PRACTICA 1

Edgar A. Ramos Mesas 2013090243

2CV10 DISEÑO DE SISTEMAS DIGITALES

CODIGO IMPLEMENTADO

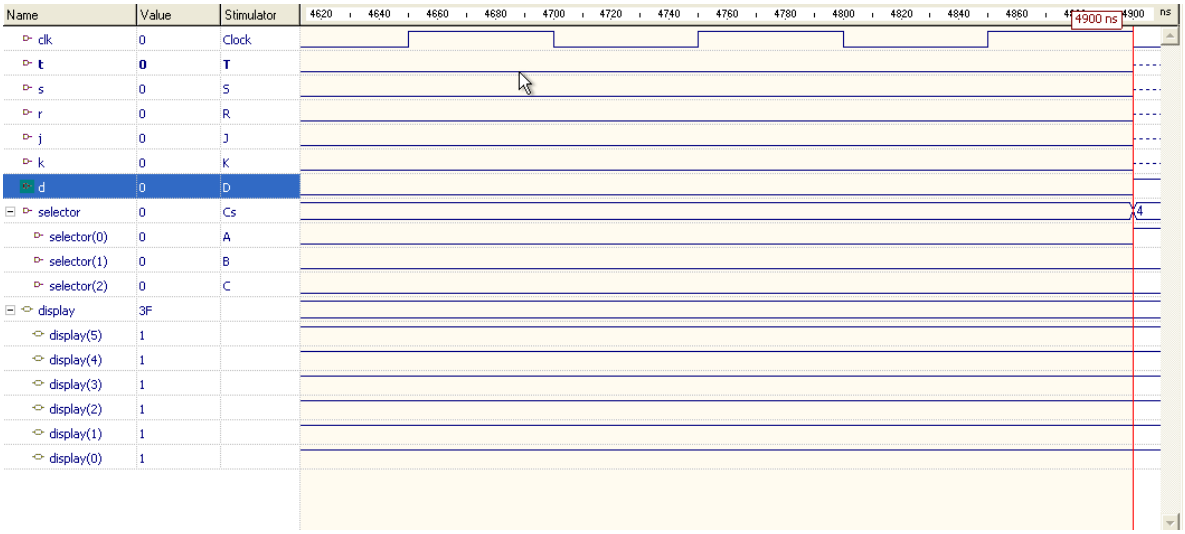
```
library ieee;
use ieee.std_logic_1164.all;
entity ff is port(
	S, R ,J ,K, CLK, D, T: in std_logic;
	selector:  in std_logic_vector(0  to 2);
	display: out std_logic_vector(5 downto 0)
);
attribute pin_numbers of ff:entity is
"CLR:10";
end ff;

architecture aff of ff is
	SIGNAL qsr, qjk, qd, qt, mux_out: std_logic;
	begin
	process(CLK, selector(0), selector)
	begin
		if selector(0) = '1' then
			qjk    <= '0';
			qt     <= '0';
			qd     <= '0';
			qsr    <= '0';
		elsif CLK'event and CLK='1' then
			qjk <= ((not K) and qjk) or (J and(not qjk));
			qt  <= ((not T) and qt) or (T and (not qt));
			qd  <= D;
			qsr    <= ((not R) and qsr) or S;
		end if;
	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 process;
end aff;
```

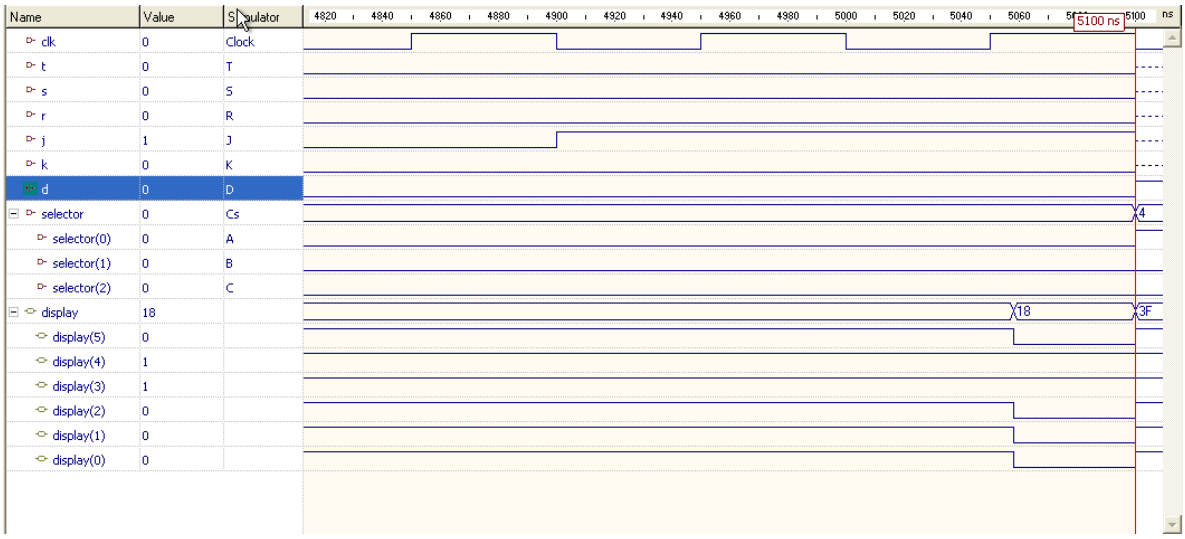
SIMULCIONES

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.

