

# PRACTICA 1

Edgar A. Ramos Mesas 2013090243  
2CV10 DISEÑO DE SISTEMAS DIGITALES

## CODIGO IMPLEMENTADO EN LA PRACTICA

```
library ieee;
use ieee.std_logic_1164.all;

entity ff is port(
    S, R, J, K, CLK, D, T: in std_logic;
    selector: in std_logic_vector(0 to 2);
    display: out std_logic_vector(5 downto 0)
);
attribute pin_numbers of ff:entity is
"CLR:10";
end ff;

architecture aff of ff is
    SIGNAL qsr, qjk, qd, qt, mux_out: std_logic;
begin
    qsr <= ((not R) and qsr) or S;
    qjk <= ((not K) and qjk) or (J and (not qjk));
    qd <= D;
    qt <= ((not T) and qt) or (T and (not qt));
    process(CLK, selector(0))
    begin
        if selector(0) = '1' then
            selector <= "100";
            qjk <= '0';
            qsr <= '0';
            qd <= '0';
            qt <= '0';
        elsif CLK'event and CLK='1' then
            case(selector)is
                when "000" => mux_out <= qjk;
                when "001" => mux_out <= qt;
                when "010" => mux_out <= qd;
                when "011" => mux_out <= qsr;
                when others => mux_out <= '0';
            end case;

            if mux_out = '1' then
                display <= "011000";
            else
                display <= "111111";
            end if;
        end if;
    end process;
end aff;
```

```

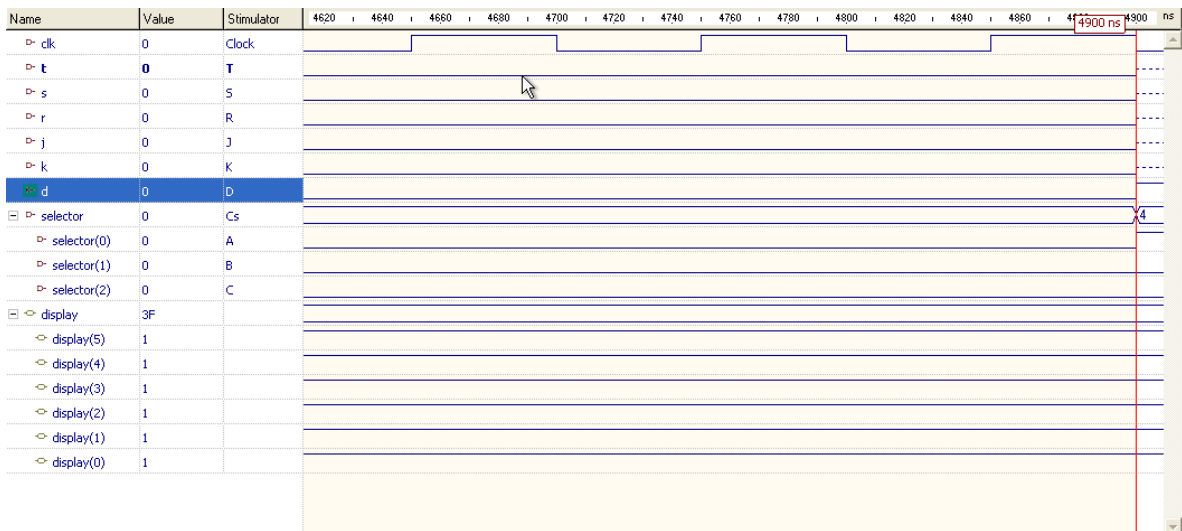
        end if;
    end if;
end process;
end aff;

```

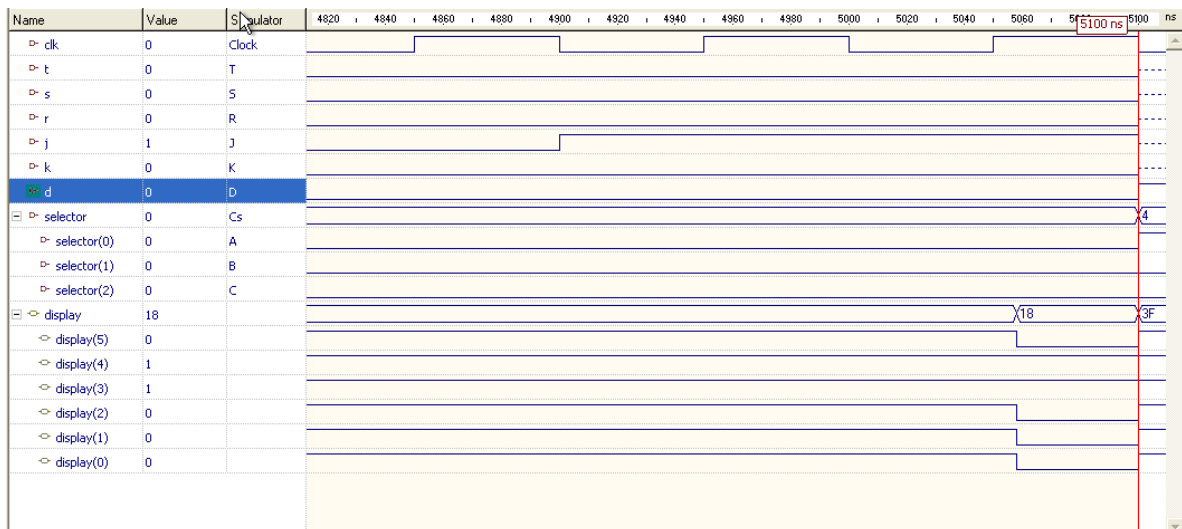
## SIMULCIONES

### FF JK

Dado que el estado inicial de nuestros selectores y entradas están definidos en '0' observamos que por defecto el selector de nuestro multiplexor esta predefinido en la opción de salida del Flip-Flop JK.

