1 D4 - TEKKOM B

FOR LOOP STATEMENT

Nama	: Septian Bagus Jumantoro
Kelas	: 1 – D4 Teknik Komputer B
NRP	: 3221600039
Dosen	: Reni Soelistijorini B.Eng., MT.
Mata Kuliah	: Praktikum Rangkaian Logika 2
Hari/Tgl. Praktikum	: Jumat, 22 April 2022

Percobaan 5 – For Loop Statement

1. Tujuan

Dapat melakukan pemrogaman dasar VHDL menggunakan *for statement*

Dapat membuat rangkaian logika sederhana menggunakan *for statement*

2. Teori

***For loop* Statement**

VHDL for loop digunakan untuk mendeskripsikan behavior dari beberapa system digital. Ekspresi forloop digunakan untuk sebuah proses perulangan. Bentuk umum dari foor loop adalah

```
[label:] -- optional label
For <identifier> in <range> loop
      <sequential statement>
end loop[label]; -- optional label
```

Dalam *for* loop statement iterasi *counter* dan *range* di spesifikasikan. *Statement* yang berada di dalam *loop body* dieksekusi sedangkan *counter* berada dalam *range* tertentu. Setelah iterasi, *counter* ditugaskan ke nilai selanjutnya dari range. *Range* nya bisa jadi *ascending* (urutan naik), dispesifikasikan oleh kata kunci *to*, atau *descending* (urutan turun) yang dispesifikasikan oleh kata kunci ***downto.*** *Range* ini dapat juga di spesifikasikan sebagai tipe atau *sub*-tipe pencacahan, ketika batasan *range* tidak disebutkan secara spesifik ***for loop*** *statement*. Batasan *range* ditentukan oleh *compiler* dari tipe atau *sub*-tipe deklarasi.

for loop *statement* di sript 5.1 menghitung kuadrat dari nilai integer antara 1 sampai 10, dan menyimpan mereka dalam array ***i_square***.

```
for i in 1 to 10 loop
      i_square (i) <= i * i;
end loop;
```

Script 5. 1 for loop statement

pada contoh ini, iterasi *count* secara default bertipe ***integer***, karena tipe mereka belum didefinisikan secara eksplisit. Bentuk lengkap dari *domain* deklarasi untuk iterasi *count* adalah sama dengan tipenya:

for i **in** integer **range** 1 t o 10 **loop**

Di beberapa bahasa pemrogaman, dalam nilai *loop* bisa jadi ditugaskan untuk iterasi *count* (dalam contoh sebelumnya, i). bahasa VHDL, bagaimanapun, tidak mengijinkan untuk memberi nilai pada iterasi *count* atau untuk menggunakan nya sebagai *input* atau *output* parameter dari sebiah prosedur. *Counter* bisa jadi digunakan di sebuah ekspresi asalkan nilainya tidak dimodifikasi. Aspek lain berhubungan dengan iterasi *count* yang tidak membutuhkan mendeklarasikan nya secara eksplisit di dalam prosesnya. *Counter* ini dideklarasikan secara implisit sebagai variable lokal dari ***loop*** *statement* Jika ada variable lain dengan nama yang sama dalam prosesnya, mereka akan diperlakukan sebagai bariabel pemisah.

Interpretasi sintetis dari ***for loop*** *statement* adalah bahwa *copy* baru digenerasi untuk *circuit* yang dideskripsikan oleh *statement* di setiap iterasi dari *loop.* Penggunaan ***for loop*** *statement* untuk me-generate *circuit* yang diilustrasikan pada script 5.2

```
entity match_bits is
 port( a, b: in bit_vector (7 downto 0);
       matches: out bit_vector (7 downto 0));
end match_bits;

architecture functional of match_bits is
begin

 process (a, b)
    begin

      for i in 7 downto 0 loop
         matches (i) <= not (a(i) xor b(i));
      end loop;

 end process;
end functional ;
```

Script 5. 2 for loop statement match bits

Proses dari contoh sebelumnya yang me-generate satu set 1-bit *comparator* untuk membandingkan bit dari urutan vector a dan b yang sama. Hasilnya disimpan di dalam vector yang cocok, yang akan bernilai ‘1’ dimanapun bit dari dua vektor cocok dan ‘0’ untuk kondisi selain itu.

proses dari contoh sebelumnya setara dengan proses berikut:

```
process (a, b)
begin

 matches (7) <= not (a(7) xor b(7));
 matches (6) <= not (a(6) xor b(6));
 matches (5) <= not (a(5) xor b(5));

 matches (4) <= not (a(4) xor b(4));
 matches (3) <= not (a(3) xor b(3));
 matches (2) <= not (a(2) xor b(2));
 matches (1) <= not (a(1) xor b(1));
 matches (0) <= not (a(0) xor b(0));

end process;
```

a. Decoder 3 to 8

Ekspresi for loop untuk 3-to-8 Decoder

Tabel 5. 1 *: 3 to 8 Decoder*

A2	A1	A0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0		0	0	0	0	0	0	0
0	0	1	0		0	0	0	0	0	0
0	1	0	0	0		0	0	0	0	0
0	1	1	0	0	0		0	0	0	0
1	0	0	0	0	0	0		0	0	0
1	0	1	0	0	0	0	0		0	0
1	1	0	0	0	0	0	0	0		0
1	1	1	0	0	0	0	0	0	0	

```
                    y(i)<= '0';
                end if;
            end loop;
        end process;
end Decode_for;
```

