1 D4 - TEKKOM B

PRAKTIKUM 4

IF AND CASE STATEMENT

Nama	:	Septian Bagus Jumantoro
Kelas	:	1D4 - Teknik Komputer B
NRP	:	3221600039
Dosen	:	Reni Soelistijorini B.Eng., MT.
Mata Kuliah	:	Praktikum Rangkaian Logika 2
Hari/Tgl. Praktikum	:	Jumat, 01 April 2022

1. TUJUAN

- Mahasiswa dapat membuat progam dasar VHDL menggunakan if dan case statement.
- Mahasiswa dapat membuat rangkaian logika sederhana menggunakan if dan case statement.
- Mahasiswa dapat membuat RTL dari rangkaian logika.
- Mahasiswa dapat membuat simulasi testbench dari rangkaian logika.
- Mahasiswa dapat membuat Top Module dari rangkaian logika.

2. TEORI

Pemrograman *VHDL* menggunakan *if statement*

Ekspresi *if* adalah salah satu ekspresi kondisional. Pada pemrograman VHDL, ekspresi ini digunakan untuk mendeskripsikan perilaku dari rangkaian logika. Bentuk umum dari statemen *if* adalah sebagai berikut :

```
[label:]                          -- optional label
If <Boolean expression> then
      <sequential statement>
elsif <Boolean expression>then
      <sequential statement>
else
      <sequential statement>
end if[label];                   --optional label
```

If statement adalah contoh sequential statement yang letaknya berada di dalam blok proses pada sebuah pemrograman VHDL

```
[label:]                -- optional label
process     (<sensitivity list>)
--declarations
<variable declarations>
begin
      --ekspresi if statement dituliskan disini
      <sequential statements>
end process[label];
```

Sebuah blok proses diawali dengan :

Process (<sensitivity list>)

Dimana *sensitivity list* berisi semua sinyal yang akan di-*generate* di dalam proses blok.

VHDL* menggunakan *case statement

V*HDL case statement* digunakan untuk mendeskripsikan behavioral dari system digital. Case statement sering digunakan untuk memilih keadaan dengan banyak pilihan berdasarkan kondisi tertentu. Bentuk umum dari case statement adalah

```
[label:]    --optional label
case <expression>is
      when<choices>=><sequential statement>
      when<choices>=><sequential statement>
      ……..
end case[label];
```

Case statement adalah contoh sebuah *sequential statement* yang harus terjadi pada ***process.*** Pada case statement, semua <*choices*> dari <*expression*>harus dimasukkan. Sebuah ekspresi <*expression*> sering menggunakan tipe STD_LOGIC. Selanjutnya untuk pilihan case yang terakhir adalah

when others =><*sequential statement*>

Dimana *Null* juga dapat digunakan pada <sequential statement>. Berikut contoh prioriti encoder dan decoder yang ditulis dalam ***if*** statement dan ***case*** statement.

a. if Statement

Program yang sama prioriti encoder dan decoder yang ditulis dengan *if statement* ditunjukan pada *script* 4.1 dan 4.2. *if statement* harus dimasukan di dalam blok *process.*

```
architecture if_arch of prio_encoder is
begin
      process(r)
      begin
            if (r(4)='1' then pcode <= "100";
            elsif (r(3)='1' then pcode <= "011";
            elsif (r(2)='1' then pcode <= "010";
            elsif (r(1)='1' then pcode <= "001";
            else pcode <= "000";
            end if;
      end process;
end if_arch;
```

Script 4. 1 Priority encoder menggunakan *If Statement*

```
architecture if_arch of decoder_2_4 is
begin
process(en,a)
begin
if (en='0') then y<="0000";
elsif (a="00") then y<= "0001";
elsif (a="01") then y<= "0010";
elsif (a="00") then y<= "0100";
else <= "1000";
end if;
end process;
end if_arch;
```

Script 4. 2 Decoder menggunakan *If Statement*

b. Case Statement

Program yang sama prioriti encoder dan decoder yang ditulis dengan *case statement* ditunjukan *script* 4.3 dan 4.4. Kode-kode tersebut mencantumkan semua kombinasi input yang mungkin dan nilai output yang sesuai.

```
architecture case_arch of prio_encoder is
begin
	process(r)
	begin
		case r is
			when "1000" | "1001" | "1010" | "1011" | "1100" | "1101" | "1110" | "
			1111" =>
				pcode <= "100";
			when "0100" | "0101" | "0110" | "0111" =>
				pcode <= "011";
			when "0010" | "0011" =>
				pcode <= "010";
			when "0001" =>
				pcode <= "001";
			when others =>
				pcode <= "000";

		end case;
	end process;
end case_arch;
```