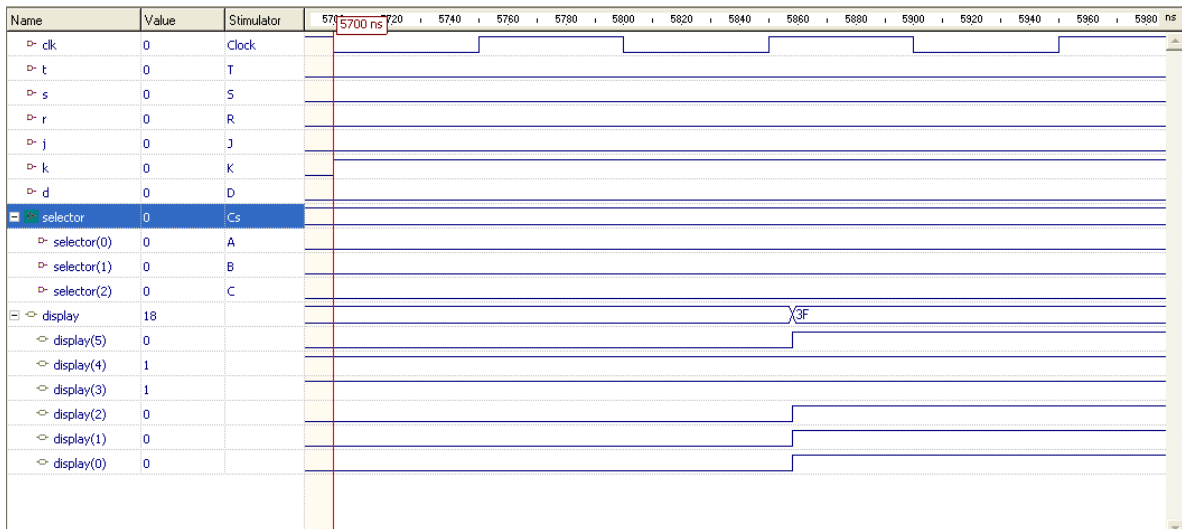


Entonces, para fines prácticos comenzaremos a hacer las pruebas con el FF JK, estableciendo, en principio, un estado set oprimiendo el botón J.

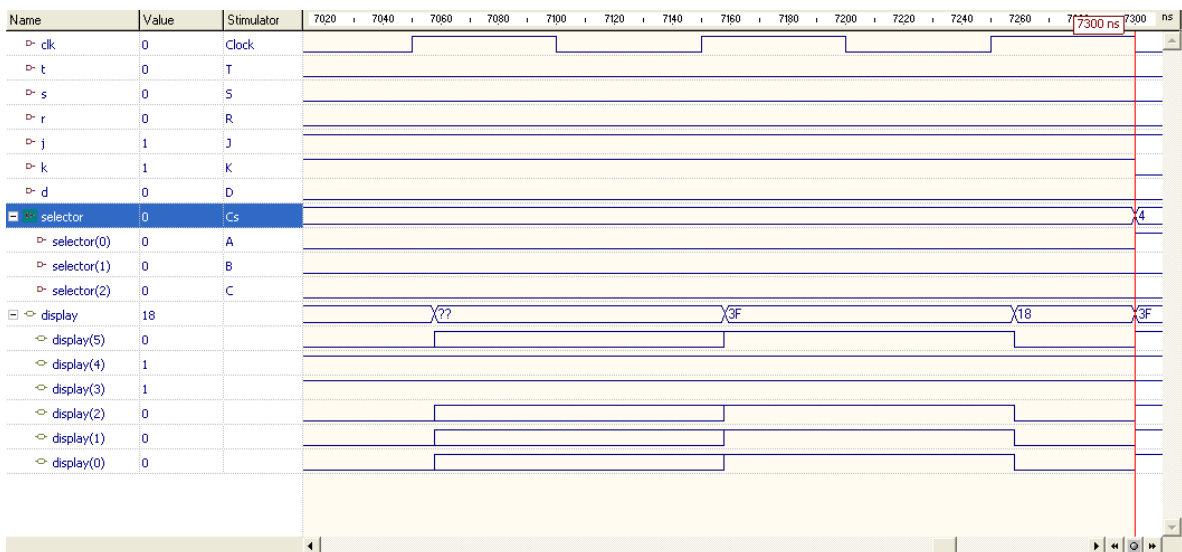


Como podemos observar la salida en nuestro display muestra el número 1 en su notación de 7 segmentos.

Ahora al oprimir resetear la entrada K a '0' observamos que existe retención del dato a la salida.