Script 5. 3 *Decoder 3 to 8*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.all;
use IEEE.STD_LOGIC_unsigned.all;


entity Decode_for is
      port(
            a :inSTD_LOGIC_VECTOR(2 downto 0);
            y :outSTD_LOGIC_VECTOR(0 downto 7)
      );
end Decode_for;


architecture Behavioral of Decode_for is
begin
      process(a)
         variable j :integer;
      begin
            j := conv_integer(a);
            for i in 0 to 7loop
                  if(i = j) then
                        y(i)<= '1';

                  else
```

b. Binary to BCD Converter

Shift and Add 3 Algorithm

Salah satu cara untuk melakukan konversi biner ke BCD adalah dengan metode shift and Add 3 Algorithm, yang step-stepnya adalah sebagai berikut :

a. Menggeser satu bit bilangan biner.
b. Jika 8 pergeseran telah terjadi, jumlah BCD adalah di kolom ratusan, puluhan dan satuan.
c. Jika nilai biner dalam salah satu kolom BCD adalah lebih besar dari 4, tambahkan 3 pada nilaidi kolom tersebut.
d. Kembali ke step a

Algoritma secara detail dapat diilustrasikan pada tabel 5.2.

Tabel 5. 2 ***: Algoritma shift and add 3 binary to BCD***

Operation	Hundreds	Tens	Units	Binary	
B				**7 4**	**3 0**
HEX				**F**	**F**
Start				**1 1 1 1**	**1 1 1 1**
Shift 1			**1**	**1 1 1 1**	**1 1 1**
Shift 2			**1 1**	**1 1 1 1**	**1 1**
Shift 3			**1 1 1**	**1 1 1 1**	**1**
Add 3			**1 0 1 0**	**1 1 1 1**	**1**
Shift 4		**1**	**0 1 0 1**	**1 1 1 1**	
Add 3		**1**	**1 0 0 0**	**1 1 1 1**	
Shift 5		**1 1**	**0 0 0 1**	**1 1 1**	
Shift 6		**1 1 0**	**0 0 1 1**	**1 1**	
Add 3		**1 0 0 1**	**0 0 1 1**	**1 1**	
Shift 7	**1**	**0 0 1 0**	**0 1 1 1**	**1**	
Add 3	**1**	**0 0 1 0**	**1 0 1 0**	**1**	
Shift 8	**1 0**	**0 1 0 1**	**0 1 0 1**		
BCD	**2**	**5**	**5**		
P	**9 8**	**7 4**	**3 0**		
Z	**17 16**	**15 12**	**11 8**	**7 4**	**3 0**

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;

entity Bin2BCD is
      port(
            b:in STD_LOGIC_VECTOR(7 downto 0);
            p:out STD_LOGIC_VECTOR(9 downto 0)
      );
end Bin2BCD;

architecture Behavioral of Bin2BCD is
begin
      bcd1:process(b)
      variable z:STD_LOGIC_VECTOR(17 downto 0);
      begin
            for i in 0 to 17 loop
                  z(i):='0';
            endloop;
            z(10 downto 3):=b;
            for i in 0 to 4 loop
                  if z(11 downto 8)> 4 then
                        z(11downto8):= z(11 downto 8)+3;
                  endif;
                  if z(15 downto 12)>4 then
                        z(15 downto 12):= z(15 downto 12)+3;
                  endif;
                  z(17 downto 1):= z(16 downto 0);
            end loop;
            p <= z(17 downto 8);
      end process bcd1;
end Behavioral;
```

Script 5. 4 ***Ekspresi for untuk biner to BCD Decoder menggunakan shift and Add 3 Algorithm***

c. Gray Code Converter - Biner to Gray

Berikut ini adalah hubungan antara 3 bit binary dengan 3 bit gray code.

Binary code {0..7} : {000, 001, 010, 011, 100, 101, 110, 111}

Gray code {0..7} : {000, 001, 011, 010, 110, 111, 101, 100}

Konversi Biner ke Gray :

- Salin semua bit

- untuk setiap i : ***g(i) = b(i+1)^b(i)***

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity BinToGray is
	port(
		b:in STD_LOGIC_VECTOR(3 downto 0);
		g:out STD_LOGIC_VECTOR(3 downto 0)
	);
end BinToGray;

architecture Behavioral of BinToGray is
begin
	process(b)
	begin
		g(3)<= b(3);
		for i in 2 downto 0 loop
			g(i)<= b(i+1) xor b(i);
		end loop;
	end process;
end Behavioral;
```

Script 5. 5 *Konversi Biner ke Gray code*

d. Gray Code Converter - Gray to Biner

Konversi Gray ke Biner :alin semua bit

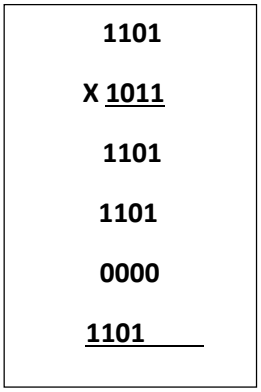
- Untuk setiap i terkecil : ***b(i)= b(i+1) ^ g(i)***

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

entity GrayToBin is
	port(
		g:in STD_LOGIC_VECTOR(3 downto 0);
		b:out STD_LOGIC_VECTOR(3 downto 0)
	);
end GrayToBin;

architecture Behavioral of GrayToBin is
begin
	process(g)
		variable bv :STD_LOGIC_VECTOR(3 downto 0);
	begin
		bv(3) := g(3);
		for i in 2 downto 0 loop
			bv(i) := bv(i+1) xor g(i);
		end loop;
		b <= bv;
	end process;
end Behavioral;
```

