d_flipflop d_flipflop_Inst(.*);

    always #5 clk <= ~clk; // Her 5ns de bir clk sinyalini değiştirir (tersini
alır).

    initial begin
        clk = 0; // clk sinyalini başlangıçta 0 yapar.
        reset <= 1;
        d <= 0;

        #20
        reset <= 0; #10;
        d <= 1; #10;
        d <= 0; #10;
        d <= 1; #10;
        d <= 0; #10;
        d <= 1; #10;
        d <= 0;

        #50
        $stop;
    end
endmodule
```

Simüle ettiğimizde dalga formu şöyle gözükür:

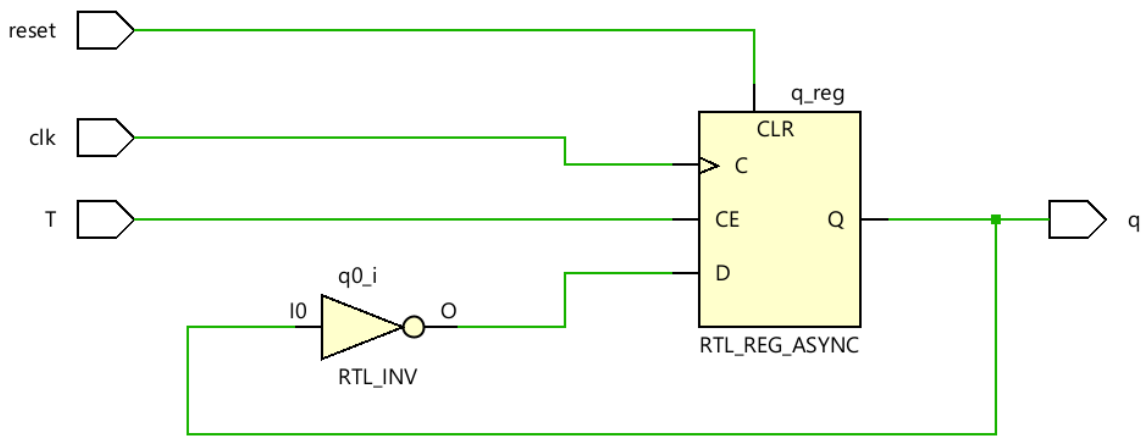
T flip flop için:



Design kod:

```
module t_flipflop(
    input logic clk,
    input logic reset, // Reset sinyali
    input logic T,     // Toggle sinyali
    output reg q
);

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            q <= 1'b0;  // Reset durumunda Q sıfırlanır
        end else begin
            if (T) begin
                q <= ~q;  // Toggle durumu
            end
            //T = 0 ise Q değişmez
        end
    end
endmodule
```

Şema:

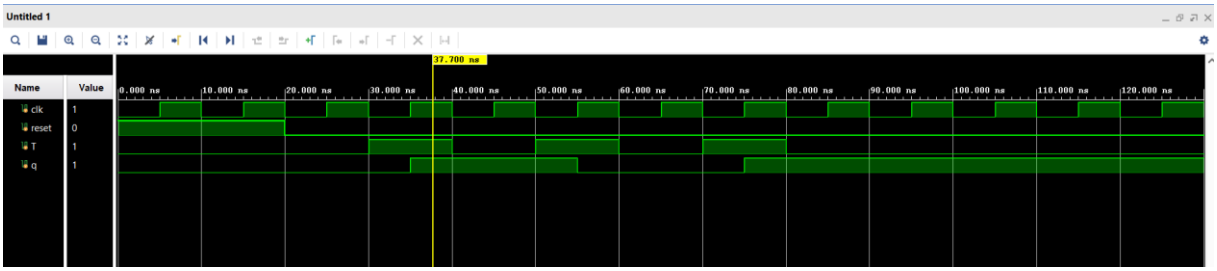


(NOT: Direkt T ff gelmemesinin sebebi çoğu FPGA ve ASIC kitaplığında doğrudan bir T ff bulunmamasıdır. Aynı davranış bir not ve D ff ile oluşturulabilir.)

Testbanch kodu için:

```
module tb_t_flipflop();
logic clk, reset, T, q;

t_flipflop t_flipflop_Inst(.*);

    always #5 clk <= ~clk; // Her 5ns de bir clk sinyalini değiştirir (tersini
alır).

    initial begin
        clk = 0; // clk sinyalini başlangıçta 0 yapar.
        reset <= 1;
        T <= 0;

        #20
        reset <= 0; #10;
        T <= 1; #10;
        T <= 0; #10;
        T <= 1; #10;
        T <= 0; #10;
        T <= 1; #10;
        T <= 0;

        #50
        $stop;
    end
endmodule
```

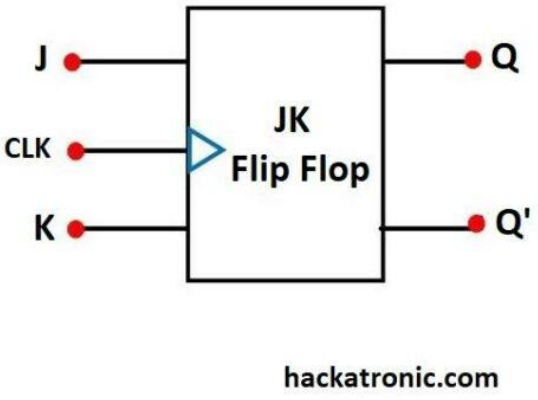
Simülasyon sonucumuz da aşağıdaki gibidir:



JK flip flop için:

Truth Table of JK Flip Flop

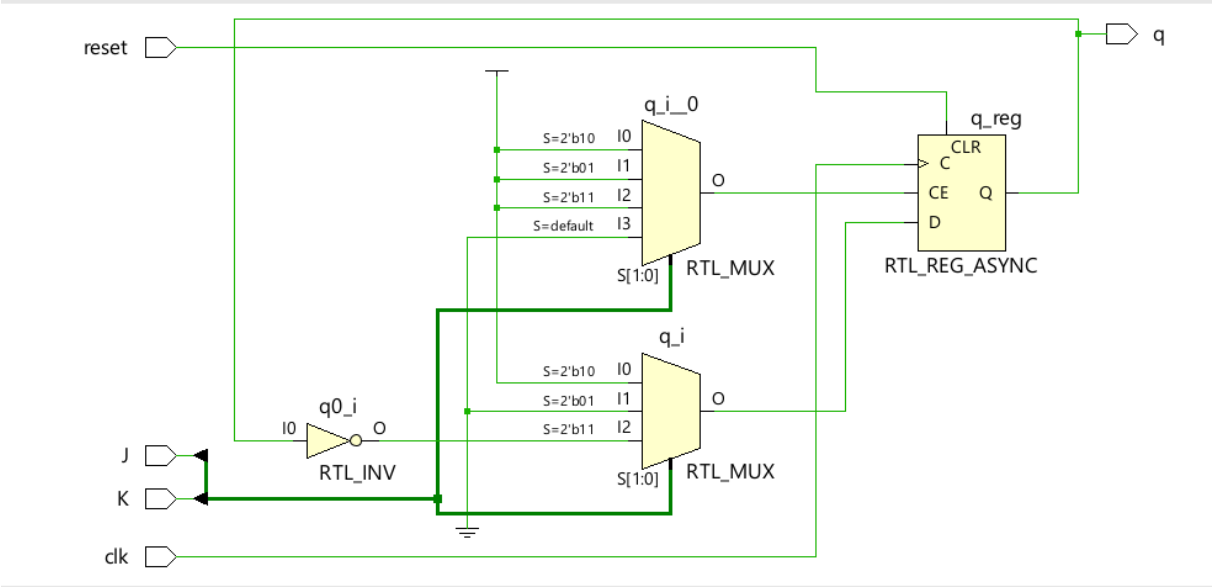
C	J	K	Q	Q`
HIGH	0	0	Latch	Latch
HIGH	0	1	0	1
HIGH	1	0	1	0
HIGH	1	1	Toggle	Toggle
LOW	0	0	Latch	Latch
LOW	0	1	Latch	Latch
LOW	1	0	Latch	Latch
LOW	1	1	Latch	Latch



Design kod:

