Nombre: Brayan José Castillo Alvarado
Carné: 19700
Sección: 10

Laboratorio 09

Ejercicio 01

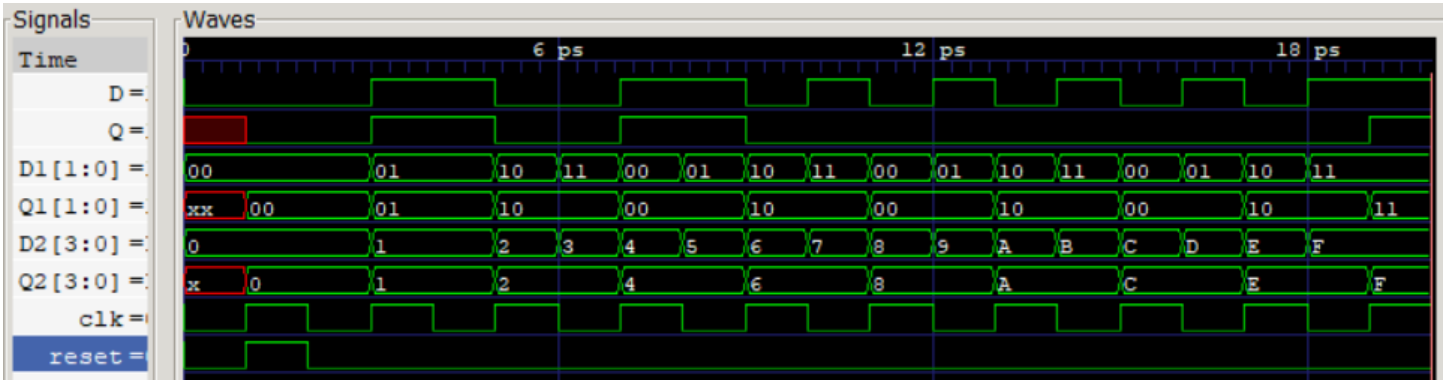
El primer modulo es un Flip Flip Tipo D de 1 bit, con las entradas clock, reset, enabled (E), y d, con la unica salida q. En el siguiente modulo para crear un Flip Flop Tipo D de 2 bits se llama al modulo del Flip Flop de 1 bit dos veces utilizando los bits significativos en cada uno respectivamente. En el caso del Flip Flop de 4 bits se realiza el mismo procedimiento en este caso llamando 4 veces al modulo de 1 bit. En el testbench se probaron diferentes combinaciones para probar los Flip Flops.

```
// Ejercicio 1 Flip Flop tipo D de 1 bit
module FFD1(input clk , input reset, E, input d,  output reg q);

  always@(posedge clk, posedge reset) begin
    if(E == 1) begin
    q <= d;
    end
    if(reset) begin
    q <= 0;
    end
  end
endmodule

// Ejercicio 1 Flip Flop tipo D de 2 bits
module FFD2(input clk , input reset, E, input [1:0]d, output wire [1:0]q);

  FFD1 M1(clk, reset, E, d[0], q[0]);
  FFD1 M2(clk, reset, E, d[1], q[1]);
endmodule

// Ejercicio 1 Flip Flop tipo D de 4 bits
module FFD4(input clk , input reset, E, input [3:0]d, output wire [3:0]q);

  FFD1 G1(clk, reset, E, d[0], q[0]);
  FFD1 G2(clk, reset, E, d[1], q[1]);
  FFD1 G3(clk, reset, E, d[2], q[2]);
  FFD1 G4(clk, reset, E, d[3], q[3]);
endmodule
```

```
module testbench();
  reg clk, reset;
  reg D, E;
  reg [1:0]D1;
  reg [3:0]D2;
  wire Q;
  wire [1:0]Q1;
  wire [3:0]Q2;

  FFD1 P1(clk, reset, E, D, Q);
  FFD2 P2(clk, reset, E, D1, Q1);
  FFD4 P3(clk, reset, E, D2, Q2);

  initial begin
  clk = 0; reset = 0; D=0; D1=2'b00; D2=4'b0000; E=0;
  #1 reset = 1;
  #1 reset = 0;
  #1 D=1; D1=2'b01; D2=4'b0001; E=1;
  #2 D=0; D1=2'b10; D2=4'b0010;
  #1 D=0; D1=2'b11; D2=4'b0011;
  #1 D=1; D1=2'b00; D2=4'b0100;
  #1 D=1; D1=2'b01; D2=4'b0101;
  #1 D=0; D1=2'b10; D2=4'b0110;
  #1 D=1; D1=2'b11; D2=4'b0111;
  #1 D=0; D1=2'b00; D2=4'b1000;
  #1 D=1; D1=2'b01; D2=4'b1001;
  #1 D=0; D1=2'b10; D2=4'b1010;
  #1 D=1; D1=2'b11; D2=4'b1011;
  #1 D=0; D1=2'b00; D2=4'b1100;
  #1 D=1; D1=2'b01; D2=4'b1101;
  #1 D=0; D1=2'b10; D2=4'b1110;
  #1 D=1; D1=2'b11; D2=4'b1111;
  end
  always
    #1 clk = ~clk;
  initial
    #20 $finish;
  initial begin
    $dumpfile("EJ1_tb.vcd");
    $dumpvars(0, testbench);
  end
endmodule
```