Script 5. 6 *Konversi Gray Code ke Biner*

e. Multiplier

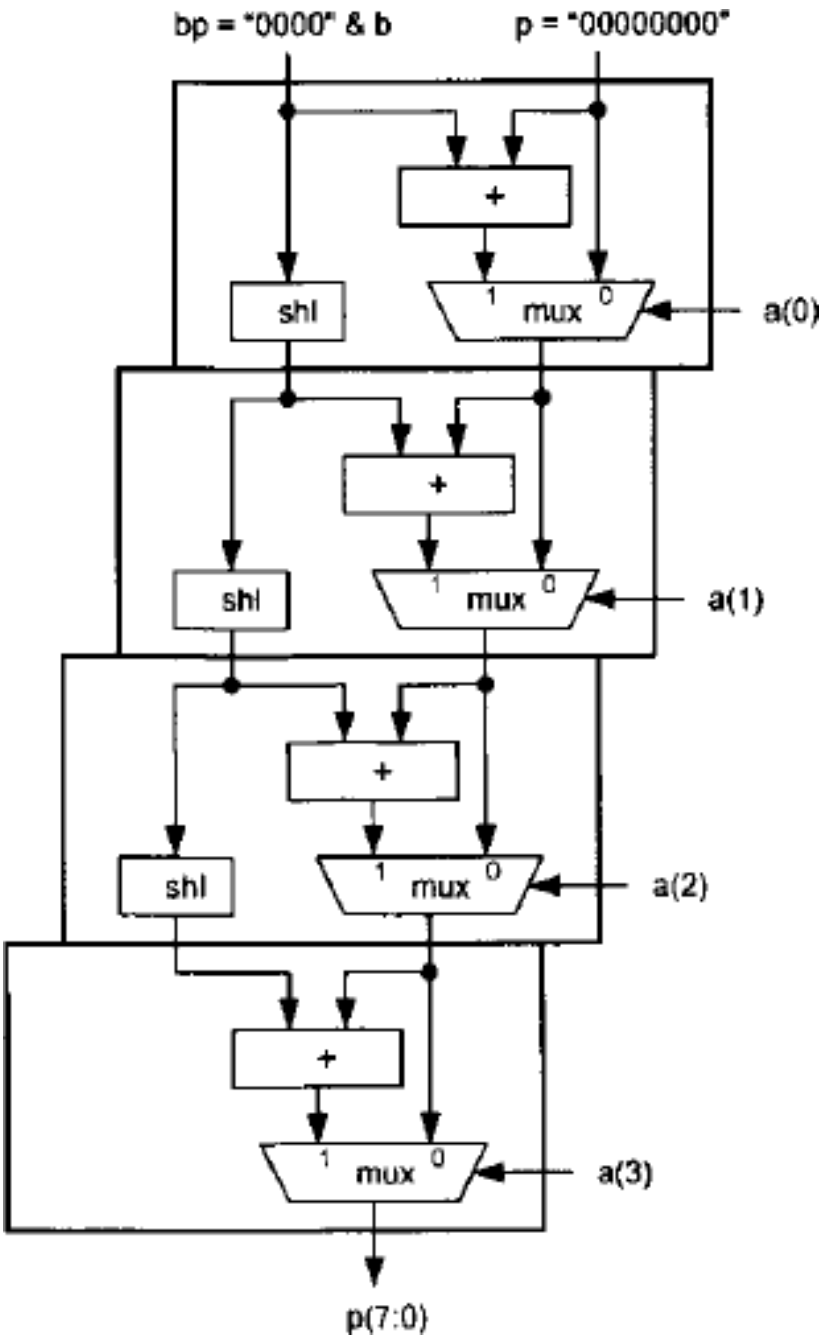
Fungsi loop juga bisa digunakan untuk fungsi multiplier. Pada script 5.7 adalah aplikasi loop untuk perkalian 4 bit dan hasilnya ditampung dalam 8 bit.



Gambar 5. 1 *Binary multiplication*

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.all;


entity MulShift is
      port(
            a:in STD_LOGIC_VECTOR(3 downto 0);
            b:in STD_LOGIC_VECTOR(3 downto 0);
            p:out STD_LOGIC_VECTOR(7 downto 0)
      );
end MulShift;


architecture Behavioral of MulShift is
begin
      process(a,b)
      variable pv,bp :STD_LOGIC_VECTOR(7 downto 0);
      begin
            pv :="00000000";
            bp :="0000"& b;
            for i in 0 to 3 loop
                  if a(i)='1' then
                        pv := pv + bp;
                  endif;
                  bp := bp(6 downto 0) & '0';
            endloop;
      p <= pv;
      endprocess;
 end Behavioral ;
```

Script 5. 7 Multiplier dengan operator (&)



Gambar 5. 2 Logika Diagram Multiplier

Untuk menggunakan multiplication operator(*) kita harus menggunakan tambahan library :

use *IEEE.STD_LOGIC_****unsigned****.all;*

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 entity Mul is
       port(
              a:in STD_LOGIC_VECTOR(3downto0);
              b:in STD_LOGIC_VECTOR(3downto0);
              p:out STD_LOGIC_VECTOR(7downto0)
       );
 end Mul;

 architecture Behavioral of Mul is
begin
              p <= a*b;

 end Behavioral;
```

Script 5. 8 *Multiplier dengan operator (*)*

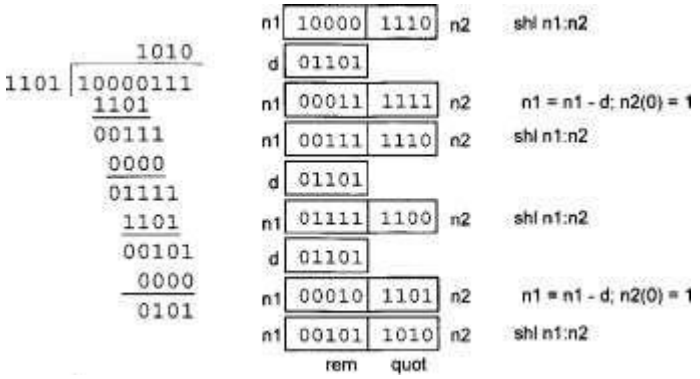
```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.all;


entity MulFunc is
      port(
            sw:in STD_LOGIC_VECTOR(7 downto 0);
            ld:out STD_LOGIC_VECTOR(7 downto 0)
      );
end MulFunc;

architecture Behavioral of MulFunc is

function mul(a,b :in STD_LOGIC_VECTOR)
            return STD_LOGIC_VECTOR is
variable pv,bp :STD_LOGIC_VECTOR(7 downto 0);


begin
      pv :="00000000";
      bp :="0000"& b;
      for i in 0 to 3 loop
            if a(i)='1' then
                  pv := pv + bp;
            endif;
            bp := bp(6 downto 0)& '0';
      end loop;
      return pv;
end function;
```

```
signal a1, a2:STD_LOGIC_VECTOR(3 downto 0);
begin

        a1 <= sw(7 downto 4);
        a2 <= sw(7 downto 4);
        ld <= mul(a1,a2);
```

