**RESUMEN**

**Compuertas logicas**

**Compuerta NOT**

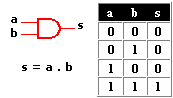
Realiza la función [booleana](http://es.wikipedia.org/wiki/%C3%81lgebra_de_Boole) de inversión o negación de una variable lógica.

La [ecuación](http://es.wikipedia.org/wiki/Ecuaci%C3%B3n) característica



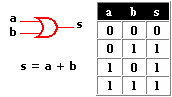
**Compuerta AND**

Realiza la función booleana de producto lógico. La [ecuación](http://es.wikipedia.org/wiki/Ecuaci%C3%B3n) característica . Una compuerta que entrega un 1 lógico sólo si todas las entradas están a *nivel alto* 1.



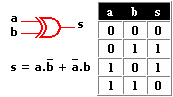
**Compuerta OR**

Realiza la operación de suma lógica. La [ecuación](http://es.wikipedia.org/wiki/Ecuaci%C3%B3n) característica la puerta proporciona a su salida un 1 lógico si al menos una de sus entradas está a 1.



**Compuerta XOR**

Realiza la función booleana A'B+AB'. La [ecuación](http://es.wikipedia.org/wiki/Ecuaci%C3%B3n) característica esta puerta da por resultado uno, cuando los valores en las entradas son distintos



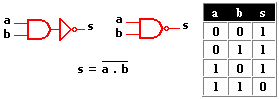
**Compuerta NAND**

Realiza la operación de [producto lógico](http://es.wikipedia.org/wiki/Puerta_l%C3%B3gica#Puerta_Y_.28AND.29) negado. La [ecuación](http://es.wikipedia.org/wiki/Ecuaci%C3%B3n) característica

, proporciona a su salida un 0 lógico únicamente cuando todas sus entradas están a 1.



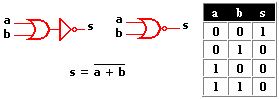
.



**Compuerta NOR**

Realiza la operación de suma lógica negada. La [ecuación](http://es.wikipedia.org/wiki/Ecuaci%C3%B3n) característica

proporciona a su salida un 1 lógico sólo cuando todas sus entradas están a 0.

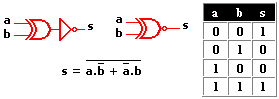


**Compuerta *XNOR***

Realiza la función booleana AB+A'B'. La [ecuación](http://es.wikipedia.org/wiki/Ecuaci%C3%B3n) característica:



Proporciona un 1 lógico, sólo si las dos entradas son iguales, esto es, 0 y 0 ó 1 y 1.



**Buffer's**

Cumple la función booleana igualdad,



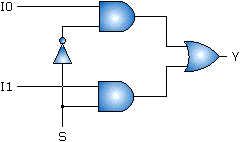
**Multiplexor**

Un multiplexor es un circuito combinacional que selecciona una de *n* líneas de entrada y transmite su información binaria a la salida. La selección de la entrada es controlada por un conjunto de líneas de selección. La relación de líneas de entrada y líneas de selección está dada por la expresión *2n*, donde *n* corresponde al número de líneas de selección y *2n* al número de líneas de entrada.

*Ejemplo: un multiplexor de dos entradas*

|  |  |
| --- | --- |
| S | Y |
| 0 | I0 |
| 1 | I1 |

*Tabla de verdad de un multiplexor de dos entradas*



Un demultiplexor es un circuito combinacional que recibe información en una sola línea y la transmite a una de *2n* líneas posibles de salida. La operación es contraria al multiplexor.

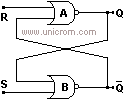
**FLIP FLOP**

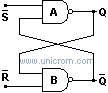
* Asíncronos: sólo tienen entradas de control. El más empleado es el [biestable RS](http://es.wikipedia.org/wiki/Biestable#Biestable_RS).
* Síncronos: además de las entradas de control posee una entrada de sincronismo o de reloj. Si las entradas de control dependen de la de sincronismo se denominan síncronas y en caso contrario asíncronas.