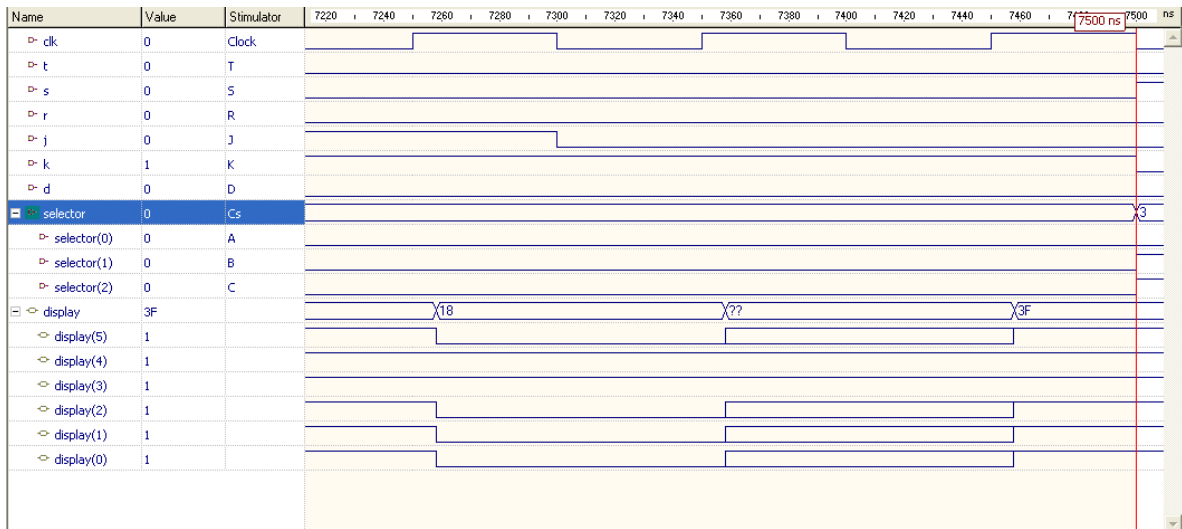


Ahora activaremos tanto J como K para activar el estado de conmutación.



Cabe recalcar que el estado de conmutación estará variando entre '1' y '0' ya que, de hecho, QJK también estará variando de esa manera.

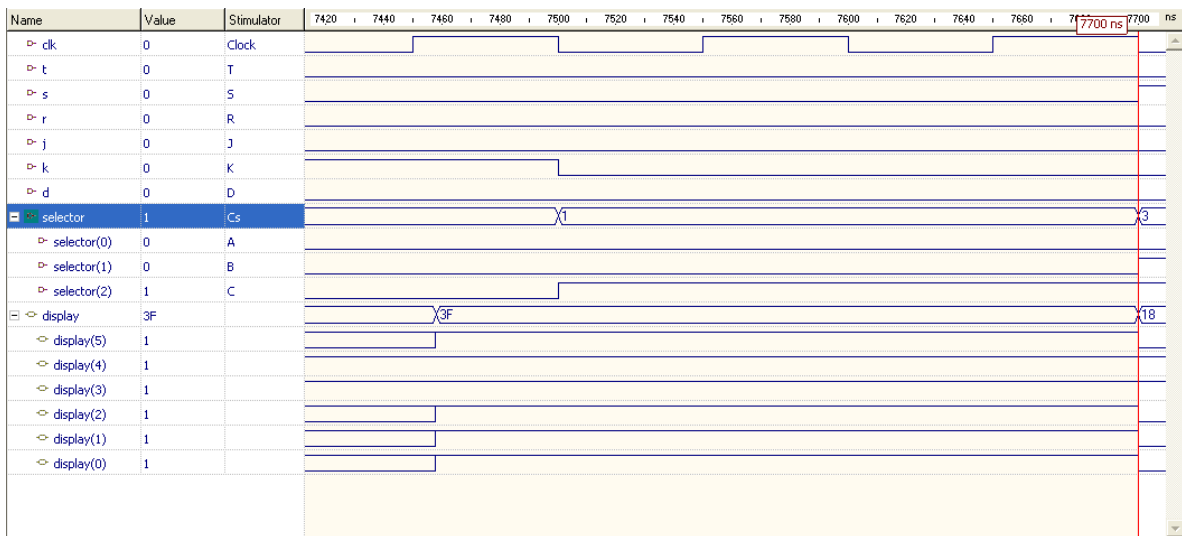
Finalmente dejaremos simplemente activada la entrada K, para "resetear" el flip-flop.