Script 5. 9 *Multiplication Function*

f. Divider

Ekspresi for loop untuk fungsi divider :



Gambar 5. 3 Contoh fungsi pembagian

Algoritma divider sebagai berikut :

1. Simpan pembilang dalam gabungan dari *n1:n2*
2. Simpan denominator di *d*
3. Ulangi 4 kali :

   Geser n1:n2 kekiri 1 bit

   If n1>d

   n1 = n1 – d;

   n2(0) =1;
4. quot = n2;

   rem = n1(3:0);

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_unsigned.all;
```

```
            numer  :in STD_LOGIC_VECTOR(7 downto 0);
            denom :in  STD_LOGIC_VECTOR(3 downto 0);
            quotient :out STD_LOGIC_VECTOR(3 downto 0);
            remainder  :out STD_LOGIC_VECTOR(3 downto 0)

      );
end Devide;


architecture Behavioral of Devide is
begin

      process(numer,denom)
      variable d, n1 :STD_LOGIC_VECTOR(4 downto 0);
      variable n2 :STD_LOGIC_VECTOR(3 downto 0);
      begin

            d   := '0'  & denom;
            n2  := numer(3 downto 0);
            n1  := '0'  & numer  (7 downto 4);
            for i in 0 to 3 loop

                  n1 := n1(3 downto 0)& n2(3);
                  n2 := n2(2 downto 0)& '0';
                  if n1 >= d then

                        n1 :=  n1-d;
                        n2(0) := '1';

                  end if;
            end loop;

            quotient <= n2;
            remainder  <= n1(3 downto 0);
      end process;

end Behavioral ;
```

Script 5. 10 *Divider*

3. Alat dan Bahan

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

4. Langkah Percobaan

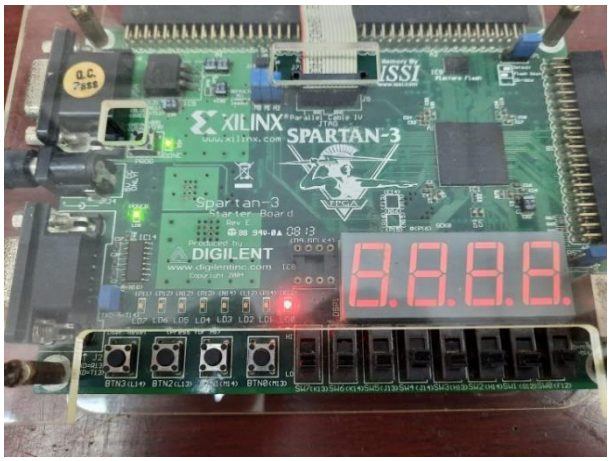
1. Tuliskan program perkalian dan pembagian.
2. Dengan menggunakan *function*, buatlah program sesuai dengan persamaan x = (a * b)/2.
3. Tampikan RTL Schematicnya. Buatlah program simulasinya.
4. Tampilkan hasil simulasi, jika nilai a=7; b=4

5. Hasil Percobaan

a) Decoder 3 to 8

Pin Layout

```
NET a(0) LOC=F12;
NET a(1) LOC=G12;
NET a(2) LOC=H14;

NET y(0) LOC=K12;
NET y(1) LOC=P14;
NET y(2) LOC=L12;
NET y(3) LOC=N14;
NET y(4) LOC=P13;
NET y(5) LOC=N12;
NET y(6) LOC=P12;
NET y(7) LOC=P11;
```



Gambar 1:Decoder 3 to 8 dengan input $0000_2 = 0_{10}$

Gambar 2:Decoder 3 to 8 dengan input $1000_2 = 4_{10}$

b) Binary to BCD Converter

Pin Layout

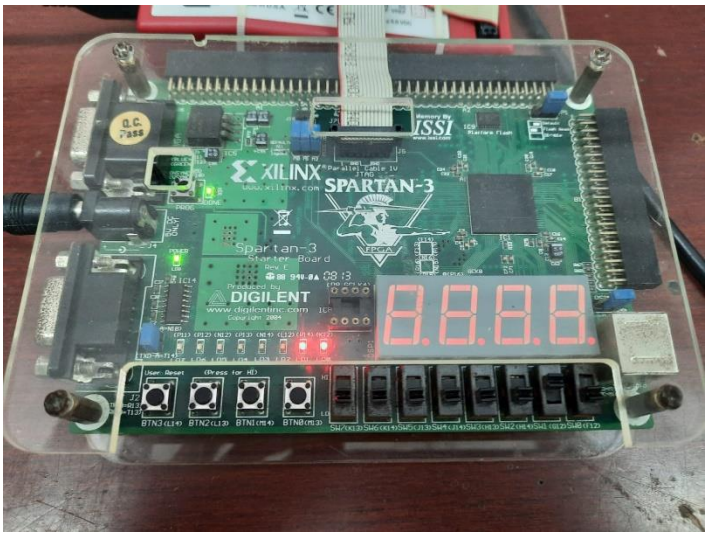
```
1  NET b(0) LOC=F12;
2  NET b(1) LOC=G12;
3  NET b(2) LOC=H14;
4  NET b(3) LOC=H13;
5  NET b(4) LOC=J14;
6  NET b(5) LOC=J13;
7  NET b(6) LOC=K14;
8  NET b(7) LOC=K13;
9
10 NET p(0) LOC=K12;
11 NET p(1) LOC=P14;
12 NET p(2) LOC=L12;
13 NET p(3) LOC=N14;
14 NET p(4) LOC=P13;
15 NET p(5) LOC=N12;
16 NET p(6) LOC=P12;
17 NET p(7) LOC=P11;
```

Gambar 3:Biner to BCD dengan input 0001 $1111_2 = 31_{10}$

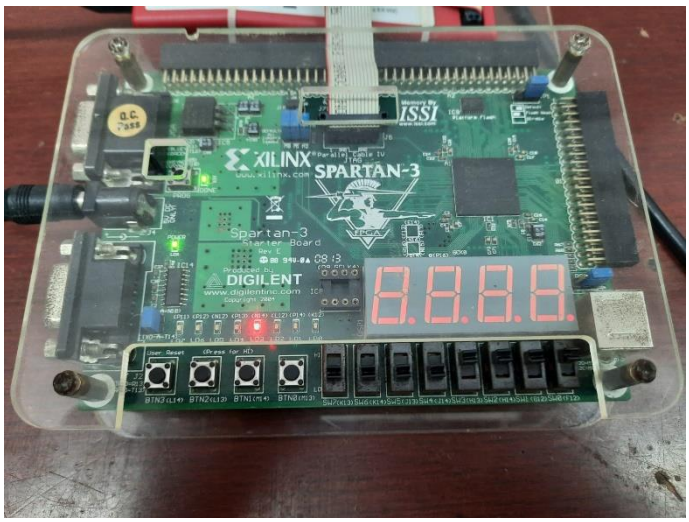
Gambar 4:Biner to BCD dengan input 1111 $1111_2 = 255_{10}$

c) Biner to Gray Converter
Pin Layout