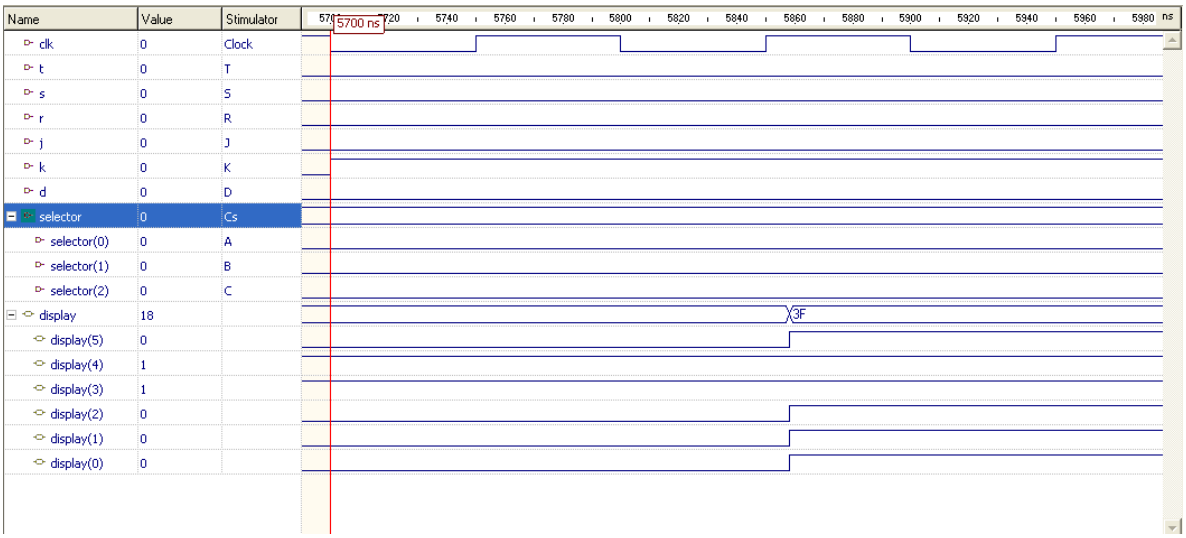


Entonces, para fines prácticos comenzaremos a hacer las pruebas con el FF JK, estableciendo, en principio, un estado set oprimiendo el botón J.

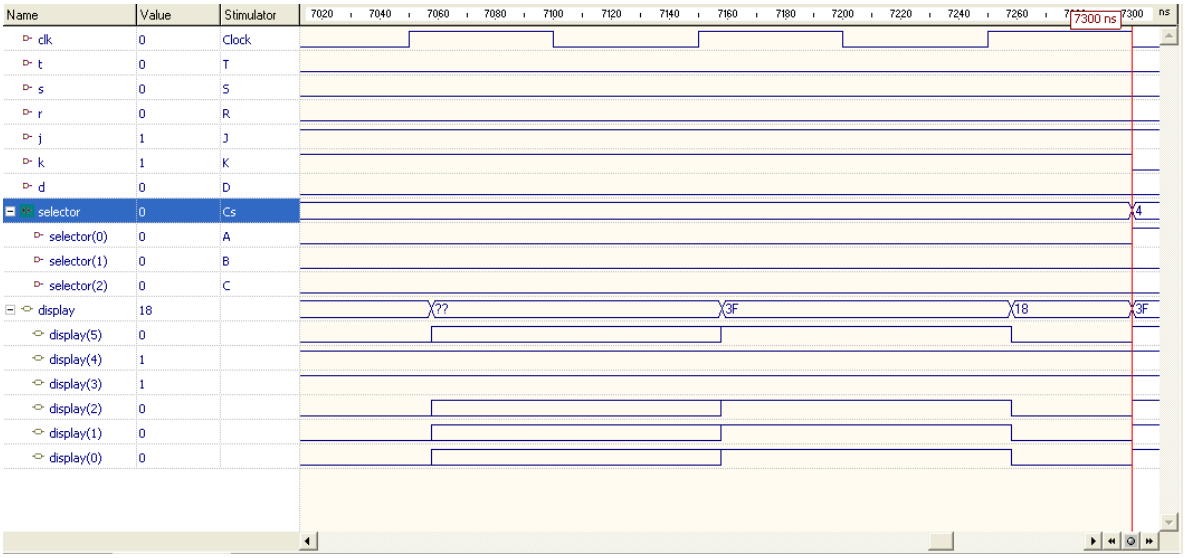


Como podemos observar la salida en nuestro display muestra el número 1 en su notación de 7 segmentos.

