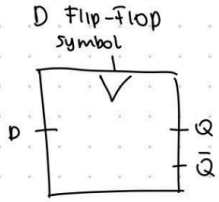
Ejercicio 1: Sequential, flip flops

1.

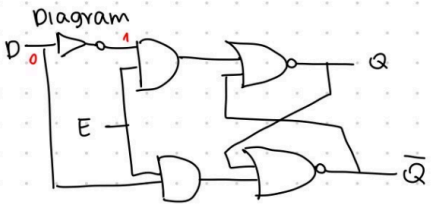
Laura Nagamine — S8: 13/05/25

Ejercicio 1 Sequential: Flip Flops

1. Definir un D Flip-Flop, tabla de verdad, diagrama y código verilog — 1 hora

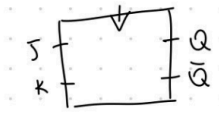


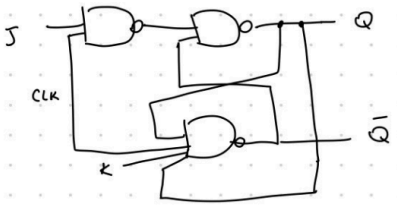
Input	Output
D	Q_{n+1}
0	0
1	1



Input	Output	
D	Q_n	Q_{n+1}
0	0	0
1	0	1
0	1	0
1	1	1

2. Definir un JK Flip Flop, tabla de verdad, diagrama y código verilog





Input		Output
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

Input		Output	
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	1	1
1	0	0	1
1	1	0	1
1	1	1	0

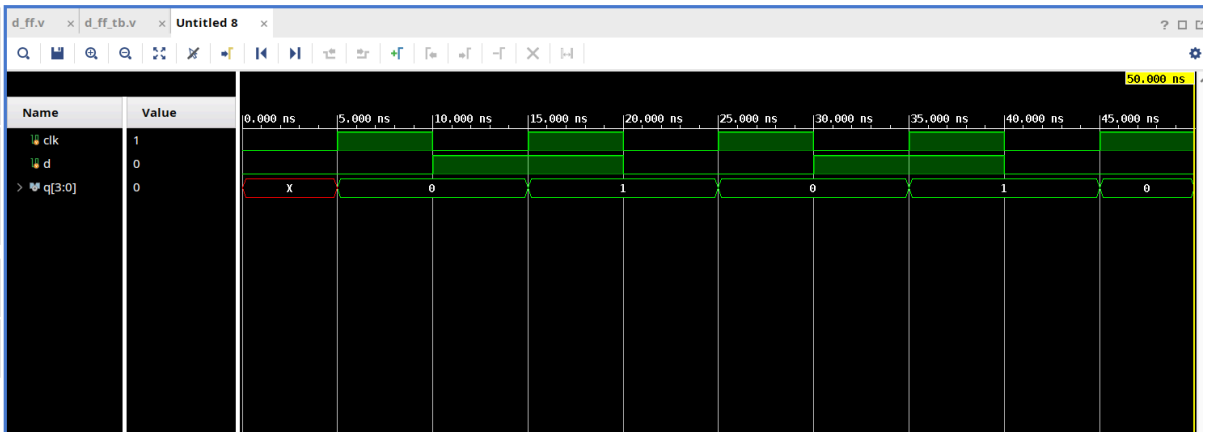
D flip flop
Design

module d_ff(input clk, input d, output reg [3:0] q);

```
    always @ (posedge clk)
        q <= d;

endmodule
```

testbench:

```
module d_ff_tb();
    reg clk;
    reg d;
    wire [3:0] q;

    always #5 clk = ~clk;

    d_ff d_ff_test(.clk(clk), .d(d), .q(q));

    initial begin
        clk = 0;
        d = 0;
        #10;

        d = 1;
        #10;

        d = 0;
        #10;

        d = 1;
        #10;

        d = 0;
        #10;

        $finish;

    end

    initial begin
        $display("Time(ns)  clk  d    q");
        $monitor("%0d        %b    %b    %b", $time, clk, d, q);

    end
```



