BAB 2

Tipe Data

Pada bab ini, praktikan akan mempelajari konsep dasar architecture dalam VHDL, yang digunakan untuk mendeskripsikan perilaku internal dari sebuah entity. Praktikan akan mempelajari cara membuat architecture sederhana dan bagaimana architecture berfungsi dalam merancang desain yang lebih besar dan kompleks. Asisten praktikum atau praktikan diharapkan membaca tujuan dan persyaratan bab ini agar praktikum dapat berjalan sesuai prosedur.

Tujuan

Tujuan	Penjelasan
Mengenal tipe data pada VHDL	Praktikan diharapkan dapat memahami berbagai tipe data yang tersedia dalam VHDL, seperti std_logic, bit, integer, dan boolean, serta bagaimana setiap tipe data berfungsi dalam mendesain rangkaian digital.
Membedakan penggunaan tipe data	Praktikan diharapkan dapat membedakan penggunaan berbagai tipe data sesuai dengan kebutuhan desain, memahami kapan menggunakan tipe data tertentu untuk memodelkan perilaku dan karakteristik rangkaian yang berbeda.
Menggunakan tipe data array untuk membuat architeture	Praktikan diharapkan dapat menggunakan tipe data array dalam VHDL untuk membuat architecture yang lebih kompleks.

Persyaratan

Disarankan praktikan menggunakan hardware dan software sesuai pada dokumentasi ini. Apabila terdapat versi hardware atau software yang cukup lama dari versi yang direkomendasikan maka sebaiknya bertanya kepada Asisten Mengajar Shift.

HARDWARE YANG DIBUTUHKAN PRAKTIKUM

FPGA Board Nexys A7 dan Nexys 4

Kabel Power USB

SOFTWARE YANG DIBUTUHKAN PRAKTIKUM

Vivado Design Suite

⚠ Diusahakan untuk memakai **Versi** dan **Aplikasi** yang sama agar tidak terjadinya kesalahan yang tidak diinginkan!

Teori

2.1. 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, variabel, konstanta, dan generik. Untuk menuliskan kode VHDL secara efisien, sangatlah penting untuk mengetahui tipe-tipe data yang diperbolehkan, bagaimana, serta kapan penggunaannya.

2.2. Tipe Data VHDL

VHDL memiliki seperangkat tipe data standar (*predefined*/*built-in*) dan juga terdapat tipe data dan subtipe yang didefinisikan pengguna (*user defined*). Artinya, dalam VHDL tipe data dapat diklasifikasikan sebagai berikut.

2.2.1. Tipe Data Predefined

VHDL memiliki beberapa tipe data standar (*predefined*) yang sudah didefinisikan sebelumnya oleh bahasa pemrograman. Tipe data ini menyediakan dasar yang kuat untuk membangun berbagai jenis rangkaian digital. Berikut adalah beberapa tipe data standar yang umum digunakan dalam VHDL :

2.2.1.1. Bit

Bit merupakan tipe data paling sederhana dan terpenting untuk sistem digital dan memiliki nilai '0' dan '1'. Setiap objek data dapat dideklarasikan sebagai tipe bit

sebelum digunakan sebagai Sinyal, Variabel, atau Konstanta.

Variable temp : BIT := 0;

Signal CLK : BIT := 1;

2.2.1.2. Boolean

Boolean hanya memiliki satu nilai yaitu "TRUE" atau "FALSE". Operasi yang dapat dilakukan pada variabel Boolean dengan gerbang logika adalah gerbang "AND, OR, NOT, NAND, NOR, dan XOR".

Variable DONE : BOOLEAN := FALSE;

Signal enable : BOOLEAN := TRUE;

2.2.1.3. Numeric

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 didefinisikan 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 dibelakangkoma. Contoh :

  Constant Pi : real := 3.14159;

2.2.1.4. Character

Tipe data ini dapat menampung karakter baik itu yang biasa maupun karakter khusus.

