

**1 D4 - TEKKOM B**

**FOR LOOP STATEMENT**



Nama Kelas NRP

Dosen

Mata Kuliah

: Septian Bagus Jumantoro

: 1 – D4 Teknik Komputer B

: 3221600039

: Reni Soelistijorini B.Eng., MT.

: Praktikum Rangkaian Logika 2

Hari/Tgl. Praktikum : Jumat, 22 April 2022



Percobaan 5 – For Loop Statement

# Tujuan

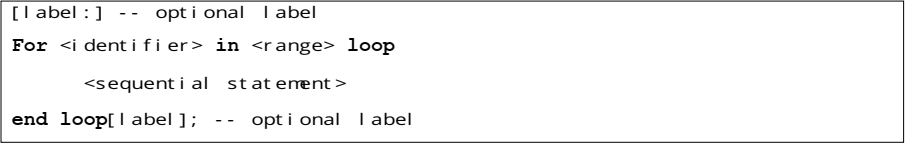
Dapat melakukan pemrogaman dasar VHDL menggunakan *for statement*

Dapat membuat rangkaian logika sederhana menggunakan *for statement*

# 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



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 \_ squar e **(**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** mat ch\_ bi t s **is**

**port(** a**,** b**: in** bi t \_ vect or **(**7 **downto** 0**);**

mat ches **: out** bi t \_ vect or **(**7 **downto** 0**)); end** mat ch\_ bi t s **;**

**architecture** f unc t i onal **of** mat ch\_ bi t s **is begin**

**process (**a**,** b**) begin**

**for** i **in** 7 **downto** 0 **loop**

mat ches **(**i **) <=** not **(**a**(**i **)** xor b**(**i **)); end loop;**

**end process; end** f unct i onal **;**

**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** | | |  |  |  |
| mat ches **(**7**)** | | | **<=** | not **(**a**(**7**)** | xor b**(**7**));** |
| mat ches **(**6**)** | | | **<=** | not **(**a**(**6**)** | xor b**(**6**));** |
| mat ches **(**5**)** | | | **<=** | not **(**a**(**5**)** | xor b**(**5**));** |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
| **end** |  | **process;** |  | |  |

## 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** Dec ode\_ f or **;**

**Script 5. 3** *Decoder 3 to 8*





**);**







**else**

## 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 :

* + 1. Menggeser satu bit bilangan biner.
    2. Jika 8 pergeseran telah terjadi, jumlah BCD adalah di kolom ratusan, puluhan dan satuan.
    3. Jika nilai biner dalam salah satu kolom BCD adalah lebih besar dari 4, tambahkan 3 pada nilaidi kolom tersebut.
    4. 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** I EEE**;**

**use** I EEE**.**STD\_ LOGI C\_ 1164**.ALL;**

**use** I EEE**.**st d\_ l ogi c\_ unsi gned**.all;**

**entity** Bi n2BCD **is port(**

b**:in** STD\_ LOGI C\_ VECTOR**(**7 **downto** 0**);** p**:out** STD\_ LOGI C\_ VECTOR**(**9 **downto** 0**)**

**);**

**end** Bi n2BCD**;**

**architecture** Behavi or al **of** Bi n2BCD **is begin**

bcd1**:process(**b**)**

**variable** z **:**STD\_ LOGI C\_ 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 **(**11**downto**8**):=** 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** Behavi or al **;**

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

## 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** I EEE**;**

**use** I EEE**.**STD\_ LOGI C\_ 1164**.ALL;**

**entity** Bi nToGr ay **is port(**

b**:in** STD\_ LOGI C\_ VECTOR**(**3 **downto** 0**);** g**:out** STD\_ LOGI C\_ VECTOR**(**3 **downto** 0**)**

**);**

**end** Bi nToGr ay**;**

**architecture** Behavi or al **of** Bi nToGr ay **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** Behavi or al **;**

