FPGA INTERNSHIP ASSIGNMENT

Submitted By : Sankaliya Mirali

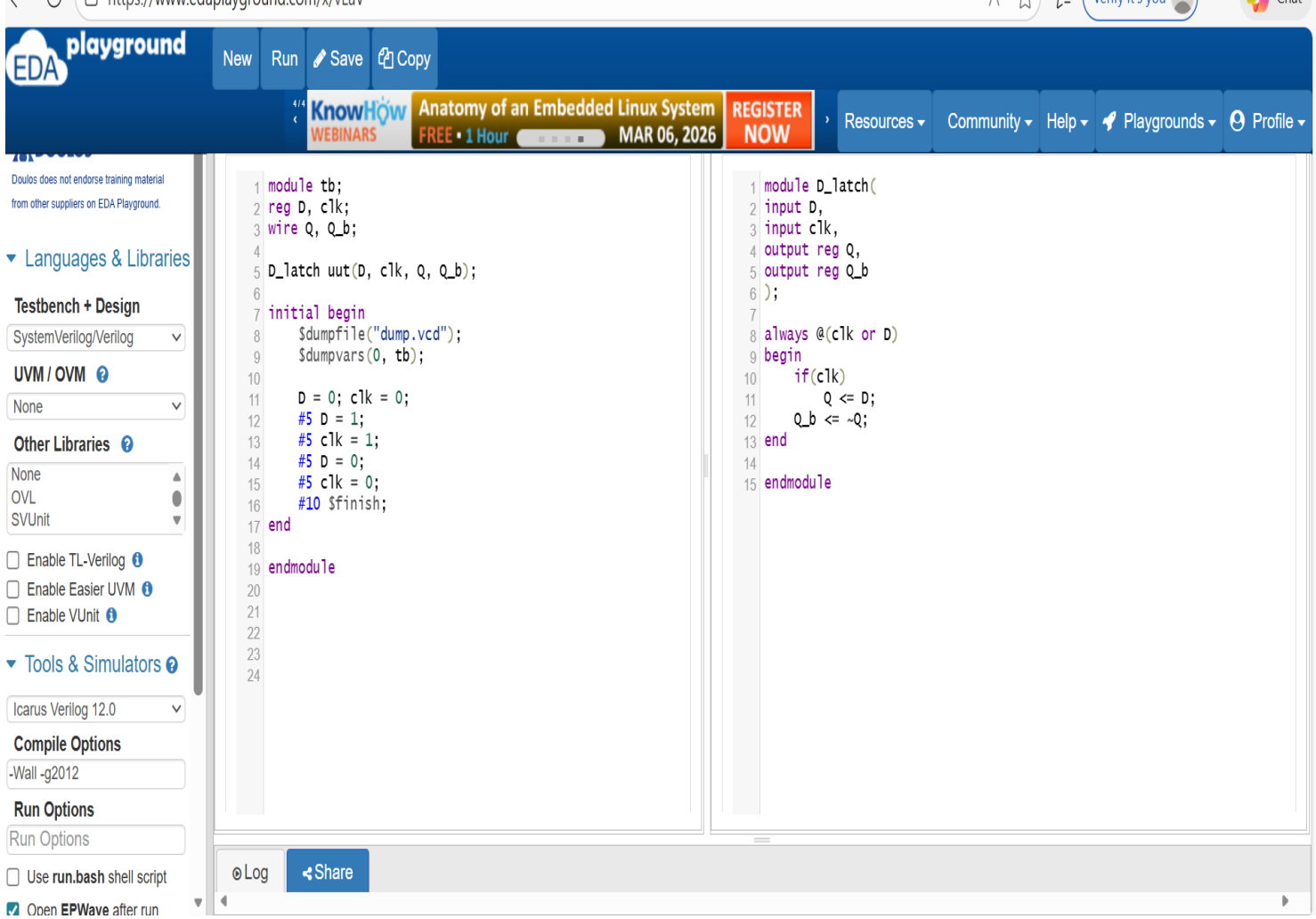
Submitted To : Anoushka ma’am

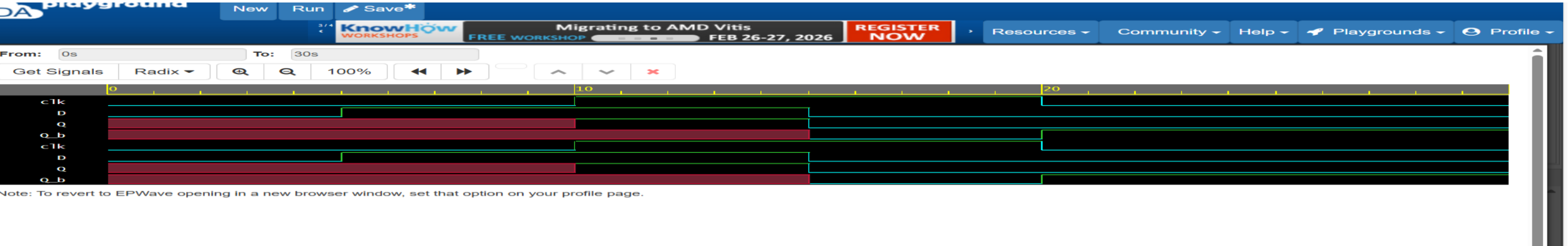
Topic : Flip-Flop & Counter Coding

1. D Flip-Flop

Explanation:

This waveform shows the behavior of a D flip-flop where the output depends on the clock signal. When the clock is active, the output follows the input value. When the clock becomes inactive, the flip-flop stores the last value and ignores further changes in the input. This allows the circuit to hold data for later use.