```
1  NET b(0) LOC = F12;
2  NET b(1) LOC = G12;
3  NET b(2) LOC = H14;
4  NET b(3) LOC = H13;
5
6  NET g(0) LOC = K12;
7  NET g(1) LOC = P14;
8  NET g(2) LOC = L12;
9  NET g(3) LOC = N14;
```



Gambar 5:biner to grey dengan input 0010



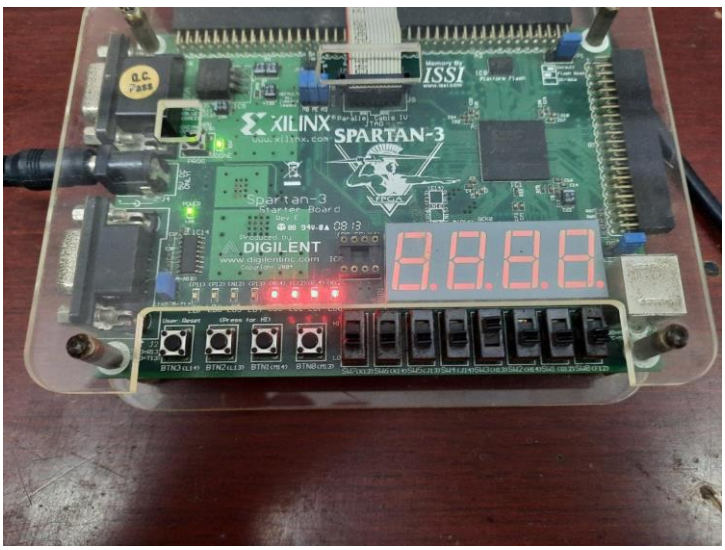
Gambar 6:biner to grey dengan input 1111

d) Gray to Biner Converter

Pin layout

```
1  NET g(0) LOC = F12;
2  NET g(1) LOC = G12;
3  NET g(2) LOC = H14;
4  NET g(3) LOC = H13;
5
6  NET b(0) LOC = K12;
7  NET b(1) LOC = P14;
8  NET b(2) LOC = L12;
9  NET b(3) LOC = N14;
```

Gambar 7:Gray to Biner dengan input 0011

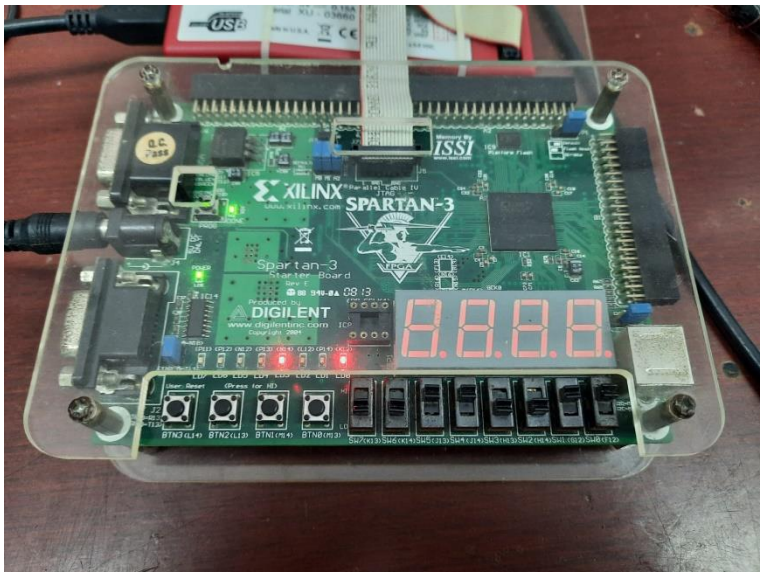


Gambar 8:Gray to Biner dengan input 1000

e) Multiplier

Pin layout

