AC2 - PRACTICA 1 - SUMADOR DE 1 BIT

Practica de introducción, con un modelo sencillo de un FA

Estructura básica de un modelo en el lenguaje VHDL

En VHDL es buena practica dividir en partes la implementación:

Interfaz (entity)

Define su interfaz con el exterior, como una lista de puertos.

```
entity NAME_ENTITY is
    port    (signal_name : mode type;
             signal_name : mode type;
             ...
             signal_name : mode type
            );
end [NAME_ENTITY]; --opcional poner otra vez el nombre
```

El mode indica si la señal es de entrada o salida (in | out), el type indica el tipo de señal, puede ser un tipo básico de vhdl o uno definido por el usuario, en este entorno tendremos los siguientes:

bit, bit_vector, U(no definido), Z(alta impedancia), don't care o - (no importa), X(desconocido), std_logic(un bit)

Arquitectura (Architecture)

Aquí se define el comportamiento de la entidad definida en la parte de 'entity' mediante sentencias de asignación de señales concurrentes.

```
architecture nombre_arch of NAME_ENTITY is
begin
    s <= (x xor y) xor cen;
    -- mas Concurrent Statements
end nombre_arch
```

Se suele utilizar el 'Modelo de comportamiento(Behavioral Model)' donde las salidas son funcion de las entradas. En los "Concurrent Statements" se utilizan sentencias de asignación de señales, donde se utilizan operadores logicos (and, or, nand, nor, xor, xnor, not) y el operador de asignación <=

Librerias (Library)

En VHDL un package es un fichero que contiene declaraciones de objetos, tipos de datos y otros. Una libreria almacena packages, antes de utilizarse se deben declarar utilizando 'library' y mediante la palabra 'use' se hacen accesibles.

```
library ieee;
use ieee.std_logic_1164.all;
```

- El lenguaje VHDL no es CaseSense (no distingue de mayusculas y minusuculas en los identificadores)
- Un caracter se especifica entre comillas
- No distingue de uno o más espacios en blanco

Utilización Quartus

Para utilizarlo, se utilizaran cuatro directorios(CODIGO, PRUEBAS, RESULTADOS, QUARTUS) para no mezclar datos.

Crear proyecto, editar ficheros vhdl, compilar, RTL

File -> New Project Wizard -> working directory (QUARTUS) + name of project (nombre arbitrario) + entity(nombre de la entidad mas externa).

File -> New -> Design files -> VHDL file -> File -> Save As (CODIGO) + Add file to current project.

Ejemplo primer codigo vhdl:

```
library ieee;                                  Especificación de la libreria que alberga las definiciones que deben
use ieee.std_logic_1164.all;                   utilizarse
entity s1bit is                                Declaración "entity". Nombre del módulo y descripción de la inter-
 port (   x, y, : in   std_logic;              face del módulo (entradas y salidas).
          cen: in     std_logic;
          s: out      std_logic;
          csal: out   std_logic );
end s1bit;
architecture funcional of s1bit is             Declaración "architecture" donde debe describirse el comporta-
begin                                          miento del circuito.
   s <= (x xor y) xor cen;
  csal <= (x and y) or (x and cen) or (y and cen);
end funcional;
```

Processing -> Start -> Start Analysis & Elaboration

Tools -> Netlist viewers -> RTL Viewer

Utilización Modelism

Para activar la simulación: Tools -> Run Siumlation Tool -> RTL Simulation

Selecionando rtl_work y la entity correspondiente y añadiendolo a la wave se podra ver la evolución de las señales.

Para comprobar el funcionamiento logico se puede aplicar la tecnica de "señales periodicas" utilizando wave -> Clock(Duty : 50, Period 10ns, Falling) para cada señal.

File -> Save Format -> wave.do

Para guardar la ventana temporal Export -> Image (tenindo selecionada la ventana).

La ventana textual se puede guardar tambien desde Export

Modelo VHDL con retardo

El behavioral Model no incorpora ningun tipo de elementos de retardo por la tecnologia, en el 'Data Flow Model' si que se añaden. No se modificara la parte de entity, solo la architecture. En este sitio es donde se declararan señales intermedias (adicionales) y los retardos se especifican en cada sentencia de asignación de señal:

```
--parte declarativa:
architecture flujodatos of s1bit is
    --constant name : type := initialValue; Valor siempre constante
    -- signal name : type [:= initial value]; Variable como siempre
    constant retardoxor: time := 15 ns;
    constant retardoand: time := 10 ns;
    constant retardoor: time := 15 ns;
    signal xorxy, andxy, andxcen, andycen: std_logic;
begin
    xorxy <= x xor y after retardoxor;
    s <= xorxy xor cen after retardoxor;
    andxy <= x and y after retardoand;
    andxcen <= x and cen after retardoand;
    andycen <= y and cen after retardoand;
    csal <= andxy or andxcen or andycen after retardoor;
end flujodatos;
```

Para estimular este modelo en la simulación se puede hacer mediante un archivo (estimulos.do) que se activa por la consola con do /path/../estimulos.do

```
force s1bit/x U 0 ns, 0 50 ns, U 100 ns, 1 150ns, <valor><tiempo>
force s1bit/y ...
etc ...
```