UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

LABORATORIO DE ORGANIZACIÓN Y ARQUITECTURA DE COMPUTADORAS

CLAVE 6867

Profesor: M. I. José Luis Cruz Mora

Integrantes del equipo:

García Fernández Jesús Alejandro

Hernández Arrieta Carlos Alberto

Moreno Guerra Marco Antonio

Reporte Práctica 2

Fecha de entrega: 04 de septiembre del 2019

Grupo: 1

Semestre: 2020-1

DISEÑO DE MÁQUINAS DE ESTADO

OBJETIVO

Familiarizar al alumno en el conocimiento de los algoritmos de las máquinas de estados utilizando Quartus y el lenguaje VHDL.

DESARROLLO

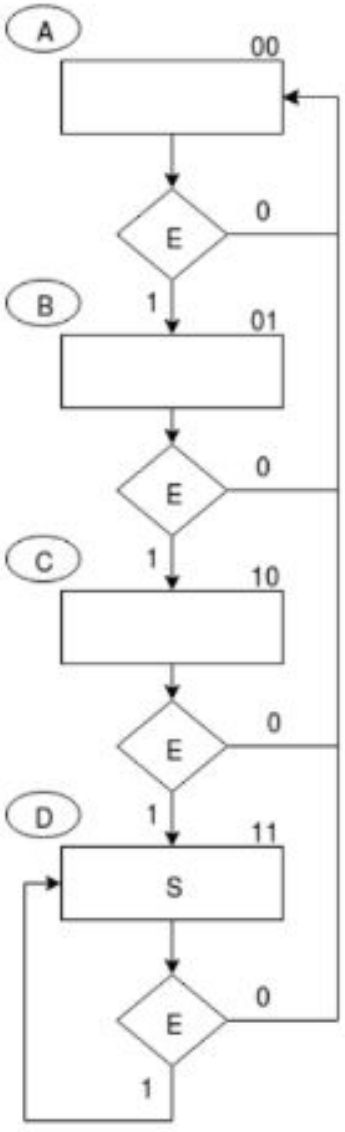


Figura 1. Diagrama de la carta ASM del problema a resolver

Para la obtención del circuito secuencial se generó primero la tabla de verdad, misma que se muestra a continuación:

ESTADO PRESENTE		ENTRADA	ESTADO SIGUIENTE		SALIDA
Q1	**Q0**	**E**	**D1**	**D0**	**S**
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Diseño de la máquina de estados con Diagramas de Bloques

Posteriormente, se obtuvieron las ecuaciones de cada uno de los bits de Estado Siguiente y la Salida.

