

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en Computación

Laboratorio de Organización y Arquitectura de Computadoras

Práctica 02: Diseño de Máquinas de Estado

Alumnos:

- Murrieta Villegas Alfonso
- Reza Chavarria Sergio Gabriel
- Valdespino Mendieta Joaquin

Profesora: Ayesha Sagrario Román García

Grupo: 7

Fecha de entrega: 24 de septiembre de 2021

Práctica 02

Objetivo

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

Introducción

Los algoritmos de máquinas de estado, son representaciones gráficas que describen el comportamiento en el tiempo de los circuitos secuenciales.

Una carta ASM es utilizado para el diseño y el análisis para las unidades de control de proyectos. Las cartas son utilizadas para la representación de algoritmos de manera secuencial.

El flip flop o un sistema básico biestable en un circuito básico utilizado para el almacenamiento de memoria, uso de registros y almacenadores de bits, esto a partir de una trayectoria de retroalimentación. Este puede ser construido con compuertas NOR o NAND.

El flip flop tipo D tiene una entrada D, una entrada de reloj y 2 salidas Q y \bar{Q} . Su comportamiento es similar al del latch D, la salida del Flip Flop se iguala a la entrada en el instante en el que se produzca el flanco (ascendente o descendente dependiendo del tipo de flip-flop) de la señal del reloj.



Ilustración 1 Flip Flop tipo D con tabla de verdad

El lenguaje VHDL se utiliza en la descripción de circuitos digitales y para la automatización de diseño electrónico. El lenguaje consta de una estructura en la programación. Esta estructura se divide en 2 partes.

- Entidades: Declaración de los puertos de entrada y salida que serán utilizados.
- Arquitectura: Las acciones o los procesos que utilizaran los puertos y las variables.

Desarrollo

Dada la carta ASM de la figura 1, elabore lo que se indica.

1. Obtenga el circuito secuencial de la carta ASM utilizando flip flops tipo D. Cree un proyecto en Quartus llamado practica2_ff, implemente su diseño en el ambiente de desarrollo Quartus y simúlelo para validar su comportamiento.
2. Cree un nuevo proyecto en Quartus llamado practica2_vhdl y diseñe la máquina de estados de la carta ASM utilizando el lenguaje de descripción de hardware VHDL. Simule su diseño para validar que funciona correctamente.

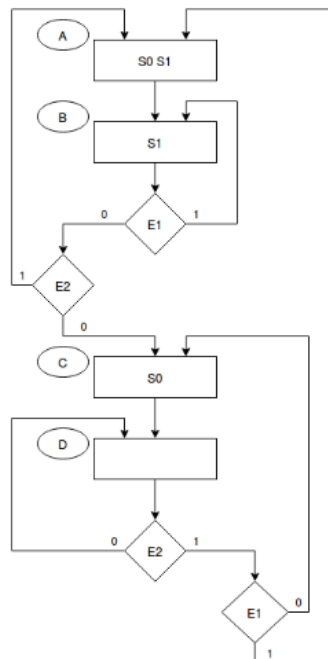


Diagrama 1: Carta ASM a desarrollar

Resultados

Ejercicio 1

Para obtener el circuito secuencial de la carta ASM se parte por hacer su tabla de verdad considerando los flip flops tipo D, quedando de la siguiente manera.

Considerando el comportamiento FFD (manejado por bits individuales)

Q (estado actual)	Q+1 (estado siguiente)	FFD
0	0	0
0	1	1
1	0	0
1	1	1

La carta ASM cuenta con 4 estados por lo que usaremos 2 bits para representarlo, además contamos con 2 salidas, aparte 2 entradas para realizar decisiones, por lo que la tabla de verdad quedaría de la siguiente forma.

