

**7 noviembre**

Practica: 4 diferentes Flip-Flop’s

Diseño Sistemas Digitales

**2018**

**Prof. Galicia**

Equipo:

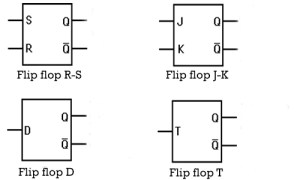
* Eduardo Alberto Pereda Guzmán
* Luis Angel Ornelas Garcia
* Christian Romero Ortega

# Objetivo general

Al terminar de la practica, los integrantes del equipo contaran con la habilidad de programar y manipular los diferentes tipos de Flip Flop’s vistos en clase. El objetivo es comprobar las tablas de verdad con el funcionamiento de los Flip-Flop’s, además de familiarizarse con la programación vhd en fpga’s.

# Introducción Teórica

El "Flip-flop" es el nombre común que se le da a los dispositivos de dos estados, que sirven como memoria básica para las operaciones de [lógica secuencial](http://hyperphysics.phy-astr.gsu.edu/hbasees/Electronic/seqoper.html#c1). Los Flip-flops son ampliamente usados para el almacenamiento y transferencia de datos digitales y se usan normalmente en unidades llamadas "registros", para el almacenamiento de datos numéricos binarios.

[](https://i2.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/flip-flop.jpg)

**Son dispositivos con memoria más comúnmente utilizados. Sus características principales son:**

* Asumen solamente uno de dos posibles estados de salida.
* Tienen un par de salidas que son complemento una de la otra.
* Tienen una o más entradas que pueden causar que el estado del Flip-Flop cambie.

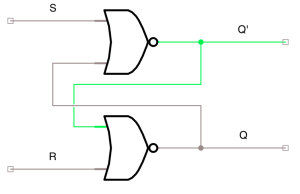
## **Los flip flops se pueden clasificar en dos:**

**Asíncronos:** Sólo tienen entradas de control. El más empleado es el flip flop S-R.  
**Síncronos:** Además de las entradas de control necesita un entrada sincronismo o de reloj.

Una vez teniendo una idea de lo que es un flip flop vamos a describir los flip flop más usados

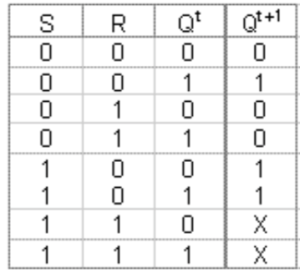
### Flip-Flop R-S (Set-Reset)

Utiliza dos compuertas NOR. S y R son las entradas, mientras que Q y Q’ son las salidas (Q es generalmente la salida que se busca manipular.)  
La conexión cruzada de la salida de cada compuerta a la entrada de la otra construye el lazo de reglamentación imprescindible en todo dispositivo de memoria.

[](https://i2.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/Flip-flopS-R.png)

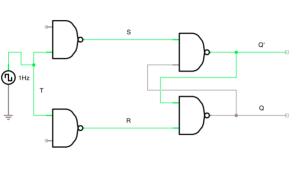
**Para saber el funcionamiento de un Flip flop se utilizan las Tablas de verdad.**

Si no se activa ninguna de las entradas, el flip flop permanece en el último estado en el cual se encontraba.

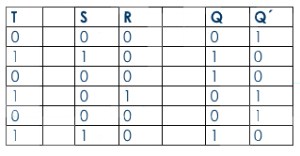
[](https://i2.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/Tabla-de-verdad-rs-.png)

### Flip-Flop T

El Flip-flop T cambia de estado en cada pulso de T. El pulso es un ciclo completo de cero a 1. Con el flip flop T podemos complementar una entrada de reloj al flip flop S-R.

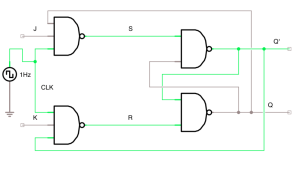
[](https://i0.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/S-R-T.png)

La siguiente tabla muestra el comportamiento del FF T y del FF S-R en cada pulso de t.

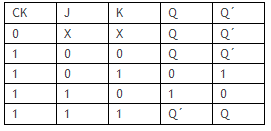
[](https://i1.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/R-S-Tabla.jpg)

### Flip-Flop J-K (Jump-Keep)

**El flip-flop J-K es una mezcla entre el flip-flop S-R y el flip-flop T.**  
A diferencia del flip flop S-R, en el caso de activarse ambas entradas a la vez, la salida adquiere el estado contrario al que tenía.

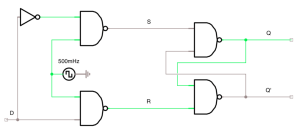
[](https://i1.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/flip-flop-JK.png)

La siguiente tabla muestra el comportamiento del flip flop JK

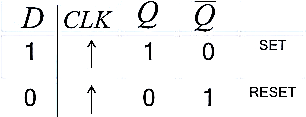
[](https://i0.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/Tabla-JK.png)

### Flip-Flop D (Delay)

El flip-flop D es uno de los FF más sencillos. Su función es dejar pasar lo que entra por D, a la salida Q, después de un pulso del reloj.

[](https://i1.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/flp-flop-d.png)

La siguiente tabla muestra el comportamiento del flip flop D



### ¿Para qué sirven las entradas Clear y Preset?

Cuando se están utilizando flip-flops en la construcción de circuitos, es necesario poder controlar el momento en el que un FF empieza a funcionar y el valor con el que inicia su secuencia. Para esto, los flip-flops cuentan con dos entradas que le permiten al diseñador seleccionar los valores iniciales del FF y el momento en el que empieza a funcionar.

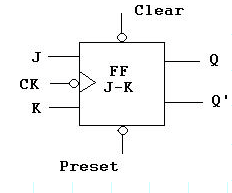
**Estas entradas son llamadas en inglés: Clear y Preset.**

* Clear – inicializa Q en cero sin importar entradas o reloj
* Preset – inicializa Q en 1 sin importar entradas o reloj

**Para ambas entradas, si reciben el valor de:**

* 0: inicializan el FF en el valor correspondiente.
* 1: el flip-flop opera normalmente

La siguiente figura muestra un FF J-K con entradas de inicialización. Note que tanto la entrada Clear, como la entrada Preset, tienen un círculo. Esto significa que la entrada funciona con un 0.

[](https://i1.wp.com/www.ingmecafenix.com/wp-content/uploads/2017/04/flip-flop-cp.png)

# Materiales empleados

* + 1 tarjeta de Desarrollo COOLRUNNER II
  + 1 tablilla de Prueba (Protoboard)
  + Cables “Jumpers” macho-macho

# Equipo empleado

* + - 1 computadora con el software “ISE” y complementos
    - Fuente de Alimentación de 5 Volts

# Circuito

# 

# Código VHD

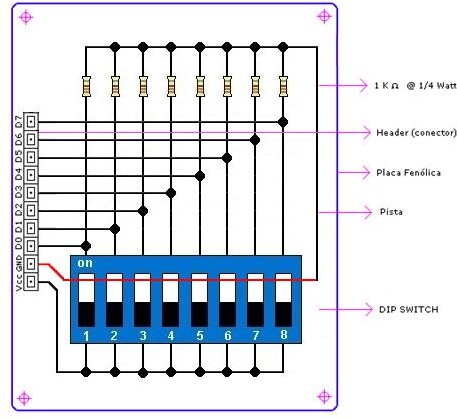
|  |
| --- |
| 1. **library IEEE;** 2. **use IEEE.STD\_LOGIC\_1164.ALL;** 3. **use IEEE.STD\_LOGIC\_ARITH.ALL;** 4. **entity F is** 5. **Port ( clk : in STD\_LOGIC;** 6. **clk\_div: inout STD\_LOGIC := '0';** 7. **cls : in STD\_LOGIC;** 8. **in1 : in STD\_LOGIC;** 9. **in2 : in STD\_LOGIC;** 10. **sel : in STD\_LOGIC\_VECTOR (1 downto 0);** 11. **qsalida : out STD\_LOGIC;** 12. **qsalidan : out STD\_LOGIC);** 13. **end F;** 14. **architecture Behavioral of F is** 15. **constant contador\_max: integer := 5000000;** 16. **signal contador: integer range 0 to contador\_max;** 17. **signal td, tt, tsr, tjk, tq: std\_logic;** 18. **begin** 19. **-------------------------------------------------------------------------------CLK** 20. **clk\_gen: process(clk) begin** 21. **if(rising\_edge(clk))then** 22. **if(contador<contador\_max) then** 23. **contador<=contador+1;** 24. **else** 25. **contador<=0;** 26. **clk\_div<=not clk\_div;** 27. **end if;** 28. **end if;** 29. **end process clk\_gen;** 30. **-------------------------------------------------------------------------------FLIPFLOP D** 31. **ftd: process (clk\_div, cls) begin** 32. **if(cls = '0') then** 33. **td <='0';** 34. **elsif (rising\_edge(clk\_div)) then** 35. **td <= in1;** 36. **else** 37. **td <= td;** 38. **end if;** 39. **end process ftd;** 40. **-------------------------------------------------------------------------------FLIPFLOP T** 41. **ftt: process (clk\_div, cls) begin** 42. **if(cls = '0') then** 43. **tt <='0';** 44. **elsif (rising\_edge(clk\_div)) then** 45. **if(in1 = '1') then** 46. **tt <= not tt;** 47. **else** 48. **tt <= tt;** 49. **end if;** 50. **end if;** 51. **end process ftt;** 52. **-------------------------------------------------------------------------------FLIPFLOP SR** 53. **ftsr: process (clk\_div, cls) begin** 54. **if(cls = '0') then** 55. **tsr <='0';** 56. **elsif(rising\_edge(clk\_div)) then** 57. **if(in1='1' and in2='1') then** 58. **tsr<=tsr;** 59. **elsif(in1 = '1' and in2='0') then----set** 60. **tsr<='1';** 61. **elsif (in2 = '1' and in1='0') then--reset** 62. **tsr<='0';** 63. **end if;** 64. **end if;** 65. **end process ftsr;** 66. **-------------------------------------------------------------------------------FLIPFLOP JK** 67. **ftjk: process (clk\_div, cls) begin** 68. **if(cls = '0') then** 69. **tjk <= '0';** 70. **elsif(rising\_edge(clk\_div)) then** 71. **if (in1/=in2) then** 72. **tjk <= in1;** 73. **elsif(in1='1' and in2='1') then** 74. **tjk <= not tjk;** 75. **else** 76. **tjk <= tjk;** 77. **end if;** 78. **end if;** 79. **end process ftjk;** 80. **-------------------------------------------------------------------------------SELECTOR** 82. **tq <= td when sel = "00"** 83. **else tt when sel = "01"** 84. **else tsr when sel = "10"** 85. **else tjk when sel = "11";** 87. **qsalida <= not tq when (sel = "10" and (in1 = '1' and in2 = '1')) else tq;** 88. **qsalidan <= not tq;** 89. **end Behavioral;** |

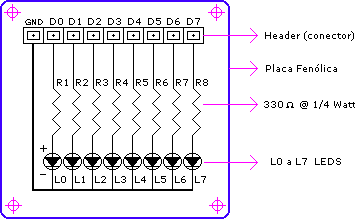
# Conclusiones Individuales.

**Eduardo Pereda**: Gracias al entendimiento de los flip flop’s de manera teórica, se pudo determinar el código a programar en la tarjeta COOLRUNNER II. El flip flop más versátil es el J-K, ya que se comporta como un flip flop tipo d y como un flip flop Toggle.

**Luis angel:** Esta práctica me ayudo bastante a entender de forma más fácil y sencilla el comportamiento de los flip-flop, además de que aprendí un poco de programación en VHDL.

1. **Referencias.**
   1. Sanchez-Elez, Marcos (2017): *INTRODUCCION A LA PROGRAMACION EN VHDL.* Universidad Complutense de Madrid: facultad de informática.
   2. G. Maxinez, David (2002): VHDL *El arte de programar sistemas digitales.* Campus Estado de México: Instituto Tecnológico y de Estudios Superiores de Monterrey.
2. **ANEXOS.**





**LIGAS DE INTERÉS:**

**Hojas de especificaciones de las compuertas lógicas** [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls00rev5.pdf](http://maven.smith.edu/%7Ethiebaut/270/datasheets/sn74ls00rev5.pdf) [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls02rev5.pdf](http://maven.smith.edu/%7Ethiebaut/270/datasheets/sn74ls02rev5.pdf) [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls04rev5.pdf](http://maven.smith.edu/%7Ethiebaut/270/datasheets/sn74ls04rev5.pdf) [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls08rev5.pdf](http://maven.smith.edu/%7Ethiebaut/270/datasheets/sn74ls08rev5.pdf) [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls32rev5.pdf](http://maven.smith.edu/%7Ethiebaut/270/datasheets/sn74ls32rev5.pdf) [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls86rev5.pdf](http://maven.smith.edu/%7Ethiebaut/270/datasheets/sn74ls86rev5.pdf)

**Manual Completo de compuertas digitales de Motorola “FAZ AND LS TTL”** [https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxmYXNpcG58Z3g6Mzg0OTUyYzkyOD](https://docs.google.com/viewer?a=v&amp;pid=sites&amp;srcid=ZGVmYXVsdGRvbWFpbnxmYXNpcG58Z3g6Mzg0OTUyYzkyODU2NjBmMg) [U2NjBmMg](https://docs.google.com/viewer?a=v&amp;pid=sites&amp;srcid=ZGVmYXVsdGRvbWFpbnxmYXNpcG58Z3g6Mzg0OTUyYzkyODU2NjBmMg)