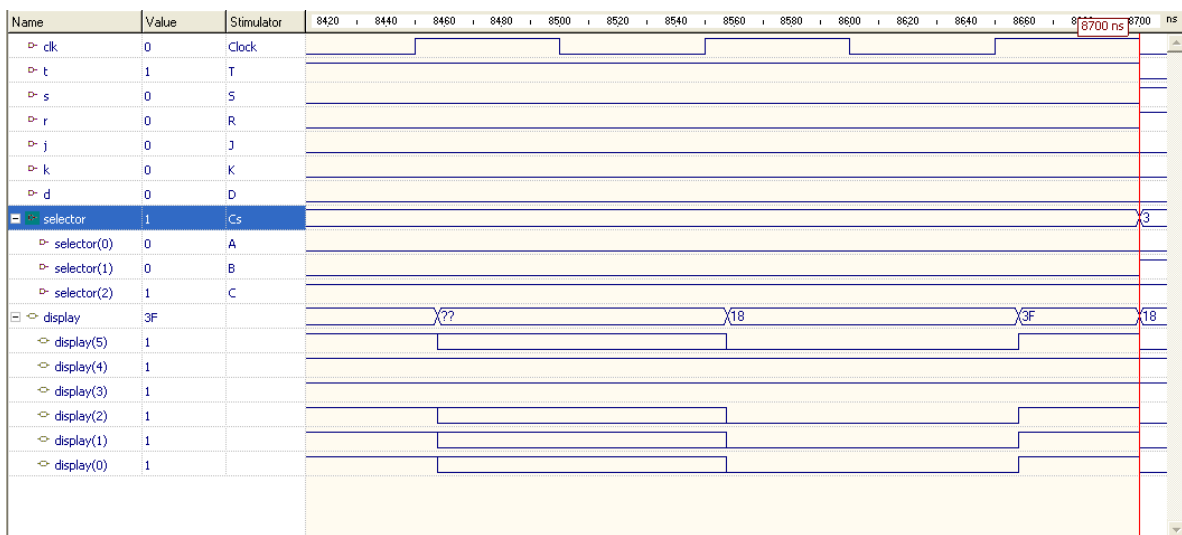
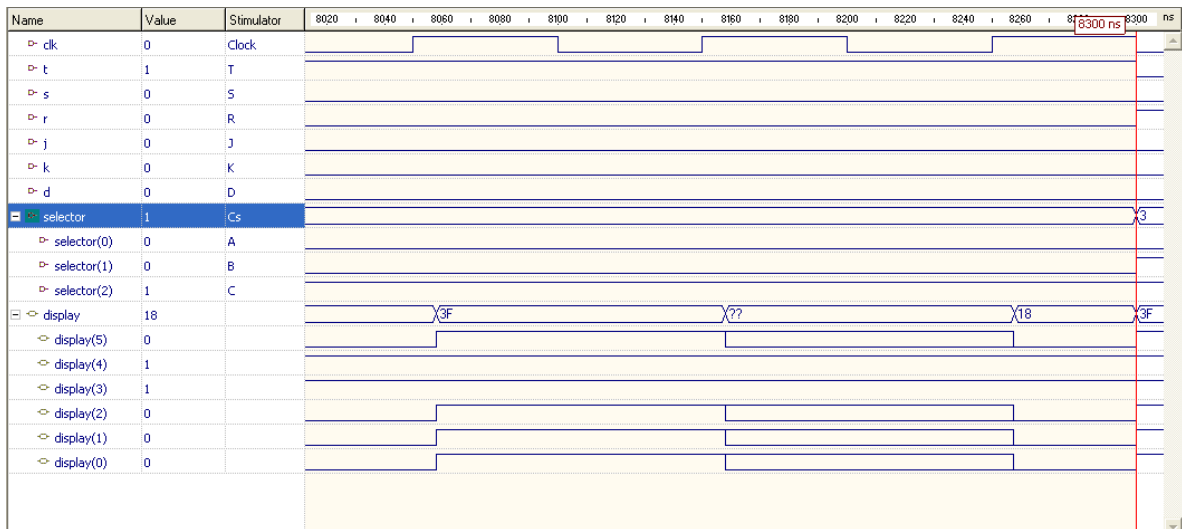
## Flip- Flop tipo T

El siguiente FF a simular será el tipo T. El funcionamiento del Flip-Flop tipo T se resume a un estado de complementación cuando este esta establecido en '1', y a un comportamiento "amo-esclavo" cuando esta establecido en '0'.

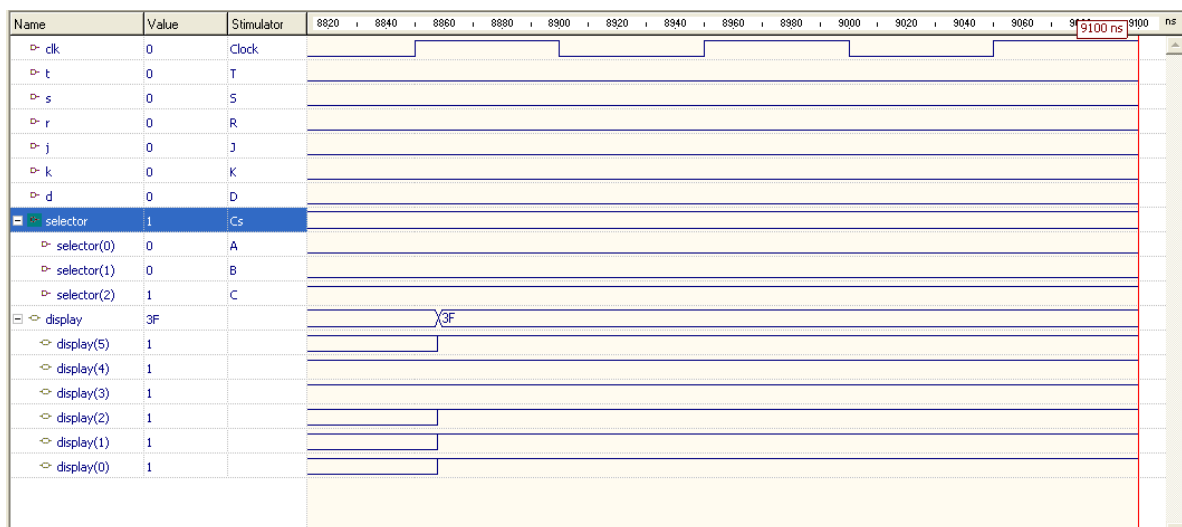
Hemos configurado el selector en la opción "01", por lo cual ahora analiza la entrada T y la señal de salida QT. Dado que todas se encuentran en '0' obtendremos eso mismo a la salida.



Ahora pondremos '1' en la entrada T, observar que en este estado existe un estado de conmutación, pues cada que cambia QT, al existir una retroalimentación, se afecta a si misma directamente.