Combinaciones Q1 Y Q0

00-A, 01-B, 10-C, 11-D

Tabla de verdad y Mapas de Karnaugh

Q1	Q0	E2	E1	Q1	Q0	S1 (D)	S0 (D)	D1	D0
0	0	*	*	0	1	1	0	0	1
0	1	0	0	1	0	0	1	1	0
0	1	*	1	0	1	1	0	0	1
0	1	1	0	0	0	1	1	0	0
1	0	*	*	1	1	0	0	1	1
1	1	0	*	1	1	0	0	1	1
1	1	1	0	1	0	0	1	1	0
1	1	1	1	0	0	1	1	0	0

$$I = \text{not} Q1 * \text{not} Q0 + \text{not} Q1 * E1 + \text{not} Q1 * E2 + Q0 * E2 * E1$$

E2E1	00	01	11	10
Q1Q0				
00	1	1	1	1
01		1	1	1
11			1	
10				

$$J = \text{not} Q1 * Q0 * \text{not} E1 + Q1 * Q0 * E2$$

E2E1	00	01	11	10
Q1Q0				
00				
01	1			1
11			1	1
10				

$$I = Q0 * \text{not} E2 * \text{not} E1 + Q1 * \text{not} E1 + Q1 * \text{not} E2 + Q1 * \text{not} Q0$$

E2E1	00	01	11	10
Q1Q0				
00				
01	1			
11	1	1		1
10	1	1	1	1

$$D0 = \text{not } Q0 + \text{not } Q1 * E1 + Q1 * \text{not } E2$$

E2E1	00	01	11	10
Q1Q0				
00	1	1	1	1
01		1	1	
11	1	1		
10	1	1	1	1

Una vez obtenido esto procedemos a pasarlo en este caso a VHDL para crear el circuito por módulos en Quartus, quedando de la siguiente forma, usando los FFD creados (de un curso anterior DDM y VLSI)

Código

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity FFD is
5  port (D,CLK: in std_logic;
6        Q: out std_logic
7        );
8  end FFD;
9
10 architecture arqF of FFD is
11 begin
12
13     process(CLK)
14     begin
15         if CLK'event and CLK='1' then
16             Q<=D;
17         end if;
18     end process;
19
20 end architecture arqF;
21
22

```

Código 1: Flip Flop D (usando flanco positivo o Rising Edge)

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity combinacional is
5  port (E2,E1,Q1,Q0: in std_logic;
6        D1,D0,S1,S0: out std_logic
7        );
8  end combinacional;
9
10 architecture arqc of combinacional is
11 begin
12
13     D1<= (Q0 AND not E2 AND not E1) OR (Q1 AND not E1) OR (Q1 AND not E2) OR (Q1 AND not Q0);
14     D0<= (not Q0) OR (not Q1 AND E1) OR (Q1 AND not E2);
15     S1<= (not Q1 AND not Q0) OR (not Q1 AND E1) OR (not Q1 AND E2) OR (Q0 AND E2 AND E1);
16     S0<= (not Q1 AND Q0 AND not E1) OR (Q1 AND Q0 AND E2);
17
18     --S1<= (not Q1 and not Q0) or (not Q1 and Q0);
19     --S0<= (not Q1 and not Q0) or (Q1 and not Q0);
20
21 end architecture arqc;
22
23