Diagrama de Timing

En el diagrama podemos observar que en cada flanco de reloj los valores Q copian los valores de D tanto en los Flip Flops de 1, 2 y 4 bits.



Nombre: Brayan José Castillo Alvarado
Carné: 19700
Sección: 10

Ejercicio 02

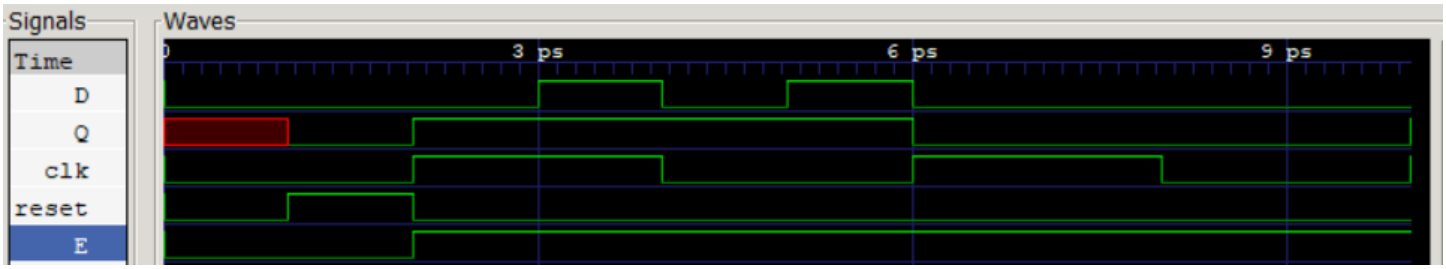
Usando el módulo de un Flip Flop tipo D de 1 bit creamos el módulo de un Flip Flop tipo T donde crearemos un wire para negar la salida Q, ya que en cada flanco de reloj buscamos obtener el negado de la entrada D.

```
// Ejercicio 2 Flip Flop tipo D de 1 bit

module FFD1(input clk , input reset, E, input d,  output reg q);

  always@(posedge clk, posedge reset) begin
    if(E == 1) begin
    q <= d;
    end
    if(reset) begin
    q <= 0;
    end
  end
endmodule

// Ejercicio 2 Flip Flop tipo T

module FFT(input clk , input reset, E, input d1, output wire q1);
  wire dn = ~d1;
  FFD1 M1(clk, reset, E, dn, q1);
endmodule
```

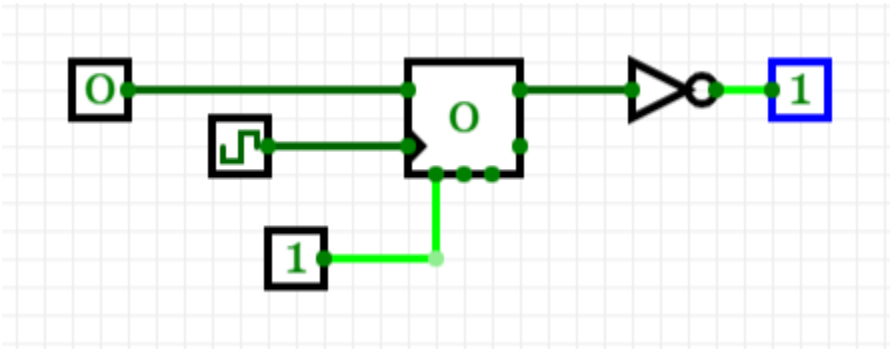
```
module testbench();
  reg clk, reset;
  reg D, E;
  wire Q;

  FFT P1(clk, reset, E, D, Q);

  initial begin
  clk = 0; reset = 0; D=0; E=0;
  #1 reset = 1;
  #1 reset = 0; E=1;
  #1 D=1;
  #1 D=0;
  #1 D=1;
  #1 D=0;

  end
  always
    #2 clk = ~clk;
  initial
    #10 $finish;
  initial begin
    $dumpfile("EJ2_tb.vcd");
    $dumpvars(0, testbench);
  end

endmodule
```

Diagrama de Timing

En el diagrama de timing podemos observar que en cada flanco de reloj la salida Q será el negado de la entrada D siempre y cuando el Enabled este activado.



Circuitverse



Nombre: Brayan José Castillo Alvarado
Carné: 19700
Sección: 10

Ejercicio 03

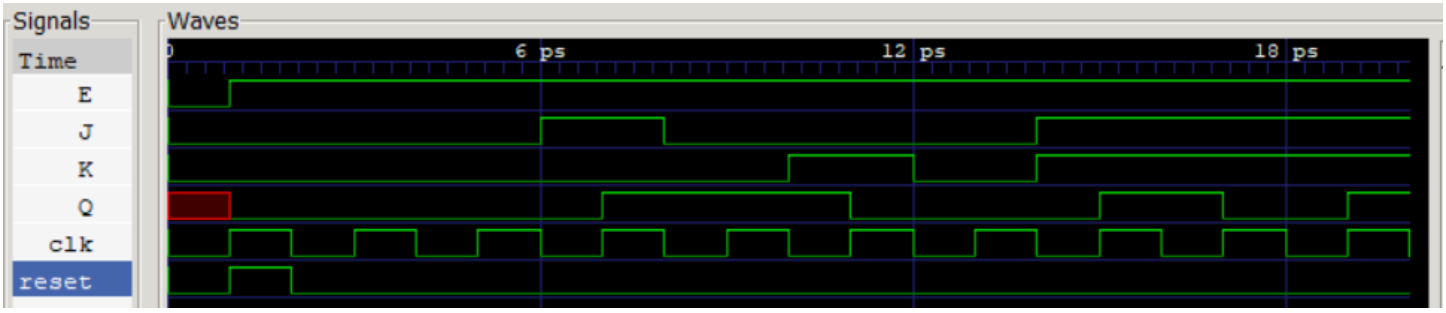
Para crear el Flip Flop tipo JK se utilizará un módulo de Flip Flop tipo D de 1 bit, junto a las entradas J y K que controlaran la salida Q, siempre teniendo en cuenta el clock y el reset. Además de esto se definieron unos cables para conectar de manera estructural la nube combinacional, que proporcionará la salida que ira al modulo del Flip Flop tipo D, la cual proveerá la salida Q.

```
// Ejercicio 3 Flip Flop tipo D de 1 bit

module FFD1(input clk , input reset, E, input d,  output reg q);

  always@(posedge clk, posedge reset) begin
    if(E == 1) begin
    q <= d;
    end
    if(reset) begin
    q <= 0;
    end
  end
endmodule

// Ejercicio 3 Flip Flop JK de 1 bit

module FFJK(input clk , input reset, E, J, K, output Q);
  wire nK, nQ, w1, w2, D;
  not(nK, K);
  not(nQ, Q);
  and(w1, nK, Q);
  and(w2, nQ, J);
  or (D, w1, w2);

  FFD1 M1(clk, reset, E, D, Q);
endmodule
```