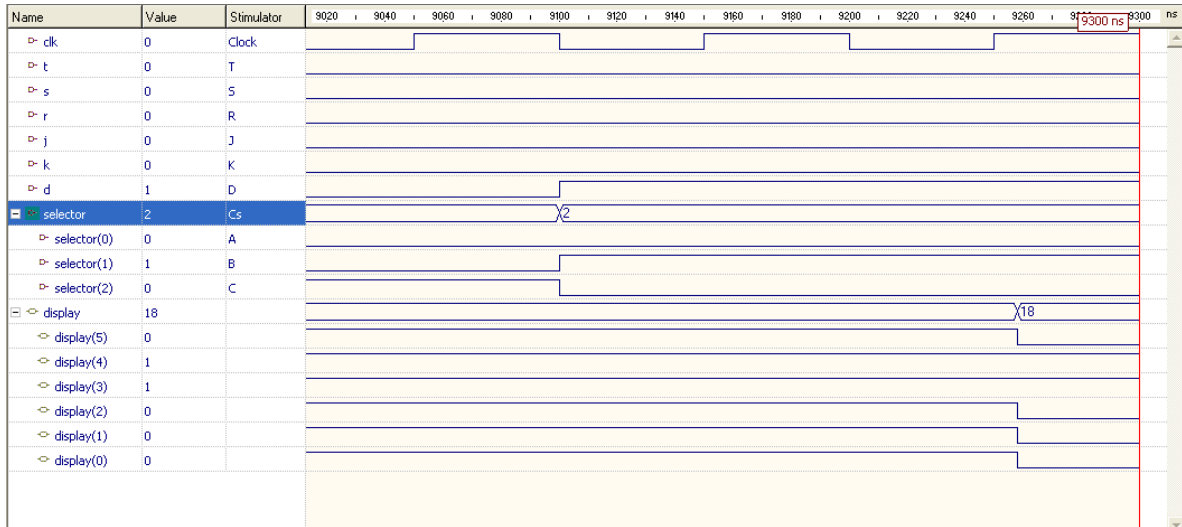


Finalmente, al desactivar T se quedara simplemente RETENIDO el ultimo dato guardado en la salida del FF.

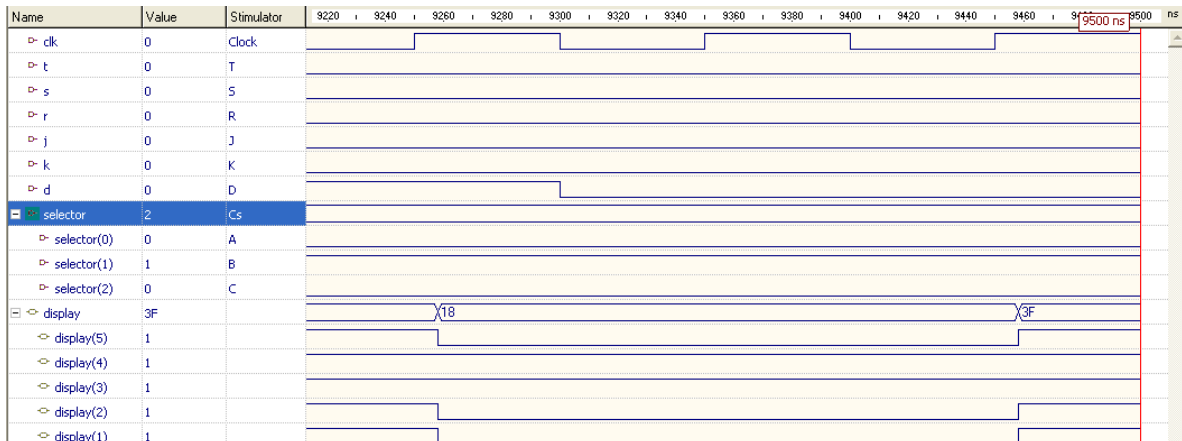


## FF tipo D

El FF tipo D es quizá el mas sencillo de usar de todos ya que su comportamiento es simplemente el de amo esclavo, es decir que la señal de salida QD siempre será igual a la entrada D. A continuación, hemos seleccionado la señal QD como salida del multiplexor, y hemos introducido un '1' a la entrada D.



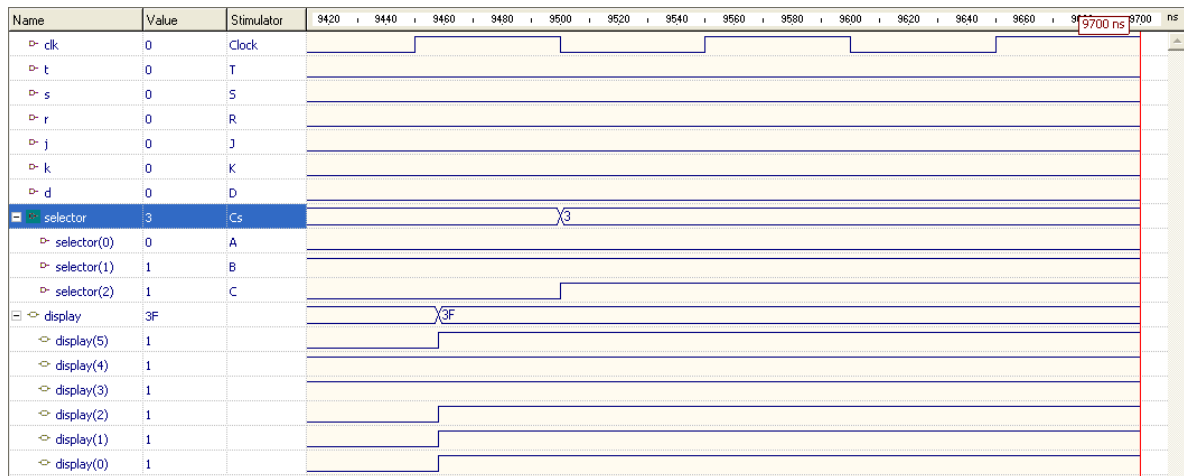
Finalmente, cambiaremos la señal de entrada D a '0'.