**Script 5. 5** *Konversi Biner ke Gray code*

## Gray Code Converter ‑ Gray to Biner

lx

Konversi Gray ke Biner :alin semua bit

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

**library**I EEE**;**

**use** I EEE**.**STD\_ LOGI C\_ 1164**.ALL;**

**entity** Gr ayToBi n **is port(**

g**:in** STD\_ LOGI C\_ VECTOR**(**3 **downto** 0**);** b**:out** STD\_ LOGI C\_ VECTOR**(**3 **downto** 0**)**

**);**

**end** Gr ayToBi n**;**

**architecture** Behavi or al **of** Gr ayToBi n **is begin**

**process(**g**)**

**variable** bv **:**STD\_ LOGI C\_ 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** Behavi or al **;**

**Script 5. 6** *Konversi Gray Code ke Biner*

## 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.

**1101**

**X 1011**

**1101**

**1101**

**0000**

**1101**

**Gambar 5. 1** *Binary multiplication*

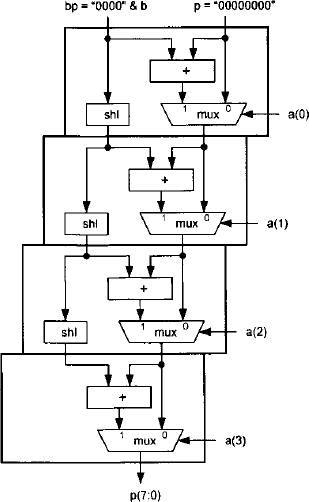
**);**



**endloop;**

**endprocess;**

**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;*

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

**Library** I EEE**;**

**use** I EEE**.**STD\_ LOGI C\_ 1164**.ALL;**

**entity** Mul **is**

**port(**

a**:in** STD\_ LOGI C\_ VECTOR**(**3**downto**0**);** b**:in** STD\_ LOGI C\_ VECTOR**(**3**downto**0**);** p**:out** STD\_ LOGI C\_ VECTOR**(**7**downto**0**)**

**);**

**end** Mul **;**

**architecture** Behavi or al **of** Mul **is begin**

p **<=** a**\***b**;**

**end** Behavi or al **;**



**);**

**begin**

**then**

**end function;**

**signal** a1**,** a2**:**STD\_ LOGI C\_ VECTOR**(**3 **downto** 0**); begin**

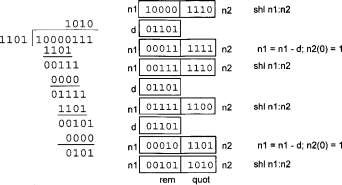
a1 **<=** s w**(**7 **downto** 4**);**

a2 **<=** s w**(**7 **downto** 4**);** l d **<=** mul **(**a1**,**a2**);**

**Script 5. 9** *Multiplication Function*

## 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;

1. quot = n2; rem = n1(3:0);







lxiv

numer **:in** STD\_ LOGI C\_ VECTOR**(**7 **downto** 0**);** denom **:in** STD\_ LOGI C\_ VECTOR**(**3 **downto** 0**);** quot i ent **:out** STD\_ LOGI C\_ VECTOR**(**3 **downto** 0**);** r emai nder **:out** STD\_ LOGI C\_ VECTOR**(**3 **downto** 0**)**

**);**

**end** Devi de**;**

**architecture** Behavi or al **of** Devi de **is begin**

**process(**numer **,**denom**)**

**variable** d**,** n1 **:**STD\_ LOGI C\_ VECTOR**(**4 **downto** 0**); variable** n2 **:**STD\_ LOGI C\_ 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;**