```
1  NET a(0) LOC=F12;
2  NET a(1) LOC=G12;
3  NET a(2) LOC=H14;
4  NET a(3) LOC=H13;
5
6  NET b(0) LOC=J14;
7  NET b(1) LOC=J13;
8  NET b(2) LOC=K14;
9  NET b(3) LOC=K13;
10
11 NET p(0) LOC=K12;
12 NET p(1) LOC=P14;
13 NET p(2) LOC=L12;
14 NET p(3) LOC=N14;
```



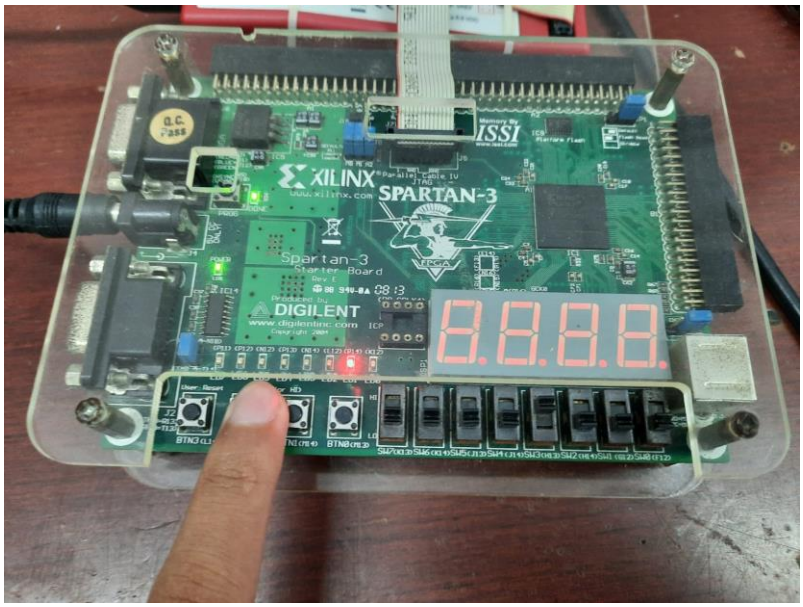
Gambar 9:Multiplier dengan input $0011_2 * 0011_2 = 9_{10}$

Gambar 10:Multiplier dengan input $0100_2 * 0100_2 = 16_{10}$

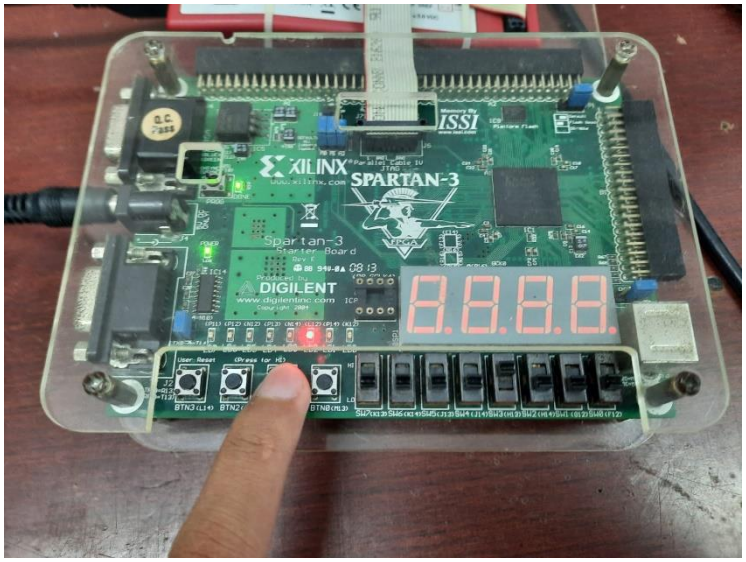
f) Divider

Pin Layout

```
NET numer(0) LOC=F12;
NET numer(1) LOC=G12;
NET numer(2) LOC=H14;
NET numer(3) LOC=H13;
NET numer(4) LOC=J14;
NET numer(5) LOC=J13;
NET numer(6) LOC=K14;
NET numer(7) LOC=K13;

NET denom(0) LOC=M13;
NET denom(1) LOC=M14;
NET denom(2) LOC=L13;
NET denom(3) LOC=L14;

NET quotient(0) LOC=K12;
NET quotient(1) LOC=P14;
NET quotient(2) LOC=L12;
NET quotient(3) LOC=N14;