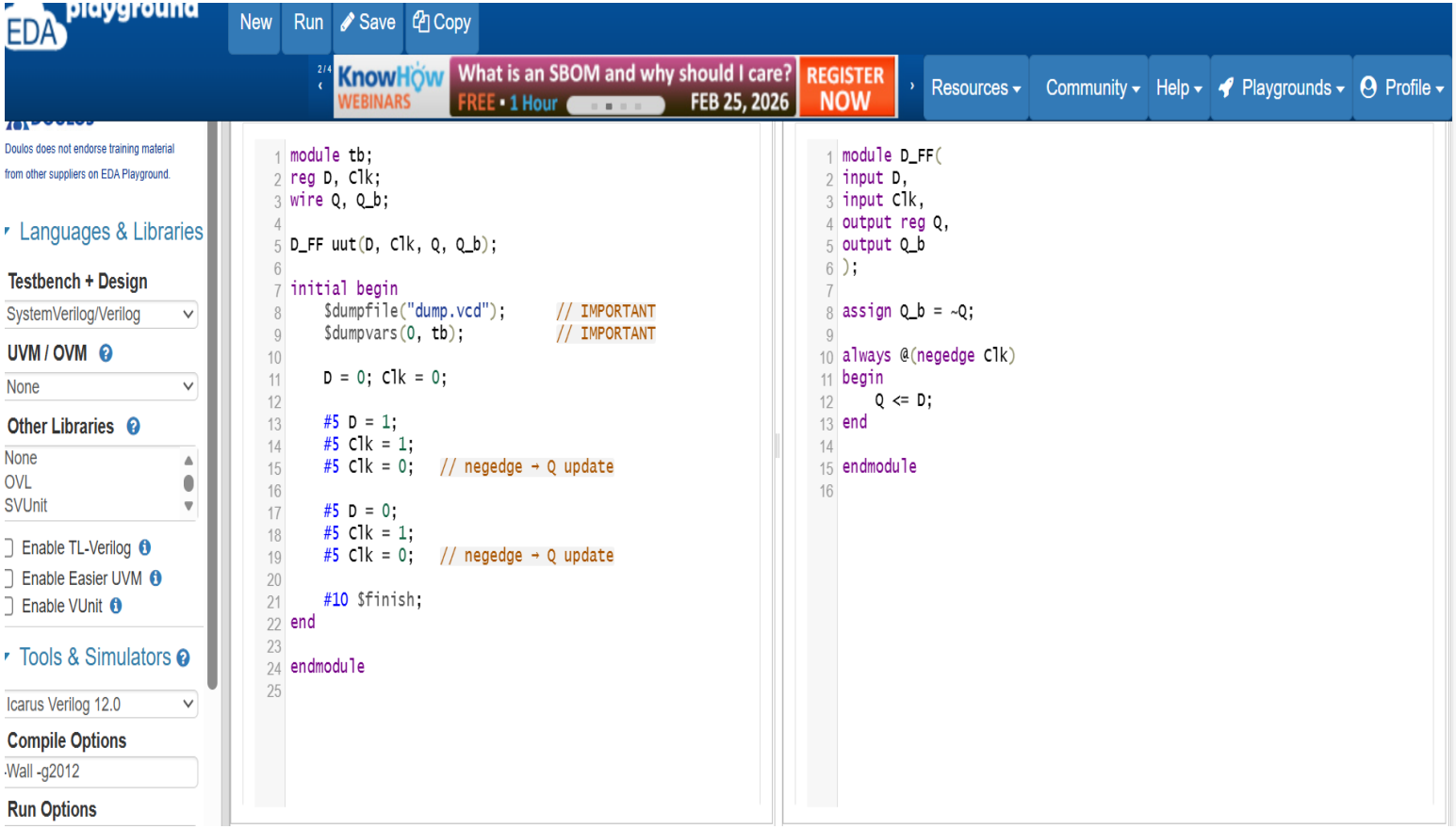


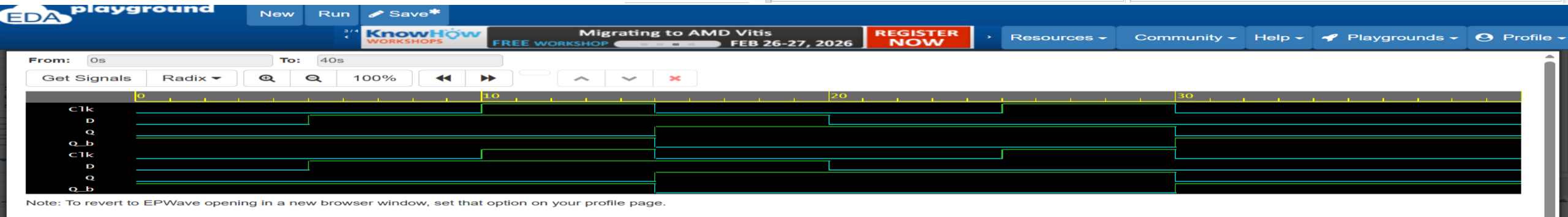


2.D-Flip-Flop(Negative edge Triggered)

Explanation:

A negative edge triggered D flip-flop stores the input value only when the clock changes from high to low (falling edge). At that moment, the output updates according to the input and then remains unchanged until the next falling edge. Any changes in the input between clock edges do not affect the output. This provides stable operation in digital circuits.