E\Q1Q0	00	01	11	10
0				
1		1	1	1

$$D_1 = (Q_1 E) + (\neg\, Q_1)(Q_0 E)$$

E\Q1Q0	00	01	11	10
0				
1	1		1	1

$$D_0 = (Q_1 E) + (\neg\, Q_1)(\neg\, Q_0) E$$

E\Q1Q0	00	01	11	10
0			1	
1			1	

$$S = Q_1 Q_0$$

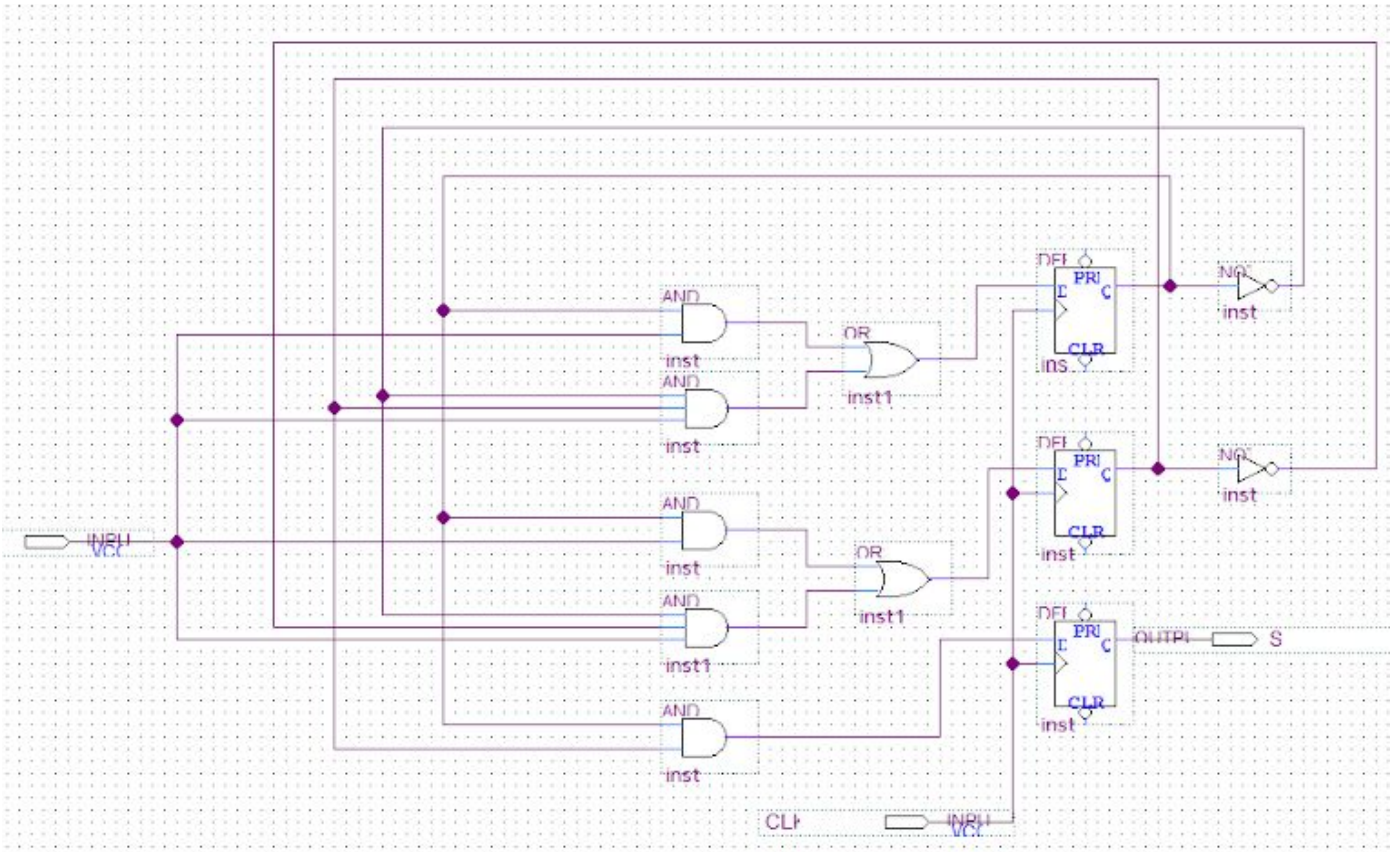


Figura 2. Diagrama de bloques del sistema

Para validar que el funcionamiento es el deseado, se procedió a realizar la simulación del diagrama de bloques de la Figura 2. Dicho resultado se muestra en la siguiente Figura:

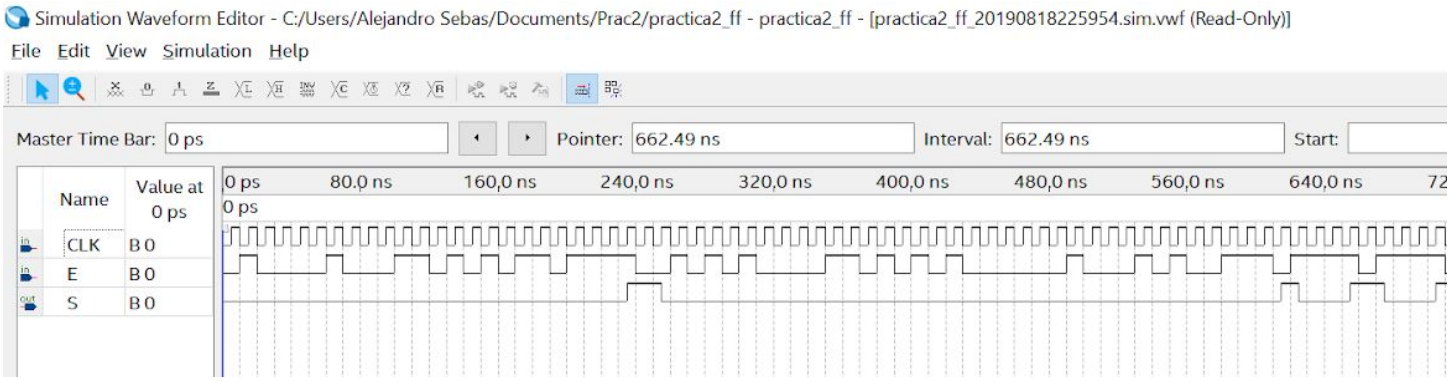


Figura 3. Resultados de la simulación del diagrama de bloques de *practica2_ff*

En la figura anterior de la simulación se aprecia que el comportamiento es el deseado, es decir, que dada una secuencia de al menos tres bits en alto (en "1") se genera una señal de salida en alto. Sin embargo, es importante mencionar que la señal de salida se retrasa en un ciclo de reloj, esto debido a que se cuenta con un Flip Flop en la señal de salida y, al estar diseñada la carta ASM en bajo *Moore*, la salida está presente en función del estado actual; por ello, la salida se atrasa en un ciclo de reloj, en lo que se avanza al Estado D, que contiene la salida igual a "1".

Diseño de la máquina de estados con VHDL

Por otra parte, se creó un nuevo proyecto llamado ***practica2_vhdl***, que consistió en resolver el mismo problema planteado con la carta ASM que detecta la secuencia "111" pero con un enfoque diferente.

Se realizó a través de código VHDL, en el que se define la entidad y su arquitectura. Dentro de esta última se establece el comportamiento del sistema, que es el variar de un estado a otro en función de las entradas y del estado presente. Esto se detalla en el siguiente código que se realizó para solucionar este problema:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity practica2_vhdl is
	Port( RELOJ : in STD_LOGIC;
	      E  : in STD_LOGIC;
```

```
                        S  : out STD_LOGIC;
                        nextState : out STD_LOGIC_VECTOR(1 downto 0);
                        currentState : out STD_LOGIC_VECTOR(1 downto 0));
end practica2_vhdl;
architecture Behavioral of practica2_vhdl is
        signal esiguiente : std_logic_vector (1 downto 0) := B"00";
        constant A : std_logic_vector(1 downto 0) := B"00";
        constant B : std_logic_vector(1 downto 0) := B"01";
        constant C : std_logic_vector(1 downto 0) := B"10";
        constant D : std_logic_vector(1 downto 0) := B"11";
        begin
        process (E, RELOJ)
                begin
                        if rising_edge (RELOJ) then
                                case esiguiente is
                                        when A =>
                                                S <= '0';
                                                currentState <= A;
                                                if E = '0' then
                                                        esiguiente <= A;
                                                        nextState <= A;
                                                elsif E = '1' then
                                                        esiguiente <= B;
                                                        nextState <= B;
                                                end if;
                                        when B =>
                                                S <= '0';
                                                currentState <= B;
                                                if E = '1' then
                                                        esiguiente <= C;
                                                        nextState <= C;
                                                elsif E = '0' then
                                                        esiguiente <= A;
```

```
                                        nextState <= A;
                                end if;
                        when C =>
                                S <= '0';
                                currentState <= C;
                                if E = '0' then
                                        esiguiente <= A;
                                        nextState <= A;
                                elsif E = '1' then
                                        esiguiente <= D;
                                        nextState <= D;
                                end if;
                        when D =>
                                S <= '1';
                                currentState <= D;
                                if E = '0' then
                                        esiguiente <= A;
                                        nextState <= A;
                                elsif E = '1' then
                                        esiguiente <= D;
                                        nextState <= D;
                                end if;
                        when others => null;
                end case;
        end if;
    end process;
end Behavioral;
```

En cada uno de los estados se asignan las señales a los puertos de salida, estado presente y estado siguiente. Además, se procesa el siguiente estado dependiendo de la entrada.

Para poder visualizar el funcionamiento de este código se hizo la simulación y funcionó de manera adecuada e idéntica al realizado en el proyecto ***practica2_ff***.

Ahora se decidió probar el código grabandolo en la FPGA DE10-Lite de Intel, pero antes fue necesario agregar un nuevo módulo que funciona como un clock manual para detectar cada una de las señales de Entrada en la máquina de estados, ya que de no ser así y usarse el reloj de la FPGA (50MHz) o un divisor de frecuencia, haría el proceso de detección de Entrada de la máquina de manera automática.

Para implementar este módulo que indica cuándo se debe tomar una nueva entrada a la máquina de estados, se tomó el código VHDL propuesto en el manual de la práctica 2, que lleva por nombre *sensa_boton.vhdl*

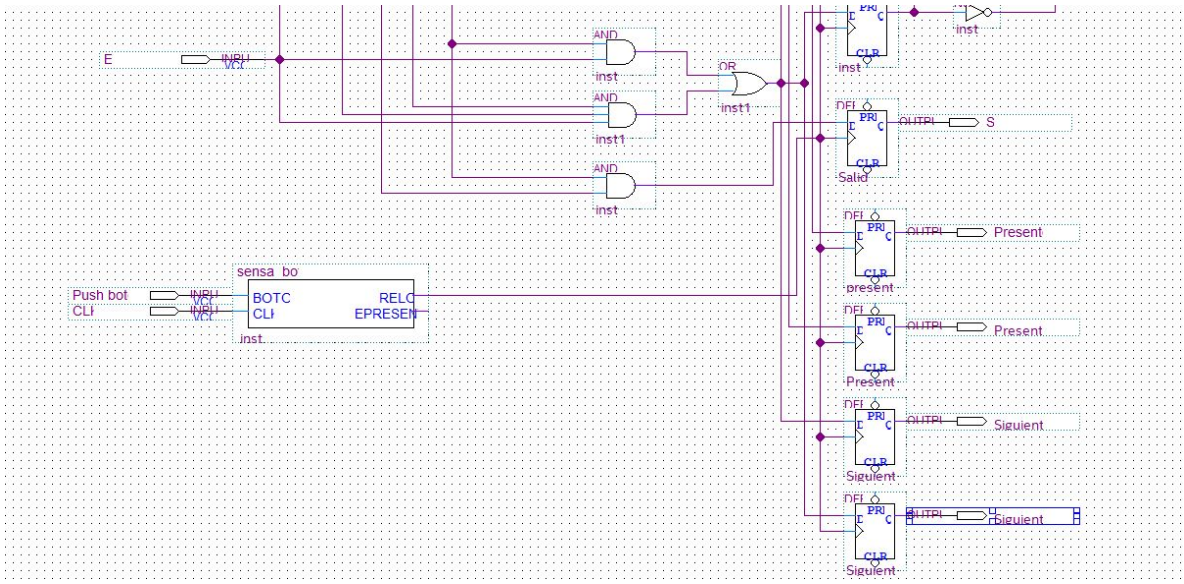
El siguiente código, como anteriormente se mencionó, realiza la función de un reloj manual, esto se desarrolló con el objetivo de poder visualizar en los leds de las tarjetas el número de estado presente y el siguiente dependiendo de el valor de la entrada.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity sensa_boton is
	Port (BOTON : in STD_LOGIC;
		CLK : in STD_LOGIC;
		RELOJ : out STD_LOGIC;
		EPRESENTE: buffer STD_LOGIC);
end sensa_boton;


architecture Behavioral of sensa_boton is
	signal ESIGUIENTE: STD_LOGIC;


		begin


		process (ESIGUIENTE, BOTON)
```

```
                        begin


                        if rising_edge (CLK) then
                                case ESIGUIENTE is
                                        when '0' =>
                                                RELOJ <= '0';
                                                if BOTON ='0' then
                                                        ESIGUIENTE <= '0';
                                                else
                                                        ESIGUIENTE <= '1';
                                                end if;
                                        when '1' =>
                                                if BOTON ='1' then
                                                        ESIGUIENTE <= '1';
                                                        RELOJ <= '0';
                                                else
                                                        ESIGUIENTE <= '0';
                                                        RELOJ <= '1';
                                                end if;
                                        when others => null;
                                end case;
                        end if;
                        EPRESENTE <= ESIGUIENTE;
end process;
end Behavioral;
```

Después de lograr general el reloj manual o específicamente generar un flanco de bajada se necesita tomar los valores correctos para visualizar el estado presente y el estado siguiente, para esto, en el diagrama obtenido de la reducción de la carta ASM, se tomaron los valores de Q1 Y Q0 como salidas para poder representar los estados presentes y tomamos los valores de D1 y D0 como salidas para poder representar los estados siguientes.

Como se puede observar, optamos por colocar un flip flop tipo d antes de tomar cada salida que representará cada estado, esto para agregar estabilidad, es decir, que aunque la entrada E esté variando sólo procesará tal valor hasta que el botón sea presionado, ya que ahí ocurrirá un flanco de bajada, por el contrario, sin utilizar los flip flops al variar la entrada E los estados siguientes variarán sin esperar la señal del reloj.



CONCLUSIONES

García, Fernández Jesús Alejandro: En esta práctica seguí recordando y conociendo nuevas funciones de Quartus y además comencé de nuevo a adentrarme en el lenguaje VHDL, el principal concepto que aprendí es el funcionamiento como estabilizador de los flip flops D antes de las salidas, ya que en clase de teoría no me quedaba completamente claro a qué se referían con el "Retraso", con la práctica pude notar que sin los ff antes de las salidas se procesan los estados en el circuito sin importar las señales que mande el reloj y eso podría ocasionar eventos inesperados.

Hernández, Carlos: En esta práctica hicimos uso de Cartas ASM que nos permiten tener una secuencia de pasos de como nuestra máquina de estados cambia de un estado a otro y cuáles con las salidas que provoca ese cambio estado, conseguimos obtener su respectivo circuito para representar el funcionamiento de la máquina de estados. A su vez además hicimos el mismo funcionamiento de la máquina de estados pero usando VHDL por lo que fue más fácil la implementación ya que solo consiste en cambios de estados en el código. Usamos nuevamente lo aprendido en la práctica anterior para obtener el funcionamiento deseado en la Tarjeta.

Moreno, Marco: Con la realización de esta práctica, se retomó todo lo visto en la práctica 1, pues fue necesario recordar el manejo del Software Quartus para la generación de un nuevo proyecto, así como la simulación. Para fines de esta práctica únicamente se realizó la simulación interna con un módulo *University Program VWF*. Aunado a lo anterior, también se realizó la implementación del funcionamiento de una máquina de estados en Quartus a través del lenguaje VHDL, indicando para cada estado presente (codificado en bits) cual es su salida y en función de las entradas, para el modelo de Moore, cuál es el siguiente estado.

También con la implementación del módulo *sensa_boton* descubrí la manera de implementar una clock manual, es decir, un módulo que determine cuándo debe sensar la siguiente entrada la máquina de estados; lo cual es muy útil para poder observar el funcionamiento general del sistema.

Finalmente, al presentar la práctica y que el profesor nos solicitara agregar el módulo de *sensa_boton* a la primera parte de la práctica (practica2_ff) y desplegar las salidas para visualizarlas al grabarlas en la FPGA, pude ver la gran importancia que tiene el implementar Flip Flops en cada uno de los bits de salida del sistema, pues éstos brindan estabilidad a las señales de salida;

es decir, que conservan el valor de las salidas hasta que ocurra otro flanco de subida en el CLOCK a pesar de que ocurra un cambio en las entradas aunque no haya ocurrido un nuevo ciclo en el CLOCK.

REFERENCIAS

"Laboratorio de Organización y Arquitectura de Computadoras Práctica No. 2. Diseño de máquinas de estado"