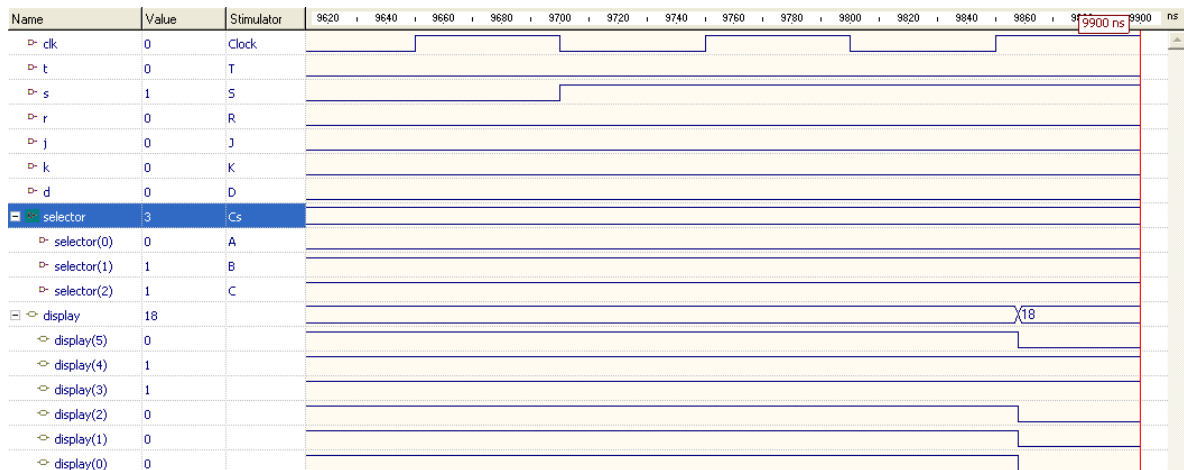
## FF tipo SR

El FF tipo SR será el ultimo FF a analizar, pues posee características un tanto distintas. Al igual que el resto este FF cuenta con un estado de retención, sin embargo, a diferencia del tipo JK, este no puede conmutar directamente con los pulsos de reloj.

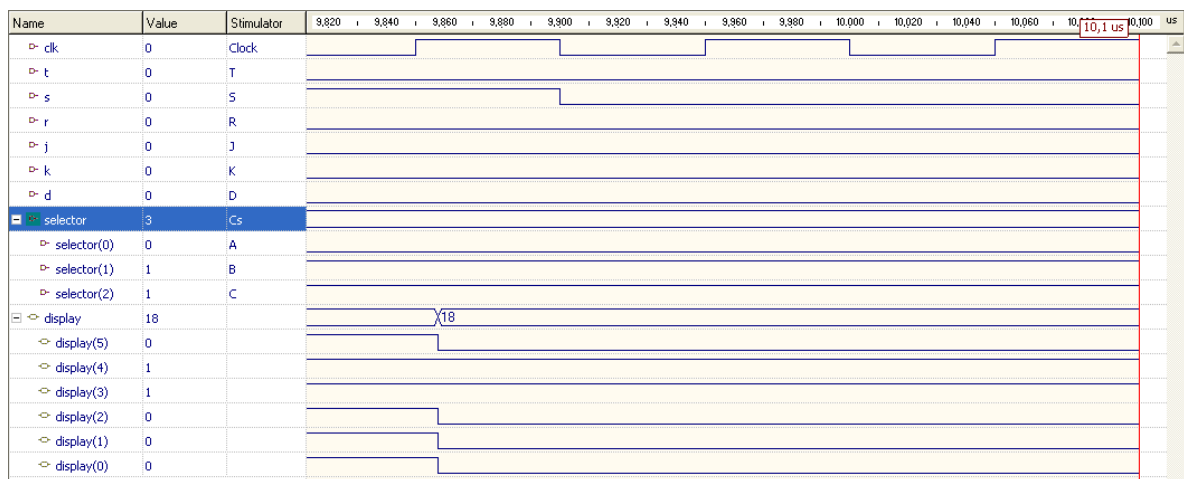
Observemos que al inicio, tanto S como R están establecidos en '0' a las entradas, por lo cual en la salida esperaremos ver simplemente un '0'.



Ahora configuraremos la entrada S en '1', estableciendo de ese modo, el mismo valor a la salida.