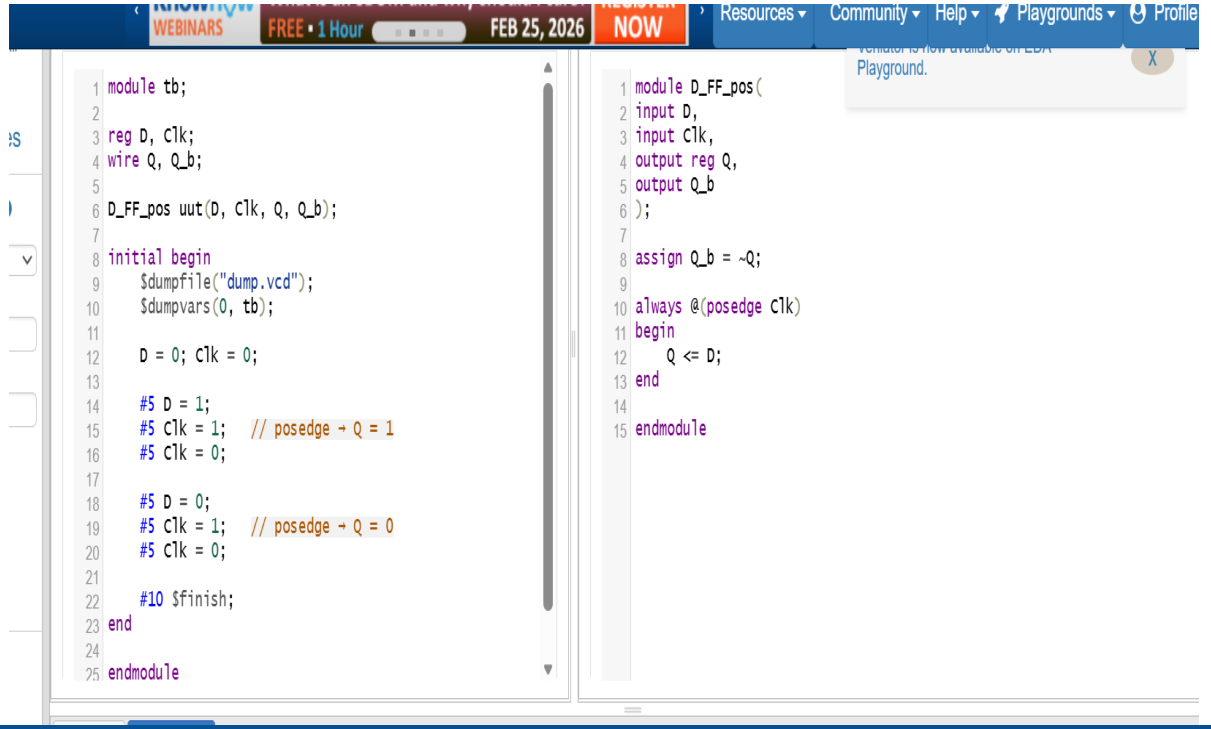


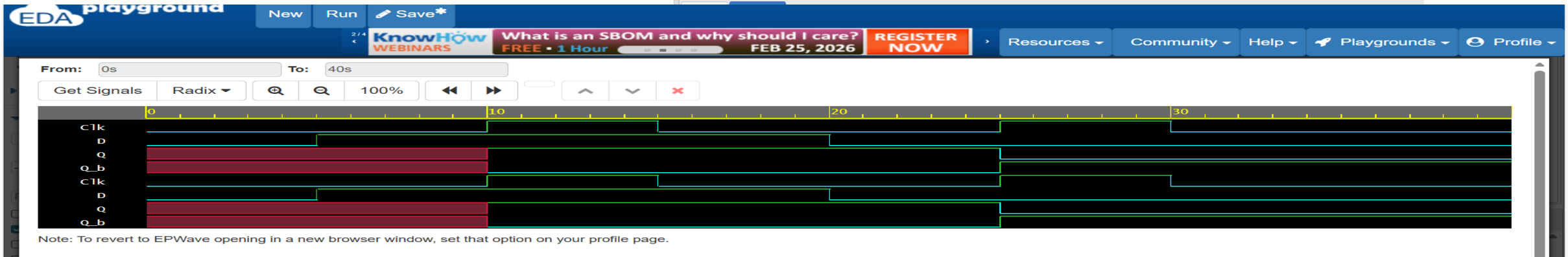


3.Positive Edge D Flip-Flop

EXPLANATION:

In this experiment, a positive edge triggered D Flip-Flop was simulated. The output