**Objetivo**

Demostrar a los estudiantes la forma de declarar tipos de datos diferentes a los definidos en el lenguaje VHDL mediante el diseño del sistema de control de un tren eléctrico.

**Introducción**

Para la realización de esta práctica será importante tener en mente varios conceptos, tanto del lenguaje VHDL, como conceptos vistos en la materia de Diseño Digital Moderno. Algunos de estos conceptos y conocimientos son:

* Una sólida comprensión de VHDL y el entorno de trabajo, incluyendo la sintaxis básica, las estructuras de diseño, funcionamiento del programa Quartus y Vivado para la realización del código y de las simulaciones respectivamente.
* Se deberá declarar tipos de datos personalizados en VHDL utilizando el comando "type" y sus ventajas en comparación con los tipos de datos predefinidos.
* Tendremos que comprender los conceptos fundamentales de los sistemas de control.
* Para la realización del diseño del sistema de control en código, deberemos definir claramente los requisitos y especificaciones del sistema de control del tren eléctrico, como la detección de obstáculos. Más adelante en la práctica veremos de manera más específica esto.
* Algunas de estructuras de control importantes durante la realización de esta práctica serán "if-else", "case", y "while", que nos serán necesarias para implementar la lógica de control del tren eléctrico.

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 diseñara un sistema que controle la apertura y cierre de un puente en el cruce de barcos que van de norte a sur y viceversa, y de autos que van de este a oeste y viceversa. Los barcos tienen preferencia, por lo que se requiere que el sistema manipule sensores con el fin de que cuando se detecte un barco, se envié una señal a unos semáforos que pasen de luz verde, a la amarilla y luego a la roja. Cuando el barco ya no se encuentre cerca del puente, la luz roja se apagará y se encenderá la verde.

Al mismo tiempo que se active el detector de barcos, se activara una señal sonora para que los conductores distraídos pongan atención al cambio de luces en los semáforos y se empiece a abrir el puente dando paso a los barcos.

Imagen que contiene pizarrón, texto, reloj

Descripción generada automáticamente

Diagrama de Estados

Diagrama

Descripción generada automáticamente

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sensores | | | | | | | |
|  | Estado Actual | | Entradas | | Estado siguiente | |  |
|  | Q1 | Q2 | TA | TB | X | Y |  |
| S0 | 0 | 0 | 0 | \* | 0 | 1 | S1 |
| S0 | 0 | 0 | 1 | \* | 0 | 0 | S0 |
| S1 | 0 | 1 | \* | \* | 1 | 0 | S2 |
| S2 | 1 | 0 | \* | 0 | 1 | 1 | S3 |
| S2 | 1 | 0 | \* | 1 | 1 | 0 | S2 |
| S3 | 1 | 1 | \* | \* | 0 | 0 | S0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cambio de Luces | | | | | | |
|  | Estado Actual | | Salidas | |  | |
|  | Q1 | Q0 | LA1 | LA0 | LB1 | LB2 |
| S0 | 0 | 0 | 0 - verde | 0 | 1 | 0 |
| S1 | 0 | 1 | 0 - amarillo | 1 | 1 | 0 |
| S2 | 1 | 0 | 1 - rojo | 0 | 0 | 0 |
| S3 | 1 | 1 | 1 - rojo | 0 | 0 | 1 |

**Hacer mapas de Karnaugh**

**Para Y**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TATB/Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 0 | 1 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 0 | 0 |

**Para X**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TATB/Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 1 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 0 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Circuito Secuencial | | | | | | |
|  | Estado Actual | | Entrada  X | Estado futuro | | Salida |
| A | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| B | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| C | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| D | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |

**Hacer mapas de Karnaugh**

**Para Q0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 / Q0 X | 00 | 01 | 11 | 10 |
| 0 |  |  | 1 |  |
| 1 | 1 | 1 |  |  |

**Para Q1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 / Q0 X | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 1 |  |
| 1 |  |  |  |  |

**Para Z**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 / Q0 X | 00 | 01 | 11 | 10 |
| 0 |  |  |  |  |
| 1 |  |  | 1 |  |

