BAB 3 TIPE DATA

I. Tujuan Praktikum :

- Praktikan Dapat Mengenal Tipe Data Pada VHDL.
- Praktikan Dapat Membedakan Penggunaan Tipe Data.

II. Dasar Teori :

- Pengenalan Tipe Data.
- Penggunaan Tipe Data Pada Program VHDL.

III. Peralatan :

- FPGA Xilinx Artix 7 Board Nexys 4.
- Kabel Power USB.
- 1 buah PC.
- Software ISE Design Suite 14.5.

3.1 Pengertian

Tipe data adalah atribut yang dilampirkan ke data untuk menyusun VHDL dalam memahami cara memperlakukan data tersebut. Sederhananya, ini merupakan perintah khusus yang memberi tahu compiler pada VHDL apa yang harus dilakukan pada data tersebut dan bagaimana mengolahnya. Pada VHDL, kita mendefinisikan tipe data pada saat menginisialisasi sinyal, variable, konstanta, dan generic. Untuk menuliskan kode VHDL secara efisien, sangatlah penting untuk mengetahui tipe-tipe data yang diperbolehkan, bagaimana, serta kapan penggunaannya.

3.2 Tipe Data Pada Standard Library

Kode VHDL mengandung sederatan tipe data yang telah ditentukan melalui aturan standar IEEE 1076 dan IEEE 1164. Untuk lebih jelasnya, beberapa tipe data telah tercantum ke dalam masing-masing jenis library/package dan kita kelompokkan menjadi 4 bagian, yaitu :

1. Enumerated type.
2. Numeric type.
3. Array.
4. Variasi lainnya.

3.2.1 Enumerated Type

Tipe data ini dapat diambil dari nilai yang ada pada library standar. Tipe data yang termasuk kedalamnya adalah :

- Boolean, hanya memiliki satu nilai yaitu “TRUE” atau “FALSE”. Operasi yang dapat dilakukan pada variable Boolean dengan gerbang logika adalah gerbang “AND, OR, NOT, NAND, NOR, dan XOR”. Contoh :

  Variable DONE : BOOLEAN := FALSE;
  Signal enable : BOOLEAN := TRUE;

- BIT, tipe data ini hanya dapat memiliki nilai ‘0’ atau ‘1’. Contoh :

  Variable temp : BIT := 0;
  Signal CLK : BIT := 1;

- Character, sesuai dengan namanya tipe data ini dapat menampung karakter baik itu yang biasa maupun khusus. Contoh :

  Variable VAL : character := ‘$’;

  Yang termasuk kedalam tipe data ini :

```
NUL, SOH, STX, ETX, EOT, ENQ, ACK,
BEL, BS, HT, LF, VT, FF, CR,
SO, SI, DLE, DCI, DC2, DC3, DC4,
NAK, SYN, ETB, CAN, EM, SUB, ESC,
FSP, GSP, RSP, USP,
' ' , ' ! ', ' " ', ' # ', ' $ ', ' % ', ' & ',
' ' ', ' (' , ' ) ', ' * ', ' + ', ' , ', ' - ',
' . ', ' / ', ' 0 ', ' 1 ', ' 2 ', ' 3 ', ' 4 ',
' 5 ', ' 6 ', ' 7 ', ' 8 ', ' 9', ' : ', ' ; ',
' < ', ' = ', ' > ', ' ? ',
' @ ', ' A ', ' B ', ' C ', ' D ', ' E ', ' F ',
' G ', ' H ', ' I ', ' J ', ' K ', ' L ', ' M ',
' N ', ' O ', ' P ', ' Q ', ' R ', ' S ', ' T ',
' U ', ' V ', ' W ', ' X ', ' Y ', ' Z ', ' [ ',
' \ ', ' ] ', ' ^ ', ' _ ',
' ` ', ' a ', ' b ', ' c ', ' d ', ' e ', ' f ',
' g ', ' h ', ' i ', ' j ', ' k ', ' l ', ' m ',
' n ', ' o ', ' p ', ' q ', ' r ', ' s ', ' t ',
' u ', ' v ', ' w ', ' x ', ' y ', ' z ', ' { ',
' | ', ' } ', ' ~ ', DEL
```

Gambar 3.1 Kategori tipe data character

3.2.2 Numeric Type

Sesuai dengan namanya, tipe data ini hanya dapat menampung nilai-nilai numerik. Tipe data numerik terdiri dari :

- Integer, tipe data ini dapat menampung nilai dari -2.147.483.647 s/d +2.147.483.647. Selain itu, integer juga memiliki dua buah sub tipe yang sudah di definisikan pada library, yaitu :
  a. Natural, memiliki range nilai dari 0 s/d +2.147.483.647.
  b. Positif, memiliki range 1 s/d +2.147.483.647.