NET reminder(0) LOC=P13;
NET reminder(1) LOC=N12;
NET reminder(2) LOC=P12;
NET reminder(3) LOC=P11;
```



Gambar 11:Divider dengan input $1000_2 : 0100_2 = 0010_2$

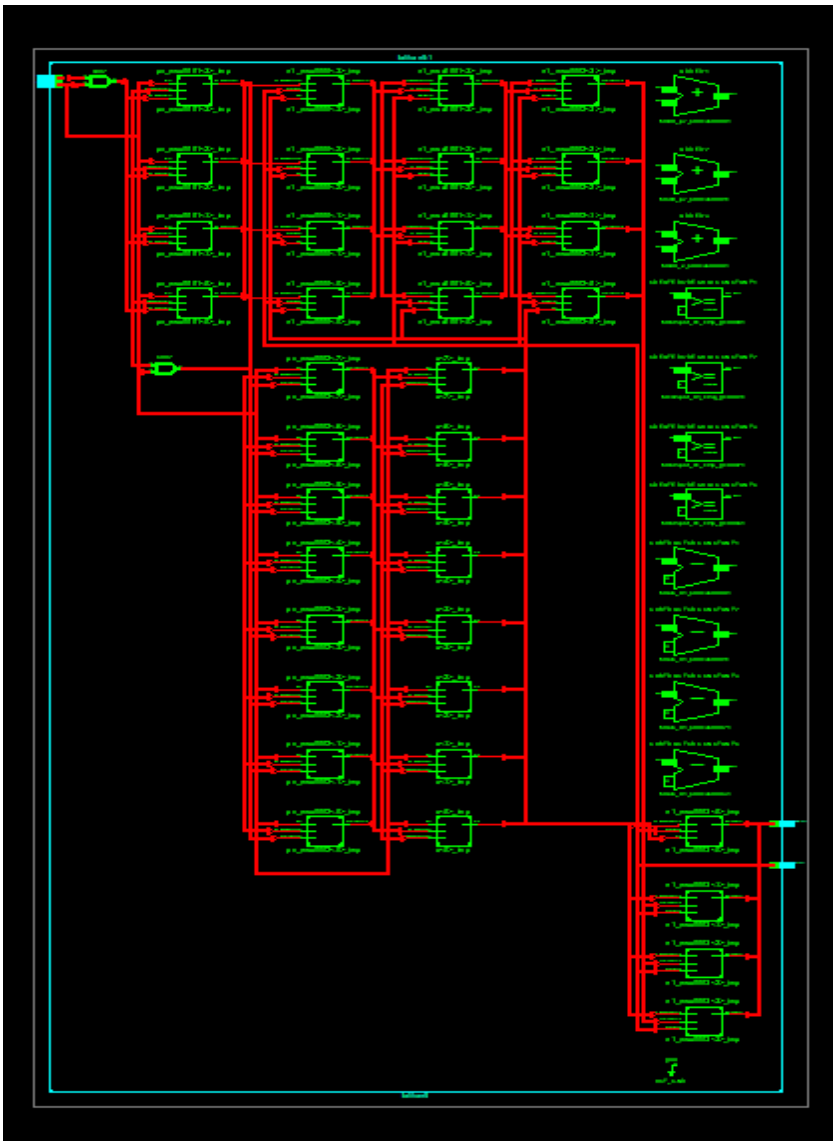


Gambar 12:Divider dengan input $1000_2 : 0010_2 = 0100_2$

g) Program Persamaan x = (a*b)/2
Kode Program

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

entity latihan5 is
port( a,b: IN STD_LOGIC_VECTOR(3 downto 0);
            quotient :out STD_LOGIC_VECTOR(3 downto 0);
            reminder :out STD_LOGIC_VECTOR(3 downto 0)
      );
end latihan5;

architecture Behavioral of latihan5 is
signal x: STD_LOGIC_VECTOR(7 downto 0);
signal z: STD_LOGIC_VECTOR(3 downto 0);
begin
      process(a,b)
      variable pv,bp: STD_LOGIC_VECTOR(7 downto 0);
      begin
            pv := "00000000";
            bp := "0000" & b;
            for i in 0 to 3 loop
                  if a(i) = '1' then
                        pv := pv + bp;
                  end if;
                  bp:= bp(6 downto 0) & '0';
            end loop;
            x <= pv;
      end process;
      z <= "0010";


      process(x,z)
      variable d, n1 :STD_LOGIC_VECTOR(4 downto 0);
      variable n2 :STD_LOGIC_VECTOR(3 downto 0);
      begin
            d  := '0' & z ;
            n2 := x(3 downto 0);
            n1 := '0' & x (7 downto 4);
            for i in 0 to 3 loop
                  n1 := n1(3 downto 0)& n2(3);
                  n2 := n2(2 downto 0)& '0' ;
                  if n1 >= d then n1 := n1-d;
                        n2(0):= '1';
                  end if;
            end loop;
      quotient <= n2;
      reminder <= n1(3 downto 0);
      end process;
end Behavioral;
```

RTL Skematik



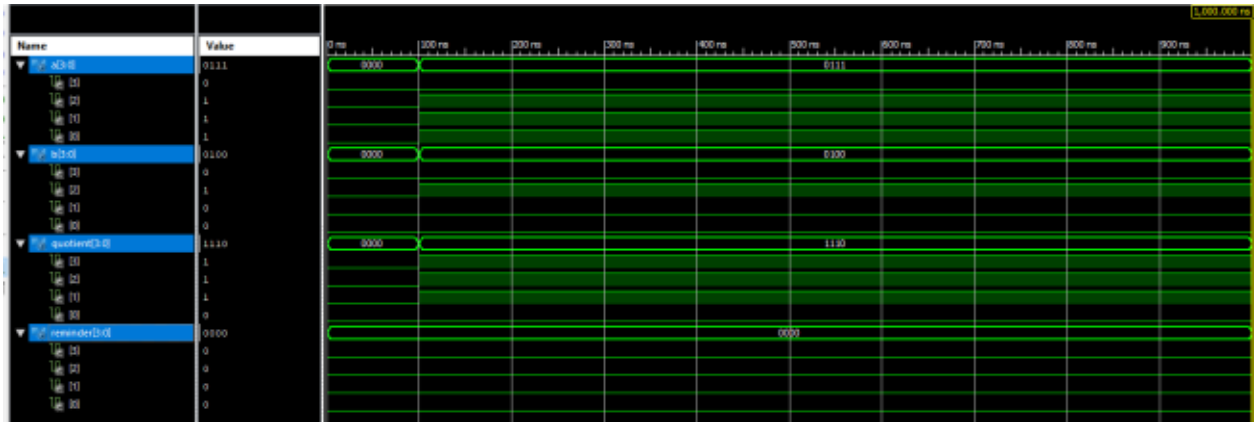
Gambar 13:RTL Skematik

Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY latihantb IS
END latihantb;
ARCHITECTURE behavior OF latihantb IS
COMPONENT latihan5
    PORT(
         a : IN  std_logic_vector(3 downto 0);
         b : IN  std_logic_vector(3 downto 0);
         quotient : OUT std_logic_vector(3 downto 0);
         reminder : OUT  std_logic_vector(3 downto 0));
    END COMPONENT;
```

```
   --Inputs
   signal a : std_logic_vector(3 downto 0) := (others => '0');
   signal b : std_logic_vector(3 downto 0) := (others => '0');