quot i ent **<=** n2**;**

r emai nder **<=** n1**(**3 **downto** 0**); end process;**

**end** Behavi or al **;**

**Script 5. 10** *Divider*

# Alat dan Bahan

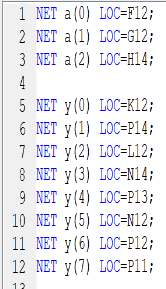
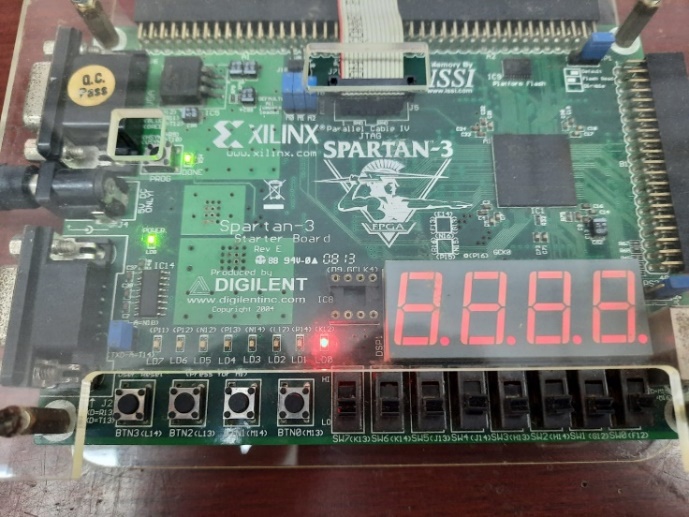
* 1. PC yang sudah terinstal ISE 13.1
  2. Xilinx Spartan 3
  3. Downloader JTAG USB
  4. Power Supply 5v

# 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

# Hasil Percobaan

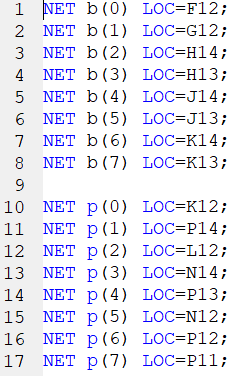
1. Decoder 3 to 8 Pin Layout



***Gambar 1:Decoder 3 to 8 dengan input 00002 = 010***



***Gambar 2:Decoder 3 to 8 dengan input 10002 = 410***

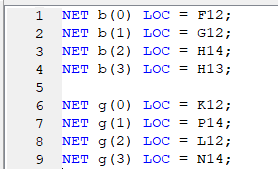
1. Binary to BCD Converter Pin Layout



***Gambar 3:Biner to BCD dengan input 0001 11112 = 3110***



***Gambar 4:Biner to BCD dengan input 1111 11112 = 25510***

1. Biner to Gray Converter Pin Layout



***Gambar 5:biner to grey dengan input 0010***



***Gambar 6:biner to grey dengan input 1111***

1. Gray to Biner Converter Pin layout

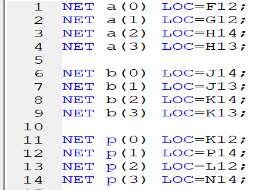


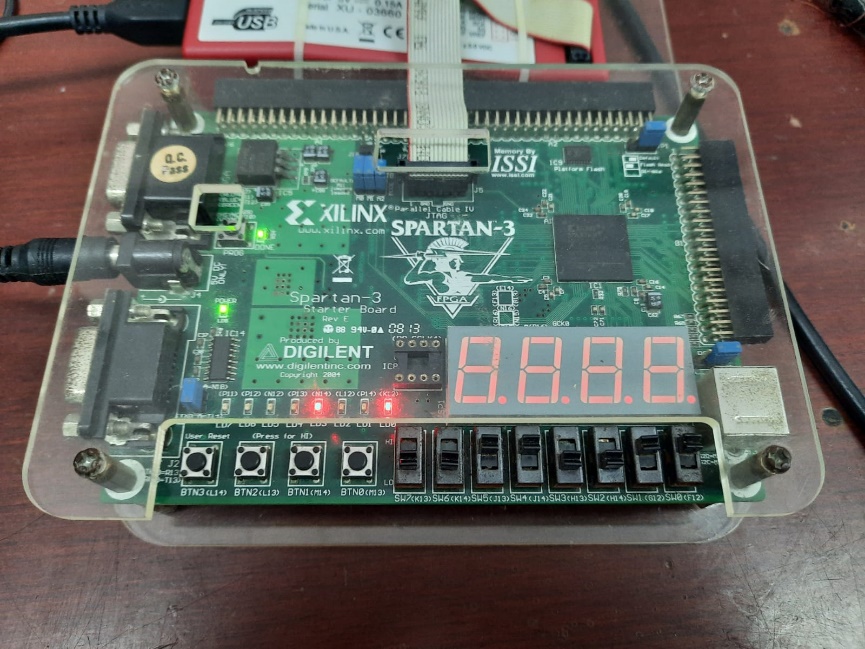
***Gambar 7:Gray to Biner dengan input 0011***