Variable VALUE : natural := 2;

Variable VALUE : positive := 2;

- Real, tipe data ini dapat menampung floating point mulai dari -1.0E38 hingga +1.0E38. Selain itu, tipe data ini juga dapat membaca bilangan pi (π) untuk beberapa perhitungan. Bilangan real dapat menyimpan hingga 6 angka dibelakang koma. Contoh :

  Constant Pi : real := 3.14159;

3.2.3 Array

Array merupakan kumpulan objek dengan tipe yang sama. Ada dua jenis array yang sudah ditentukan dalam library standar VHDL, yaitu :

- String, merupakan tipe data yang menggunakan huruf. Contoh :

  Variable MESSAGE : STRING (1 to 10) := "Acsl";

- Bit vector, merupakan sebuah array dari tipe data BIT yang dikelompokkan. Biasanya digunakan untuk mendefinisikan banyak input pin. Jika kita ingin membuat sebuah rangkaian dengan 4 input, daripada mendefinisikannya satu persatu, kita dapat membuatnya lebih sederhana dengan menggunakan BIT vector ini. Contoh :

  Signal input : BIT_VECTOR (3 downto 0);

  Pernyataan tersebut mendefinisikan input 4 bit. Untuk mengaturnya, kita dapat menggunakan input (0) untuk mengakses bit pertama dan (1) untuk bit kedua, begitu seterusnya. Contoh :

  Input <= "0101";

3.2.4 Tipe Data Lainnya

Salah satu jenis tipe data lainnya adalah yang berkaitan dengan waktu. Ini digunakan untuk menyimpan nilai yang dapat digunakan untuk operasi waktu seperti membuat delay, dan sejenisnya. Beberapa tipe data jenis ini sudah didefinisikan pada library standar seperti dibawah ini :

```
fs;              -- femtosecond
ps = 1000 fs;    -- picosecond
ns = 1000 ps;    -- nanosecond
us = 1000 ns;    -- microsecond
ms = 1000 us;    -- microsecond
sec = 1000 ms;   -- seconds
min = 60 secs;   -- minutes
hr = 60 min;     -- hours
```

Gambar 3.2 Beberapa jenis satuan waktu pada VHDL

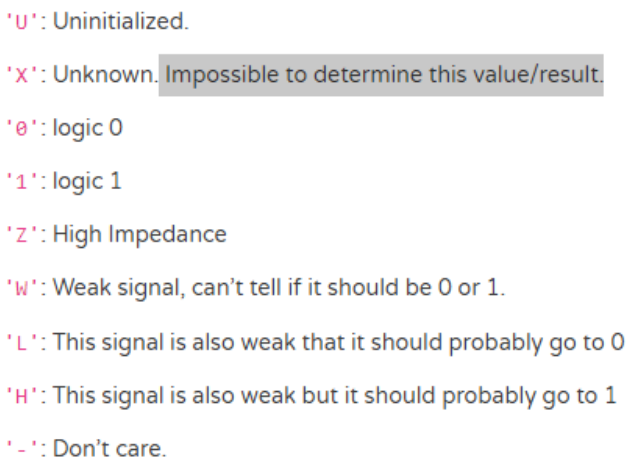
Contoh :

Variable DELAY : time := 5 ns;

3.3 Tipe Data Diluar Dari Standard Library

Tipe data ini dapat mengambil beberapa nilai yang semuanya terdaftar atau disebutkan di library masing-masing. Tipe data ini didefinisikan dalam sebuah library, untuk menggunakannya harus menggunakan perintah "**use IEEE.std_logic-1164.all**. Tipe data dari library yang cukup banyak digunakan yaitu :

- STD_logic, biasanya digunakan untuk merepresentasikan detail dari sinyal digital dalam circuit dan kabel atau penghubungnya. Tipe data yang termasuk kedalam ini adalah :



Gambar 3.3 Tipe data std_logic

Variable temp : std_logic := 1;
Signal CLK : std_logic := 0;

Tipe data ini juga dapat menggunakan array sama seperti sebelumnya. Contoh :

Signal address : std_logic_vector(3 downto 0);

3.4 Contoh Program

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

entity mux is
port (
input : in bit_vector(1 downto 0);
sel : in bit;
output : out bit);
end mux;

architecture behavioral of mux is
begin
process (sel)
begin
case sel is
when "0" => output <= input(0);
when "1" => output <= input(1);
when others => output <= '0';
end case;
end process;
end behavioral;
```

Gambar 3.4 Contoh program

Analisa :

Dari program tersebut, terdapat 3 port yang didefinisikan yaitu 2 input dan 1 output. Kita membutuhkan 2 bit input maka kita dapat mendeklarasikannya dengan :

```
input1, input2 in bit;
```

Untuk memberikan nilai kedua input diatas kita dapat menggunakan dua baris kode yaitu :

```
input1 <= '1';
input2 <= '1';
```

Namun cara ini tidak efektif, kita dapat menggunakan array dengan fungsi menuliskan program *bit_vector* seperti berikut :

```
input : in bit_vector(1 downto 0);
```