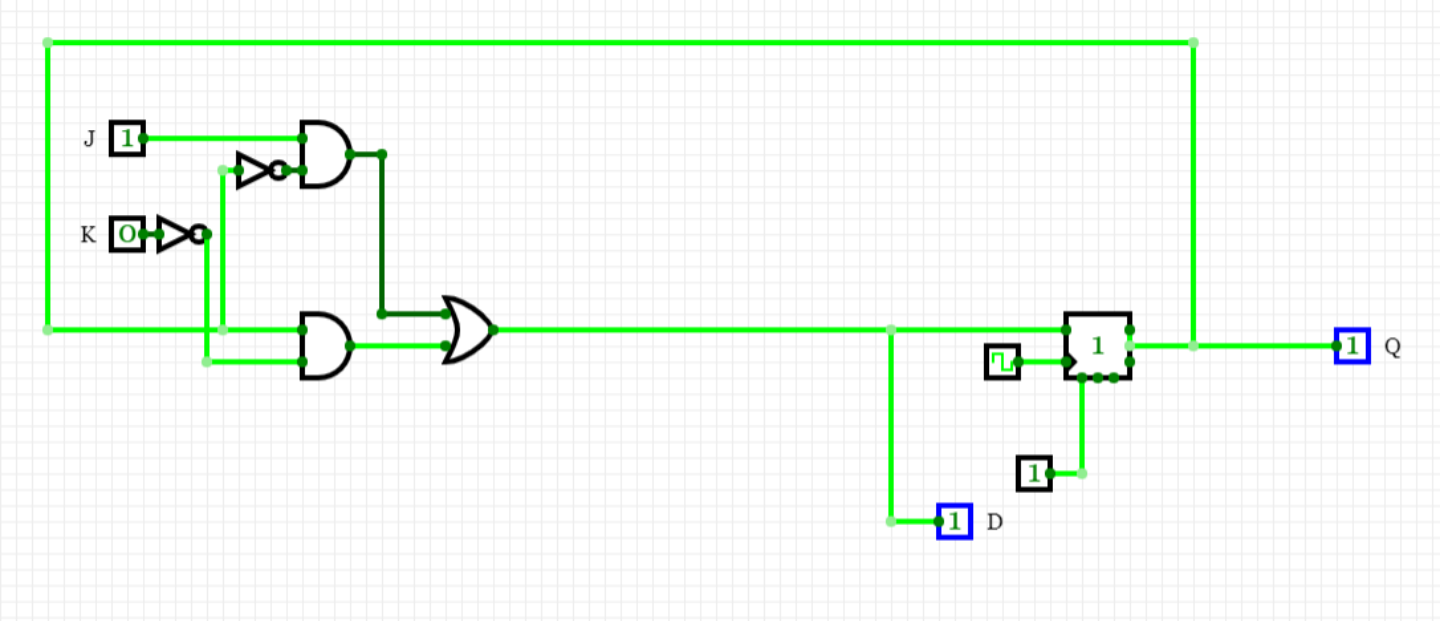
```
module testbench();
  reg clk, reset;
  reg J, K, E;
  wire Q;

  FFJK P1(clk, reset, E, J, K, Q);

  initial begin
  clk = 0; reset = 0; J = 0; K = 0; E = 0;
  #1 reset = 1; E=1;
  #1 reset = 0;
  #2 J=0; K=0;
  #2 J=1; K=0;
  #2 J=0; K=0;
  #2 J=0; K=1;
  #2 J=0; K=0;
  #2 J=1; K=1;

  end
  always
    #1 clk = ~clk;
  initial
    #20 $finish;
  initial begin
    $dumpfile("EJ03_tb.vcd");
    $dumpvars(0, testbench);
  end

endmodule
```

Diagrama de Timing

En el diagrama podemos observar las diferentes combinaciones del Flip Flop, ya que cuando J y K están en 0 la salida Q permanece en estado de memoria, en cambio sí J está en 1 y K en 0 la salida Q será 1, en cambio si la salida J es 0 y la salida K es 1 la salida Q será 0, y en el último caso donde ambos estén encendidos la salida será Q negado.