***Gambar 8:Gray to Biner dengan input 1000***

1. Multiplier Pin layout



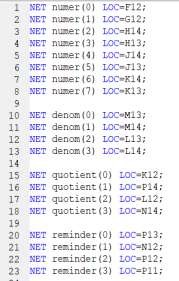


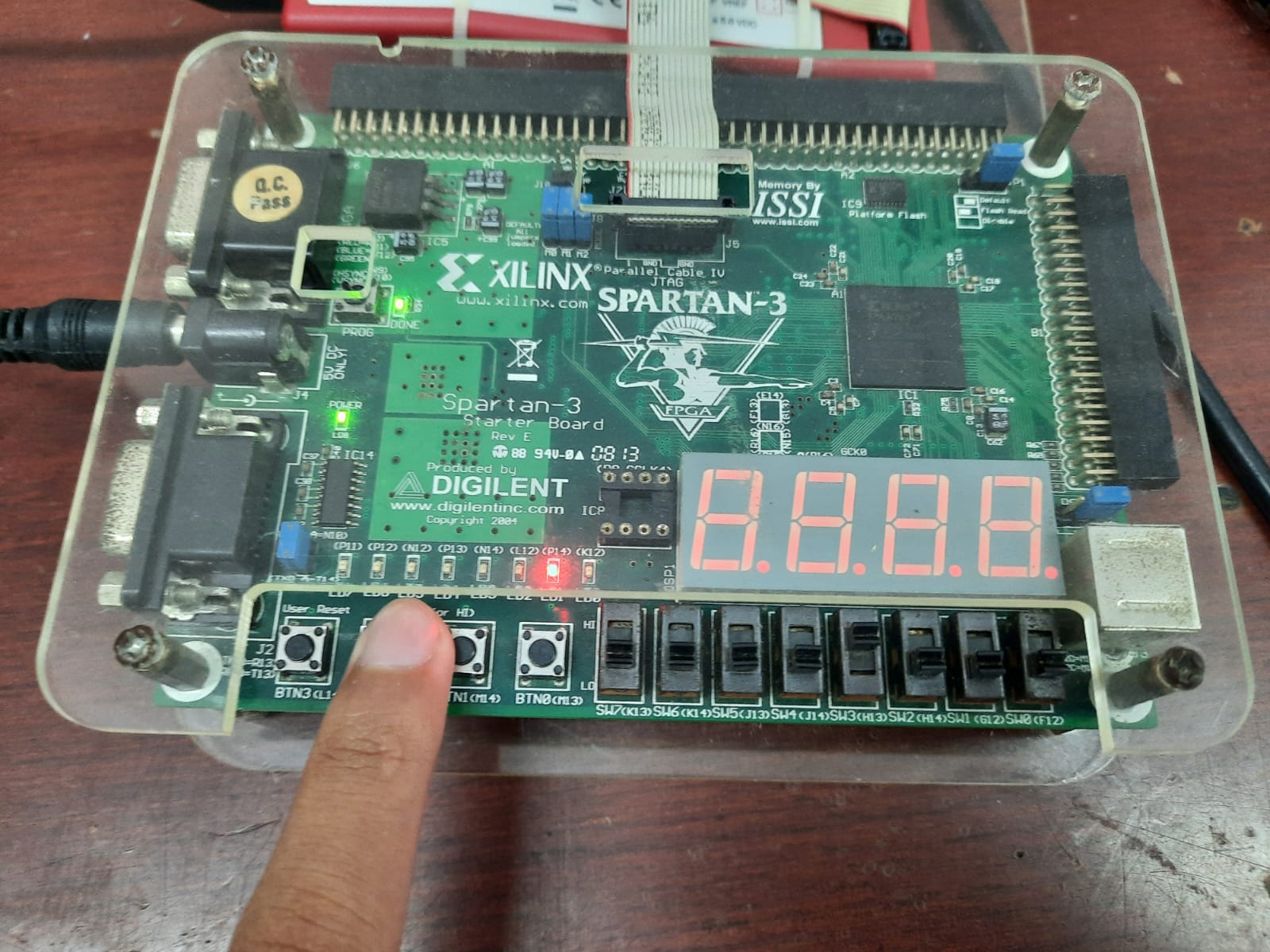
***Gambar 9:Multiplier dengan input 0011­2 \* 00112 = 910***