```
module jk_flipflop(
    input logic clk,
    input logic reset, // Reset sinyali
    input logic J,     // J girişi
    input logic K,     // K girişi
    output logic q
);

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            q <= 1'b0;  // Reset durumunda Q sıfırlanır
        end else begin
            case ({J, K})
                2'b10: q <= 1'b1;       // Set durumu
                2'b01: q <= 1'b0;       // Reset durumu
                2'b11: q <= ~q;         // Toggle durumu
                default: q <= q;        // Q değişmez
            endcase
        end
    end
endmodule
```

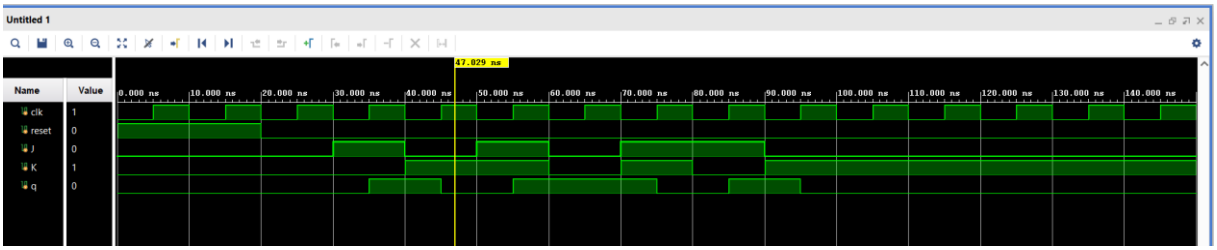
Şemamız şu şekildedir:



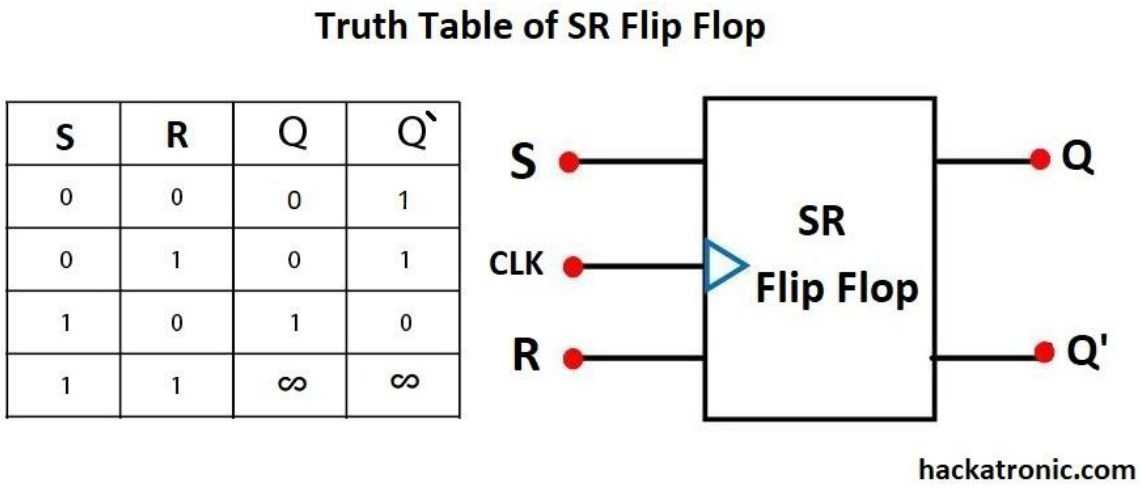
Testbanch kod:

```
module tb_jk_flipflop();
logic clk, reset, J, K, q;

jk_flipflop jk_flipflop_Inst(.*);

always #5 clk <= ~clk;

initial begin
    clk = 0;
    reset <= 1;
    J <= 0;
    K <= 0;