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity P02 is

Port (

clk: in STD\_LOGIC; -- Reloj del sistema

sensorBarco: in STD\_LOGIC; -- Sensor de barcos

sensorAuto: in STD\_LOGIC; -- Sensor de autos

luzVerde: out STD\_LOGIC; -- Luz verde para autos

luzAmarilla: out STD\_LOGIC; -- Luz amarilla para autos

luzRoja: out STD\_LOGIC; -- Luz roja para autos

puenteAbierto: out STD\_LOGIC -- Señal para abrir el puente

);

end P02;

architecture Behavioral of P02 is

type Estados is (EsperandoBarco, CerrandoPuente, AbriendoPuente, PasandoBarco, PasandoAuto);

signal estadoActual, estadoSiguiente: Estados;

signal contador: integer := 0; -- Contador para temporizar la luz amarilla

begin

process (clk)

begin

if rising\_edge(clk) then

estadoActual <= estadoSiguiente;

contador <= contador + 1;

end if;

end process;

process (estadoActual, sensorBarco, sensorAuto, contador)

begin

case estadoActual is

when EsperandoBarco =>

if sensorBarco = '1' then

estadoSiguiente <= CerrandoPuente;

elsif sensorAuto = '1' then

estadoSiguiente <= PasandoAuto;

else

estadoSiguiente <= EsperandoBarco;

end if;

when CerrandoPuente =>

puenteAbierto <= '0';

if sensorBarco = '0' then

estadoSiguiente <= PasandoBarco;

else

estadoSiguiente <= CerrandoPuente;

end if;

when AbriendoPuente =>

puenteAbierto <= '1';

estadoSiguiente <= EsperandoBarco;

when PasandoBarco =>

if sensorBarco = '1' then

estadoSiguiente <= PasandoBarco;

else

estadoSiguiente <= AbriendoPuente;

end if;

when PasandoAuto =>

if contador = 50000000 then -- Temporizador para luz amarilla (ajusta según necesites)

estadoSiguiente <= AbriendoPuente;

contador <= 0;

else

estadoSiguiente <= PasandoAuto;

end if;

when others =>

estadoSiguiente <= EsperandoBarco;

end case;

end process;

-- Control de semáforos para autos

process (estadoActual)

begin

case estadoActual is

when EsperandoBarco =>

luzVerde <= '1';

luzAmarilla <= '0';

luzRoja <= '0';

when CerrandoPuente =>

luzVerde <= '0';

luzAmarilla <= '1';

luzRoja <= '0';

when AbriendoPuente =>

luzVerde <= '0';

luzAmarilla <= '1';

luzRoja <= '0';

when PasandoBarco =>

luzVerde <= '0';

luzAmarilla <= '0';

luzRoja <= '1';

when PasandoAuto =>

luzVerde <= '0';

luzAmarilla <= '1';

luzRoja <= '0';

when others =>

luzVerde <= '1';

luzAmarilla <= '0';

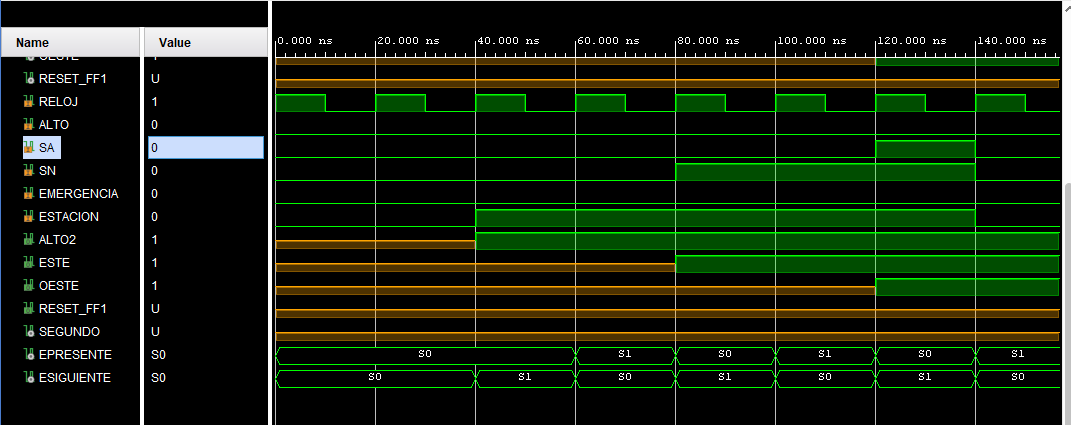
luzRoja <= '0';

end case;

end process;

end Behavioral;

**Resultados**



En esta simulación podemos observar lo siguiente: el cambio de estados de s1 a s0 esto debido a la lógica utilizada en el programa ya que el estado se actualiza cada que la variable reloj cambia, además al momento de que se tiene un 1 en estación causa que pase del estado inicial S0 al estado S1 en este estado revisa si hay algún alto o cambio en las variables SA, SN o emergencia y según esta ira cambiando su valor.

**Bibliografía**

* Wikipedia. (2023, 5 de septiembre). VHDL. <https://en.wikipedia.org/wiki/VHDL>