Script 4. 3 Priority Encoder menggunakan Case statement

```
architecture case_arch of decoder_2_4 is
signal s: std_logoc_vector (2 downto 0);
begin
s <= en & a;
process(a)
begin
case s is
when "000" | "001" | "010" | "01" | =>
y <= "0001";
when "100" =>
y <= "0001";
when "101" =>
y <= "0010";
when "110" =>
y <= "0100";
when others =>
y <= "1000";
end case;
end process;
end case_arch;
```

Script 4. 4 Decoder menggunakan Case Statement

3. ALAT & BAHAN

1. PC yang sudah terinstal ISE 13.1
2. Xilinx Spartan 3
3. Downloader JTAG USB
4. Power Supply 5v

4. LATIHAN

a. Buatlah program decoder 2 to 4 dengan if dan case statement.
b. Buatlah program Multiplex 4x1; 4 bit dengan if statement.
c. Buatlah program simulasinya.
d. Tampilkan hasil timing diagram simulasinya.

5. HASIL PERCOBAAN

1. Decoder 2 to 4 dengan if dan case statement
   a. if Statemenr
      i. Kode VHDL

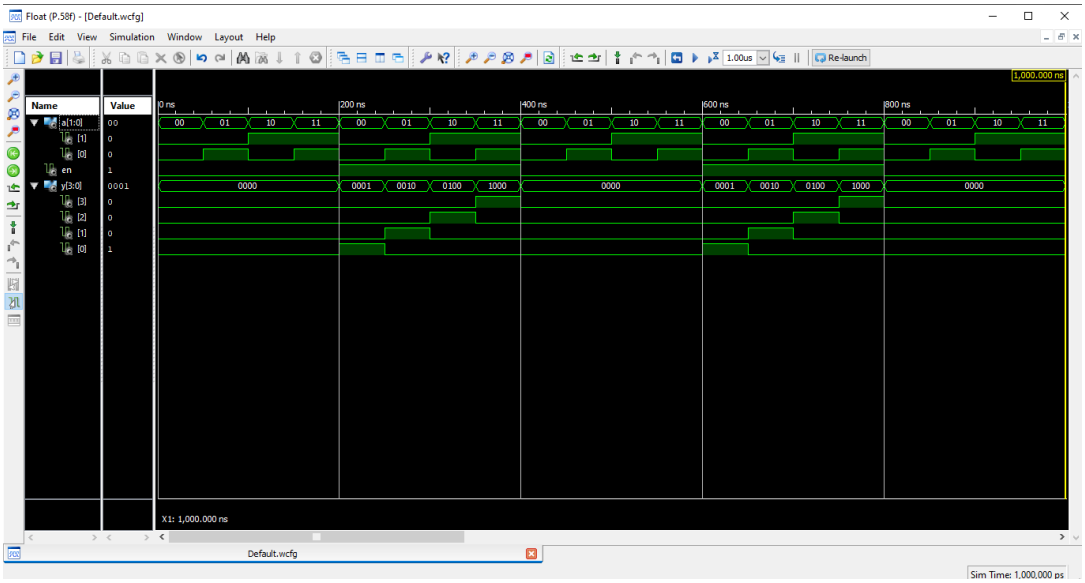
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity kode4a1 is
    port(a: in std_logic_vector(1 downto 0);
          en: in std_logic;
          y: out std_logic_vector(3 downto 0));
end kode4a1;

architecture Behavioral of kode4a1 is
begin
    process(en, a)
    begin
        if (en='0') then y <="0000";
        elsif (a="00") then y <= "0001";
        elsif (a="01") then y <= "0010";
        elsif (a="10") then y <= "0100";
        else y <= "1000";
        end if;
    end process;
end Behavioral;
```

      ii. Kode Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY testLat4a1 IS
END testLat4a1;

ARCHITECTURE behavior OF testLat4a1 IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT kode4a1
    PORT(
         a : IN  std_logic_vector(1 downto 0);
         en : IN  std_logic;
         y : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal a : std_logic_vector(1 downto 0) := (others => '0');
```

```
    signal en : std_logic := '0';

     --Outputs
    signal y : std_logic_vector(3 downto 0);
BEGIN

     -- Instantiate the Unit Under Test (UUT)
    uut: kode4a1 PORT MAP (
            a => a,
            en => en,
            y => y
          );
    -- Stimulus process
    stim_proc_a0: process
    begin
        wait for 50 ns;
          a(0) <= not a(0);
    end process;

     stim_proc_a1: process
    begin
        wait for 100 ns;
          a(1) <= not a(1);
    end process;

     stim_proc_en: process
    begin
        wait for 200 ns;
          en <= not en;
    end process;
END;
```