Variable VAL : character := '$';

Semua karakter yang termasuk dalam tipe data ini adalah sebagai berikut :

```
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
```

2.2.2. Tipe Data User Defined

Tipe data *user defined* dalam VHDL memungkinkan pengguna untuk mendefinisikan sendiri tipe data khusus yang sesuai dengan rangkaian yang akan dibuat. Ini memberikan fleksibilitas yang lebih besar dalam memodelkan dan mengelola data dalam desain rangkaian digital**.** Tipe data *user defined* dibagi menjadi 4 tipe yaitu :

2.2.2.1. Tipe Scalar

Tipe scalar adalah jenis data yang hanya dapat menyimpan satu nilai pada satu waktu, maksudnya adalah bahwa sebuah variabel atau sinyal yang dideklarasikan dengan tipe data skalar hanya bisa menyimpan satu nilai tunggal pada saat tertentu, bukan beberapa nilai sekaligus.

- Enumerated, sangat fleksibel dan dapat digunakan untuk merepresentasikan berbagai jenis data yang memiliki jumlah nilai yang terbatas dan terdefinisi. Tipe data ini dapat berupa Bit, Boolean, Numeric dan character.

  signal bit_val : std_logic;

  bit_val <= '1';

- Physical, berisi nilai-nilai yang mewakili pengukuran suatu kuantitas fisik, seperti waktu, panjang, tegangan, dan arus. Nilai-nilai dari tipe ini dinyatakan sebagai kelipatan bilangan bulat dari satuan dasar.

  type CURRENT is range 0 to 1 E-9

  units

  nA; -- (base unit) nano-ampere

  uA = 1000 nA; -- micro-ampere

  mA = 1000 µA; --milli-ampere

  Amp = 1000 mA; -- ampere

  end units;

  dan

  t*ype TIMEis range 0 to -1 E18 to 1 E18*

  units

  pS; -- (base unit) Pico Sec

  nS = 1000pS;

  µS = 1000nS;

  mS = 1000µS;

  Sec= 1000mS

  Min= 60Sec

  end units;

2222 Tipe Composites

Ini adalah tipe data yang digunakan untuk menggabungkan beberapa nilai, selain tipe data standar. memungkin representasi struktur data yang lebih kompleks dalam desain rangkaian digital. tipe data ini memiliki dua macam yaitu :

- Array, digunakan untuk merepresentasikan sebuah objek yang berisi lebih dari satu nilai dengan tipe yang sama. Misalnya, sebuah register delapan bit dapat diwakili sebagai sebuah objek dari tipe array yang terdiri dari nilai-nilai delapan bit. Jadi, delapan nilai dari tipe bit dikelompokkan menjadi satu objek dari tipe array.

  typeADDRESS_WORD is array (0 to 7) of BIT

- *Record*, memungkinkan pengelompokan beberapa elemen yang berbeda jenis menjadi satu entitas tunggal. Dengan kata lain, record memungkinkan kita untuk menggabungkan berbagai tipe data (seperti integer, boolean, std_logic, dll.) dalam satu struktur, yang memudahkan organisasi dan pengelolaan data yang kompleks dalam desain perangkat keras.

  type MODULE is record
  SIZE: INTEGER range 20 to 200;
  CRITICAL_DLY: TIME;
  NO_INPUTS: PIN_TYPE: NO_OUTPUTS:
  PIN_TYPE;
  end record;
  type DATEis record
  DAY: integer_range 1 to 31;
  MONTH:month_name;
  YEAR:integer_range0 to 4000;
  end record;

- Tipe Access

Tipe *Access* digunakan untuk mendeklarasikan *pointer*, yaitu referensi ke objek lain dalam memori. Tipe *data access* memungkinkan akses dinamis ke objek dan data.

variable p : int_ptr; p := new integer;
p.all := 5; -- Mengisi nilai 5 ke memori yang ditunjuk oleh p
report integer'image(p.all); -- Menampilkan nilai yang dirujuk oleh p, yaitu 5

- Tipe File

Tipe File digunakan untuk membaca dan menulis data dari dan ke file eksternal selama proses simulasi. Ini memungkinkan pengujian, debugging, dan pengumpulan data dengan cara yang lebih fleksibel dan terstruktur.

file my_file : text is in "input_data.txt";

2.2.3. Operator

Operator dalam VHDL terdiri dari berbagai berbagai macam operator untuk melakukan operasi matematika, logika, dan perbandingan. Berikut adalah beberapa jenis operator yang umum digunakan dalam VHDL.