        --Outputs
   signal quotient : std_logic_vector(3 downto 0);
   signal reminder : std_logic_vector(3 downto 0);
BEGIN
        -- Instantiate the Unit Under Test (UUT)
   uut: latihan5 PORT MAP (
          a => a,
          b => b,
          quotient => quotient,
          reminder => reminder
        );
   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
                a <= "0111";
                b <= "0100";
      wait;
   end process;
END;
```

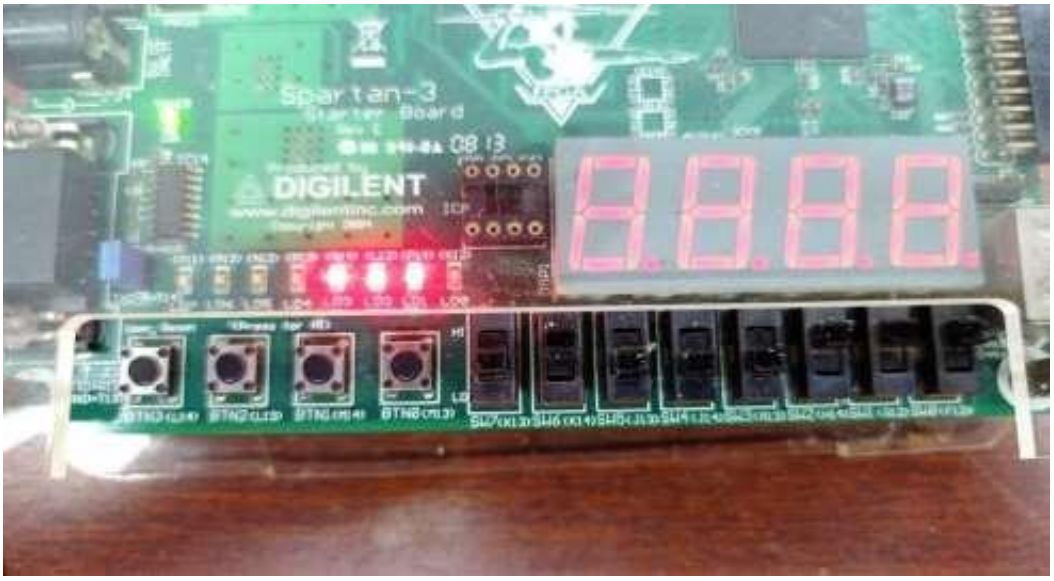
Simulasi Timing Diagram dengan input a = 7, b = 4



Gambar 14:Timing Diagram

Pin Layout

```
NET a(0) LOC=F12;
NET a(1) LOC=G12;
NET a(2) LOC=H14;
NET a(3) LOC=H13;

NET b(0) LOC=J14;
NET b(1) LOC=J13;
NET b(2) LOC=K14;
NET b(3) LOC=K13;


NET quotient(0) LOC=K12;
NET quotient(1) LOC=P14;
NET quotient(2) LOC=L12;
NET quotient(3) LOC=N14;

NET reminder(0) LOC=P13;
NET reminder(1) LOC=N12;
NET reminder(2) LOC=P12;
NET reminder(3) LOC=P11;
```



Gambar 15:Latihan dengan input 0100*0111

6. ANALISA

Pada program pertama (Decoder 3 to 8) menggunakan for loop statement. Statement tersebut memiliki input biner dan output desimal. Sehingga ketika menerima input 0000_2 maka Led yang menyala adalah paling kanan, karena urutannya dari kanan ke kiri mulai dari 0. Jika inputnya 1000_2 maka led yang menyala yaitu nomor 5 yang bernilai 4.

Pada program kedua (Binary to BCD Converter) yang memiliki input biner dan output desimal yang berbentuk biner terpisah. Jadi ketika menerima input 00011111_2 (31) maka led yang menyala 0011_2(3) dan 0001_2(1), yang berarti nilai 31 dipisah menjadi biner dari 3 dan 1.

Pada program ketiga (Biner to Gray Converter) yang memiliki input biner dan output kode Gray. Ketika menerima input 0010 maka LED yang menyala 0011 kode Gray.

Pada program keempat (Gray for Biner Converter) yang hampir sama dengan percobaan ke tiga, hanya berbeda input-output yang dibalik. Jadi ketika diberi input kode Gray 0011 maka LED yang menyala 0010 biner.

Pada program kelima (Multiplier) memiliki 2 input terpisah yang berasal dari 4 switch kanan dan 4 switch kiri. Untuk outputnya dari LED yang menampilkan biner. Cara kerjanya seperti perkalian, jika memberi input 0011_2 yang berarti 3 dan juga 0011_2 yang berarti 3 maka ulput pada LED akan menampilkan 9_{10} dalam bentuk biner yaitu 1001_2.

Pada program ke enam (Divider) hampir sama dengan program ke lima, hanya saja ini pembagian. Jadi, ketika menerima input 1000_2 yang bernilai 8 dan input 0100_2 yang bernilai 4. Maka 8:4=2, jadi pada LED akan menampilkan. biner dari 2 yaitu 0010_2.

No.

Date:

Pada program Latihan menghitung hasil dari a * b / 2, dimana a = 7 dan b = 4. Untuk outputnya menggunakan metode quetient dan reminder. Metodenya sendiri gabungan dari perkalian dan pembagian. Alur kerja program berawal dari input divider yang berasal dari hasil perkalian a * b, dan denom yang telah di deklarasikan menjadi "0010$_2$". Maka ketika input a yaitu 0111 dan input b yaitu 0100, dan outputnya akan $0111_2 * 0100_2 / 0010_2$ menjadi 1110_2

7. KESIMPULAN

Berdasarkan praktikum tersebut telah dilakukan percobaan menggunakan Spartan 3 dengan menjalankan beberapa program, seperti Decoder 3 to 8, Binary to BCD Converter, Biner to Gray Converter, Gray to Biner Converter, Multiplier, Divider dan juga program Latihan yang merupakan gabungan dari multiplier dan divider.