Nombre: Brayan José Castillo Alvarado
Carné: 19700
Sección: 10
Circuitverse



Nombre: Brayan José Castillo Alvarado
Carné: 19700
Sección: 10

Ejercicio 04

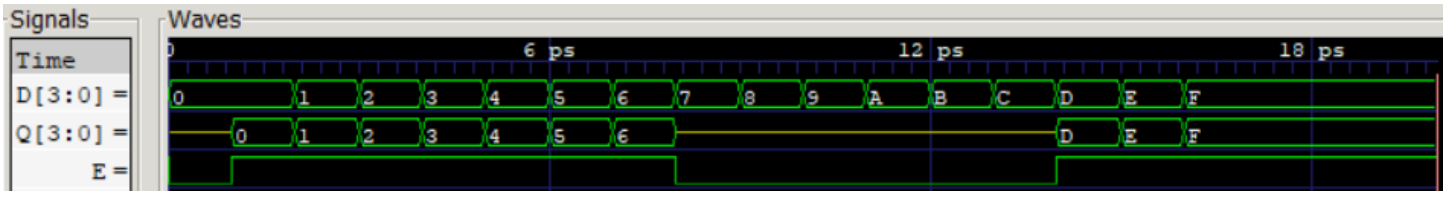
Para crear el Buffer Tri- estado de 4 bits únicamente utilizamos un ternario para definir que cuando Enabled sea 1 las entradas D serán directamente las salidas Q, y en el caso Enabled sea 0 las salidas estarán indeterminadas con alta impedancia.

```
// Ejercicio 4 buffer Tri-estado de 4 bits

module BTE(input E, input [3:0]D,  output [3:0]Q);

    assign Q = E==1?D:4'bzzzz;

endmodule
```

```
module testbench();
  reg E;
  reg [3:0]D;
  wire[3:0]Q;

  BTE P1(E, D, Q);

  initial begin
  E = 0; D=0;
  #1 E = 1;
  #1 D = 4'b0001;
  #1 D = 4'b0010;
  #1 D = 4'b0011;
  #1 D = 4'b0100;
  #1 D = 4'b0101;
  #1 D = 4'b0110;
  #1 D = 4'b0111;E = 0;
  #1 D = 4'b1000;
  #1 D = 4'b1001;
  #1 D = 4'b1010;
  #1 D = 4'b1011;
  #1 D = 4'b1100;
  #1 D = 4'b1101;E = 1;
  #1 D = 4'b1110;
  #1 D = 4'b1111;
  end

  initial
    #20 $finish;
  initial begin
    $dumpfile("EJ4_tb.vcd");
    $dumpvars(0, testbench);
  end

endmodule
```

Diagrama de Timing

En el diagrama podemos observar que cuando Enabled es 1 las salidas son iguales a las entradas, mientras que cuando Enabled es 0 las salidas están en alta impedancia