2.2.3.1. Operator Aritmatika

- Penjumlahan : +
- Pengurangan : -
- Perkalian : *
- Pembagian : /
- Modulus : mod (menghasilkan sisa pembagian)

2.2.3.2. Operator Logika

- AND: perlu semua operand bernilai TRUE untuk menghasilkan TRUE.
- OR : perlu setidaknya satu operand bernilai TRUE untuk menghasilkan TRUE.
- NOT: membalikkan nilai boolean.

2.2.3.3. Operator Perbandingan

- Sama dengan : =
- Tidak sama dengan : /=
- Lebih besar : >

- Lebih kecil : <
- Lebih besar atau sama dengan : >=
- Lebih kecil atau sama dengan : <=

2.2.3.4. Operator Khusus

- Negasi : - (untuk bilangan)
- Konkatenasi : & (untuk menggabungkan string)
- Akses elemen array : () (untuk mengakses elemen tertentu dalam array)

2.3. Array

Array adalah struktur data yang terdiri dari kumpulan elemen yang memiliki tipe data yang sama, disusun dalam urutan tertentu, dan diakses menggunakan indeks. Dalam konteks FPGA, array digunakan untuk menyimpan dan mengelola data secara efisien.

2.3.1. Jenis Array

Terdapat 2 jenis array yang ditentukan dalam standar VHDL, yaitu:

- String, merupakan karakter yang digunakan untuk menyimpan teks, seperti pesan atau data komunikasi.

  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";

23.2. Karakteristik Array

- Semua elemen dalam array memiliki tipe data yang sama.
- Setiap elemen dalam array diakses menggunakan indeks, yang biasanya dimulai dari 0.
- Ukuran array biasanya ditentukan saat deklarasi dan tidak dapat diubah selama *runtime* atau saat sedang dijalankan.

24. Contoh Program

1. Contoh dibawah ini adalah BIT vector sederhana yang meringkas 2 *input* A.

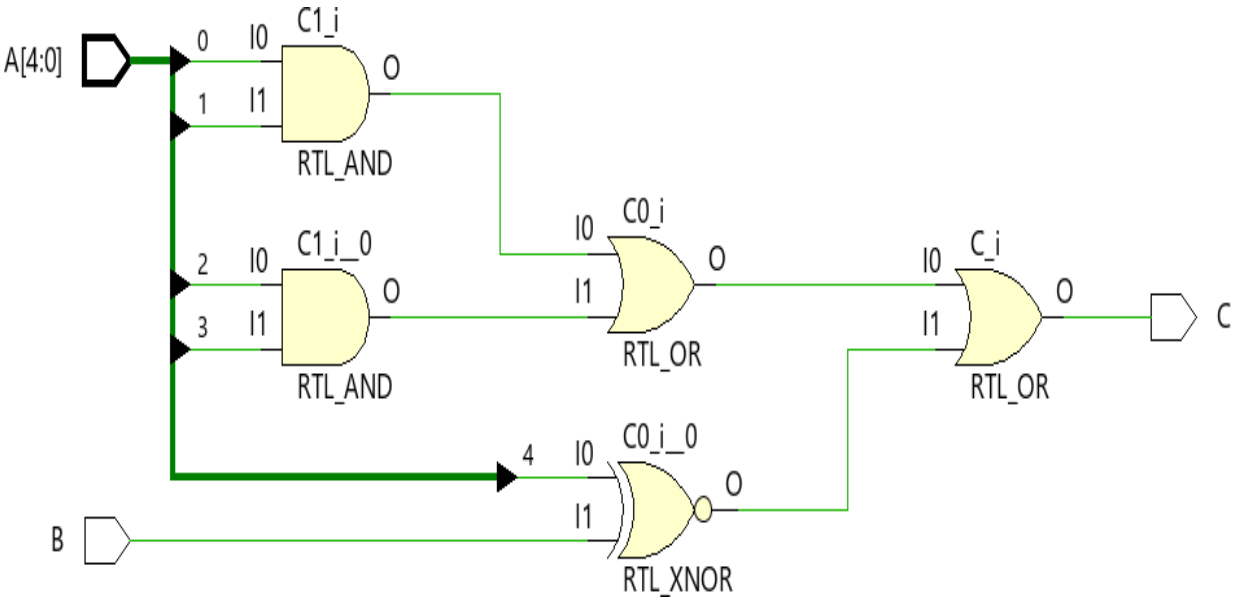
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity bab3array is
    Port ( A : in std_logic_vector (1 downto 0);
           B : in std_logic;
           C : out std_logic);
end bab3array;