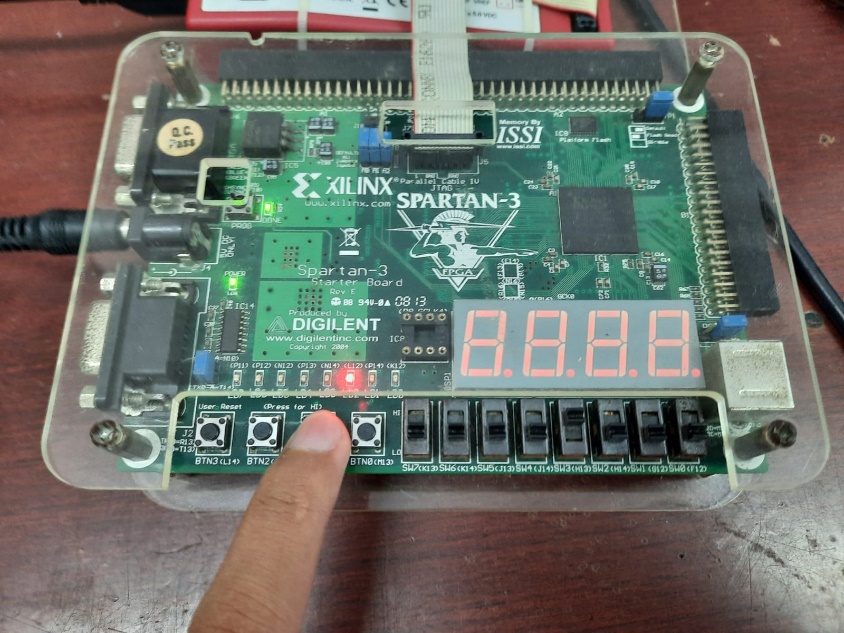
***Gambar 10:Multiplier dengan input 01002 \* 01002 = 1610***

1. Divider Pin Layout





***Gambar 11:Divider dengan input 10002 : 01002 = 00102***



***Gambar 12:Divider dengan input 10002 : 00102 = 01002***

1. 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';

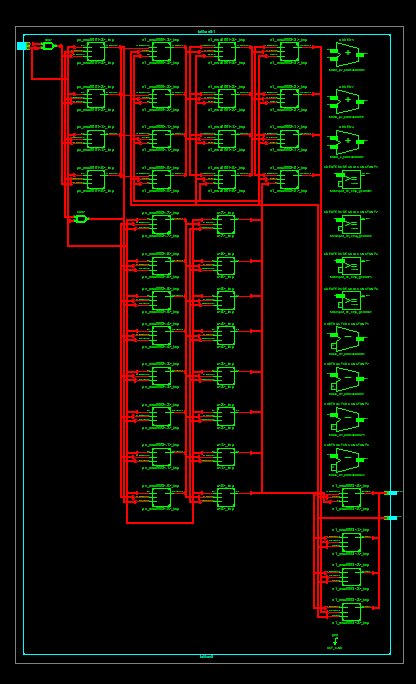
end if; end loop;

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;

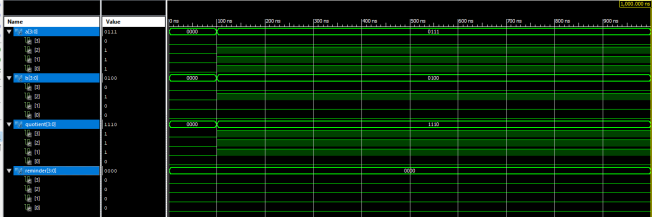
a <= "0111";

b <= "0100";