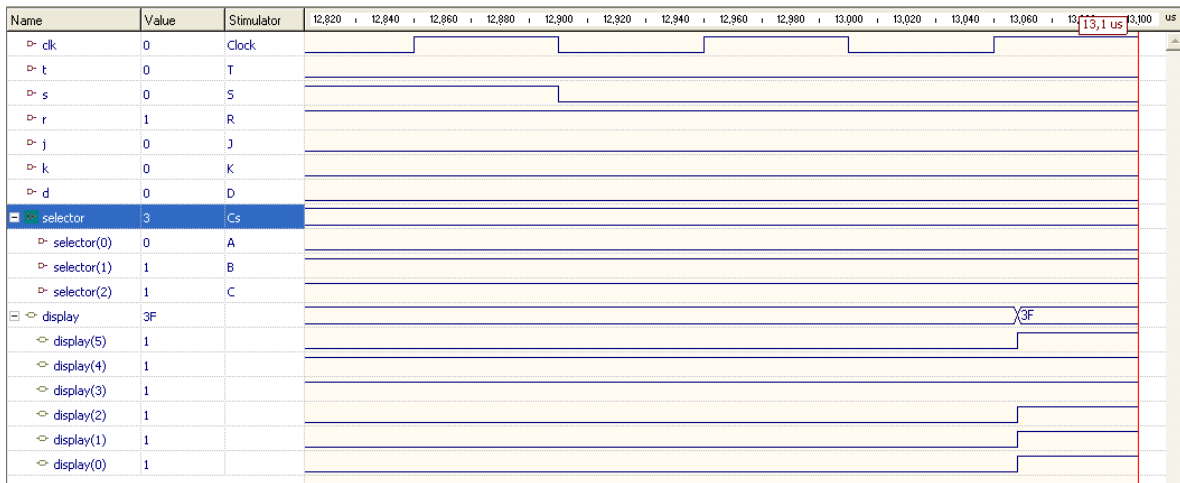


Ahora observamos que, aunque regresemos S a '0' existe el estado de retención. Es decir que dicho estado se establece con S = R = '0'.

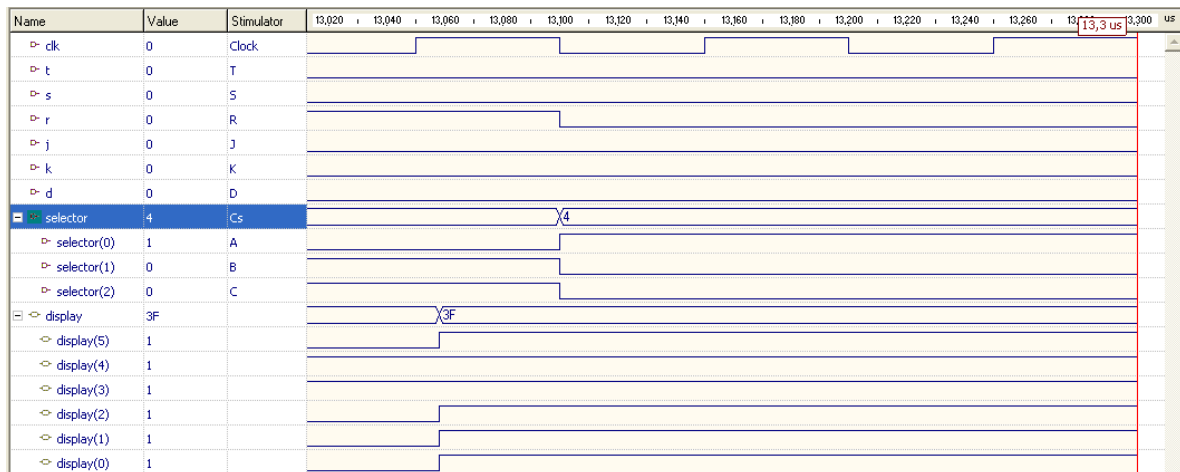




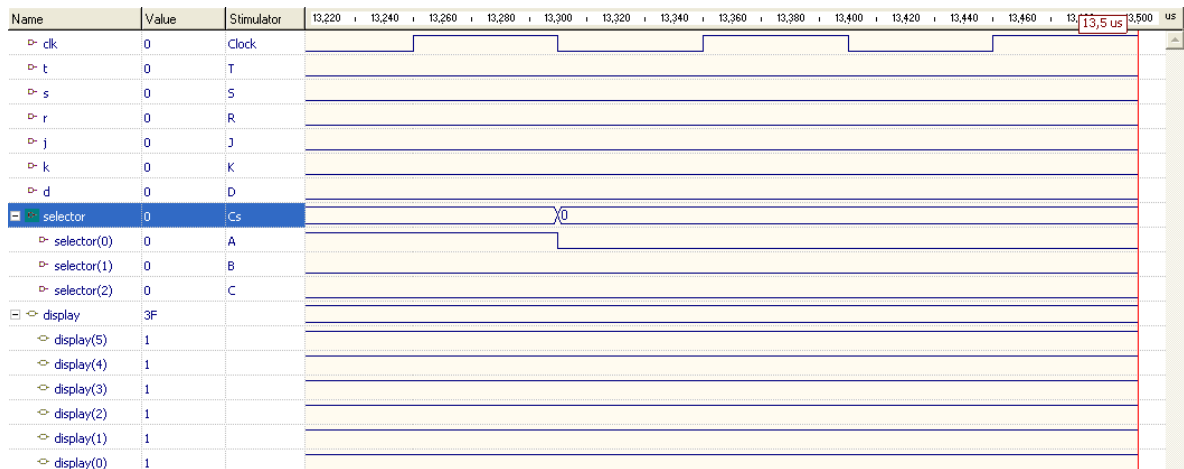
Finalmente regresaremos la señal de salida QSR a '0' asignándole a S un valor de '0' y a R= '1'.



