22-2-2024

Programación de sistemas reconfigurables

Tarea 5. Estructuras de programación 2

Ingeniería en electrónica y computación

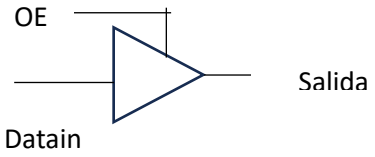
ANDRADE SALAZAR, IGNACIO
CENTRO UNIVERSITARIO DE LOS VALLES, UNIVERSIDAD DE GUADALAJARA

ESTRUCTURAS DE PROGRAMACIÓN

1.- ¿Cuáles son las estructuras elementales de las declaraciones secuenciales? Solo cuáles son

- If-then-else
- If-then-elsif
- case

2.-En la sección Buffers tri-estado: ponga figura 2.13 (dibujo y código) escrito o tecleado y mencione que significa Z y OE.



OE	Salida	Comentario
0	Z	Z=Alta impedancia
1	Datain	

```
library ieee;
use ieee.std_logic_1164.all;
entity tri_est is
port( OE, datain: in std_logic;
            salida: out std_logic);
end tri_est;
architecture arq_buffer of tri_est is
begin
    process (OE, datain) begin
        if OE = '0' then
            salida <= 'Z';
        else
            salida <= datain;
        end if;
    end process;
end arq_buffer;
```

3.-Dibuje la tabla 2.2 no imagen y teclee el párrafo inicial de la página 38. En apariencia…

‘U’	**Valor indefinido “undefined”**
‘X’	**Valor fuerte desconocido**
‘0’	**0 Fuerte**
‘1’	**1 Fuerte**
‘Z’	**Alta impedancia**
‘W’	**Valor débil desconocido**
‘L’	**0 débil la L debe estar entre comilla ‘L’**
‘H’	**1 débil**
‘-‘	**No importa (don´t care);**

4.-Operadores relacionales se utilizan para....., ponga todo el párrafo completo. Dibuje la tabla 2.3

Los operadores relacionales se utilizan para evaluar la igualdad, desigualdad o la magnitud en una expresión. Los operadores de igualdad y desigualdad (= y /=) son definidos en todos los tipos de datos. En tanto, los operadores de magnitud (<,<= y >=) están definidos solo dentro del tipo escalar. Sin embargo, en ambos casos debe considerarse que el tamaño de los vectores en los que se aplicarán dichos operadores debe ser de igual magnitud.

Operador	Significado
=	Igual
/=	Diferente
<	Menor
<=	Menor o igual
>	Mayor
>=	Mayor o igual

5.-Página 41- complete ->En la línea 4 del código que se muestra ..... complete todo el párrafo. Al final de la Figura 2.19.

En la línea 4 del código que se muestra en la figura 2.19 se ve que la entrada a se declara como de tipo entero (integer), con un rango (range) de 0 a 9 (debido a que son diez los dígitos que se usarán), lo que nos permite manejar los datos directamente de la forma decimal. Por otro lado, el perto de salida es un arreglo del tipo std_logic_vector, esto nos permite seleccionar un dato en forma decimal y obtener a la salida su equivalente binario. Al igual que en el listado anterior, el funcionamiento del circuito se basa en un proceso, el cual se ejecuta mediante una declaración secuencial y la instrucción elsif.

6.-En la sección Enteros (integer): El tipo integer se utiliza en VHDL .... ponga los cinco renglones.

El tipo integer se utiliza en VHDL para representar números enteros con o sin signo. Los Valores soportados por el lenguaje se encuentran definidos en el rango de -2,147-483,647(-2^31-1) hasta 2,147,483,647(2^31-1).

Un rango (range) es una palabra reservada por VHDL y utilizada para definir el conjunto de valores que el entero tomará; en conjunto con las palabras reservadas to y downto se puede especificar si el rango es creciente o decreciente, respectivamente.

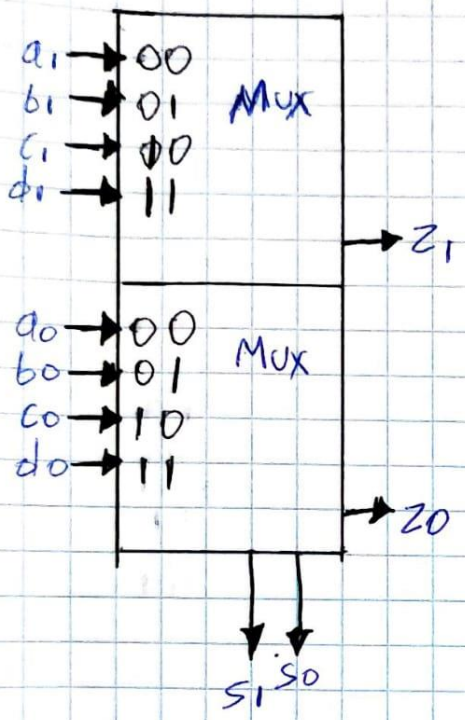
7.- Resuma la página 43, ponga las imágenes de la Figura 2.22, así como los códigos del listado 2.12 a y b y diga sus diferencias y similitudes

La descripción case tiene mucha similitud con el manejo de la declaración with-select-when, pues ambas se utilizan en la comparación de tablas con valores constantes; la única diferencia entre estas radica en que la instrucción case es del tipo secuencial y permite la asignación multiple.