architecture Behavioral of bab3array is

begin
C <= (A(0) and A(1))or B;

end Behavioral;
```

2. Contoh dibawah ini memiliki *output* setiap array vector merupakan kebalikan dari *input*-nya.

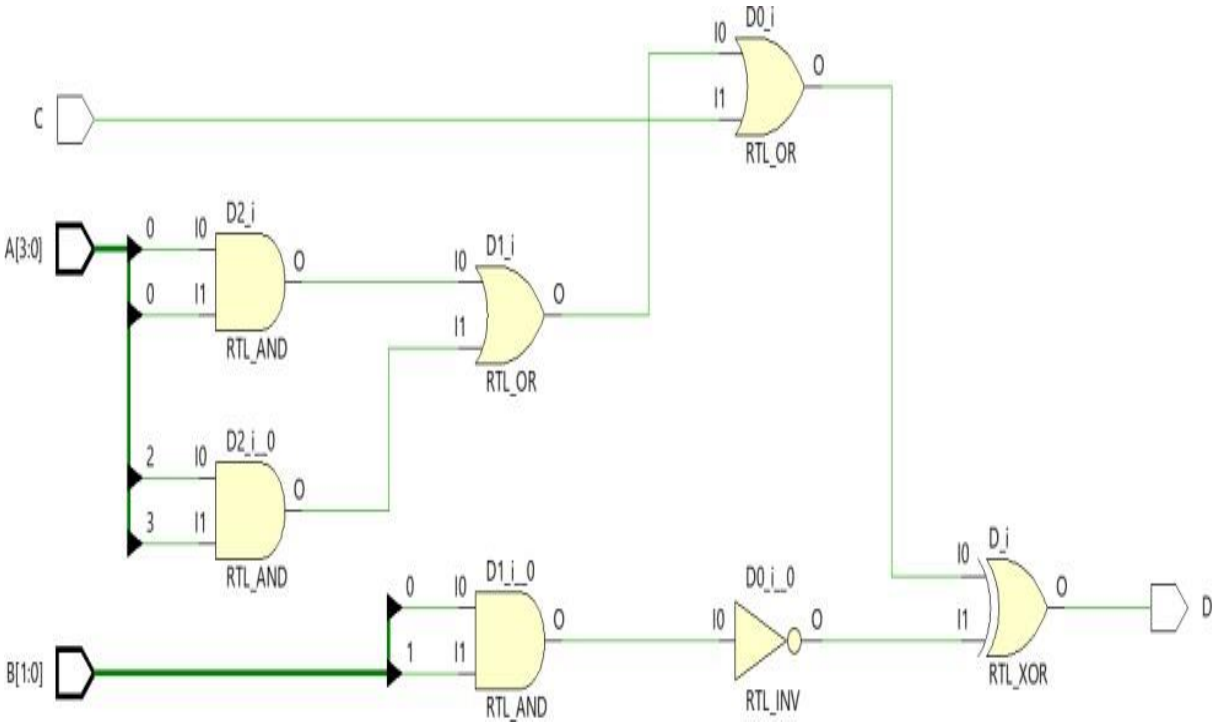
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity bit_vector_example is
    Port (
        input_vector  : in  bit_vector(3 downto 0);
        output_vector : out bit_vector(3 downto 0)
    );
end bit_vector_example;

architecture Behavioral of bit_vector_example is
begin

    process(input_vector)
    begin
        output_vector <= not input_vector;
    end process;

end Behavioral;
```

