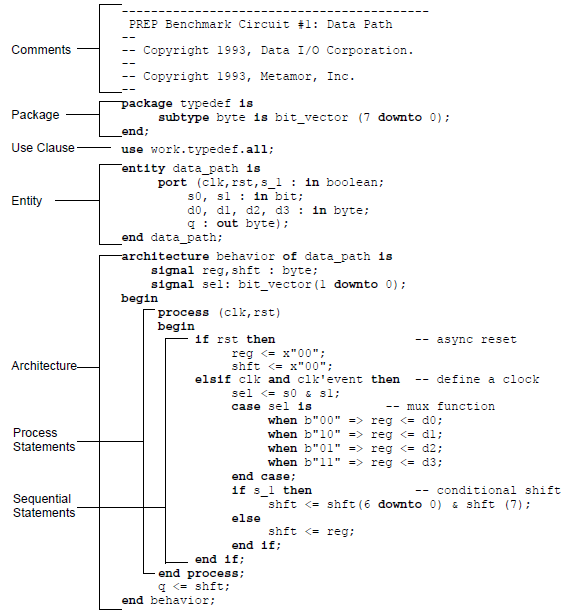
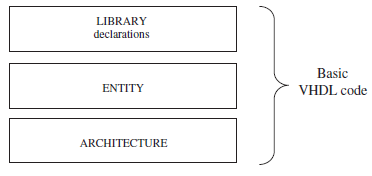
**VHDL. Lenguaje para la descripción de hardware. Elementos del lenguaje.**

Este lenguaje se usa para describir el comportamiento de un circuito o sistema electrónico. La letra V se usa para expresar el concepto de Very High Speed Integrated Circuits , este resulto de una iniciativa del ejercito de los Estados Unidos en la década de los 80’s. Posteriormenete l organismo IEEE creo el Estándar 1076 para sistemas de lógica multivaluada. El lenguaje se usa para la síntesis de circuitos, así puede usarse para simulación de estos, normalmente para implantación de estos diseños en CPLD (dispositivos Lógicos Complejos)y FPGA (Field Programable Gate Array). Un código de lenguaje VHDL se constituye de los elementos siguientes:



Aunque inicialmente no todos los elementos se necesitan en un programa se hablará de cada uno de ellos para explicar sus funciones en el mismo. En esencia un programa básico consta de los siguientes elementos.



La librería es una declaración que hace referencia a una lista de librerías a usares en un diseño. Esta así misma es un conjunto de piezas de código para reusarse o enlazarse con otros diseños. Al menos en un diseño debemos colocar tres librerías:

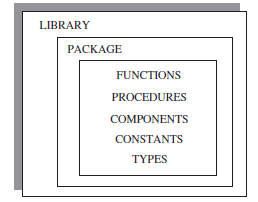
* ieee.std\_logic\_1164 (ieee library),
* standard (La std library)
* work

En una sintaxis sencilla tenemos

**LIBRARY** nombre\_de\_la\_libreria;

**USE** nombre\_de\_la\_libreria.**package**\_name.package\_parts;

Una librería tiene la estructura siguiente:



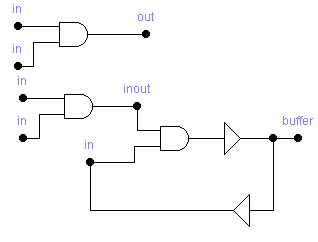
Un **PACKAGE** contiene un conjunto de funciones , procedimientos y componentes , entre otros elementos y pueden compilares dentro de una librería destino.

**ENTIDAD.**

El siguiente elemento básico de un código es la **ENTITY** (entidad) , esta define las entradas y la salidas de un diseño, para ello usa uno o más puertos, el cual es análogo con los pines en un símbolo esquemático. Cada **port** (puerto) define una dirección (entrada , salida , entrada salida o Buffer) usando las instrucciones

**in , out , inout y buffer** según corresponde.

Las variables declaradas como entrada solo pueden ser leídas y no se pueden escribir, así mismo una salida no puede ser leída , ya que solo se puede escribir. Solo se puede leer fuera de la entidad una variable **buffer (pasar un valor a otro proceso)**. La variable **inout** se usa para retroalimentar un valor entre variables.



Una **entidad** se declara:

**entity** nombre\_entidad **is**

**port** (entrada1,entrada2,entradaN : in tipo\_entrada;

salida1,salida2,salidaN : out tipo\_salida);

**end** nombre\_entidad;

el nombre de la entidad el programador lo selecciona, así mismo el nombre de cada entrada y salida se sugiere que el nombre de cada variable describa su uso en el circuito (bloquear, habilitar, salida, etc..).

Por ejemplo una entidad llamada *compuertas* siguiente:

-- Abrir Comentarios

**entity** compuertas **is**

**port** (a,b,c : in bit;

y,z : out bit);

**end** compuertas;

En la herramienta de XILINIX el código aparece

**library** IEEE;

**use** IEEE.STD\_LOGIC\_1164.ALL;

**use** IEEE.STD\_LOGIC\_ARITH.ALL;

**use** IEEE.STD\_LOGIC\_UNSIGNED.ALL;

**entity** compuertas is

**Port** (

C : in std\_logic;

B : in std\_logic;

A : in std\_logic;

Y : out std\_logic;

Z : out std\_logic);

**end** compuertas;

Las librerías colocadas por la herramienta nos indican que usaremos la estándar lógica 1164, aritmética (comparaciones e igualaciones) y lógica no signada.Quitar la librería marca errores , aunque este correcta la sentencia.

Se usa in std\_logic para normalizar las salida a 8 niveles de lógica multivaluaday nueve para datos no signados.**ARQUITECTURA.**

Un programa en VHDL debe tener una parte de **architecture ,** esta describe el diseño de forma funcional. Esta puede contener procesos concurrentes y secuenciales. El lenguaje permite más de una arquitectura para una misma entidad. Como modelo de creación de la arquitectura, **actividad** es el nombre dado a la actividad que realizara la entidad.

**architecture** **actividad** **of** nombre\_entidad **is**

**begin**

--código

**end** **actividad**;

Para crear la actividad debemos desarrollar un código basado en operadores (asignación , lógicos , aritméticos , relacionales , de corrimiento y concatenación)). El operador de asignación es

**<=** este funciona de derecha a izquierda asignando el miembro o miembros de una función hacía la varable izquierda, recordando en el caso de las variables in , out para que no haya errores.

**:=** este se usa para asignar valores de una variable , constante o genérico , así mismo para colocar un valor inicial.

**=>** Este se usa para asignar valores a elementos individuales de un tipo vector u otros **OTHERS**.

Los operadores Lógicos son:

* NOT
* AND
* OR
* NAND
* NOR
* XOR
* XNOR

Para compuertas el código completo es:

**architecture** **behaviorial** **of** nombre\_entidad **is**

**begin**

--código

Z<= not A;

Y<= B and A;

**end** **behaviorial**;