INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Ingeniería en Sistemas Computacionales

Diseño de Sistemas Digitales

Practica 10. BCDJK

Presenta:

Álvarez Hernández Gabriel Alexander
García Quiroz Gustavo Iván
Huesca Laureano Josué Alejandro
Muñoz Valdivia Irving Omar
Pedroza Villagómez Emir

FECHA DE ENTREGA: 27/06/24

Índice

Introducción 3

Desarrollo 5

Implementación en el Código 5

RTL Viewer 8

Simulación 9

Conclusiones 10

Introducción

Un BCDJK es un contador decimal codificado en binario (BCD, por sus siglas en inglés: Binary-Coded Decimal) implementado utilizando flip-flops JK. En esta práctica vamos a desglosar cómo se relaciona con el código desarrollado. Un contador BCD cuenta de 0 a 9 en formato binario. Cada número decimal se representa con un grupo de 4 bits binarios. Por ejemplo:

- 0 en decimal es 0000 en BCD.
- 1 en decimal es 0001 en BCD.
- 2 en decimal es 0010 en BCD.
- 3 en decimal es 0011 en BCD.
- 4 en decimal es 0100 en BCD.
- 5 en decimal es 0101 en BCD.
- 6 en decimal es 0110 en BCD.
- 7 en decimal es 0111 en BCD.
- 8 en decimal es 1000 en BCD.
- 9 en decimal es 1001 en BCD.

Un flip-flop JK es un tipo de flip-flop (un elemento de almacenamiento de un bit de datos) que tiene dos entradas (J y K) y puede cambiar de estado en función de estas entradas y del reloj. La tabla de verdad para un flip-flop JK es la siguiente:

- J = 0, K = 0: No cambia el estado.
- J = 1, K = 0: La salida se pone en 1.
- J = 0, K = 1: La salida se pone en 0.
- J = 1, K = 1: La salida cambia (toggle).

Desarrollo

Implementación en el Código

En el código, primero se define un módulo BCDJK utilizando flip-flops JK para implementar un contador BCD.

1.- Módulo BCDJK:

a) Entradas: clk_i, rst_i, j_i, k_i

b) Salidas: q_o (salidas del contador), qn_o (salidas complementarias)

c) Componentes:

- Cuatro instancias del módulo FFJK, que representan cada uno de los cuatro bits del contador BCD.
- Un flip-flop JK ffjk_b0 cuya entrada de reloj es ~clk_i.
- Los flip-flops ffjk_b1, ffjk_b2 y ffjk_b3 toman como entrada de reloj las salidas negadas de los flip-flops anteriores (~q_o[0], ~q_o[1], etc.).
- Una compuerta AND (and_u0) que genera una señal intermedia jb3_w usada como entrada J del flip-flop más significativo ffjk_b3.

2.- Módulo FFJK:

a) Entradas: clk_i, rst_i, j_i, k_i

b) Salidas: q_o (salida del flip-flop), qn_o (salida complementaria)

c) Comportamiento: Define el comportamiento del flip-flop JK utilizando un bloque always sensible a los flancos positivos del reloj (clk_i) y del reinicio (rst_i). Según las entradas j_i y k_i, la salida q_o se mantiene, se pone a 1, se pone a 0, o cambia de estado.

3.- Banco de Pruebas BCDJK_tb:

- Simula el funcionamiento del módulo BCDJK.
- Inicializa las señales y genera un reloj alternante (clk_i).
- Observa cómo el contador responde a los cambios de reloj y reinicio.

Básicamente, el módulo BCDJK es un contador BCD que utiliza flip-flops JK. Cuenta de 0 a 9 en formato binario utilizando 4 flip-flops JK conectados en cascada, donde cada flip-flop representa un bit del número binario. Este tipo de contador es útil en aplicaciones digitales donde es necesario contar en decimal y convertirlo a binario para procesamiento digital.

```
 3    Archivo: BCDJK.v
 4    Descripción:Se define un módulo en Verilog que implementa un contador decimal
 5    codificado en binario (BCD) utilizando flip-flops JK. Este contador cuenta de
 6    0 a 9 en formato binario, utilizando 4 bits para representar cada dígito decimal.
 7    Asignatura: DSD
 8    Profesor: Flores Escobar Jose Antonio
 9    Equipo: Alvarez Hernandez Gabriel Alexander
10    Garcia Quiroz Gustavo Ivan
11    Huesca Laureano Josue Alejandro
12    Muños Valdivia Irving Omar
13    Pedroza Villagomez Emir
14 */
15 module BCDJK(
16
17     input           clk_i,
18     input           rst_i,
19     input           j_i,
```

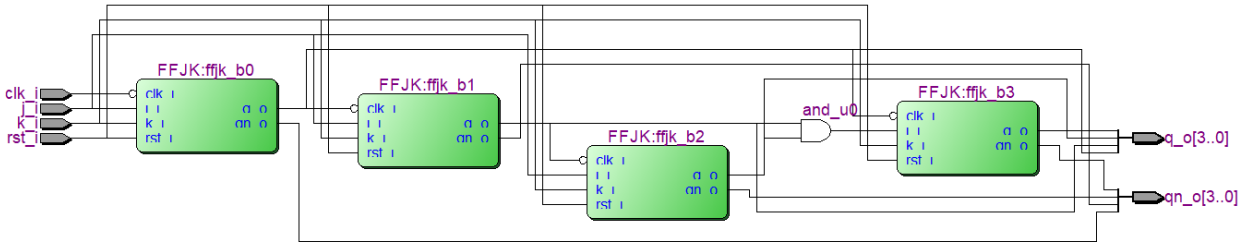
```
20     input           k_i,
21     output    [3:0] q_o,
22     output    [3:0]    qn_o
23
24     );
25     //Seccion de declaracion de señales intermedias
26     wire  jb3_w;
27
28     FFJK ffjk_b0(
29     .clk_i      (~clk_i),
30     .rst_i      (rst_i),
31     .j_i        (j_i),
32     .k_i        (k_i),
33     .q_o        (q_o[0]),
34     .qn_o       (qn_o[0])
35     );
36     FFJK ffjk_b1(
```

```
37      .clk_i      (~q_o[0]),
38      .rst_i      (rst_i),
39      .j_i        (j_i),
40      .k_i        (k_i),
41      .q_o        (q_o[1]),
42      .qn_o       (qn_o[1])
43      );
44      FFJK ffjk_b2(
45      .clk_i      (~q_o[1]),
46      .rst_i      (rst_i),
47      .j_i        (j_i),
48      .k_i        (k_i),
49      .q_o        (q_o[2]),
50      .qn_o       (qn_o[2])
51      );
52
53      //AND Gate
```

```
53      //AND Gate
54      and and_u0 (jb3_w,q_o[1], q_o[2]);
55
56      FFJK ffjk_b3(
57      .clk_i      (~q_o[0]),
58      .rst_i      (rst_i),
59      .j_i        (jb3_w),
60      .k_i        (k_i),
61      .q_o        (q_o[3]),
62      .qn_o       (qn_o[3])
63      );
64  endmodule
```

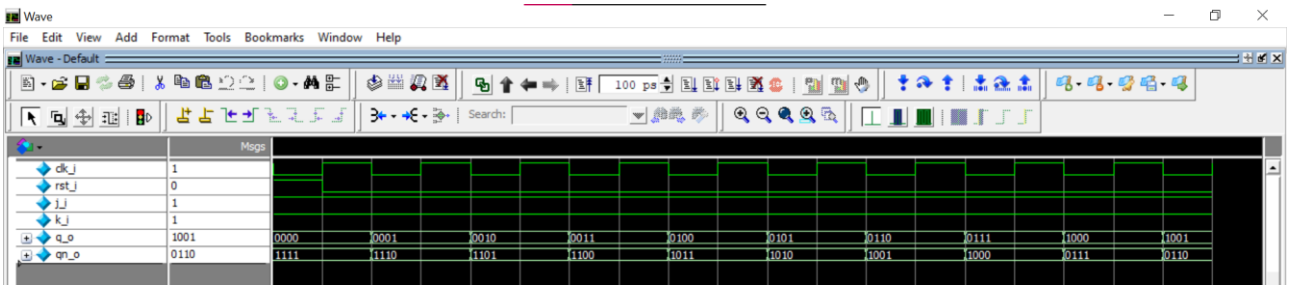
RTL Viewer

El RTL muestra cómo los flip-flops JK están conectados en cascada, cómo se genera la señal jb3_w a través de una compuerta AND, y cómo se utilizan las salidas negadas de los flip-flops anteriores como señales de reloj para los siguientes flip-flops. A continuación la explicación de lo que podemos ver:



- El primer flip-flop (ffjk_b0) recibe como entrada de reloj la señal clk_i negada y las entradas j_i y k_i. Su salida q_o[0] sirve como reloj para el siguiente flip-flop (ffjk_b1).
- El segundo flip-flop (ffjk_b1) utiliza la salida negada del primer flip-flop (~q_o[0]) como su señal de reloj.
- El tercer flip-flop (ffjk_b2) toma la salida negada del segundo flip-flop (~q_o[1]) como su señal de reloj.
- La salida de q_o[1] y q_o[2] se combinan en una compuerta AND (and_u0), cuya salida (jb3_w) se utiliza como la entrada J del cuarto flip-flop (ffjk_b3).
- El cuarto flip-flop (ffjk_b3) recibe como señal de reloj la salida negada del primer flip-flop (~q_o[0]) y jb3_w como su entrada J.

Simulación



La simulación demostró que el contador BCDJK diseñado funciona correctamente, contando de 0 a 9 en formato binario cuando se le proporciona una señal de reloj continua y las entradas J y K están activadas. La señal de reinicio (`rst_i`) fue efectiva para establecer el contador en su estado inicial. La herramienta de simulación permitió verificar el comportamiento del circuito y visualizar las transiciones de las señales de manera clara, confirmando que el diseño cumple con las especificaciones requeridas.

Conclusiones

Durante la práctica, implementamos y analizamos un contador decimal codificado en binario (BCD) utilizando flip-flops JK en Verilog. Este ejercicio permitió explorar tanto el diseño como la simulación de circuitos secuenciales. Primero, escribimos el código para un flip-flop JK (FFJK). Este módulo define el comportamiento básico del flip-flop con entradas de reloj (clk_i), reinicio (rst_i), y las entradas características J (j_i) y K (k_i). Las salidas del flip-flop, q_o y qn_o, reflejan el estado del flip-flop y su complemento, respectivamente. Luego observamos cómo, dependiendo de las entradas J y K, el flip-flop puede mantener su estado, establecerse en 1, restablecerse en 0 o alternar su estado.

Posteriormente, se diseñamos el módulo BCDJK utilizando instancias del módulo FFJK. Este módulo, que cuenta de 0 a 9 en formato binario, emplea cuatro flip-flops JK conectados en cascada. Nosotros configuramos las conexiones adecuadas, donde las salidas negadas de cada flip-flop sirvieron como señales de reloj para el siguiente flip-flop en la cadena. Además, se implementó una compuerta AND para generar una señal intermedia (jb3_w) que se usó como entrada J del flip-flop más significativo.

Para verificar el correcto funcionamiento del diseño, se utilizó un banco de pruebas (BCDJK_tb). En este banco de pruebas, se inicializaron las señales de entrada y se generó una señal de reloj alternante. Se observó cómo el módulo BCDJK respondió a los cambios de reloj y reinicio, permitiendo verificar el comportamiento esperado del contador.

Finalmente, se utilizó el RTL Viewer para visualizar el diseño a nivel de registros y lógica combinacional. Esta herramienta proporcionó una representación gráfica de las interconexiones entre los flip-flops y la compuerta AND, lo que permitió pudiéramos comprender mejor el flujo de datos y la estructura del contador BCDJK.

La práctica culminó con éxito, proporcionándonos una comprensión profunda del diseño, simulación y verificación de circuitos secuenciales en Verilog, y cómo estas técnicas se aplican en la implementación de contadores binarios codificados decimal.

Bibliografía

- https://es.fmuser.net/content/?20973.html
- https://en.wikipedia.org/wiki/Flip-flop_(electronics)
- https://espanol.libretexts.org/Vocacional/Tecnolog%C3%ADa_Electr%C3%B3nica/Libro%3A_Circuitos_El%C3%A9ctricos_IV_-_Circuitos_Digitales_(Kuphaldt)/10%3A_Multivibradores/10.06%3A_El_flip-flop_J-K