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

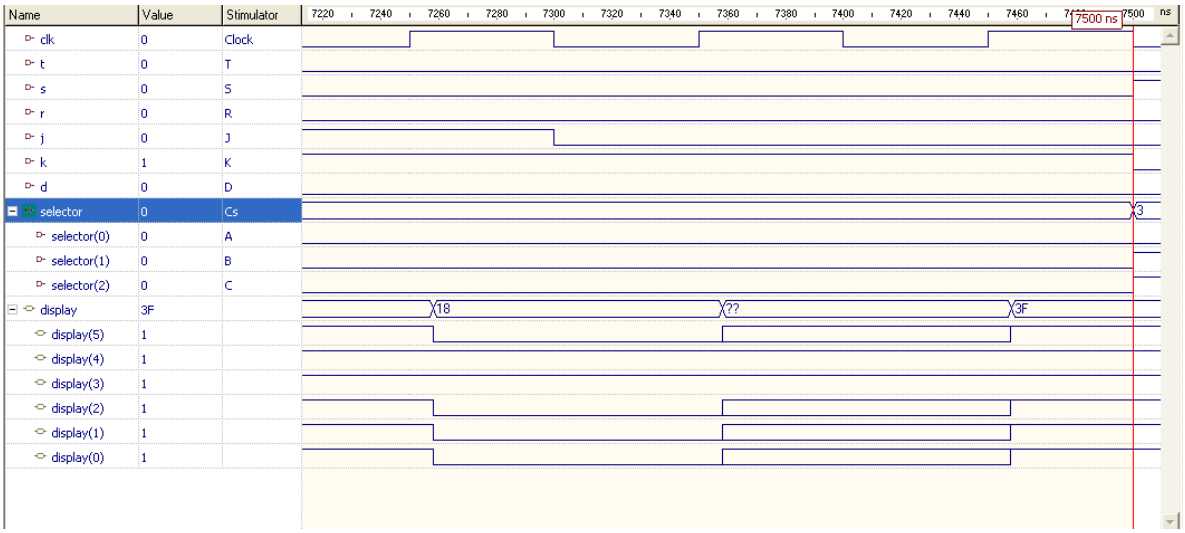


Ahora activaremos tanto J como K para activar el estado de conmutación.



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

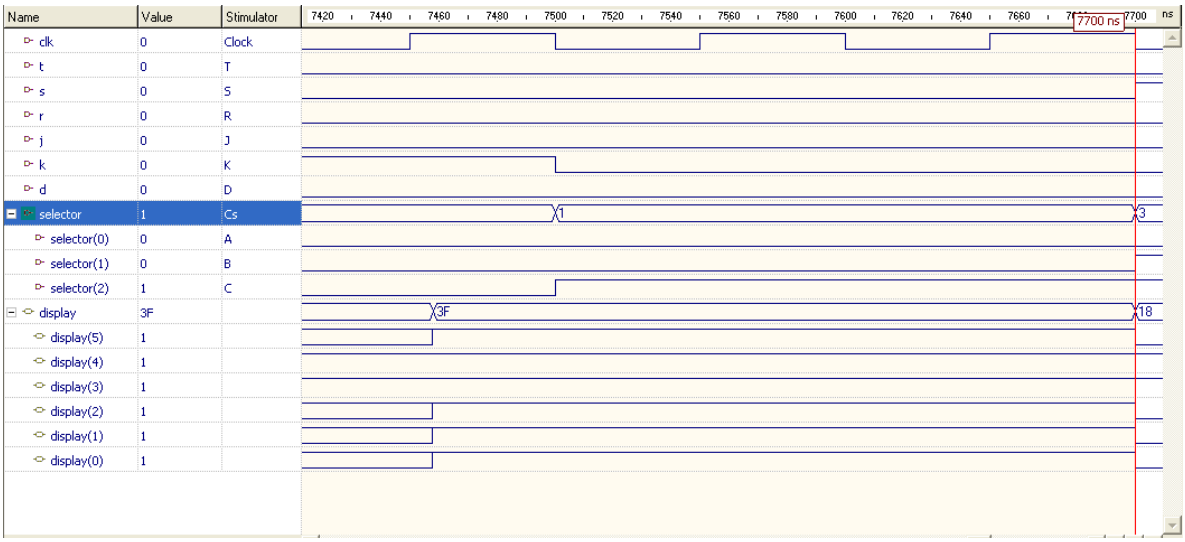
Finalmente dejaremos simplemente activada la entrada K, para “resetear” el flip-flop.