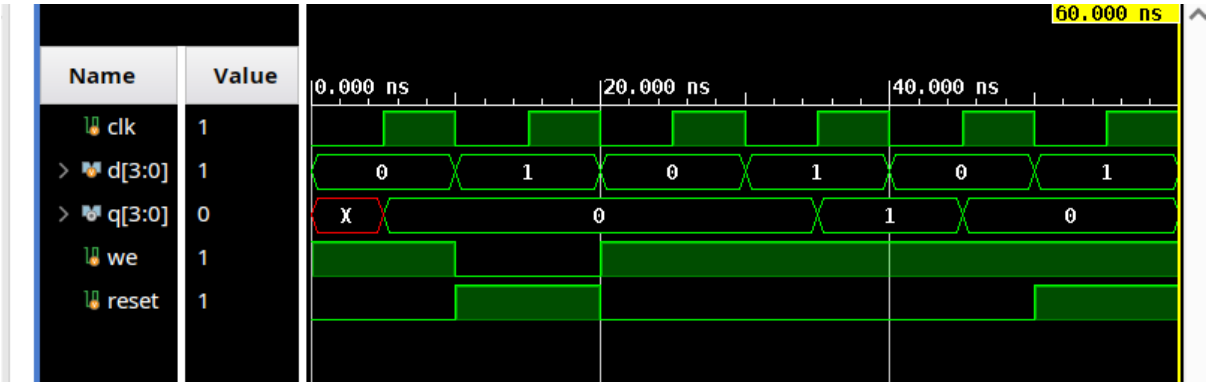
D flip flop con write enable y reset
Asíncrono:

```
// d flip flop con WE y RESET asíncrono
module d_ff_we(input clk, input [3:0] d, input we, input reset, output reg [3:0] q);
    // asynchronous reset
        always @(posedge clk or posedge reset)
        if (reset)
            q <= 0;
        else if (we)       // write habilita la escritura
             q <= d;

endmodule
```

testbench:

```
module dff_we_tb();

    reg clk;
    reg d;
    wire [3:0] q;
    reg we;
    reg reset;

    always #5 clk = ~clk;

    d_ff_we d_ff_test(.clk(clk), .d(d), .q(q), .we(we), .reset(reset));

    initial begin
        clk = 0;
        d = 0;
        we = 1;
        reset = 0;
        #10;
        we  =0;
        d = 1;
```

```
        reset = 1;
        #10;

        reset =0;
        we=1;
        d = 0;
        #10;

        d = 1;
        #10;

        d = 0;
        #10;

        reset = 1;
        #10

        d = 1;
        $finish;

    end

    initial begin
        $display("Time(ns)  clk  d    q");
        $monitor("%0d       %b    %b    %b", $time, clk, d, q);

    end

endmodule
```



Síncrono:

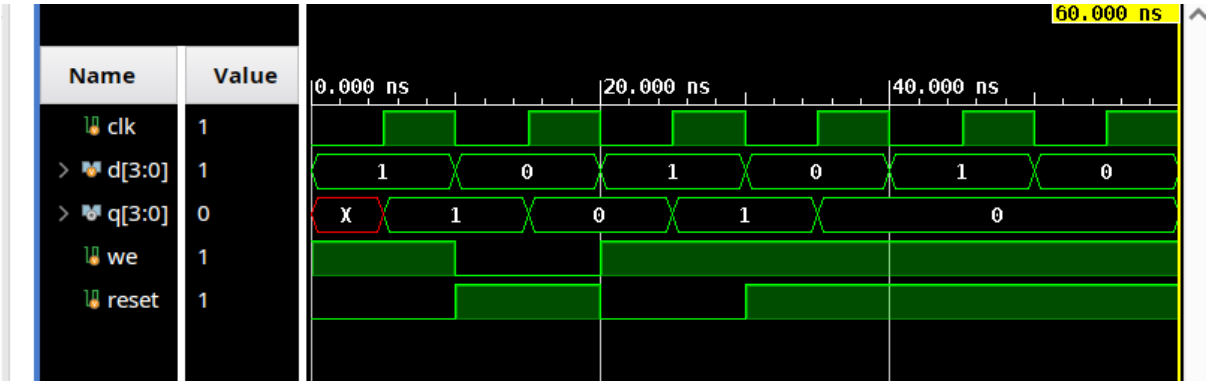
```
// d flip flop con WE y RESET síncrono
module d_ff_we_s(input clk, input [3:0] d, input we, input reset, output reg [3:0] q);
    // asynchronous reset
    always @(posedge clk)
        if (reset)
```

```
        q <= 0;
    else if (we)       // write habilita la escritura
        q <= d;

endmodule
```

testbench:

```
module dff_we_tb();

    reg clk;
    reg [3:0] d;
    wire [3:0] q;
    reg we;
    reg reset;

    always #5 clk = ~clk;

    d_ff_we_s d_ff_test(.clk(clk), .d(d), .q(q), .we(we), .reset(reset));

    initial begin
        clk = 0;
        d = 1;
        we = 1;
        reset = 0;
        #10;
        we  =0;
        d = 0;
        reset = 1;
        #10;

        reset =0;
        we=1;
        d = 1;
        #10;

        d = 0;
        reset = 1;
        #10;

        d = 1;
        #10;

        reset = 1;
        d=0;
        #10;

        d = 1;
        $finish;
```

```
    end

    initial begin
        $display("Time(ns)  clk  d    q");
        $monitor("%0d        %b    %b    %b", $time, clk, d, q);

    end

endmodule
```