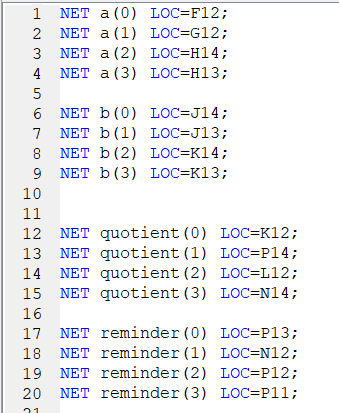
wait;

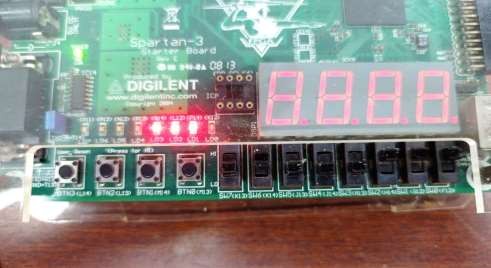
end process;

END;

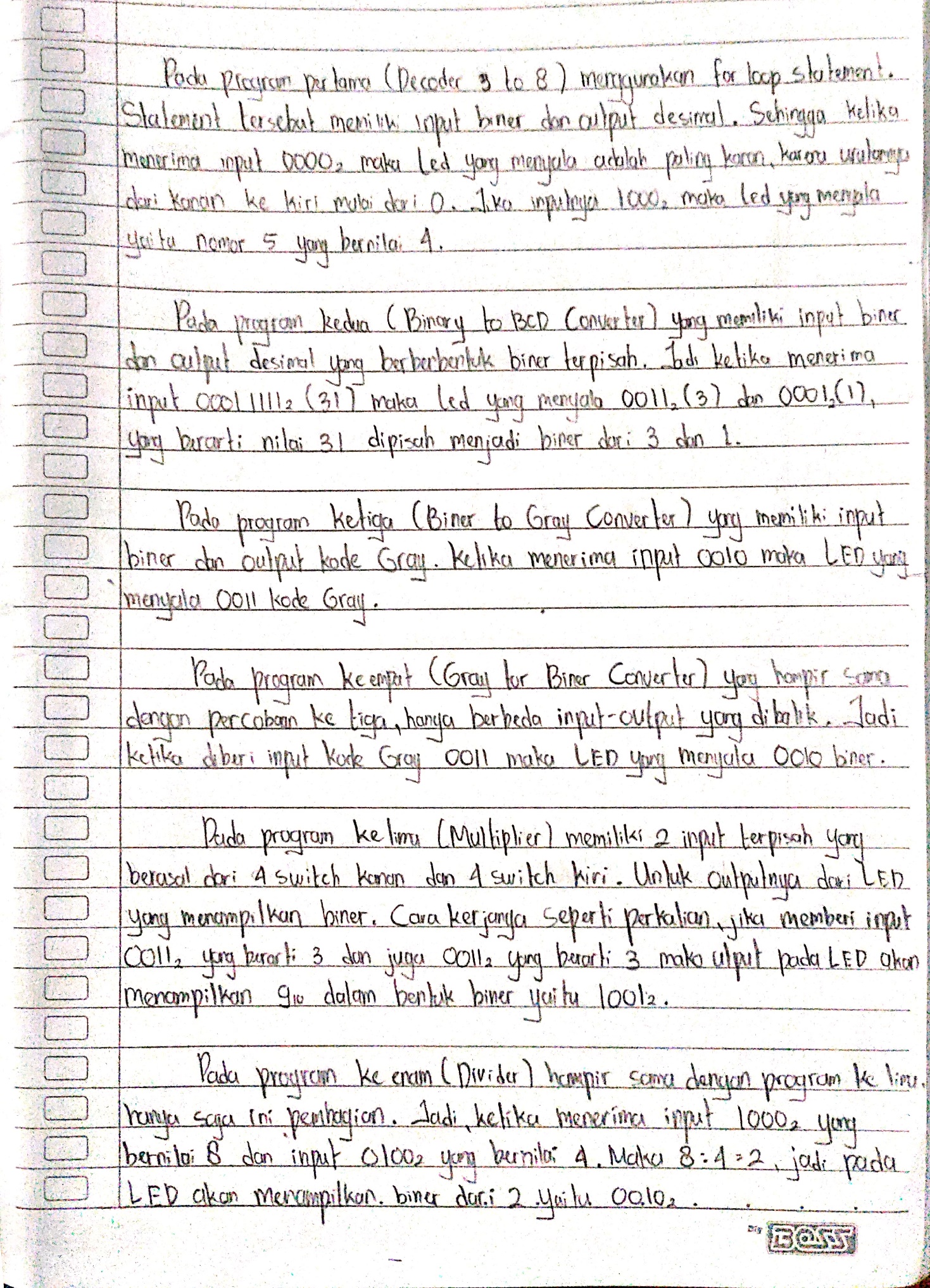
Simulasi Timing Diagram dengan input a = 7, b = 4