Soal

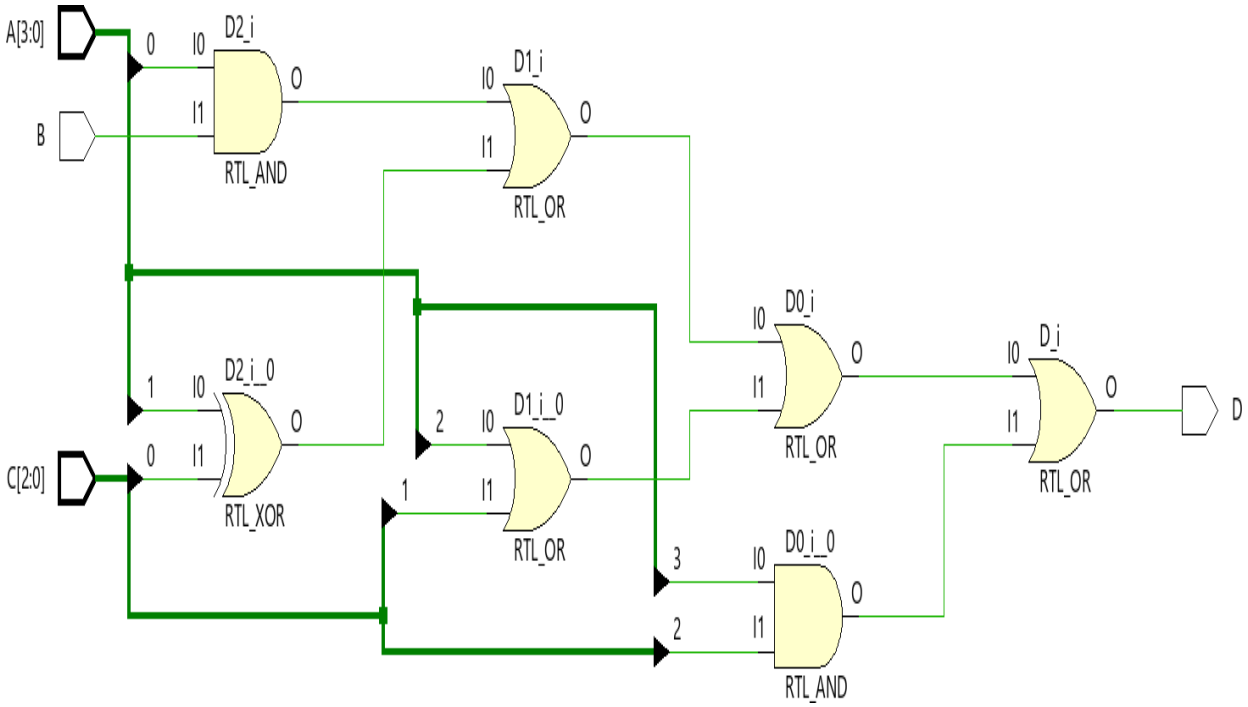
1.


A[4:0]
0
1
I0
I1
C1_i
O
RTL_AND
2
3
C1_i_0
RTL_AND
C0_i
RTL_OR
C_i
RTL_OR
C
4
C0_i_0
RTL_XNOR
B

2.


C
D0_i
RTL_OR
A[3:0]
D2_i
RTL_AND
D1_i
RTL_OR
D2_i_0
RTL_AND
B[1:0]
D1_i_0
RTL_AND
D0_i_0
RTL_INV
D_i
RTL_XOR
D

3.


A[3:0]
B
C[2:0]
D2_i
RTL_AND
D1_i
RTL_OR
D2_i_0
RTL_XOR
D1_i_0
RTL_OR
D0_i
RTL_OR
D0_i_0
RTL_AND
D_i
RTL_OR
D
I0
I1
O