updates only at the rising edge of the clock. The waveform clearly shows the output changing when the clock goes from low to high.





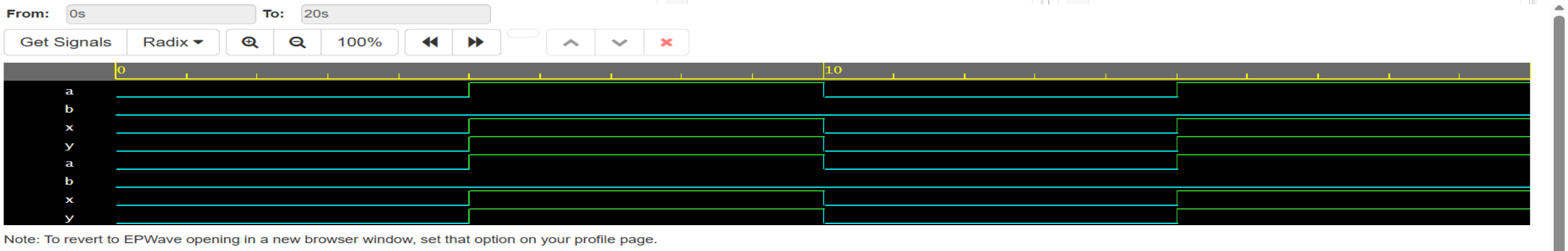
4.Blocking Assignment

EXPLANATION:

Blocking assignment (=) was implemented and simulated. In this method, statements execute sequentially and update immediately. The waveform shows how values change step-by-step based on execution order.

```
module tb;

reg a, b;
wire x, y;

blocking_example uut(a, b, x, y);

initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0, tb);

    a = 0; b = 0;
    #5 a = 1;
    #5 a = 0;
    #5 a = 1;
    #5 $finish;
end

endmodule// Code your testbench here
// or browse Examples
```

```
module blocking_example(
input a,
input b,
output reg x,
output reg y
);

always @(*)
begin
    x = a;   // Blocking assignment
    y = x;   // y ne updated x ni value turant male
end

endmodule// Code your design here
```



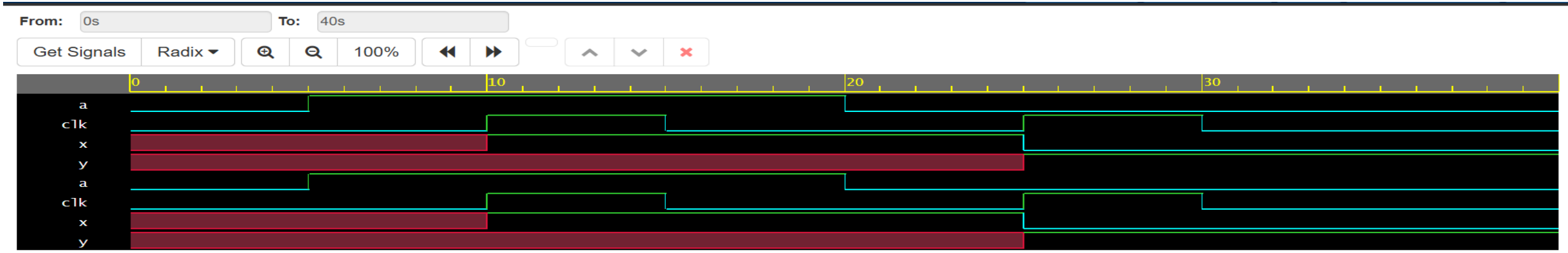
5. Non-Blocking Assignment

EXPLANATION:

Non-blocking assignment (<=) was implemented. In this case, all assignments update simultaneously at the clock edge. The simulation shows correct sequential behavior, making it suitable for flip-flop and sequential circuit design.

```
module tb;

reg clk, a;
wire x, y;

nonblocking_example uut(clk, a, x, y);

initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0, tb);

    clk = 0;
    a = 0;

    #5 a = 1;
    #5 clk = 1;   // posedge → x=1, y=0 (old x)
    #5 clk = 0;

    #5 a = 0;
    #5 clk = 1;   // posedge → x=0, y=1
    #5 clk = 0;

    #10 $finish;
end

endmodule// Code your testbench here
// or browse Examples
```

```
module nonblocking_example(
input clk,
input a,
output reg x,
output reg y
);

always @(posedge clk)
begin
    x <= a;   // Non-blocking assignment
    y <= x;   // y ne x ni old value male
end

endmodule// Code your design here
```



6.T flip-Flop (Asynchronous)

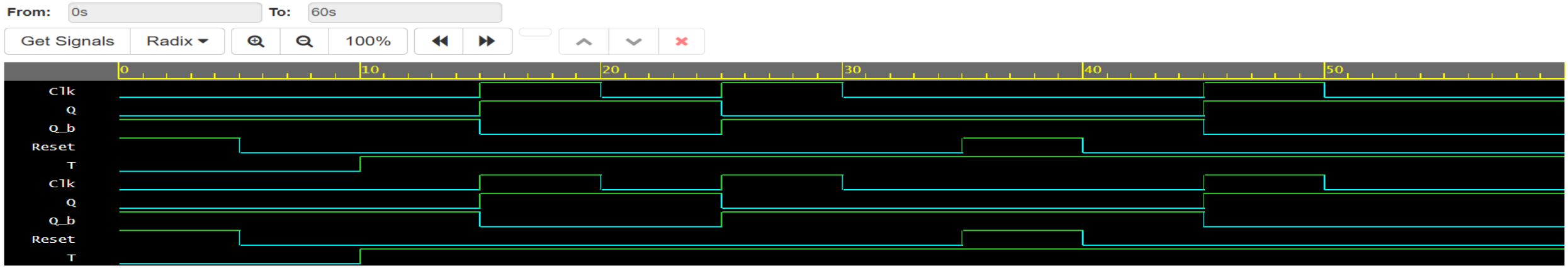
EXPLANATION:

An asynchronous T Flip-Flop with reset was designed. The output toggles when T = 1 and remains unchanged when T = 0. The waveform verifies proper toggle and reset operation.

```
module tb;

reg T, Clk, Reset;
wire Q, Q_b;

T_FF_async uut(T, Clk, Reset, Q, Q_b);

initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0, tb);

    Clk = 0;
    Reset = 1;
    T = 0;

    #5 Reset = 0;   // Release reset

    #5 T = 1;
    #5 Clk = 1;     // Toggle → Q = 1
    #5 Clk = 0;

    #5 Clk = 1;     // Toggle → Q = 0
    #5 Clk = 0;

    #5 Reset = 1;   // Asynchronous reset → Q = 0 immediately
    #5 Reset = 0;

    #5 Clk = 1;     // Toggle again
    #5 Clk = 0;

    #10 $finish;
```

Playground.

```
module T_FF_async(
input T,
input Clk,
input Reset,      // Asynchronous Reset
output reg Q,
output Q_b
);

assign Q_b = ~Q;

// Asynchronous Reset + Positive edge triggered
always @(posedge Clk or posedge Reset)
begin
    if (Reset)
        Q <= 0;
    else if (T)
        Q <= ~Q;   // Toggle
end

endmodule// Code your design here
```



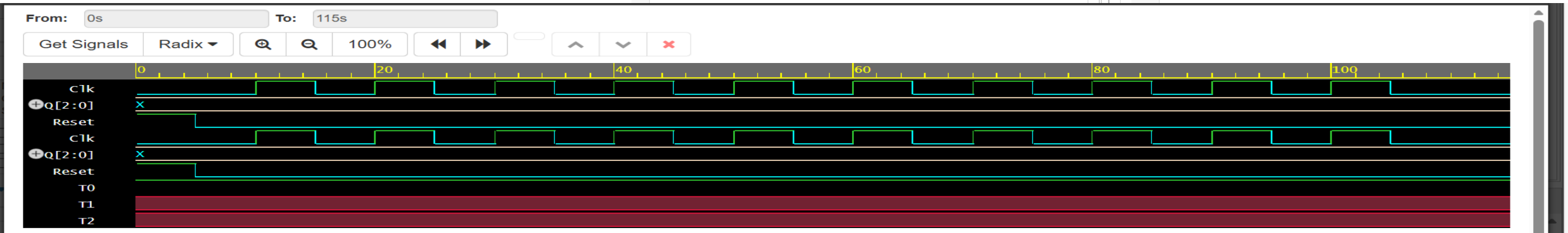
7.Synchronous Up Counter(T FF based)

EXPLANATION:

A synchronous up counter using T Flip-Flops was designed and simulated. All flip-flops operate using the same clock signal, and the counter increments sequentially. The waveform shows proper counting without ripple delay.

```
module tb;

reg Clk, Reset;
wire [2:0] Q;

sync_up_counter uut(Clk, Reset, Q);

initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0, tb);

    Clk = 0;
    Reset = 1;

    #5 Reset = 0;

    repeat (10) begin
        #5 Clk = 1;
        #5 Clk = 0;
    end

    #10 $finish;
end

endmodule// Code your testbench here
// or browse Examples
```

```
module sync_up_counter(
input Clk,
input Reset,
output reg [2:0] Q
);

// T inputs logic
wire T0, T1, T2;

assign T0 = 1'b1;
assign T1 = Q[0];
assign T2 = Q[0] & Q[1];

// Synchronous T Flip-Flop operation
always @(posedge Clk)
begin
    if (Reset)
        Q <= 3'b000;
    else begin
        if (T0) Q[0] <= ~Q[0];
        if (T1) Q[1] <= ~Q[1];
        if (T2) Q[2] <= ~Q[2];
    end
end

endmodule// Code your design here
```



Difference between Synchronous and Asynchronous

- In synchronous systems, all operations are controlled by a common clock signal, so all outputs change at the same time. This makes the circuit faster and more reliable. In asynchronous systems, there is no common clock for all stages, and each stage is triggered by the output of the previous stage. Because of this, the outputs change one after another, which may cause delay and make the system slower.