JK Flip flop (con reset y we)

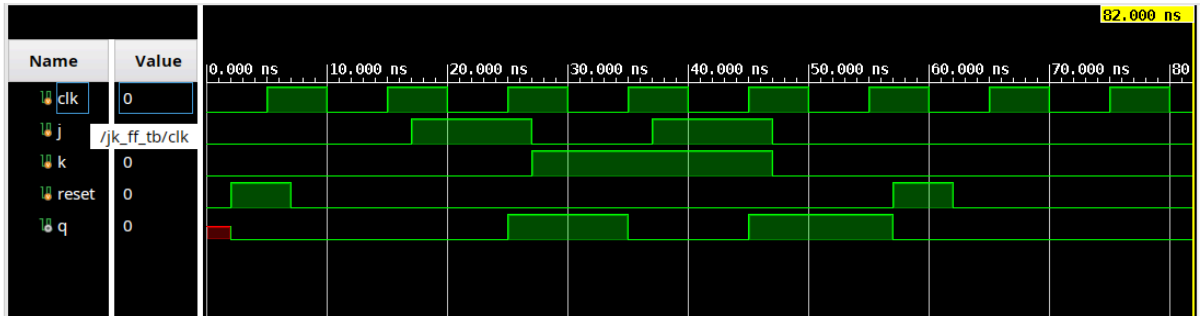
// síncrono

```
module jk_ff (
    input clk,
    input reset,
    input we,
    input j,
    input k,
    output reg q
);

    always @(posedge clk or posedge reset) begin
        if (reset)
            q <= 0;  // Reset the output to 0
        else if (we)
            if (j == 0 && k == 0)
                q <= q;     // No change
            else if (j == 0 && k == 1)
                q <= 0;     // Reset
            else if (j == 1 && k == 0)
                q <= 1;     // Set
            else if (j == 1 && k == 1)
                q <= ~q;    // Toggle
        end
```

```
endmodule
```

testbench:

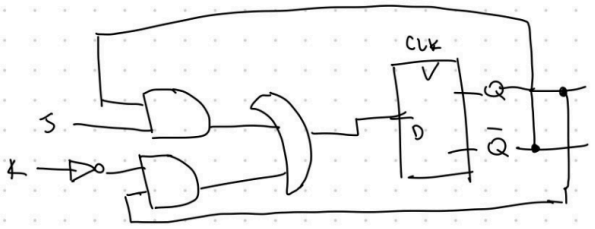
```
module jk_ff_tb;

    reg clk, j, k, reset, we;
    wire q;

    jk_ff jk (
        .clk(clk),
        .j(j),
        .k(k),
        .reset(reset),
        .we(we),
        .q(q)
    );

    always #5 clk = ~clk;

    initial begin
        clk = 0; j = 0; k = 0; reset = 0; we = 1;

        #2  reset = 1;
        #5  reset = 0;

        #10 j = 1; k = 0;

        // Reset (J=0, K=1) con we=0 => no debe cambiar q
        #10 j = 0; k = 1; we = 0;

        #10 we = 1; j = 1; k = 1;

        #10 j = 0; k = 0;


        #10 reset = 1;
        #5  reset = 0;

        #20 $finish;
    end
```


82.000 ns
Name
Value
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
70.000 ns
clk
0
j
/jk_ff_tb/clk
k
0
reset
0
q
0

3. Implementar un JK Flip-Flop usando un D Flip Flop. Diagrama y código de Verilog

Tabla de Verdad:

D	J	k	Q_n	Q_{n+1}
0	0	0	0	0
1	0	0	1	1
0	0	1	0	0
0	0	1	1	0
1	1	0	1	1
1	1	0	0	1
1	1	1	0	1
0	1	1	1	0