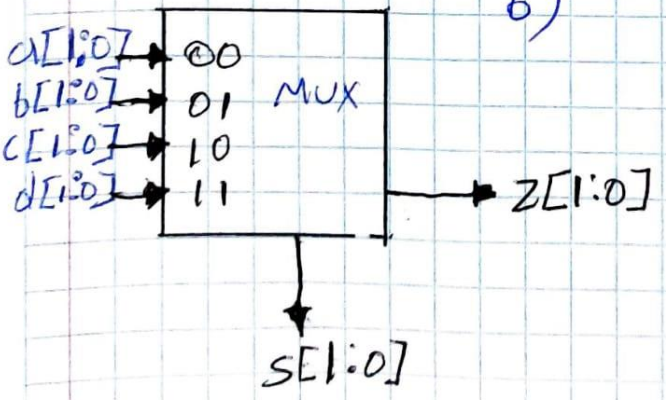
```
--Utilizando with-select-when
library ieee;
use ieee.std_logic_1164.all;
entity mux is
port(   a,b,c,d: in std_logic_vector (1 downto 0);
             s: in std_logic_vector (1 downto 0);
             z: out std_logic_vector (1 downto 0));
end mux;
architecture arqmux4 of mux is
begin
with s select
    z <= a when "00",
         b when "01",
         c when "10",
         d when others;
end arqmux4;
```

```
--Utilizando case
library ieee;
use ieee.std_logic_1164.all;
entity mux is
port(   a,b,c,d: in std_logic_vector (1 downto 0);
             s: in std_logic_vector (1 downto 0);
             z: out std_logic_vector (1 downto 0));
end mux;
architecture arqmux4 of mux is
process (s,a,b,c,d)
begin
case s is
    when "00" => z <=a;
    when "01" => z <=b;
    when "10" => z <=c;
    when others => z <=d;
    end case;
    end process;
end arqmux4;
```

a)



b)



Selección		Salida
S_1	S_0	z
0	0	a
0	1	b
1	0	c
1	1	d others

6.- Genere un mux de 3 canales de entrada, cada canal tiene 16 líneas E0, E1 y E2 = 16. Agregue las líneas de selección necesarias. Y tiene un canal de salida de 16 líneas F1 = 16

Entidad = Multiplexor3, hay que expresarlo en dos formas ->usando with select when y usando case

Entidad y Architecture completas.

```
--Utilizando with-select-when
entity Multiplexor3 is
    Port (
        E0, E1, E2 : in std_logic_vector(15 downto 0);  -- 16 líneas de
entrada para cada canal
        F1         : out std_logic_vector(15 downto 0) -- 16 líneas de
salida
    );
end entity Multiplexor3;
architecture Behavioral of Multiplexor3 is
begin
    process (E0, E1, E2)
    begin
        if E0 = "0000000000000000" then
            F1 <= E0;
        elsif E1 = "0000000000000000" then
            F1 <= E1;
        elsif E2 = "0000000000000000" then
            F1 <= E2;
        else
            F1 <= (others => '0'); -- Si ninguna entrada está
seleccionada, se establece la salida en cero
        end if;
    end process;
end architecture Behavioral;
```

```
--Utilizando case
entity Multiplexor3 is
    Port (
        E0, E1, E2 : in std_logic_vector(15 downto 0);  -- 16 líneas de
entrada para cada canal
        F1         : out std_logic_vector(15 downto 0) -- 16 líneas de
salida
    );
end entity Multiplexor3;
architecture Behavioral of Multiplexor3 is
begin
    process (E0, E1, E2)
    begin
        case (E0 & E1 & E2) is
            when "0000000000000000" =>
                F1 <= E0;
            when "0000000000000001" =>
                F1 <= E1;
            when "0000000000000010" =>
                F1 <= E2;
            -- Puedes agregar más casos según sea necesario para otras
combinaciones
            when others =>
                F1 <= (others => '0'); -- Por defecto, establece la
salida en cero si ninguna entrada está seleccionada
        end case;
    end process;
end architecture Behavioral;
```