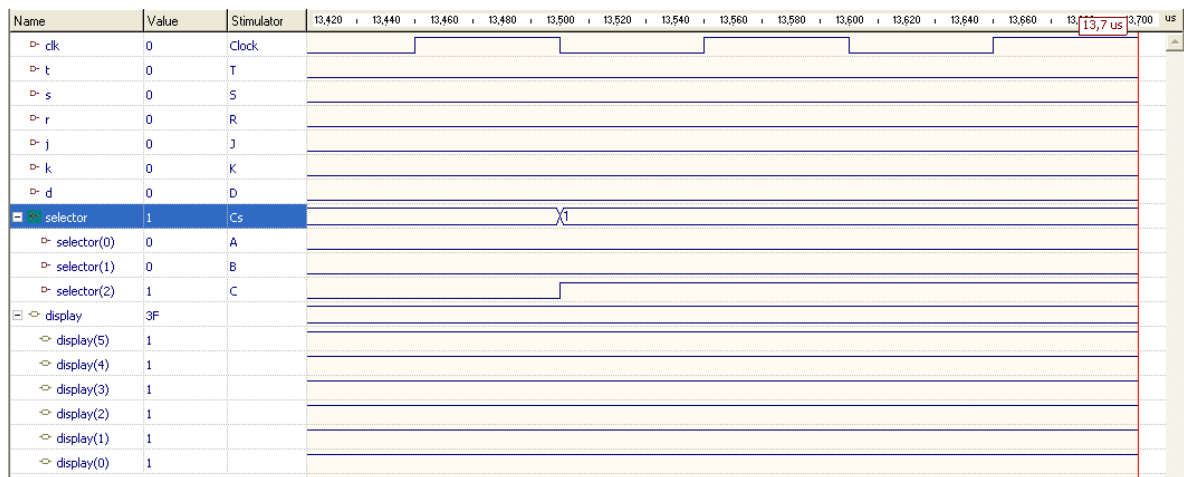
Ahora, simplemente para finalizar, observaremos que al introducir un 1 en CLR (selector(0)), se limpian todas las señales de salida, obteniendo de ese modo, simplemente 0.



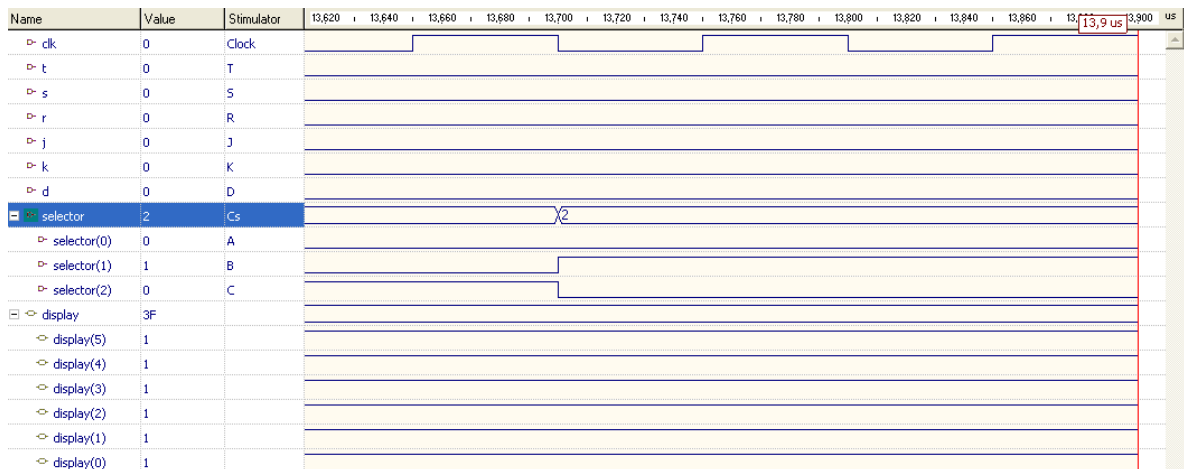
QJK



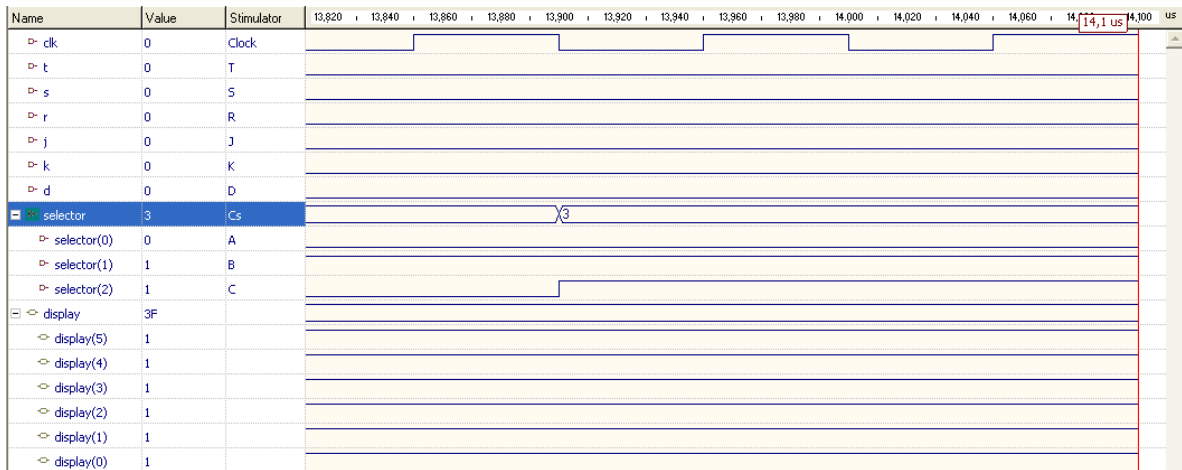
QT



QD



## QSR



De este modo probamos, mediante simulaciones, que el código implementado funciona.