Ejercicio 1:

JKFF

Behavioral

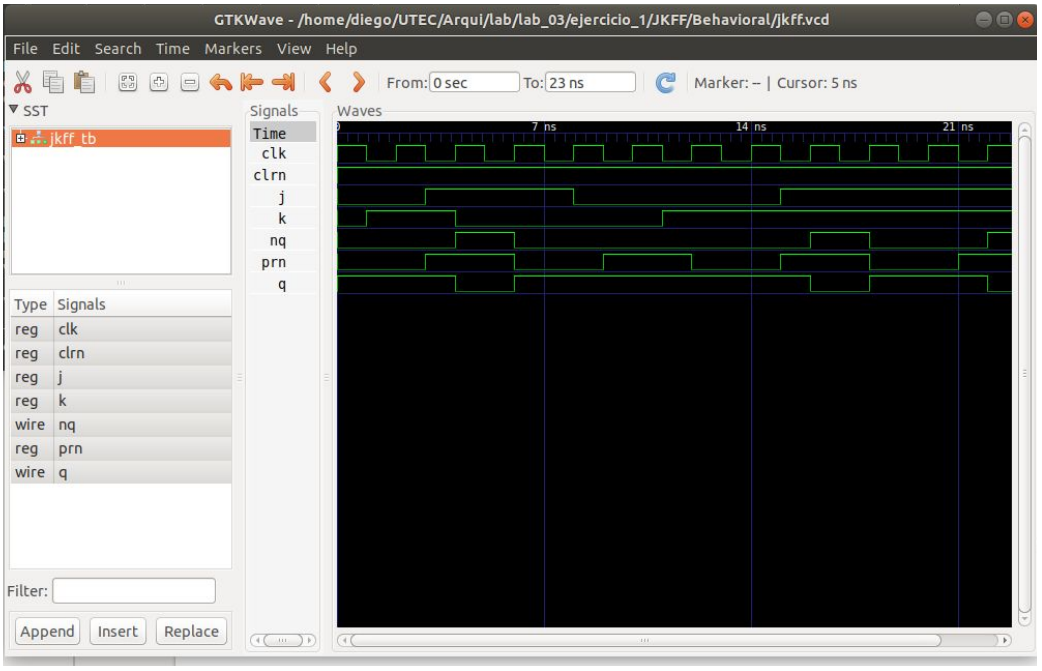
```
module jkff(prn, j, k, clk, clrn, q, nq);

        input   prn, j, k, clk, clrn;
        output  q, nq;
        wire    w1, w2, w3, w4, w5, w6;

        //Master JK Latch

        assign w1 = ~(nq & j & ~clk & clrn);
        assign w2 = ~(q & k & ~clk & prn);

        assign w3 = ~(w1 & prn & w4);
        assign w4 = ~(w2 & clrn & w3);

        //Slave D Latch

        assign w5 = ~(w3 & clrn & clk);
        assign w6 = ~(w4 & clk & prn);

        //Output

        assign q = ~(w5 & prn & nq);
        assign nq = ~(w6 & clrn & q);

endmodule
~
```

Structural

```
module jkff(prn, j, k, clk, clrn, q, nq);

        input   prn, j, k, clk, clrn;
        output  q, nq;
        wire    w1, w2, w3, w4, w5, w6;

        //Master JK Latch

        or a1(w1, ~nq, ~j);
        or a2(w2, clk, ~clrn);
        or a3(w3, w1, w2);
```

```
    //w3

    or a4(w4, ~q, ~k);
    or a5(w5, clk, ~prn);
    or a6(w6, w4, w5);

    //w6

    or a7(w7, ~w3, ~prn);
    or a8(w8, w7, ~w10);

    //w3->w8

    or a9(w9, ~clrn, ~w6);
    or a10(w10, w9, ~w8);

    //w6->w10


    //Slave D Latch

    or a11(w11, ~w8, ~clrn);
    or a12(w12, w11, ~clk);


    //w8->w12

    or a13(w13, ~w10, ~prn);
    or a14(w14, w13, ~clk);

    //w10->w14

    //Output

    or a15(w15, ~w12, ~prn);
    or a16(q, w15, ~nq);

    or a17(w16, ~w14, ~clrn);
    or a18(nq, w16, ~q);


ndmodule
```