**FLIP FLOP RS**

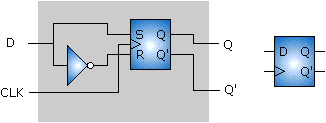
Este dispositivo de almacenamiento temporal se compone de dos [compuertas NOR](http://www.unicrom.com/Tut_compuertanor.asp) conectadas como se muestra en la figura, (ver las realimentaciones) y se llama RS porque sus entradas tiene los nombres SET (poner un "1" en la salida Q) y RESET (reponer o poner a "0" la salida Q)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operación | Entradas | | Salidas | |
| S | R | Q | Q |
| Memoria | 0 | 0 | Qo | Qo |
| Reset | 0 | 1 | 0 | 1 |
| Set | 1 | 0 | 1 | 0 |
| Prohibido | 0 | 0 | 0 | 0 |



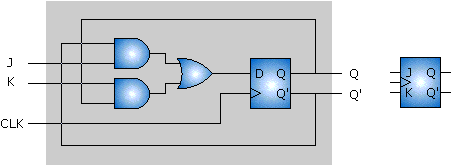
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operación | Entradas | | Salidas | |
| S | R | Q | Q |
| Prohibido | 0 | 0 | 1 | 1 |
| Reset | 0 | 1 | 0 | 1 |
| Set | 1 | 0 | 1 | 0 |
| Memoria | 0 | 0 | Qo | Qo |

**FLIP FLOP D**

Indica que el dato se tranfiere cuando ocurre un pulso de reloj.

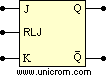
|  |  |  |
| --- | --- | --- |
| D | CLK | Qi+1 |
| 0 | ↑ | 0 |
| 1 | ↑ | 1 |

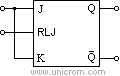
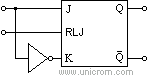
**FLIP FLOP JK**

Tabla de verdad para el FF JK

|  |  |  |  |
| --- | --- | --- | --- |
| J | K | CLK | Qi+1 |
| 0 | 0 | ↑ | Qi |
| 1 | 0 | ↑ | 1 |
| 0 | 1 | ↑ | 0 |
| 1 | 1 | ↑ | Qi' |

En el siguiente diagrama se presenta la representación de un flip-flop tipo JK y las conexiones adicionales que hay que hacer para poder implementar un [flip-flop tipo D](http://www.unicrom.com/dig_FF_D.asp) y un flip-flop tipo T

            
 FF JK                                             FF tipo D                                FF tipo T



**FLIP FLOP T**

T (*Toggle*) cambiando el estado de la salida por su complemento.

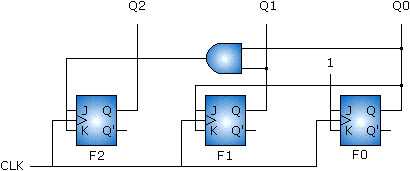
|  |  |  |
| --- | --- | --- |
| T | CLK | Qi+1 |
| 0 | ↑ | Qi |
| 1 | ↑ | Qi' |

http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/images/050507.gif

**CONTADORES**

SINCRONO

La señal de reloj se aplica simultáneamente a todos los *flip-flops*.



Contador ascendente sincrónico de 3 bits

|  |  |  |  |
| --- | --- | --- | --- |
| pulsos | Q2 | Q1 | Q0 |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 |

ASINCRONO

Un contador asincrónico es un arreglo de FF conectados en cascada. En este caso la señal de reloj se aplica sólo al primer FF. Los siguientes entradas de reloj (en los otros FF) se alimentan de la salida Q del FF anterior.

|  |
| --- |
|  |

**LOS REGISTROS DE DESPLAZAMIENTO**

Se utilizan para almacenar y transferir la información de maneras diferentes.

En un registro de desplazamiento la información puede:

- Entrar en serie y salir en serie  
- Entrar en serie y salir en paralelo  
- Entrar en paralelo y salir en serie  
- Entrar en paralelo y salir en paralelo  
- Entrar en serie y salir en serie y paralelo  
- Entrar en serie y paralelo y salir en serie

**Contador en anillo**

.

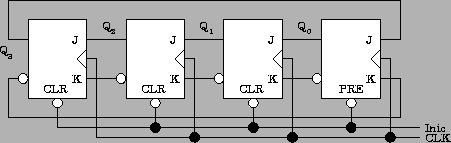


Tabla de estados de un contador en anillo de 4 bits.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pulso | *Q*3 | *Q*2 | *Q*1 | *Q*0 |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 |