iii. Timing Diagram



b. Case Statement

i. Kode VHDL

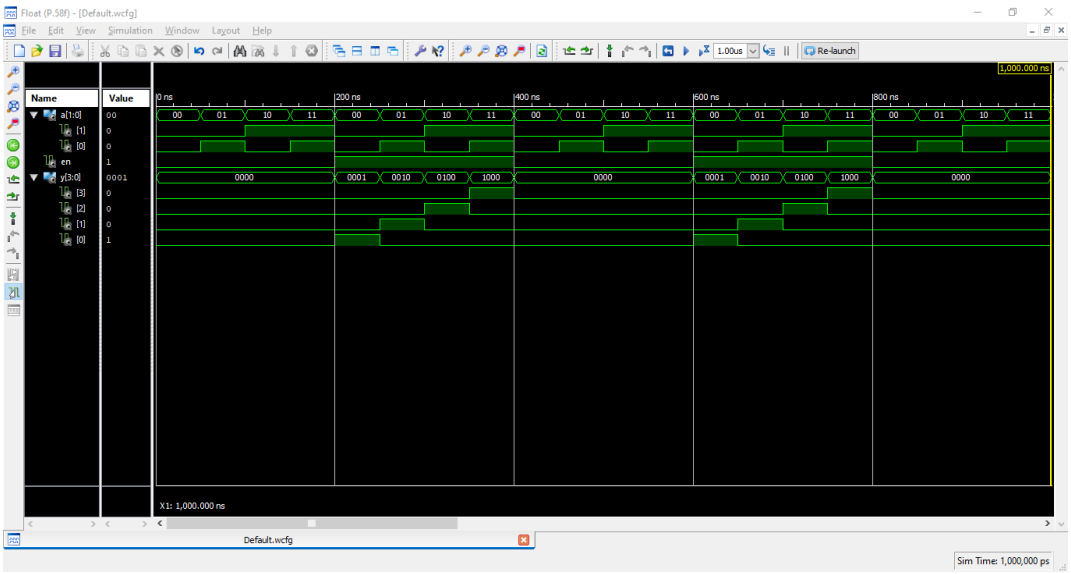
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity kodeLat4a2 is
    port( a: in std_logic_vector(1 downto 0);
            en: in std_logic;
            y: out std_logic_vector(3 downto 0));
end kodeLat4a2;

architecture Behavioral of kodeLat4a2 is
    signal s: std_logic_vector(2 downto 0);
begin
    s <= en & a;
    process(s)
    begin
        case s is
        when "000"|"001"|"010"|"011" =>
            y<= "0000";
        when "100" =>
            y<= "0001";
        when "101" =>
            y<= "0010";
        when "110" =>
            y<= "0100";
        when others =>
            y<= "1000";
        end case;
    end process;
end Behavioral;
```

ii. Kode Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY testLat4a2 IS
END testLat4a2;

ARCHITECTURE behavior OF testLat4a2 IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT kodeLat4a2
    PORT(
         a : IN  std_logic_vector(1 downto 0);
         en : IN  std_logic;
         y : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal a : std_logic_vector(1 downto 0) := (others => '0');
   signal en : std_logic := '0';
```

```
    --Outputs
   signal y : std_logic_vector(3 downto 0);
BEGIN
    -- Instantiate the Unit Under Test (UUT)
   uut: kodeLat4a2 PORT MAP (
          a => a,
          en => en,
          y => y
        );
   -- Stimulus process
   stim_proc_a0: process
   begin
      wait for 50 ns;
        a(0) <= not a(0);
   end process;

    stim_proc_a1: process
   begin
      wait for 100 ns;
        a(1) <= not a(1);
   end process;
```