```

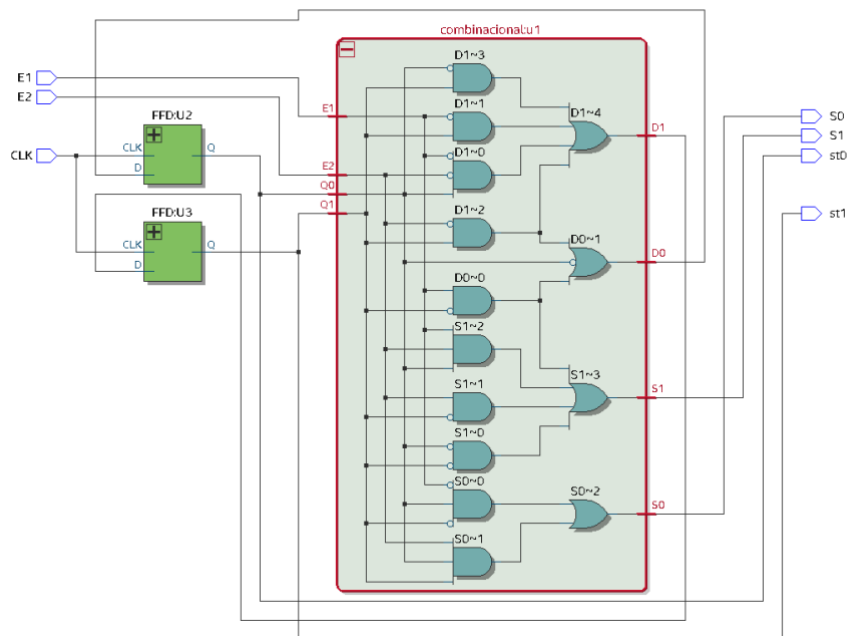
Código 2: Modulo combinacional

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity practica2_ff is
5  port (E2,E1, CLK: in std_logic;
6        S1,S0, st1,st0: out std_logic
7        );
8  end practica2_ff;
9
10 architecture arqp of practica2_ff is
11     signal D0,D1: std_logic;
12     signal Q0,Q1: std_logic;
13
14 begin
15
16
17     u1: entity work.combinacional(arqc) port map(E2,E1,Q1,Q0,D1,D0,S1,S0);
18     u2: entity work.FFD(arqF) port map(D0,CLK,Q0);
19     u3: entity work.FFD(arqF) port map(D1,CLK,Q1);
20
21
22
23     st1<=Q1;
24     st0<=Q0;
25
26 end architecture arqp;
27

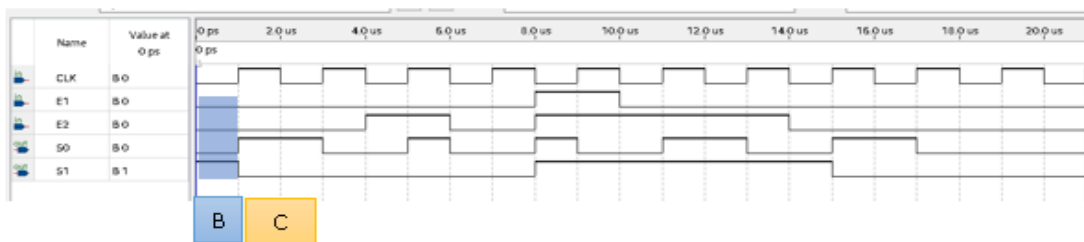
```

Código 3: Modelado del circuito (mediante código)

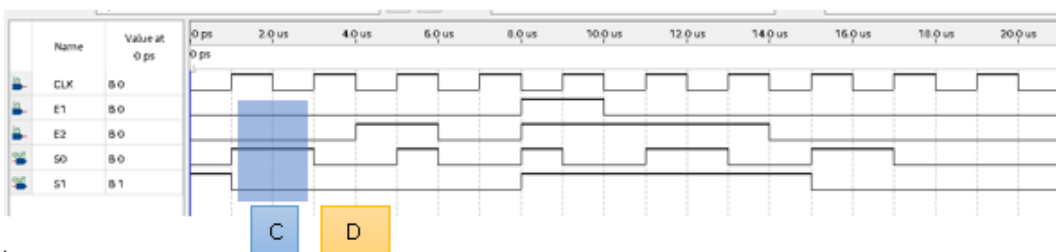


RTL 1: Circuito creado

Posteriormente se realizaron las pruebas correspondientes, empezamos con el estado B el análisis, que transición a el estado C debido a las entradas en 0.



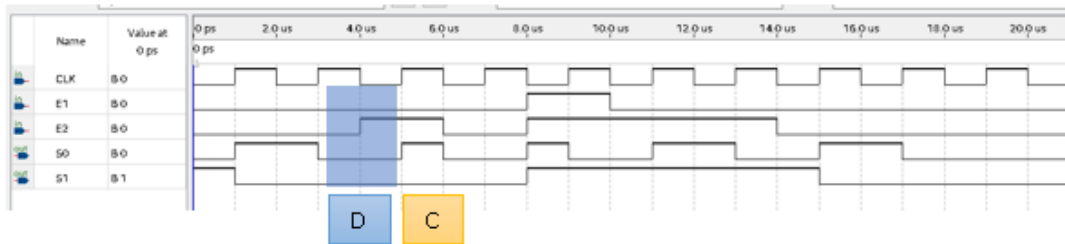
Del estado C pasamos directamente al estado D



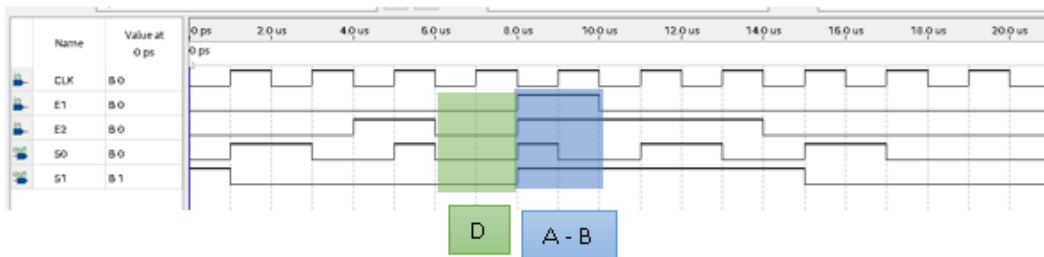
I

En

el estado D con las entradas en E1=0 y E2=1, damos un regreso a C



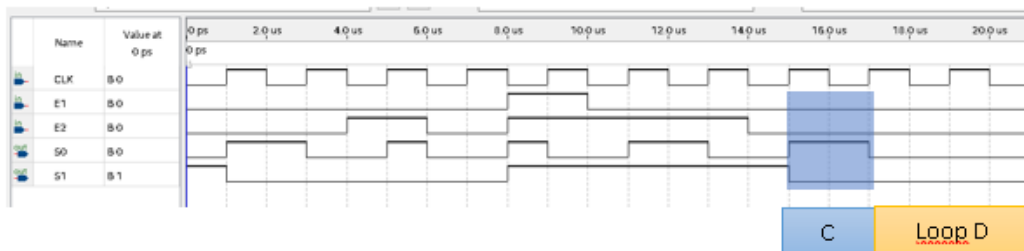
De C hacemos transición directamente a D, y aquí al tener E2=1 Y E2=1 vamos al estado A, de ahí directamente al estado B.



De B con las entradas E2 = 1 y E1=0, volvemos al estado A y otra vez al estado B posteriormente.



Del estado B con las entradas E2=0 y E1=0, vamos al estado C, y de este al estado D, que con las entradas en 00 queda en un loop en D.



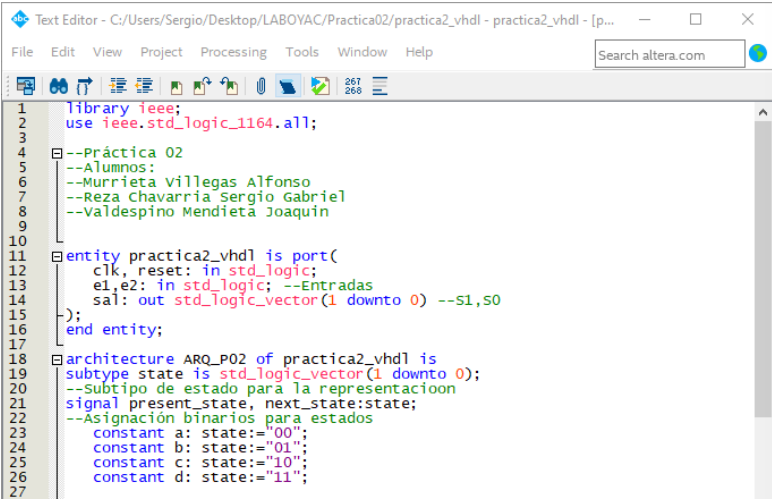
Ejercicio 2

Para la creación del proyecto se asignaron los valores de la entidad y la arquitectura del proyecto. La entidad se le declaran las entradas y salidas del proyecto, se asignaron las entradas e1 y e2, las salidas S1 y S2, y la entrada para el reloj interno y el botón de reinicio.

Para la arquitectura se creó un subtipo de dato para la representación del estado esto a partir de un vector lógico para la representación de los estados en formato binario. En la arquitectura se presentan 2 procesos, el primero depende del cambio con respecto al reloj, cuando haya un flanco de subida se revisará el estado de reset, si está arriba se reinicia el estado presente en el estado a, si no es así, el estado siguiente será el nuevo estado presente. En el segundo proceso se revisará con el cambio del estado presente. A partir de la revisión de las entradas con las sentencias de control por cada estado.

A continuación, se anexa el código vhd1 realizado.

Código



```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 --Práctica 02
5 --Alumnos:
6 --Murrieta villegas Alfonso
7 --Reza Chavarria Sergio Gabriel
8 --Valdespino Mendieta Joaquin
9
10
11 entity practica2_vhdl is port(
12     clk, reset: in std_logic;
13     e1,e2: in std_logic; --Entradas
14     sal: out std_logic_vector(1 downto 0) --S1,S0
15 );
16 end entity;
17
18 architecture ARQ_P02 of practica2_vhdl is
19     subtype state is std_logic_vector(1 downto 0);
20     --Subtipo de estado para la representacion
21     signal present_state, next_state:state;
22     --Asignación binarios para estados
23     constant a: state:="00";
24     constant b: state:="01";
25     constant c: state:="10";
26     constant d: state:="11";
27
```

```

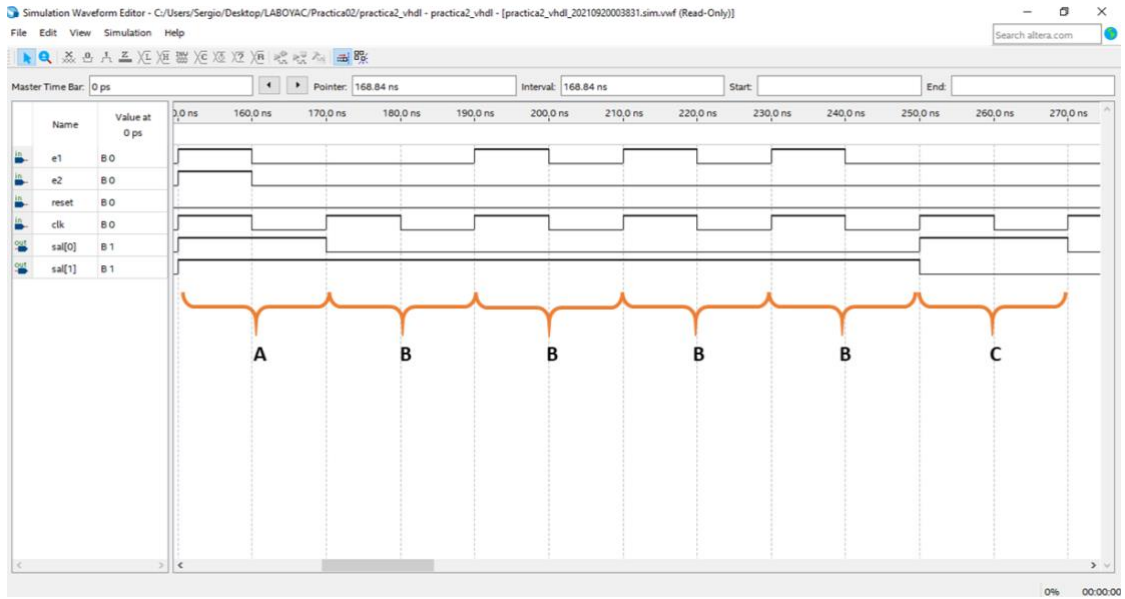
29 begin
30   process(clk)
31     --Cambio de estado con respecto al reloj interno
32     begin
33       if rising_edge(clk) then
34         if (reset='1') then
35           present_state<=a;
36         else
37           present_state<=next_state;
38         end if;
39       end if;
40     end process;
41   process(present_state)
42   begin
43     case present_state is
44       --Caso estado A, salida S1=1,S0=1
45       when a=>
46         sal<="11";
47         next_state<=b;
48       --Caso estado B, salida S1=1,S0=0
49       when b=>
50         sal<="10";
51         if (e1='0') then
52           if (e2='0') then
53             next_state<=c;
54           else
55             next_state<=a;
56           end if;
57         else
58           next_state<=b;
59         end if;
60       else
61         next_state<=b;
62       end if;
63       --Caso estado C, salida S1=0,S0=1
64       when c=>
65         sal<="01";
66         next_state<=d;
67       --Caso estado D, salida S1=0,S0=0
68       when d=>
69         sal<="00";
70         if (e2='1') then
71           if (e1='1') then
72             next_state<=a;
73           else
74             next_state<=c;
75           end if;
76         else
77           next_state<=d;
78         end if;
79       end case;
80     end process;
81   end architecture;
82 end architecture;
83
84
85
86
87
88

```

Código 4: Código VHDL de la máquina de estados

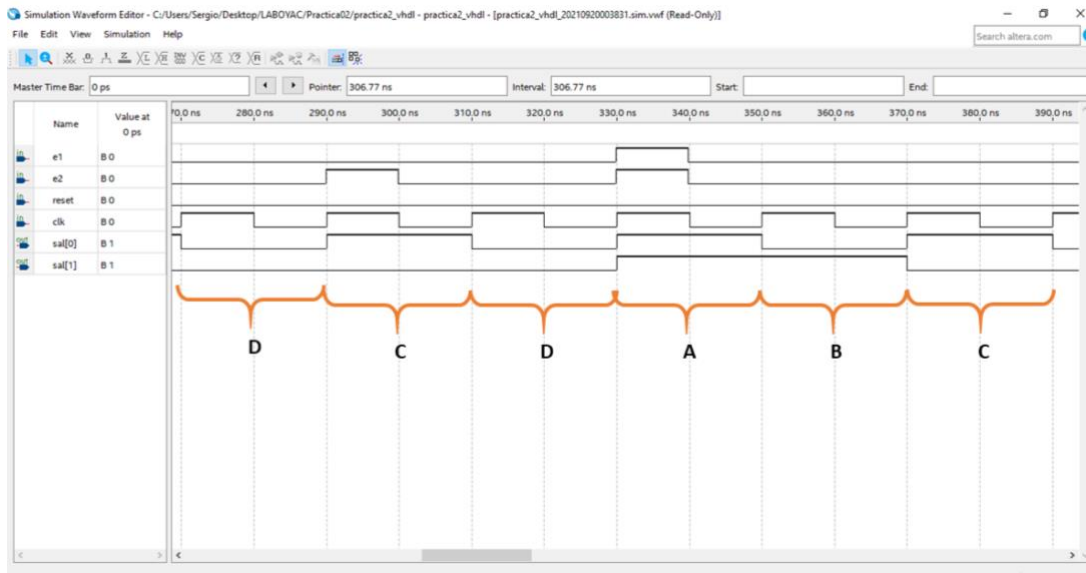
Pruebas

Las pruebas se realizaron en el simulador Waveform, a continuación, se presentan las diferentes situaciones de la carta ASM.



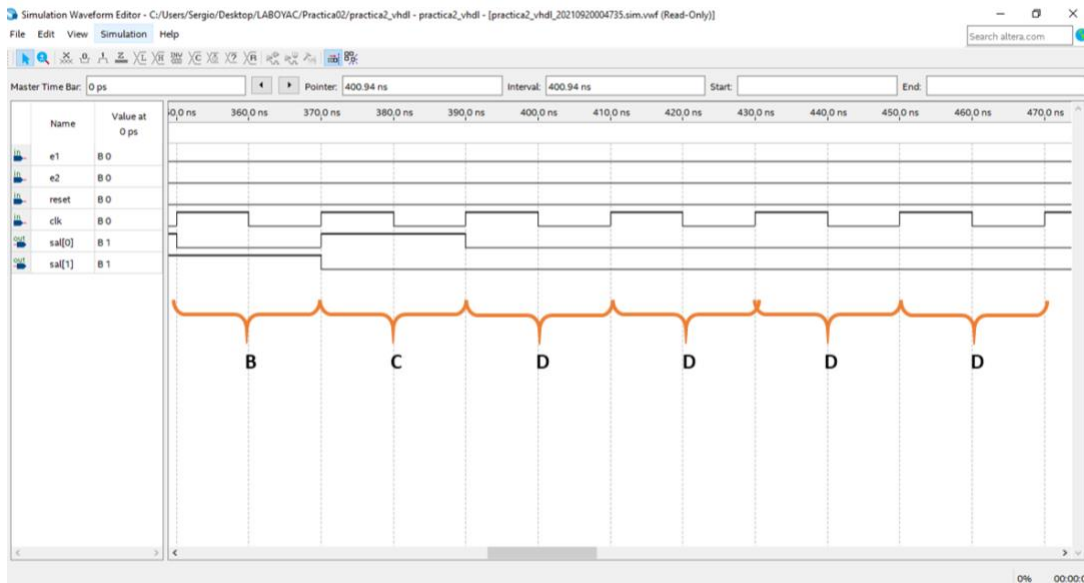
Simulación 3: Repetición del estado B por medio de la entrada E1=1

La siguiente sección es con respecto a la sección de ir del estado D al estado C con la entrada E2=1 y E1=0.



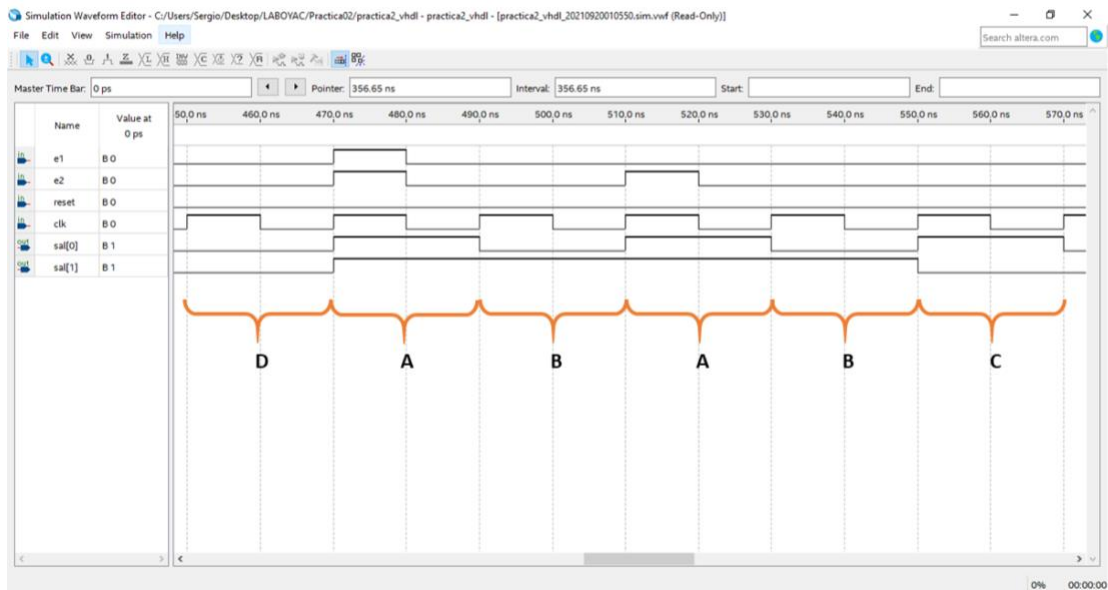
Simulación 4: Recorrido del estado D al estado C

La siguiente sección es para representar la repetición del estado D por medio de la entrada E2=0



Simulación 5: Repetición del recorrido del estado D de manera repetitiva

Al final de la simulación se revisa el recorrido del estado A al estado B por medio de las entradas E1=0 y E2=1.



Simulación 6: Recorrido del estado B al estado A

Conclusiones

Murrieta Villegas Alfonso

En la presente práctica aprendimos a realizar máquinas de estados mediante el lenguaje de desarrollo de componentes de hardware VHDL a partir de su forma Combinacional, empleando principalmente tablas de verdad, mapas de Karnaugh además de su representación de estados en forma de código.

Por último en la presente práctica repasamos el funcionamiento específico de como usar el IDE de Quartus a través de VHDL.

Reza Chavarria Sergio Gabriel

A partir de la práctica pudimos recuperar los conocimientos y el manejo de Quartus Prime con respecto a la estructura de codificación del lenguaje VHDL, vistos en cursos anteriores. Con esto se realizó la representación de una máquina de estado a partir de su forma combinacional (utilizando las tablas de verdad y los mapas de Karnaugh para su implementación) o por representación de los estados en forma de código.

Valdespino Mendieta Joaquin

En la presente practica pudimos reforzar los conocimientos previos antes vistos en otras materias como Diseño Digital Moderno o VLSI, en relación a la creación de máquinas de estado, implementación de las Cartas ASM, así como parte del lenguaje VHDL para la descripción y realización de los mismos ejercicios, con ello podemos familiarizarnos nuevamente con los elementos para trabajar en esta materia.

Bibliografía

- Fonseca E. (2020) A10 ASM. Consultado el 19 de septiembre de 2021. Enlace de video: <https://youtu.be/iV58vpy49WE>
- Estrada G. (Desconocido) Cartas ASM. Encontrado el 19 de septiembre de 2021, en <https://sites.google.com/site/cartasasm/>

- Fonseca E. (2020). A2 secuenciales2. Consultado el 19 de septiembre de 2021.
Enlace del video: <https://youtu.be/sisY0Naps8A>
- Saavedra M. (Desconocido). Circuitos Secuenciales. Flip Flop tipo D.
Consultado el 20 de septiembre de 2021, en:
<http://circuitossecuenciales.weebly.com/flip-flop-tipo-d.html>