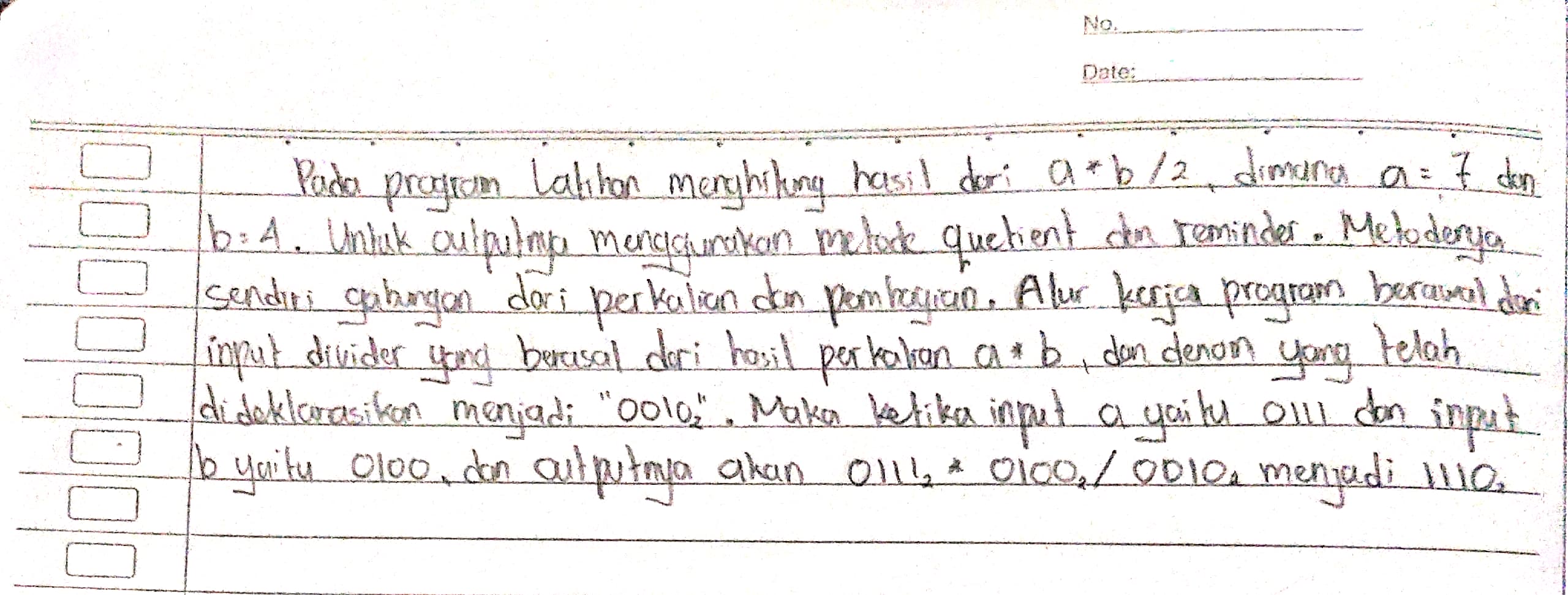
***Gambar 14:Timing Diagram***

Pin Layout



***Gambar 15:Latihan dengan input 0100\*0111***

1. **ANALISA**



1. **KESIMPULAN**