**Contador en Johnson**

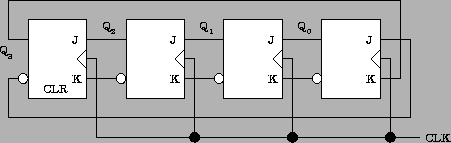


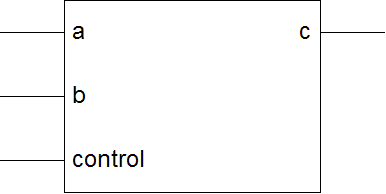
Tabla de estados y lógica de decodificación de un contador Johnson de 4 bits.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pulso | *Q*3 | *Q*2 | *Q*1 | *Q*0 | Decodificador |
| 0 | 0 | 0 | 0 | 0 |  |
| 1 | 0 | 0 | 0 | 1 |  |
| 2 | 0 | 0 | 1 | 1 |  |
| 3 | 0 | 1 | 1 | 1 |  |
| 4 | 1 | 1 | 1 | 1 | *Q*3*Q*0 |
| 5 | 1 | 1 | 1 | 0 |  |
| 6 | 1 | 1 | 0 | 0 |  |
| 7 | 1 | 0 | 0 | 0 |  |

**IMPLEMENTACION DE CODIGO**

**MULTIPLEXOR**

---- Uncomment the following library declaration if instantiating

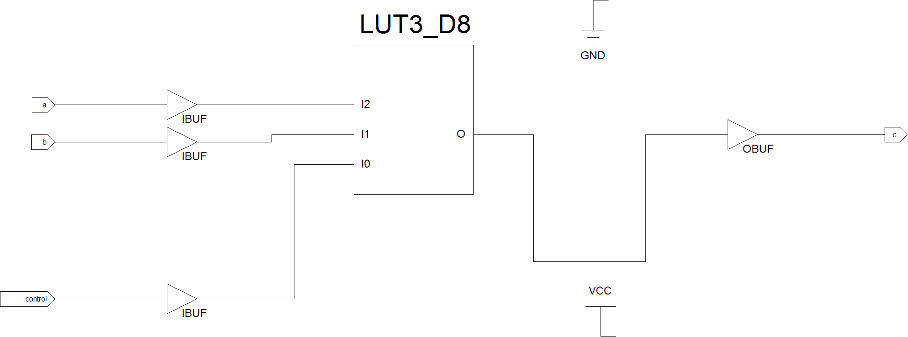
---- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity mux1 is port(

a,b : in bit;

control : in bit;

c : out bit

);

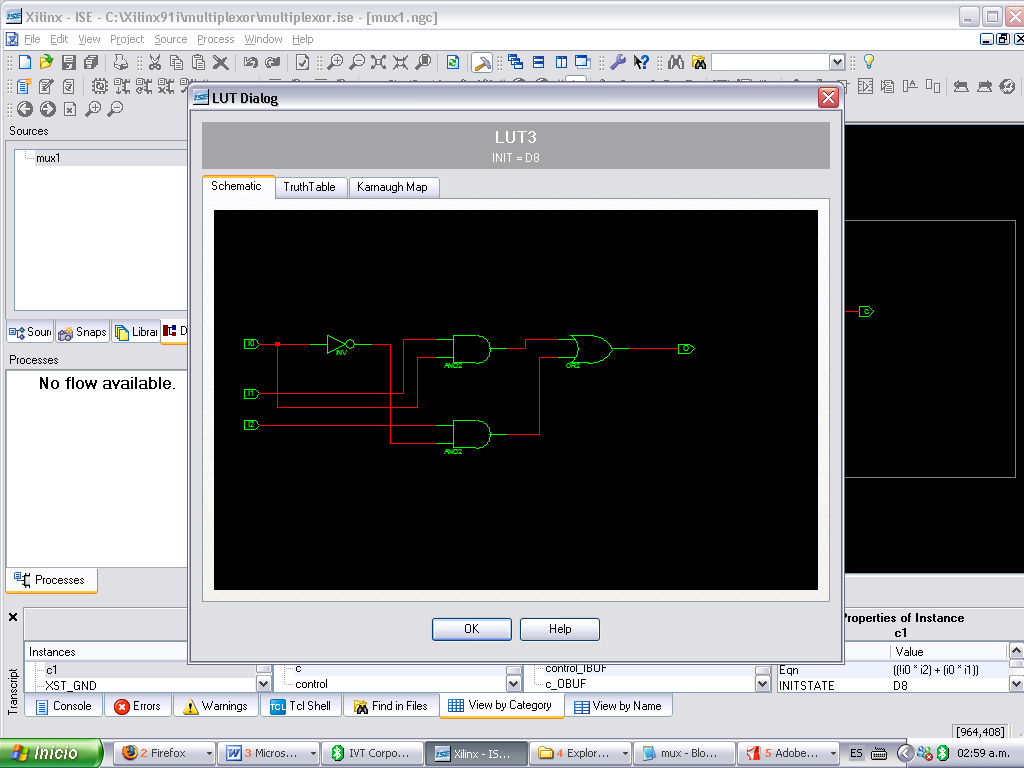
end mux1;

architecture Behavioral of mux1 is

begin

process (a,b,control)

begin

 if (control='0') then

c<=a;

else

c<=b;

end if;

end process;

end Behavioral;

**CONTADOR**

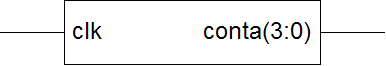
entity regis is port(

clk: in std\_logic;

conta: buffer std\_logic\_vector(3 downto 0)

);

end regis;

architecture Behavioral of regis is

begin

process (clk)

begin

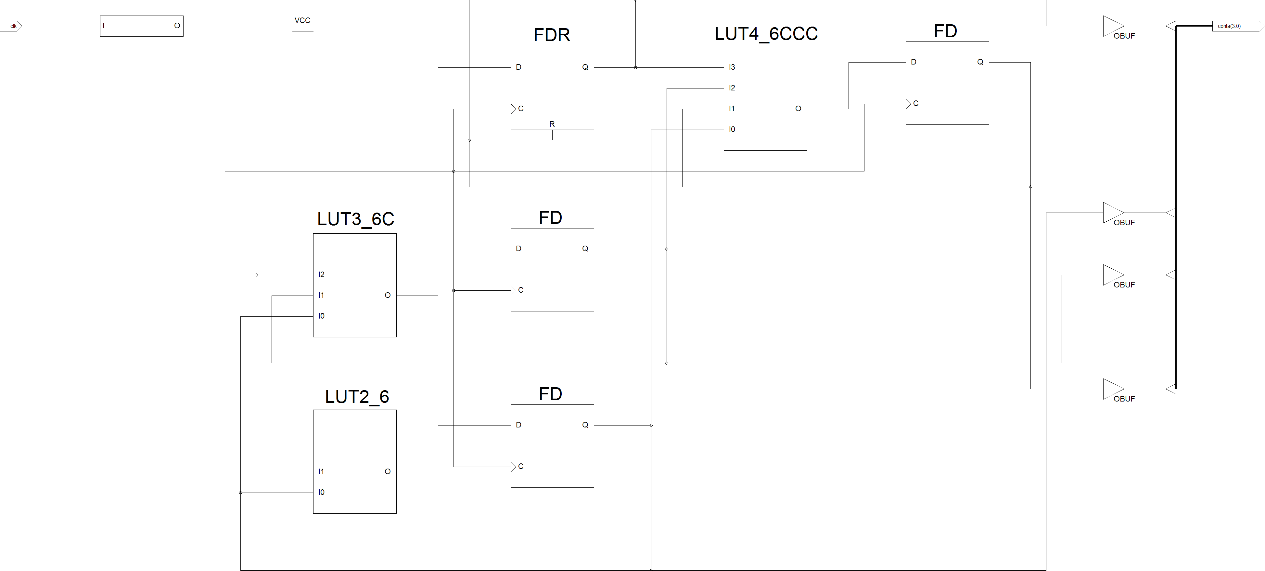
if (clk'event and clk= '1') then

conta <= conta + 1;

end if;

end process;

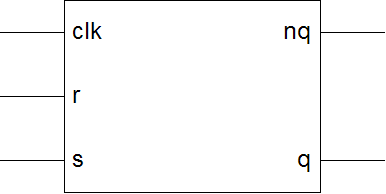
end Behavioral;



**FLIP FLOP RS**

entity ff is port (

r,s, clk: in std\_logic;

 q: buffer std\_logic;

notq: out std\_logic

);

end ff;

architecture Behavioral of ff is

begin

process (clk)

begin

if (clk'event and clk='1') then

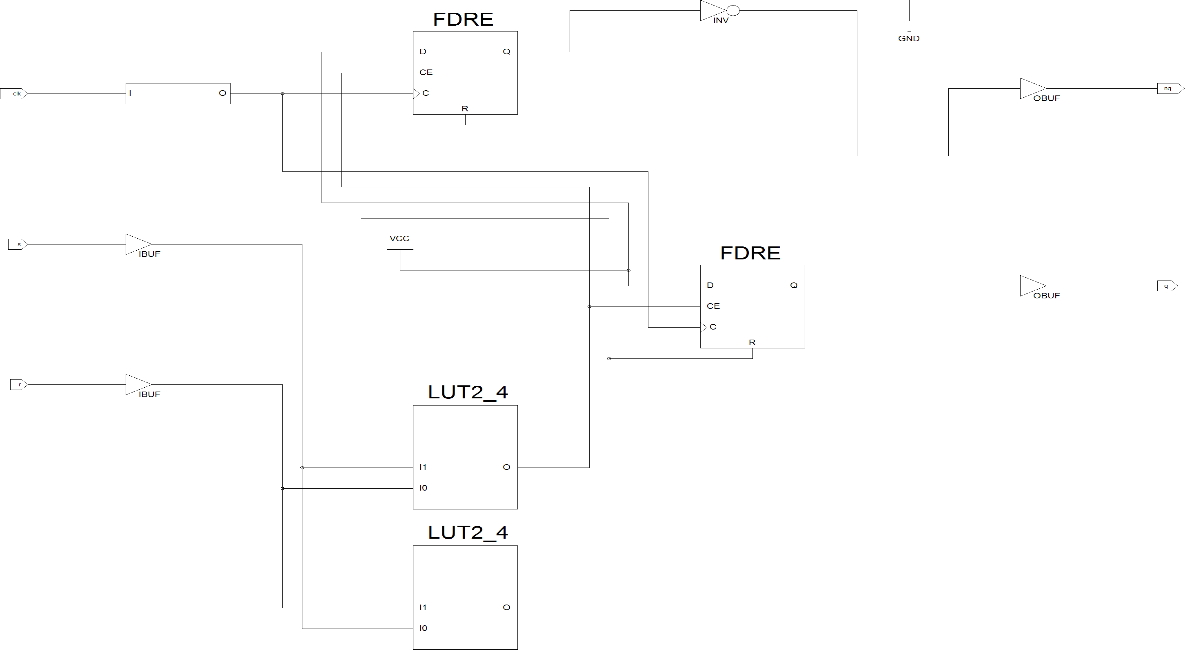
if (r='1')and(s='0') then q <= '0';

elsif (r='0')and(s='1')then q <='1';

end if; end if; end process;

notq <= not q;

end Behavioral;



**FLIP FLOP D**

entity ff is port (

d, clk: in std\_logic;

q: out std\_logic

);

end ff;

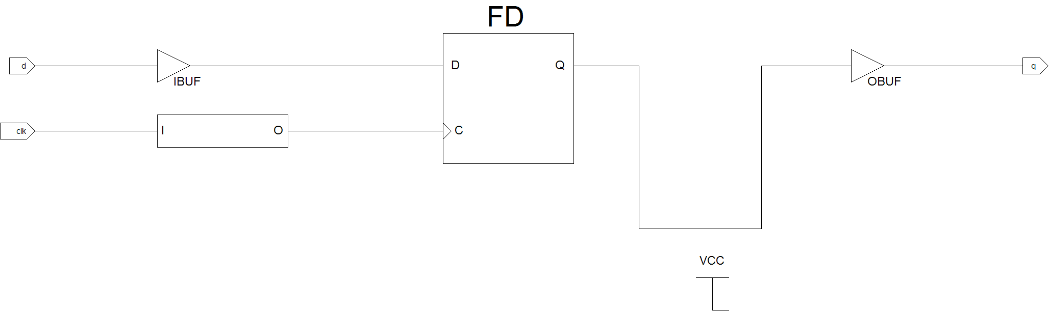
architecture Behavioral of ff is

begin

process (clk)

begin

if (clk'event and clk='1') then q <= d;

end if;

end process;

end Behavioral;

**Bibliografía**

* Cáp. 3 selección de una tarjeta de desarrollo para actividades practicas con
* <http://es.wikipedia.org/wiki/>
* VHDL lenguaje para descripción y modelado de circuitos, Fernando Pardo Carpio, Ingeniería informática, Universidad de Valencia.
* Presentación Fpga, Gabriel Sánchez Suárez, Microelectrónica, Universidad Francisco de Paula Santander.
* http://r-luis.xbot.es/edigital/ed14.html.