J	k	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q_n}$

D	Q_n	Q_{n+1}
0	0	0
1	0	1
0	1	0
1	1	1

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$D = J \cdot \bar{Q_n} + \bar{K} Q_n$$

4. Implementar un D Flip-Flop utilizando un JK Flip Flop. Diagrama y código en verilog

J	k	D	Q_n	Q_{n+1}
0	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	0	1	1	1
1	1	1	0	1
0	1	0	1	0
0	1	0	0	0
0	0	1	1	1

→

J	k	D	Q_n	Q_{n+1}
0	X	0	0	0
X	1	0	1	0
1	X	1	0	1
X	0	1	1	1

J:

Q_n \ D	0	1
0	0	1
1	X	X

$J = D$

K:

Q_n \ D	0	1
0	X	X
1	1	0

$K = \bar{D}$

JK flip flop usando un D Flip flop

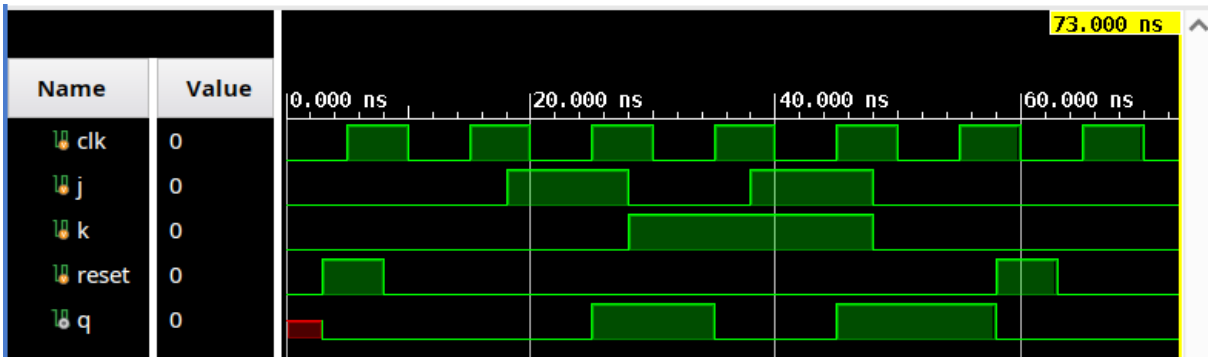
odule jk_using_d (

```
    input clk,
    input j,
    input k,
    input reset,
    output reg q
);

    wire d;

    assign d = (j & ~q) | (~k & q);  // D = J·~Q + ~K·Q

    always @(posedge clk or posedge reset) begin
        if (reset)
            q <= 0;
        else
            q <= d;
    end

endmodule
```

Testbench

```
`timescale 1ns / 1ps

module jk_using_d_tb;

    reg clk, j, k, reset;
    wire q;

    jk_using_d jk (.clk(clk), .j(j), .k(k), .reset(reset), .q(q));

    always #5 clk = ~clk;

    initial begin
        clk = 0; j = 0; k = 0; reset = 0;

        #3 reset = 1;
        #5 reset = 0;

        #10 j = 1; k = 0;  // Set
        #10 j = 0; k = 1;  // Reset
        #10 j = 1; k = 1;  // Toggle
        #10 j = 0; k = 0;  // Hold

        #10 reset = 1;
        #5  reset = 0;

        #10 $finish;
    end
```

endmodule



D flip flop usando un JK flip flop

```
module d_using_jk (
    input clk,
    input d,
    input reset,
    output reg q
);

    wire j, k;

    assign j = d;
    assign k = ~d;

    always @(posedge clk or posedge reset) begin
        if (reset)
            q <= 0;
        else begin
            if (j == 0 && k == 0)
                q <= q;     // No change
            else if (j == 0 && k == 1)
                q <= 0;     // Reset
            else if (j == 1 && k == 0)
                q <= 1;     // Set
            else if (j == 1 && k == 1)
                q <= ~q;    // Toggle
        end
    end

endmodule
```

testbench:

```
module d_using_jk_tb;

    reg clk, d, reset;
    wire q;

    d_using_jk d_jktest(.clk(clk), .d(d), .reset(reset), .q(q));

    always #5 clk = ~clk;

    initial begin
        clk = 0; d = 0; reset = 0;

        #3 reset = 1;
        #5 reset = 0;

        #10 d = 1;
        #10 d = 0;
        #10 d = 1;
        #10 d = 1;
        #10 d = 0;

        #10 reset = 1;
        #5  reset = 0;

        #10 $finish;
    end


endmodule
```