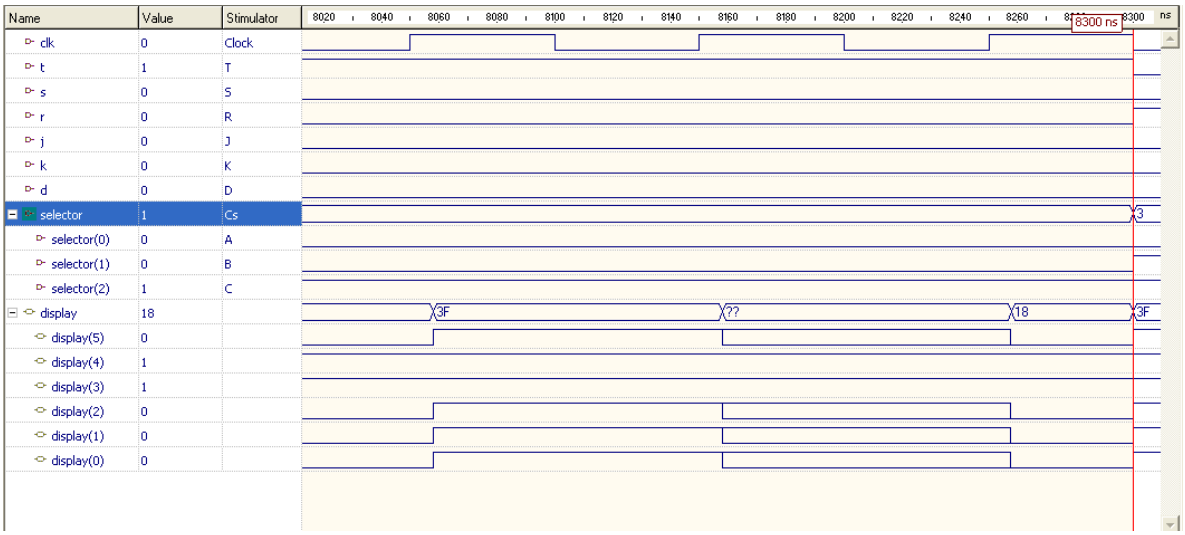
Flip- Flop tipo T

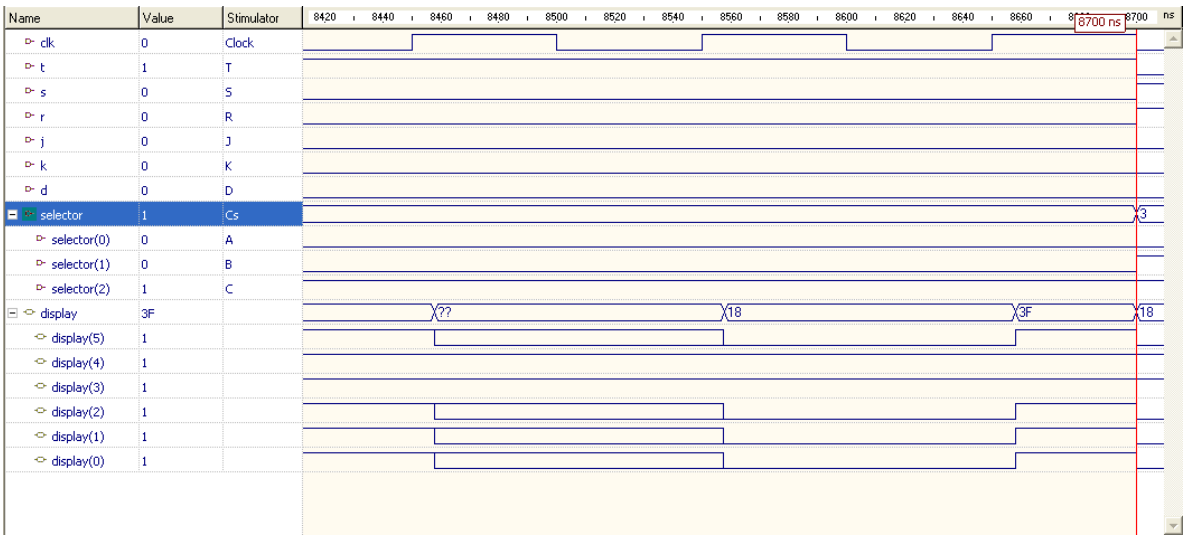
El siguiente FF a simular será el tipo T. El funcionamiento del Flip-Flop tipo T se resume a un estado de complementación cuando este esta establecido en ‘1’, y a un comportamiento “amo-esclavo” cuando esta establecido en ‘0’.

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.

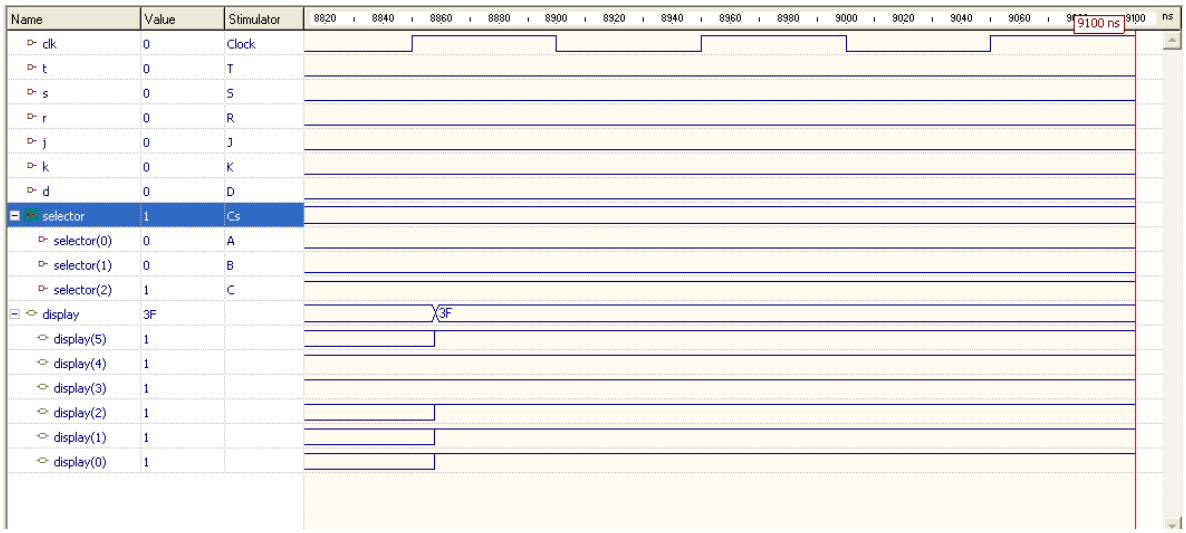


Ahora pondremos ‘1’ en la entrada T, observar que en este estado existe un estado de conmutación, pues cada que cambia QT, al existir una retroalimentación, se afecta a si misma directamente.



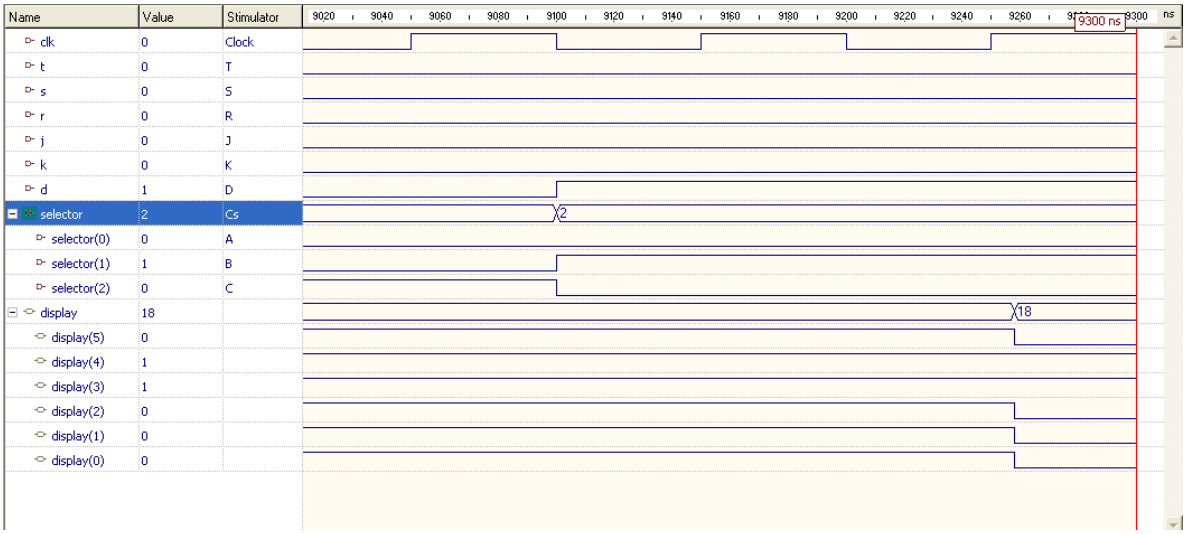


Finalmente, al desactivar T se quedara simplemente RETENIDO el ultimo dato guardado en la salida del FF.

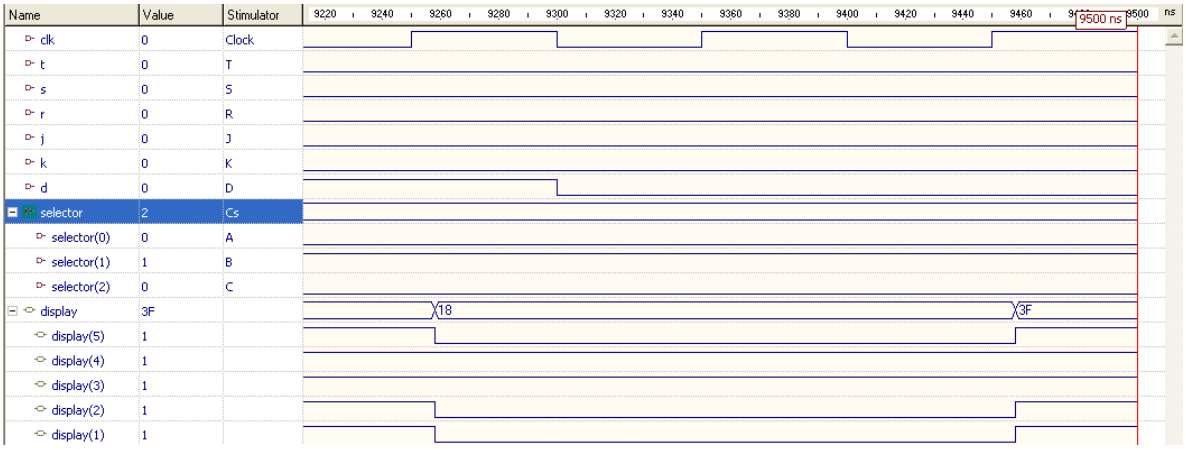


FF tipo D

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.



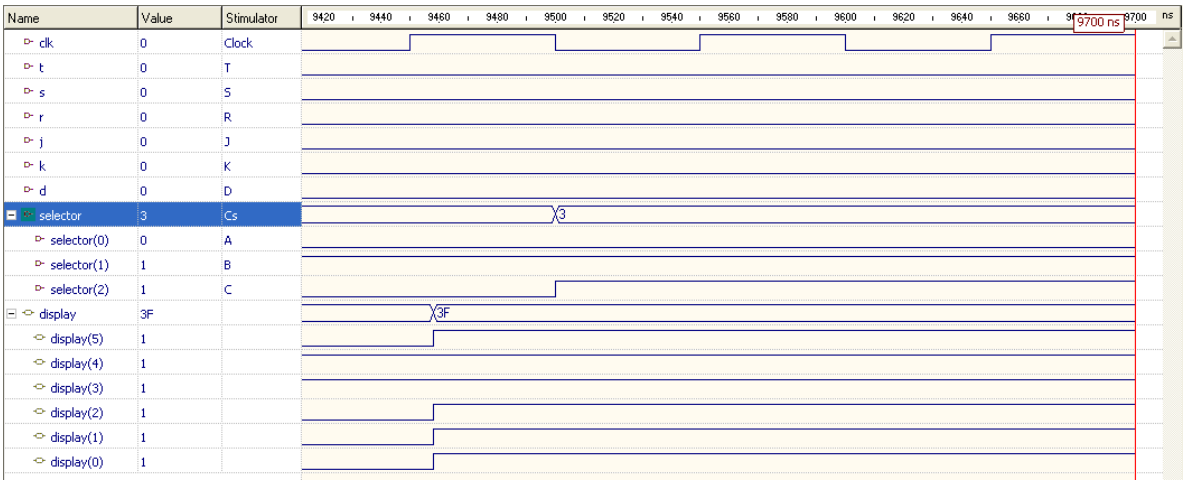
Finalmente, cambiaremos la señal de entrada D a ‘0’.



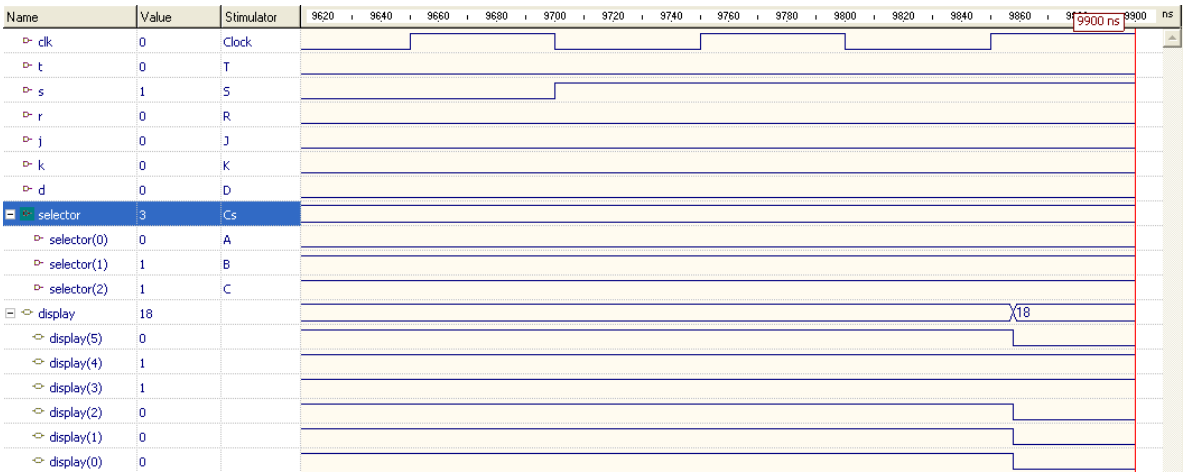
FF tipo SR

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.

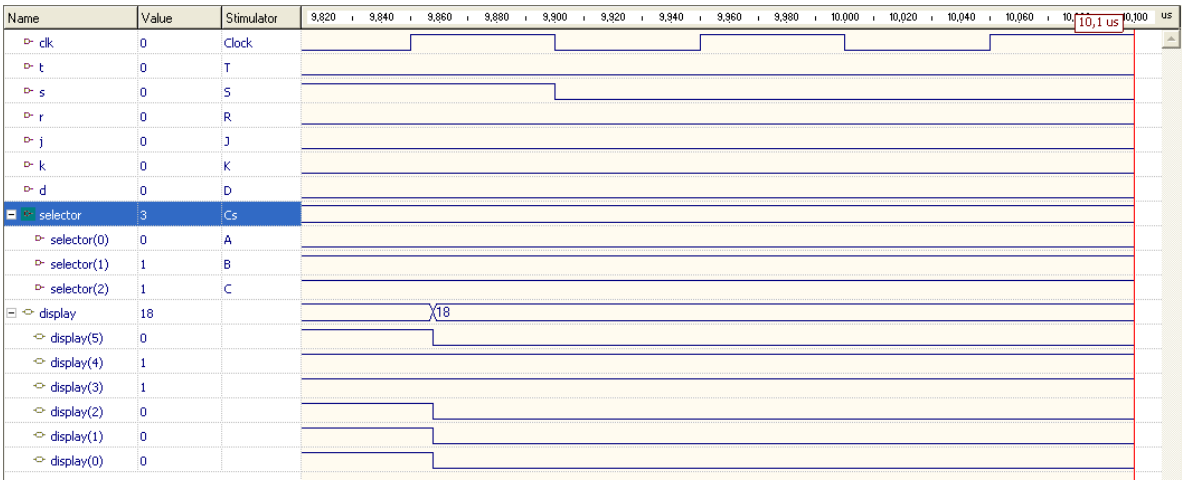
Observemos que al inicio, tanto S como R están establecidos en ‘0’ a las entradas, por lo cual en la salida esperaremos ver simplemente un ‘0’.



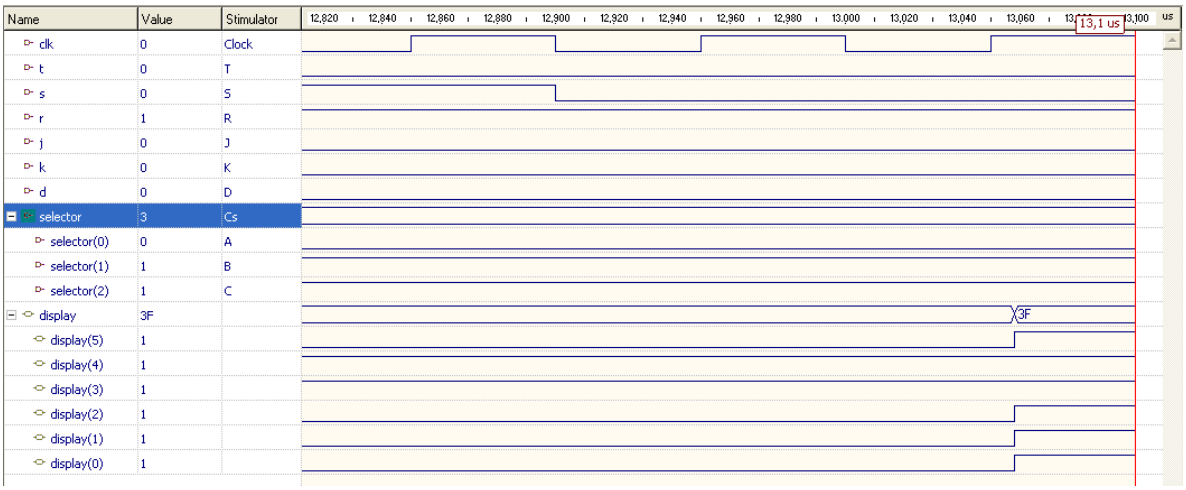
Ahora configuraremos la entrada S en ‘1’, estableciendo de ese modo, el mismo valor a la salida.



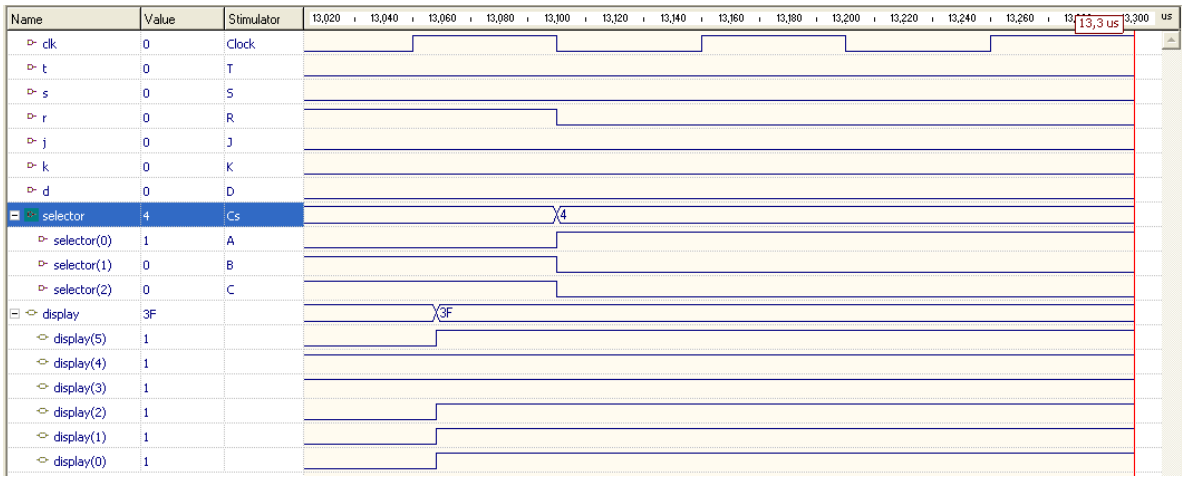
Ahora observamos que, aunque regresemos S a ‘0’ existe el estado de retención. Es decir que dicho estado se establece con S = R = ‘0’.



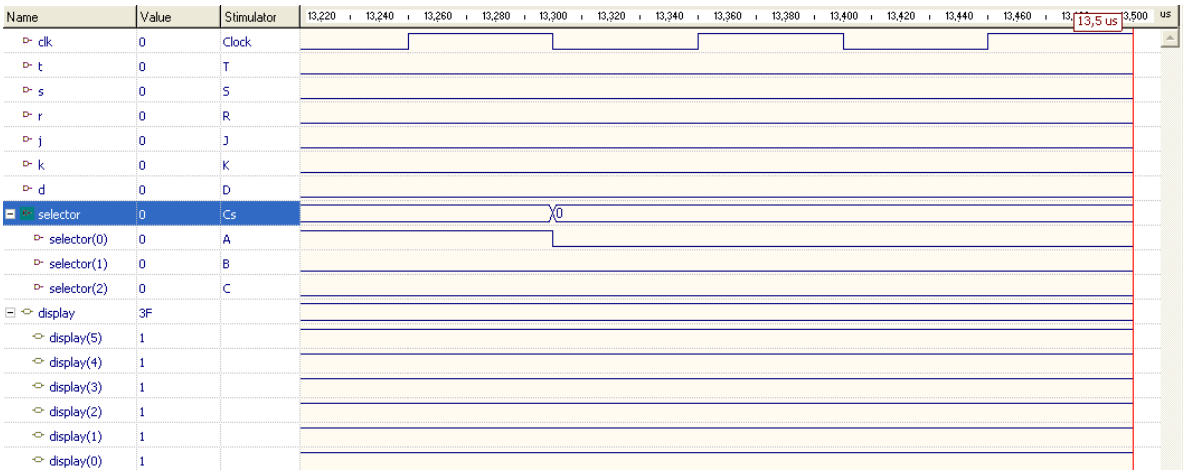
Finalmente regresaremos la señal de salida QSR a ‘0’ asignándole a S un valor de ‘0’ y a R= ‘1’.



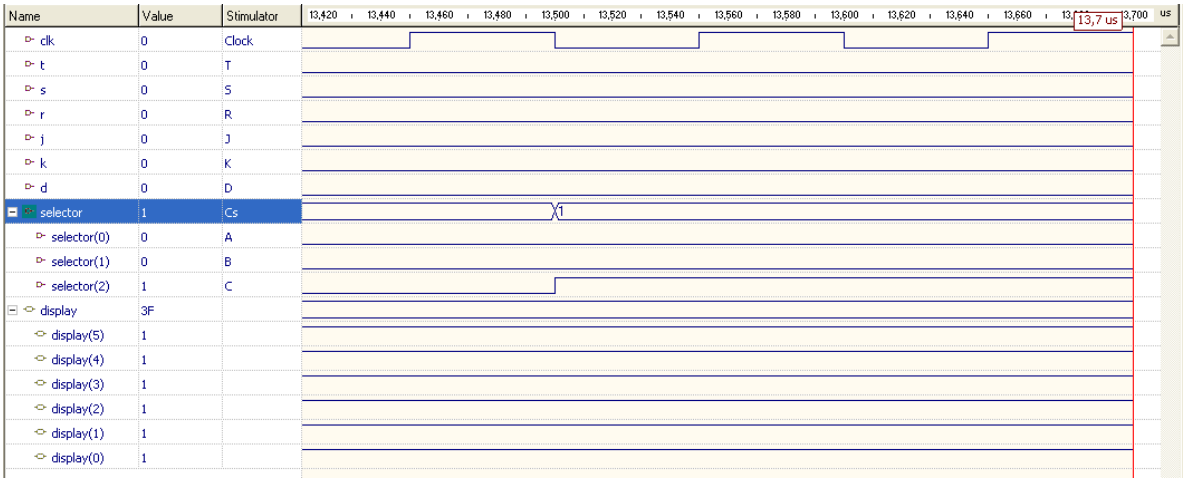
Ahora, simplemente para finalizar, observaremos que al introducir un 1 en CLR (selector(0)), se limpian todas las señales de salida, obteniendo de ese modo, simplemente 0.