    #20
    reset <= 0; #10; // Reset kaldırılır.
    J <= 1; K <= 0; #10; // Set durumu
    J <= 0; K <= 1; #10; // Reset durumu
    J <= 1; K <= 1; #10; // Toggle durumu
    J <= 0; K <= 0; #10; // No change durumu
    J <= 1; K <= 1; #10; // Toggle durumu
    J <= 1; K <= 0; #10; // Set durumu
    J <= 0; K <= 1; #10; // Reset durumu

    #50
    $stop;
end
endmodule
```

Simülasyon sonucu:



SR flip flop için:

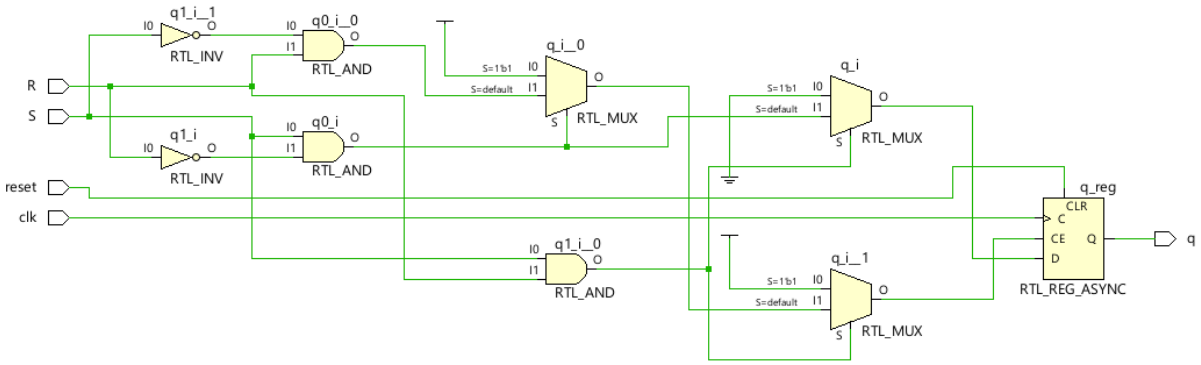


Design kod için:

```
module sr_flipflop(
    input wire clk,
    input wire reset,
    input wire S,
    input wire R,
    output reg q
);

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            q <= 1'b0;
        end else begin
            if (S && R)
            begin
                // INVALID durumu: S ve R aynı anda 1
                q <= 1'b0; //INVALID durumu için tanımlı bir durum atayalım (0 
verelim) ve tcl console'da hata mesajını yazsın.
                $display("ERROR: Invalid state (S = 1, R = 1) at time %t", 
$time);
            end
            else if (S && !R)
            begin
                q <= 1'b1;  // Set durumu
```

```
            end
            else if (!S && R)
            begin
                q <= 1'b0;  // Reset durumu
            end
            // S = 0 R = 0 ise Q değişmez
        end
    end
endmodule
```

Design çıktısı şu şekildedir:



Testbanch için:

```
module tb_sr_flipflop();
logic clk, reset, S, R, q;

// SR flip-flop instance
sr_flipflop sr_flipflop_Inst(.*);

always #5 clk <= ~clk; // 5 ns'de bir clk sinyalini değiştir.

initial begin
    clk = 0; // clk sinyalini başlangıçta 0 yapar.
    reset <= 1;
    S <= 0;
    R <= 0;

    #20
    reset <= 0; #10; // Reset kaldırılır.
    S <= 1; R <= 0; #10; // Set durumu
    S <= 0; R <= 1; #10; // Reset durumu
    S <= 0; R <= 0; #10; // No change durumu
    S <= 1; R <= 1; #10; // Invalid durumu (SR flip-flop için geçersizdir.)
    S <= 1; R <= 0; #10; // Set durumu
    S <= 0; R <= 1; #10; // Reset durumu
```

```
    #50
    $stop; // Simülasyonu sonlandır.
end
endmodule
```

Simülasyon çıktısı aşağıdaki gibidir: