**Objetivo**

Demostrar a los estudiantes la forma de diseñar máquinas de estado en lenguaje VHDL mediante el diseño de un detector de secuencia.

**Introducción**

En esta práctica repasaremos temas importantes vistos en Diseño Digital Moderno. Algunos de estos temas abarcan lo siguiente: la introducción básica a VHDL y el entorno de trabajo, el concepto de una máquina de estado, sus estados y transiciones.

Para esto, es importante que todos los miembros del equipo comprendan cómo funcionan las máquinas de estado. Esto podría hacerse mediante diagramas de estados y transiciones para representar el comportamiento de la máquina de estado que se diseñará. También es importante conocer los símbolos y las estructuras específicas de VHDL utilizadas para diseñar máquinas de estado, como "process", "estado", "estado siguiente" y "case". Además de declarar y utilizar señales en VHDL para representar entradas, salidas y estados internos de la máquina de estado.

En esta práctica en específico se diseñará un detector de secuencia basado en una tabla con los diferentes estados y entradas. Esta tabla la veremos mejor durante el desarrollo de la práctica. Una parte muy importante en la realización de esta práctica es la realización de simulaciones en VHDL para verificar el funcionamiento correcto de la máquina de estado antes de la implementación en hardware.

Durante la realización de esta práctica no haremos uso de circuitos armados ya que solo realizaremos la simulación en el programa de Vivado ya que las simulaciones en Quartus no funcionan. Estas simulaciones las explicaremos a detalle durante el desarrollo y resultados.

**Desarrollo**

El alumno programará la siguiente tabla de estados



Código que realizamos:

library IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY SECUENCIA IS

PORT (CLK: IN STD\_LOGIC;

RESET: IN STD\_LOGIC;

X: IN STD\_LOGIC;

Y: IN STD\_LOGIC;

Z: OUT STD\_LOGIC);

END SECUENCIA;

ARCHITECTURE BEHAVIORAL OF SECUENCIA IS

TYPE ESTADOS IS (A, B, C, D);

SIGNAL Epresente, Esiguiente: ESTADOS;

BEGIN

PROCESS(CLK)

BEGIN

IF (RESET = '1') THEN

Epresente <= A;

ELSIF(CLK'EVENT AND CLK='1') THEN

-- CLK'EVENT SE ACTIVA EN EVENTO (ALTO Y BAJO) PERO

-- CON EL '1' SOLO ACTIVAMOS EN ALTOS

Epresente <= Esiguiente;

END IF;

END PROCESS;

PROCESS(Epresente, X, Y)

BEGIN

CASE Epresente IS

WHEN A =>

IF (X='0' AND Y = '0') THEN

Esiguiente <= A;

Z<='0';

ELSIF (X='0' AND Y = '1') THEN

Esiguiente <= B;

Z<='0';

ELSIF (X='1' AND Y = '0') THEN

Esiguiente <= C;

Z<='1';

ELSIF (X='1' AND Y = '1') THEN

Esiguiente <= D;

Z<='1';

END IF;

WHEN B =>

IF (X='0' AND Y = '0') THEN

Esiguiente <= B;

Z<='1';

ELSIF (X='0' AND Y = '1') THEN

Esiguiente <= C;

Z<='1';

ELSIF (X='1' AND Y = '0') THEN

Esiguiente <= C;

Z<='0';

ELSIF (X='1' AND Y = '1') THEN

Esiguiente <= A;

Z<='0';

END IF;

WHEN C =>

IF (X='0' AND Y = '0') THEN

Esiguiente <= D;

Z<='1';

ELSIF (X='0' AND Y = '1') THEN

Esiguiente <= D;

Z<='0';

ELSIF (X='1' AND Y = '0') THEN

Esiguiente <= D;

Z<='1';

ELSIF (X='1' AND Y = '1') THEN

Esiguiente <= C;

Z<='0';

END IF;

WHEN D =>

IF (X='0' AND Y = '0') THEN

Esiguiente <= A;

Z<='0';

ELSIF (X='0' AND Y = '1') THEN

Esiguiente <= A;

Z<='1';

ELSIF (X='1' AND Y = '0') THEN

Esiguiente <= A;

Z<='0';

ELSIF (X='1' AND Y = '1') THEN

Esiguiente <= B;

Z<='1';

END IF;

END CASE;

END PROCESS;

END BEHAVIORAL;

El código es bastante sencillo de entender. En cada WHEN del CASE tenemos un estado (A, B, C o D). Para movernos de un estado a otro tenemos que considerar la combinación de entradas X e Y. Dentro de cada WHEN hay 4 IF, dichos IF contienen las combinaciones disponibles de X e Y, y dentro de cada IF viene el estado siguiente y el valor de Z dependiendo la combinación de X e Y.

A continuación, en los resultados, veremos la simulación realizada para una secuencia de entrada específica.

**Resultados**

Vamos a utilizar la secuencia de entrada X = 0101 e Y=1011, para que así, en la entrada X=1 e Y=1, tengamos Z=1 y, por lo tanto, se reconozca nuestra secuencia. A continuación, se muestran capturas de pantalla de la simulación paso a paso. Cabe destacar que realizamos la simulación en Vivado porque en Quartus tuvimos muchos problemas que no se pueden resolver.

1. Inicio. Estado futuro = Estado actual = A

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

1. X=0, Y=1, Z=0, estado actual = A, estado siguiente = B (revisar tabla)

Interfaz de usuario gráfica

Descripción generada automáticamente

1. X=1, Y=0, Z=0, estado actual = B, estado siguiente = C

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. X=0, Y=1, Z=0, estado actual = C, estado siguiente = D

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

1. X=1, Y=1, **Z=1**, estado actual = D, estado siguiente = B

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

Como se puede observar en esta última captura, ya llegamos a Z=1, por lo que se aceptan las secuencias de entrada de X e Y que pusimos. Si se compara cada punto enlistado anteriormente con la tabla, se podrá observar que, en efecto, nuestros resultados son correctos.

**Bibliografía**

* Wikipedia. (2023, 5 de septiembre). VHDL. <https://en.wikipedia.org/wiki/VHDL>
* Wikipedia. (2023, 5 de septiembre). State machine. <https://en.wikipedia.org/wiki/State_machine>
* Electronics Hub. (s.f.). Sequence Detector using State Machine and VHDL Code. <https://www.electronicshub.org/sequence-detector-using-state-machine-vhdl-code/>