Nombre: Brayan José Castillo Alvarado
Carné: 19700
Sección: 10

Ejercicio 05

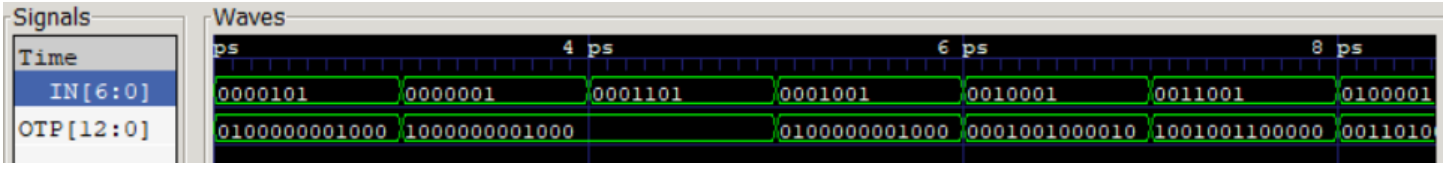
Para crear la tabla del ROM se utilizó la función casez para definir en base a las entradas una salida específica, cabe aclarar que en los casos que se repiten se utilizaron don't cares que están definidos por el signo de interrogación. En el testbench se realizaron 26 diferentes combinaciones para el diagrama de timing.

```
module ROM(input wire [6:0]IN, output logic [12:0]OTP);

reg[12:0] DATA;

  always @(*) begin
    casez (IN)
      7'b??????0: DATA = 13'b1000000001000;
      7'b00001?1: DATA = 13'b0100000001000;
      7'b00000?1: DATA = 13'b1000000001000;
      7'b00011?1: DATA = 13'b1000000001000;
      7'b00010?1: DATA = 13'b0100000001000;
      7'b0010??1: DATA = 13'b0001001000010;
      7'b0011??1: DATA = 13'b1001001100000;
      7'b0100??1: DATA = 13'b0011010000010;
      7'b0101??1: DATA = 13'b0011010000100;
      7'b0110??1: DATA = 13'b1011010100000;
      7'b0111??1: DATA = 13'b1000000111000;
      7'b1000?11: DATA = 13'b0100000001000;
      7'b1000?01: DATA = 13'b1000000001000;
      7'b1001?11: DATA = 13'b1000000001000;
      7'b1001?01: DATA = 13'b0100000001000;
      7'b1010??1: DATA = 13'b0011011000010;
      7'b1011??1: DATA = 13'b1011011100000;
      7'b1100??1: DATA = 13'b0100000001000;
      7'b1101??1: DATA = 13'b0000000001001;
      7'b1110??1: DATA = 13'b0011100000010;
      7'b1111??1: DATA = 13'b1011100100000;
      default: DATA = 13'bxxxxxxxxxxxxx;
    endcase
    assign OTP = DATA;
end
endmodule
```

```
module testbench();
  reg [6:0]IN;
  wire[12:0]OTP;

  ROM P1(IN, OTP);

  initial begin
  IN = 0;
  #1 IN = 7'b0000000;
  #1 IN = 7'b0000101;
  #1 IN = 7'b0000001;
  #1 IN = 7'b0001101;
  #1 IN = 7'b0001001;
  #1 IN = 7'b0010001;
  #1 IN = 7'b0011001;
  #1 IN = 7'b0100001;
  #1 IN = 7'b0101001;
  #1 IN = 7'b0110001;
  #1 IN = 7'b0111001;
  #1 IN = 7'b1000011;
  #1 IN = 7'b1000001;
  #1 IN = 7'b1001011;
  #1 IN = 7'b1001001;
  #1 IN = 7'b1010001;
  #1 IN = 7'b1011001;
  #1 IN = 7'b1100001;
  #1 IN = 7'b1101001;
  #1 IN = 7'b1110001;
  #1 IN = 7'b1111001;
  #1 IN = 7'b0000111;
  #1 IN = 7'b0000011;
  #1 IN = 7'b0001111;
  #1 IN = 7'b0001011;
  #1 IN = 7'b0010111;
  end

  initial
    #30 $finish;
  initial begin
    $dumpfile("EJ5_tb.vcd");
    $dumpvars(0, testbench);
  end
endmodule
```

Diagrama de Timing

En el diagrama podemos observar que dependiendo de la entrada se devolverá una salida de las definidas en nuestro casez. La primera imagen muestra las combinaciones decimales. Sin embargo, para abarcar todas las combinaciones la segunda y tercera imagen están en formato hexadecimal.



Nombre: Brayan José Castillo Alvarado
Carné: 19700
Sección: 10