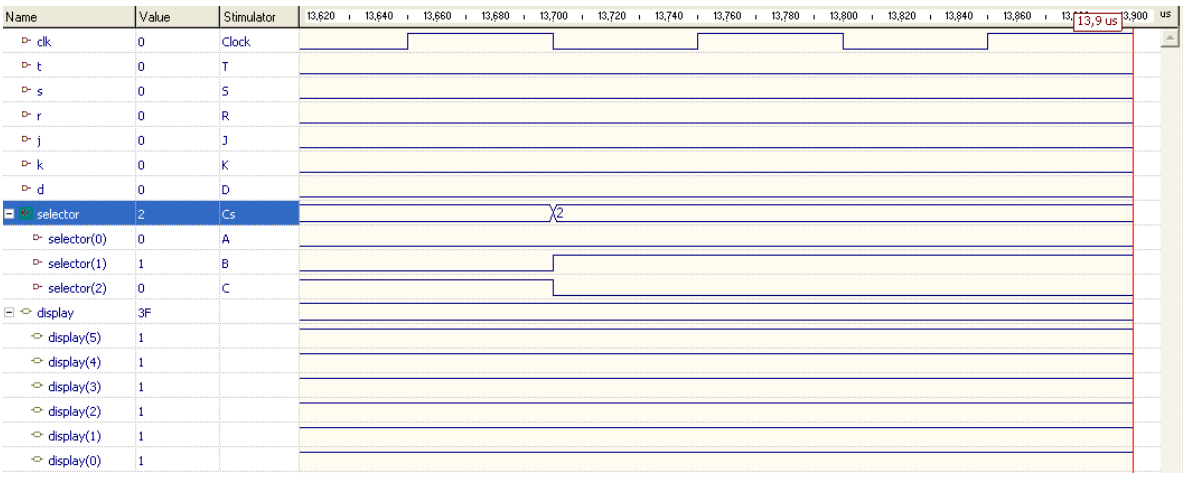
QJK


Name
Value
Stimulator
13,220 13,240 13,260 13,280 13,300 13,320 13,340 13,360 13,380 13,400 13,420 13,440 13,460 13,500 us
13,5 us
clk 0 Clock
t 0 T
s 0 S
r 0 R
j 0 J
k 0 K
d 0 D
selector 0 Cs
0
selector(0) 0 A
selector(1) 0 B
selector(2) 0 C
display 3F
display(5) 1
display(4) 1
display(3) 1
display(2) 1
display(1) 1
display(0) 1

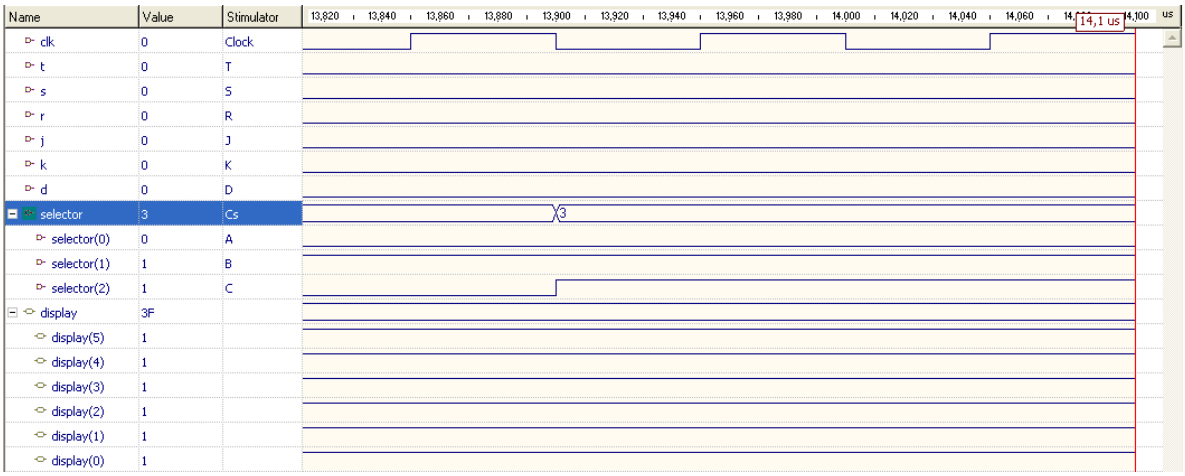
QT


Name
Value
Stimulator
13,420 13,440 13,460 13,480 13,500 13,520 13,540 13,560 13,580 13,600 13,620 13,640 13,660 13,700 us
13,7 us
clk 0 Clock
t 0 T
s 0 S
r 0 R
j 0 J
k 0 K
d 0 D
selector 1 Cs
1
selector(0) 0 A
selector(1) 0 B
selector(2) 1 C
display 3F
display(5) 1
display(4) 1
display(3) 1
display(2) 1
display(1) 1
display(0) 1

QD


Name
Value
Stimulator
13,620 13,640 13,660 13,680 13,700 13,720 13,740 13,760 13,780 13,800 13,820 13,840 13,860 13,900 us
13,9 us
clk 0 Clock
t 0 T
s 0 S
r 0 R
j 0 J
k 0 K
d 0 D
selector 2 Cs
2
selector(0) 0 A
selector(1) 1 B
selector(2) 0 C
display 3F
display(5) 1
display(4) 1
display(3) 1
display(2) 1
display(1) 1
display(0) 1

QSR



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