iii. Timing Diagram



2. Buatlah program Multiplex 4x1; 4 bit dengan if statement.
   a. if Statement
      i. Kode VHDL

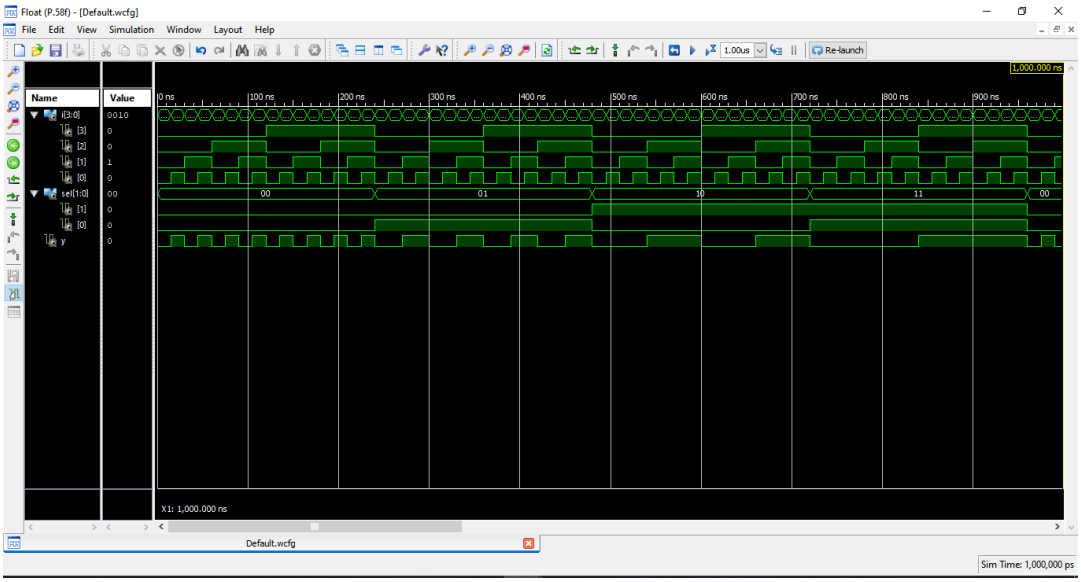
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity kodeLat4b1 is
    port(i: in std_logic_vector(3 downto 0);
         sel: in std_logic_vector(1 downto 0);
         y: out std_logic);
end kodeLat4b1;

architecture Behavioral of kodeLat4b1 is
    signal s: std_logic_vector (1 downto 0);
begin
    s <= sel(1) & sel(0);
    process(s, i)
    begin
        if (s="00") then y <= i(0);
        elsif (s="01") then y <= i(1);
        elsif (s="10") then y <= i(2);
        else y <= i(3);
        end if;
    end process;
```

      ii. Kode test bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test4b IS
END test4b;

ARCHITECTURE behavior OF test4b IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT kodeLat4b1
    PORT(
         i : IN  std_logic_vector(3 downto 0);
         sel : IN  std_logic_vector(1 downto 0);
         y : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal i : std_logic_vector(3 downto 0) := (others => '0');
   signal sel : std_logic_vector(1 downto 0) := (others => '0');

    --Outputs
   signal y : std_logic;
BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: kodeLat4b1 PORT MAP (
```

```
          i => i,
          sel => sel,
          y => y
        );
   -- Stimulus process
   stim_proc_i0: process
   begin
      wait for 15 ns;
         i(0) <= not i(0);
   end process;

    stim_proc_i1: process
   begin
      wait for 30 ns;
         i(1) <= not i(1);
   end process;

    stim_proc_i2: process
   begin
      wait for 60 ns;
         i(2) <= not i(2);
   end process;

    stim_proc_i3: process
   begin
      wait for 120 ns;
         i(3) <= not i(3);
   end process;


    stim_proc_sel0: process
   begin
      wait for 240 ns;
         sel(0) <= not sel(0);
   end process;
    stim_proc_sel1: process
   begin
      wait for 480 ns;
         sel(1) <= not sel(1);
   end process;
END;
```

iii. Timing Diagram



6. ANALISA

Berdasarkan praktikum tersebut diketahui bahwa penggunaan if statement diawali dengan process() sebagai deklarasi nilai. Untuk penulisannya yaitu if a = "00" then y <= "0000"; dimana a merupakan input, dan y merupakan output. Untuk ketentuan selanjutnya menggunakan else lalu dilanjut dengan output yang tersisa.

Lalu untuk penggunaan case statement sendiri hampir sama dengan if statement namun cara penulisannya diawali dengan case s is, selanjutnya when "loo" => y <= "0001". Dengan keterangan s sebagai input dan y sebagai output. Lalu terakhir dapat menggunakan when others.

Pada praktikum multiplexer 4 x 1 berfungsi untuk mengubah output sinyal tergantung dari switch yang ada. Untuk tabel kebenaran sebagai berikut:

S_1	S_0	Output
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

D merupakan data dari setiap sinyal input.
S merupakan switch untuk mengganti sinyal yg ada.
Z merupakan output yang nilainya bergantung pada data yang ada

7. KESIMPULAN

Berdasarkan praktikum tersebut dapat disimpulkan bahwa tiap statement memiliki kelebihan dan kekurangan. Untuk if statement menggunakan penulisan syntax if, elsif, else yang lebih sedikit daripada case statement.

8. REFERENSI

1. FPGA Prototyping by VHDL Example. Xilinx Spartan-3. Pong P. Chu
2. Haskell, Richard and Darrin, Learning By Example Using VHDL Advanced Digital Design With a NEXYS 2TM FPGA Board, 2009,Oakland university Rochester, Michigan, ISBN 978-0-9801337-4-5.