GTKWAVE



TFF

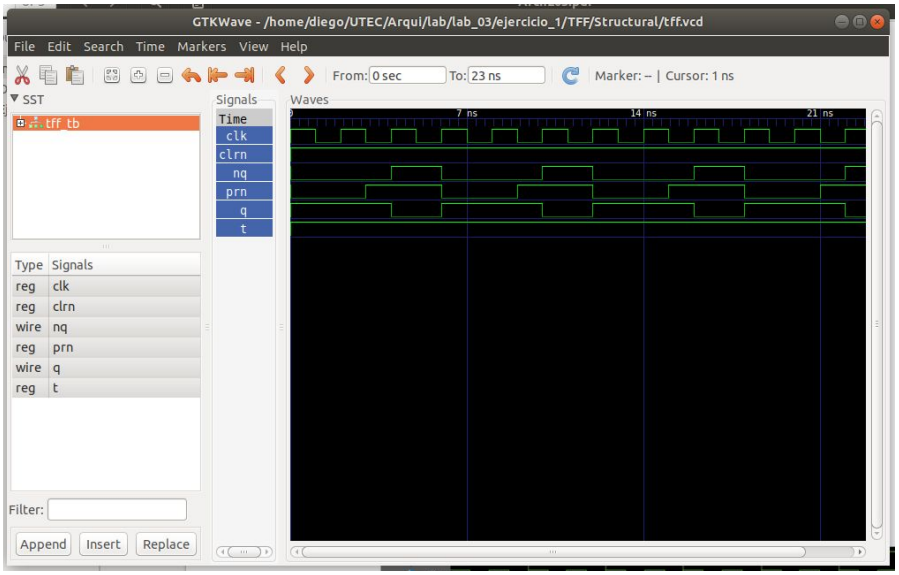
Behavioral

```
module tff(prn, t, clk, clrn, q, nq);

        input prn, t, clk, clrn;
        output q ,nq;
        wire w1, w2, w3, w4, w5, w6;

        //Tlatch master

        assign w1 = ~(t & ~clk & nq & clrn);
        assign w2 = ~(t & prn & ~clk & q);

        assign w3 = ~(w1 & prn & w4);
        assign w4 = ~(w3 & w2 & clrn);

        //Dlatch slave

        assign w5 = ~(w3 & clrn & clk);
        assign w6 = ~(w4 & prn & clk);

        //output
        assign q = ~(w5 & prn & nq);
        assign nq = ~(w6 & clrn & q);

endmodule
~
```

Structural

```
module tff(prn, t, clk, clrn, q, nq);

        input   prn, t, clk, clrn;
        output  q, nq;
        wire    w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14, w15, w16;

        //Master JK Latch

        or a1(w1, ~nq, ~t);
        or a2(w2, clk, ~clrn);
        or a3(w3, w1, w2);

        //w3

        or a4(w4, ~q, ~t);
        or a5(w5, clk, ~prn);
        or a6(w6, w4, w5);

        //w6

        or a7(w7, ~w3, ~prn);
        or a8(w8, w7, ~w10);

        //w3->w8

        or a9(w9, ~clrn, ~w6);
        or a10(w10, w9, ~w8);

        //w6->w10


        //Slave D Latch

        or a11(w11, ~w8, ~clrn);
        or a12(w12, w11, ~clk);


        //w8->w12

        or a13(w13, ~w10, ~prn);
        or a14(w14, w13, ~clk);

        //w10->w14

        //Output
tff.v
```

GTKWAVE



Ejercicio 2:

Mux 2:1

```
module mux2_1(a, b ,s , out);
        input a, b, s;
        output out;

        assign out = s? b:a;

endmodule
```

D flip-flop

```
module dff2(d, clk, clrn, q);

        input d, clk, clrn;
        output q;
        wire r, s, nq, qw, r2, s2, nq2, qu;


        // master dlatch
        assign r = (~d & ~clk);
        assign s = d & ~clk;

        assign qw = ~(r | nq);
        assign nq = ~(s | qw);

        // slave dlatch
        assign r2 = (~qw & clk);
        assign s2 = qw & clk;

        assign qu = ~(r2 | nq2);
        assign nq2 = ~(s2 | qu);

        assign q = qu & ~clrn;

        endmodule
```

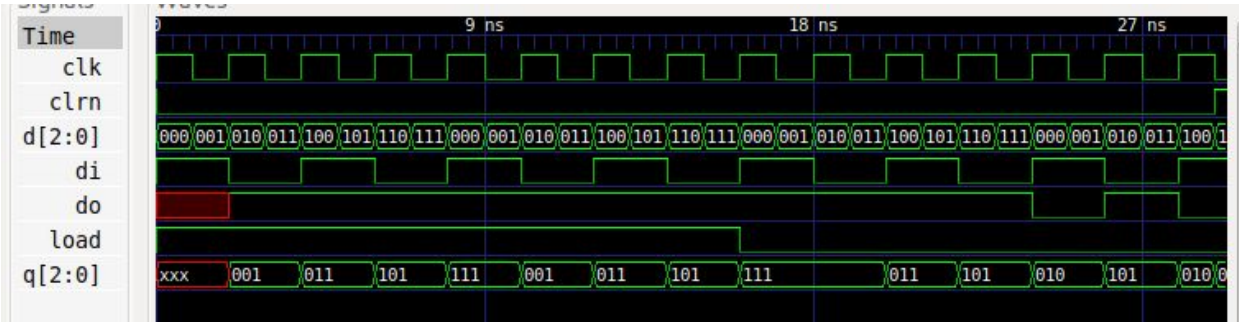
Termite

Mr. Schematic misterioso

Firefox Web Browser

```
load, clk, clrn, q, do);

    input [2:0] d;
    input di, load, clk, clrn;
    output [2:0] q;
    output do;
    wire w1, w2, w3, w4;

    mux2_1 m1(di, d[2], load, w1);
    dff2 dff1(w1, clk, clrn, w2);
    assign q[2] = w2;

    mux2_1 m2(w2, d[1], load, w3);
    dff2 dff2(w3, clk, clrn, w4);
    assign q[1] = w4;

    mux2_1 m3(w4, d[0], load, w5);
    dff2 dff3(w5, clk, clrn, do);
    assign q[0] = do;

endmodule
```

GTKWAVE



Explicación:

Por ser un montón de d flip-flop que comparten un clk es catalogado como un registro, ahora solo queda saber qué hace exactamente, cuando el load es 0 el serial input es que sale en todos los outputs, ya sean parallels y serial, si el load es 1 todos los outputs retornan los valores de los parallel inputs.

Ejercicio 3: