MODUL 7
IMPLEMENTASI RANGKAIAN DIGITAL PENCACAH DI FPGA MENGGUNAKAN VERILOG-HDL

1.1 Tujuan Praktikum Modul

Setelah mempraktekkan topik ini, praktikan diharapkan dapat :

1. Praktikan dapat mengetahui dan memahami sirkuit yang telah di rancang dan di implementasikan pada intel FPGA.
2. Praktikan dapat memahami dan menggunakan pencacah pada FPGA dengan menggunakan Verilog-HDL.
3. Praktikan dapat memiliki pemahaman dasar tentang pencacah pada FPGA untuk dapat melakukan berbagai macam implementasi

1.2 Dasar Teori Praktikum Modul 1

1.2.1 Pencacah pada FPGA

Pencacah atau yang bisa disebut Counter ini merupakan rangkaian logika sekuensial yang digunakan untuk menghitung jumlah pulsa. Pencacah dalam FPGA juga dapat berfungsi untuk menghitung berapa kali suatu peristiwa terjadi, seperti pulsa masukan atau perubahan sinyal tertentu. Counter dapat dibagi menjadi dua, yaitu:

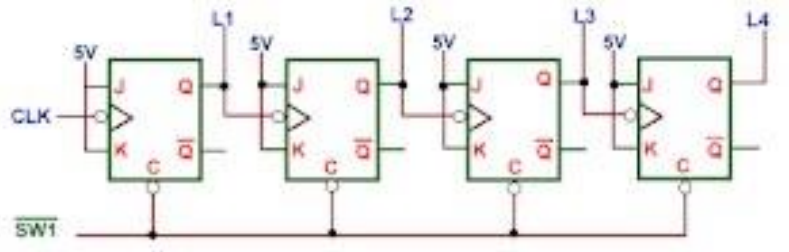
1. Asynchronous Counter
2. Synchronous Counter

Sedangkan menurut urutan hitungan yang terbentuk pada outputnya, maka counter dapat dibagi menjadi :

1. Up counter
2. Down counter
3. Up-down counter

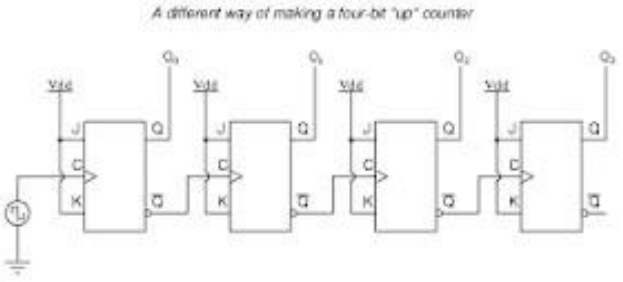
1.2.2 Asynchronous Binary Up Counter

Merupakan counter yang dapat menghitung biangan biner dengan urutan dari bawah ke atas. Apabila digunakan 4 buah flip-flop, maka kita dapat melakukan hitunga paling tinggi adalah 1111. Counter yang dapat menghitung sampai 1111 disebut 4 bit binary counter. Oleh karena dapat menghitung dengan cara ke atas, maka disebut pula asynchronous 4 binary up counter.



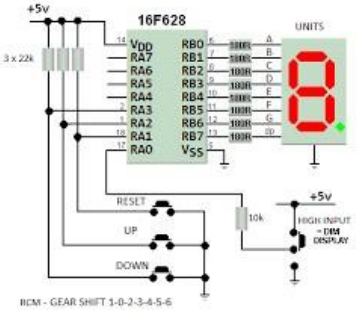
1.2.3 Asynchronous Binary Down Counter

Merupakan Counter yang memiliki prinsip kerja dari counter ini adalah kebalikan dari up counter, yaitu menghitung bilangan biner dengan urutan mulai dari atas ke bawah (dari besar ke kecil). Prinsip kerjanyapun tidak jauh berbeda dari up counter. Bedanya hanya setiap output flip-flop diambil dari output Q, sedangkan input clocknya dihubungkan dengan ouput not Q dari flip-flop sebelumnya.



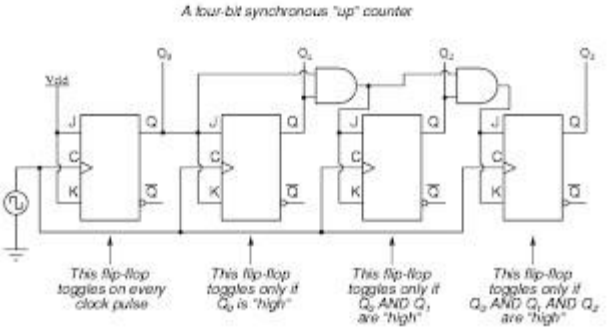
1.2.4 Asynchronous up Down Counter

Merupakan suatu rangkaian elektronik yang mempergunakan sistem digital sering memerlukan suatu alat pencacah yang dapat menghitung ke atas dan bisa juga menghitung ke bawah. Alat pencacah yang dapat melakukan penghitungan seperti ini disebut dengan binary up down counter. Alat ini dapat menghitung ke atas dan ke bawah dengan mengatur suatu alat pengontrol tertentu.



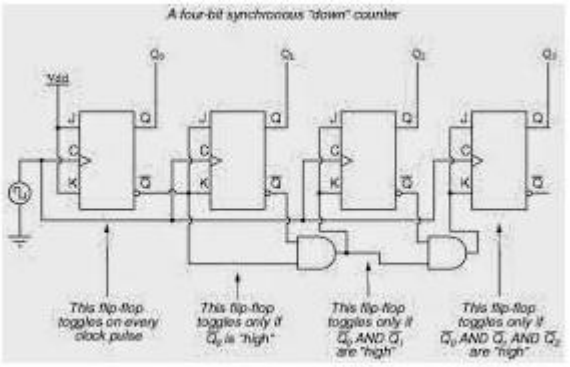
1.2.5 Synchronous Binary up Counter

Jika pada asynchronous counter pulsa yang akan dihitung datangnya tidak serentak, maka pada synchronour counter ini pulsa yang ingin dihitung ini masuk ke dalam setiap flip-flop serentak (bersama-sama) sehinga perubah output setiap flip-flop akan terjadi secara serentak.



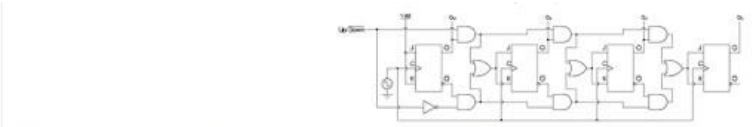
1.2.6 Synchronous Binary Down Counter

Sama dengan synchronous binary up counter di atas, hanya saja bedanyan rangkaian ini melakukan penghitungan dari atas ke bawah.

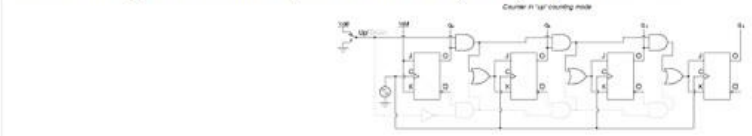


1.2.7 Synchronous Binary Up Down Counter

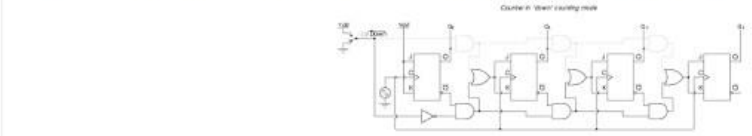
Pada rangkaian ini bisa dilakukan proses penghitungan ke atas atau ke bawah dengan memanfaatkan tombol pengatur proses penghitungan.



Jika kita menggunakan kontrol up counter maka rangkaian yang aktif adalah



Sedangkan jika kita menggunakan down counter maka rangkaian yang aktif adalah



1.3 Lembar Kegiatan Praktikum Modul 1

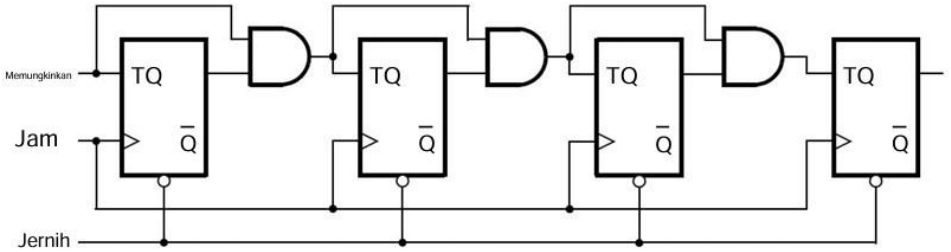
1.3.1 Alat dan Bahan

a. Software *Quartus II*
b. *ModelSim*
c. Mouse
d. Laptop

1.3.2 Langkah Praktikum Modul Quartus II

Bagian I

Perhatikan rangkaian pada Gambar 1. Ini adalah pencacah sinkron 4-bit yang menggunakan empat flip-flop tipe-T. Penghitung menaikkan nilainya pada setiap tepi positif sinyal clock jika sinyal Aktifkan tinggi. Penghitung direset ke 0 pada tepi clock positif berikutnya jika input Clear sinkron rendah. Anda harus menerapkan penghitung 8-bit jenis ini.



Gambar 1: Penghitung 4-bit.

1. Tulis file Verilog yang mendefinisikan pencacah 8-bit dengan menggunakan struktur yang digambarkan pada Gambar 1. Kode Anda harus menyertakan modul T flip-flop yang dipakai delapan kali untuk membuat pencacah. Kompilasi sirkuitnya. Berapa banyak elemen logika (LE) yang digunakan untuk mengimplementasikan rangkaian Anda?

```
1   //
2   // inputs:
3   // KEY0: manual clock
4   // SW0: active low reset
5   // SW1: enable signal for the counter
6   //
7   // outputs:
8   // HEX0 - HEX1: hex segment displays
9   module part1 (SW, KEY, HEX1, HEX0);
10     input [1:0] SW ;
11     input [0:0] KEY ;
12     output [0:6] HEX1, HEX0;
13
14     wire Clock = KEY[0];
15     wire Resetn = SW[0];
16
17     // 8-bit counter based on T-flip flops
18     wire [7:0] Count;
19     wire [7:0] Enable;
20
21     assign Enable[0] = SW[1];
22     ToggleFF ff0(Enable[0], Clock, Resetn, Count[0]);
23     assign Enable[1] = Count[0] & Enable[0];
24     ToggleFF ff1(Enable[1], Clock, Resetn, Count[1]);
25     assign Enable[2] = Count[1] & Enable[1];
26     ToggleFF ff2(Enable[2], Clock, Resetn, Count[2]);
27     assign Enable[3] = Count[2] & Enable[2];
28     ToggleFF ff3(Enable[3], Clock, Resetn, Count[3]);
29     assign Enable[4] = Count[3] & Enable[3];
30     ToggleFF ff4(Enable[4], Clock, Resetn, Count[4]);
31     assign Enable[5] = Count[4] & Enable[4];
32     ToggleFF ff5(Enable[5], Clock, Resetn, Count[5]);
33     assign Enable[6] = Count[5] & Enable[5];
34     ToggleFF ff6(Enable[6], Clock, Resetn, Count[6]);
```

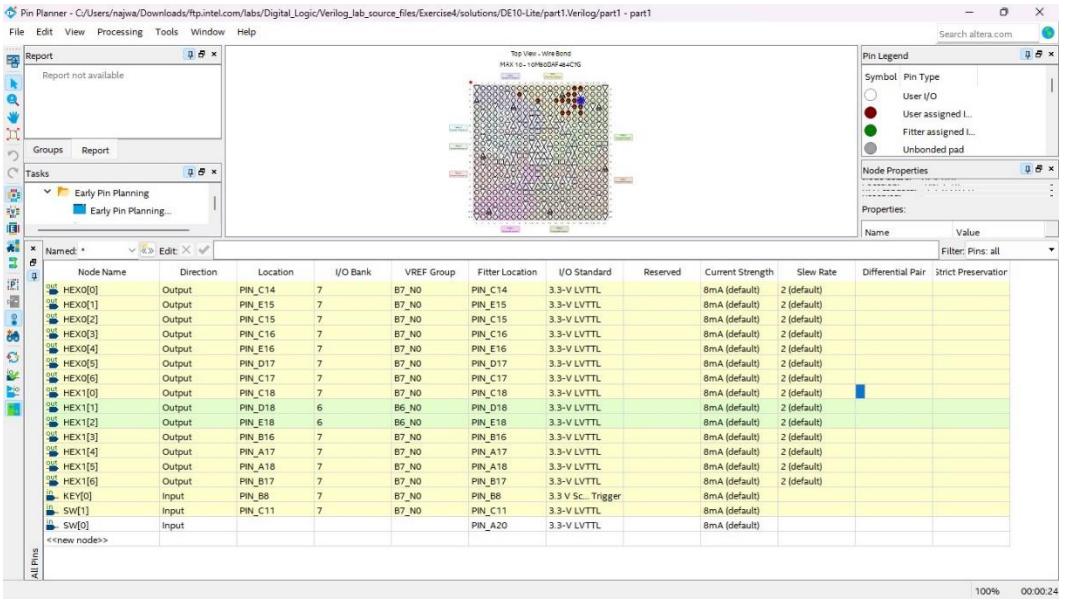
```
23      assign Enable[1] = Count[0] & Enable[0];
24      ToggleFF ff1(Enable[1], Clock, Resetn, Count[1]);
25      assign Enable[2] = Count[1] & Enable[1];
26      ToggleFF ff2(Enable[2], Clock, Resetn, Count[2]);
27      assign Enable[3] = Count[2] & Enable[2];
28      ToggleFF ff3(Enable[3], Clock, Resetn, Count[3]);
29      assign Enable[4] = Count[3] & Enable[3];
30      ToggleFF ff4(Enable[4], Clock, Resetn, Count[4]);
31      assign Enable[5] = Count[4] & Enable[4];
32      ToggleFF ff5(Enable[5], Clock, Resetn, Count[5]);
33      assign Enable[6] = Count[5] & Enable[5];
34      ToggleFF ff6(Enable[6], Clock, Resetn, Count[6]);
35      assign Enable[7] = Count[6] & Enable[6];
36      ToggleFF ff7(Enable[7], Clock, Resetn, Count[7]);
37
38      // drive the displays
39      hex7seg digit1 (Count[7:4], HEX1);
40      hex7seg digit0 (Count[3:0], HEX0);
41  endmodule
42
43  module ToggleFF(T, Clock, Resetn, Q);
44      input T, Clock, Resetn;
45      output reg Q;
46
47      always @(posedge Clock)
48          if (Resetn  == 1'b0) // synchronous clear
49              Q <= 1'b0;
50          else if(T)
51              Q <= ~Q;
52  endmodule
53
54  module hex7seg (hex, display);
55      input [3:0] hex;
56      output [0:6] display;
```

```
52  endmodule
53
54  module hex7seg (hex, display);
55      input [3:0] hex;
56      output [0:6] display;
57
58      reg [0:6] display;
59
60      /*
61       *        0
62       *       ---
63       *      |   |
64       *    5 |   | 1
65       *      | 6 |
66       *       ---
67       *      |   |
68       *    4 |   | 2
69       *      |   |
70       *       ---
71       *        3
72       */
73      always @ (hex)
74          case (hex)
75              4'h0: display = 7'b0000001;
76              4'h1: display = 7'b1001111;
77              4'h2: display = 7'b0010010;
78              4'h3: display = 7'b0000110;
79              4'h4: display = 7'b1001100;
80              4'h5: display = 7'b0100100;
81              4'h6: display = 7'b0100000;
82              4'h7: display = 7'b0001111;
83              4'h8: display = 7'b0000000;
84              4'h9: display = 7'b0000100;
```

```
61      *        0
62      *       ---
63      *      |   |
64      *    5 |   | 1
65      *      | 6 |
66      *       ---
67      *      |   |
68      *    4 |   | 2
69      *      |   |
70      *       ---
71      *        3
72      */
73     always @ (hex)
74        case (hex)
75            4'h0: display = 7'b0000001;
76            4'h1: display = 7'b1001111;
77            4'h2: display = 7'b0010010;
78            4'h3: display = 7'b0000110;
79            4'h4: display = 7'b1001100;
80            4'h5: display = 7'b0100100;
81            4'h6: display = 7'b0100000;
82            4'h7: display = 7'b0001111;
83            4'h8: display = 7'b0000000;
84            4'h9: display = 7'b0000100;
85            4'hA: display = 7'b0001000;
86            4'hb: display = 7'b1100000;
87            4'hC: display = 7'b0110001;
88            4'hd: display = 7'b1000010;
89            4'hE: display = 7'b0110000;
90            4'hF: display = 7'b0111000;
91        endcase
92  endmodule
93
```

2. Simulasikan sirkuit Anda untuk memverifikasi kebenarannya.

3. Tambahkan file Verilog Anda untuk menggunakan tombol tekan KEY0 sebagai input Jam dan alihkan SW1 dan SW0 sebagai input Aktifkan dan Hapus, dan 7-segmen menampilkan HEX1-0 untuk menampilkan hitungan heksadesimal saat sirkuit Anda beroperasi. Buatlah penetapan pin yang diperlukan untuk mengimplementasikan sirkuit pada papan seri DE Anda, dan kompilasi sirkuitnya.



4. Unduh sirkuit Anda ke dalam chip FPGA dan uji fungsinya dengan mengoperasikan sakelar.

5. Terapkan versi empat-bit dari sirkuit Anda dan gunakan Quartus® RTL Viewer untuk melihat bagaimana Quartus perangkat lunak mensintesis sirkuit. Apa perbedaannya dibandingkan dengan Gambar 1?

Modul Praktikum

Bagian II

Cara lain untuk menentukan counter adalah dengan menggunakan register dan menambahkan 1 pada nilainya. Hal ini dapat dicapai dengan menggunakan pernyataan Verilog berikut:

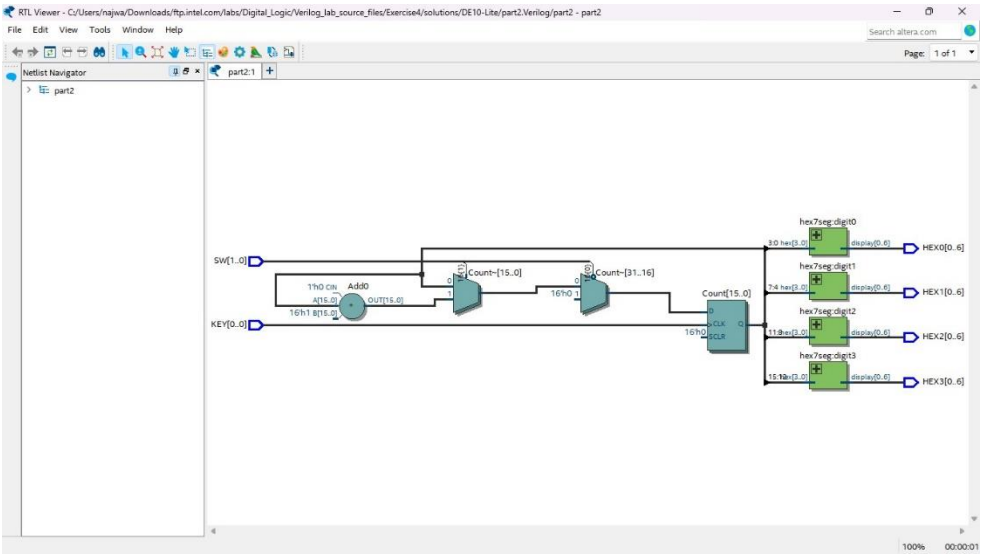
```
Q <= Q + 1;
```

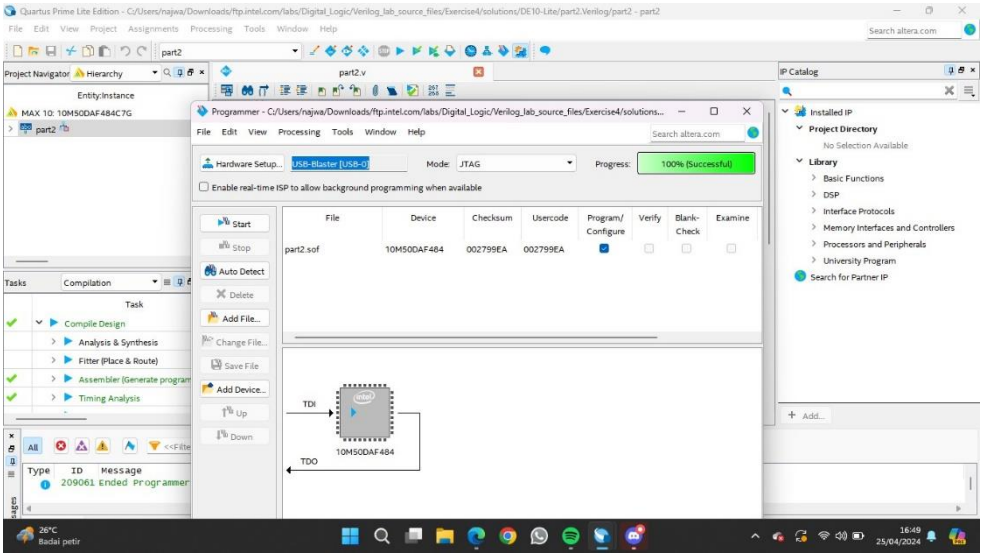
Kompilasi versi 16-bit dari penghitung ini dan tentukan jumlah LE yang dibutuhkan. Gunakan RTL Viewer untuk melihat struktur implementasi ini dan komentari perbedaannya dengan desain dari Bagian I. Implementasikan penghitung pada papan seri DE Anda, gunakan tampilan HEX3-0 untuk menunjukkan nilai penghitung.

```
1   //
2   // inputs:
3   // KEY0: manual clock
4   // SW0: active low reset
5   // SW1: enable signal for the counter
6   //
7   // outputs:
8   // HEX0 - HEX3: hex segment displays
9   module part2 (SW, KEY, HEX3, HEX2, HEX1, HEX0);
10     input [1:0] SW ;
11     input [0:0] KEY ;
12     output [0:6] HEX3, HEX2, HEX1, HEX0;
13
14     wire Clock = KEY[0];
15     wire Resetn = SW[0];
16     wire Enable = SW[1];
17
18     // 16-bit counter
19     reg [15:0] Count;
20     always @(posedge Clock)
21        if (!Resetn)
22           Count <= 0;
23        else if (Enable)
24           Count <= Count + 1'b1;
25
26     // drive the displays
27     hex7seg digit3 (Count[15:12], HEX3);
28     hex7seg digit2 (Count[11:8], HEX2);
29     hex7seg digit1 (Count[7:4], HEX1);
30     hex7seg digit0 (Count[3:0], HEX0);
31  endmodule
32
33  module hex7seg (hex, display);
34     input [3:0] hex;
```

```
33  module hex7seg (hex, display);
34     input [3:0] hex;
35     output [0:6] display;
36
37     reg [0:6] display;
38
39     /*
40      *       0
41      *      ---
42      *     |   |
43      *    5|   |1
44      *     | 6 |
45      *      ---
46      *     |   |
47      *    4|   |2
48      *     |   |
49      *      ---
50      *       3
51      */
52     always @ (hex)
53        case (hex)
54           4'h0: display = 7'b0000001;
55           4'h1: display = 7'b1001111;
56           4'h2: display = 7'b0010010;
57           4'h3: display = 7'b0000110;
58           4'h4: display = 7'b1001100;
59           4'h5: display = 7'b0100100;
60           4'h6: display = 7'b0100000;
61           4'h7: display = 7'b0001111;
62           4'h8: display = 7'b0000000;
63           4'h9: display = 7'b0000100;
64           4'hA: display = 7'b0001000;
```

```
40          *       0
41          *      ---
42          *     |   |
43          *    5|   |1
44          *     | 6 |
45          *      ---
46          *     |   |
47          *    4|   |2
48          *     |   |
49          *      ---
50          *       3
51          */
52        always @ (hex)
53           case (hex)
54               4'h0: display = 7'b0000001;
55               4'h1: display = 7'b1001111;
56               4'h2: display = 7'b0010010;
57               4'h3: display = 7'b0000110;
58               4'h4: display = 7'b1001100;
59               4'h5: display = 7'b0100100;
60               4'h6: display = 7'b0100000;
61               4'h7: display = 7'b0001111;
62               4'h8: display = 7'b0000000;
63               4'h9: display = 7'b0000100;
64               4'hA: display = 7'b0001000;
65               4'hb: display = 7'b1100000;
66               4'hC: display = 7'b0110001;
67               4'hd: display = 7'b1000010;
68               4'hE: display = 7'b0110000;
69               4'hF: display = 7'b0111000;
70           endcase
71    endmodule
```

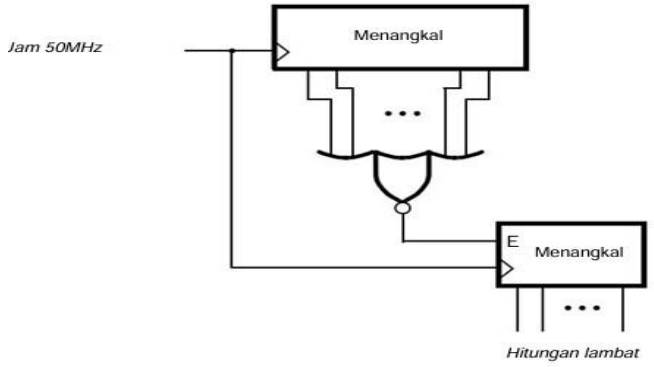




Modul Praktikum

Bagian III

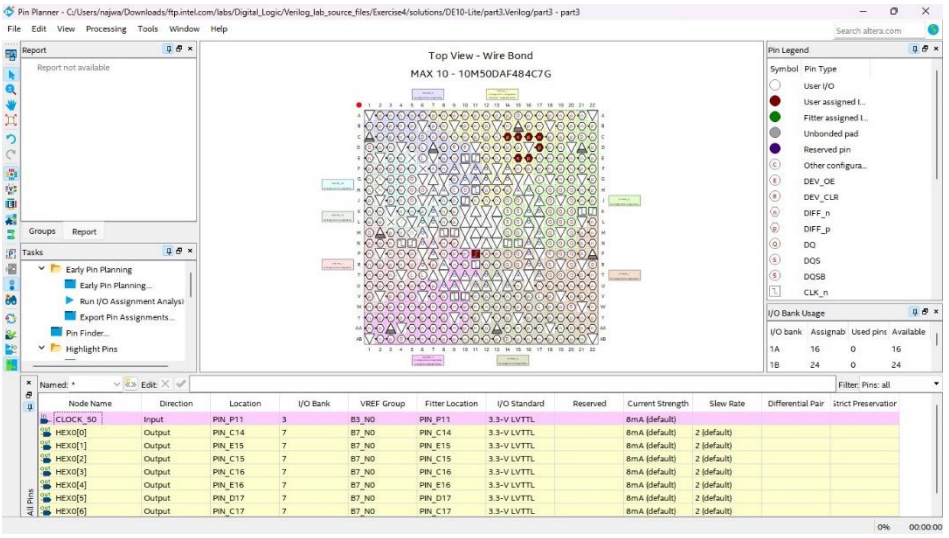
Rancang dan implementasikan rangkaian yang secara berturut-turut menampilkan angka 0 hingga 9 pada tampilan 7-segmen HEX0. Setiap digit harus ditampilkan sekitar satu detik. Gunakan penghitung untuk menentukan interval satu detik. Penghitung harus ditingkatkan dengan sinyal clock 50 MHz yang disediakan pada papan seri DE. Jangan mendapatkan sinyal clock lain dalam desain Anda – pastikan semua flip flop di sirkuit Anda memiliki clock langsung oleh sinyal clock 50 MHz. Desain sebagian dari rangkaian yang diperlukan ditunjukkan pada Gambar 2. Gambar tersebut menunjukkan bagaimana pencacah lebar bit yang besar dapat digunakan untuk menghasilkan sinyal pengaktifan untuk pencacah yang lebih kecil. Kecepatan kenaikan pencacah yang lebih kecil dapat dikontrol dengan memilih jumlah bit yang sesuai pada pencacah yang lebih besar.

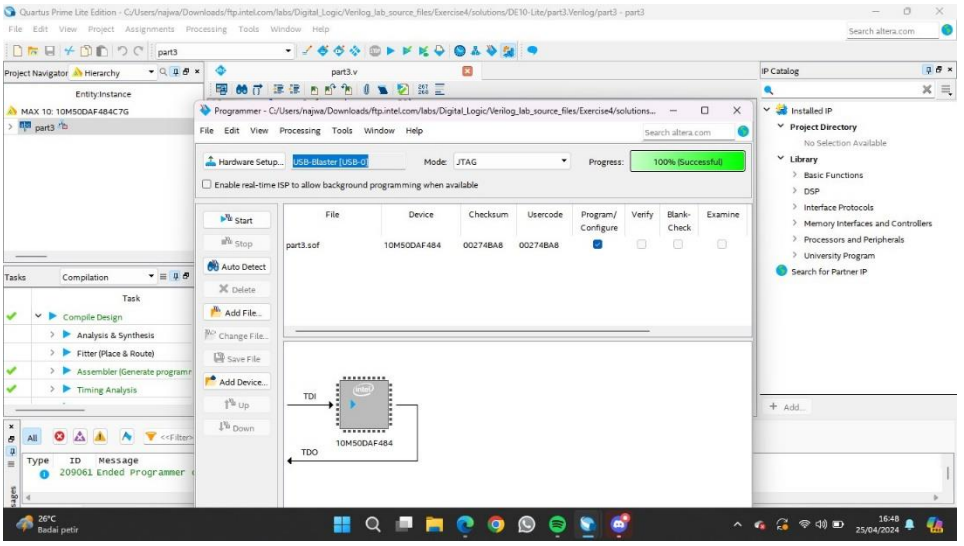


Gambar 2: Membuat penghitung lambat.

```
// uses a 1-digit bcd counter enabled at 1Hz
module part3 (CLOCK_50, HEX0);
   input CLOCK_50;
   output [0:6] HEX0;

   wire [3:0] bcd;
   parameter m = 25;
   reg [m-1:0] slow_count;

   reg[3:0] digit_flipper;

   // Create a 1Hz 4-bit counter

   // A large counter to produce a 1 second (approx) enable from the 50 MHz Clock
   always @(posedge CLOCK_50)
      slow_count <= slow_count + 1'b1;

   // four-bit counter that uses a slow enable for selecting digit
   always @ (posedge CLOCK_50)
      if (slow_count == 0)
         if (digit_flipper == 4'h9)
            digit_flipper <= 4'h0;
         else
            digit_flipper <= digit_flipper + 1'b1;

   assign bcd = digit_flipper;
   // drive the display through a 7-seg decoder
   bcd7seg digit_0 (bcd, HEX0);

endmodule

module bcd7seg (bcd, display);
   input [3:0] bcd;
   output [0:6] display;
```

```
32  module bcd7seg (bcd, display);
33     input [3:0] bcd;
34     output [0:6] display;
35
36     reg [0:6] display;
37
38     /*
39      *       0
40      *      ---
41      *     |   |
42      *    5|   |1
43      *     | 6 |
44      *      ---
45      *     |   |
46      *    4|   |2
47      *     |   |
48      *      ---
49      *       3
50      */
51     always @ (bcd)
52        case (bcd)
53           4'h0: display = 7'b0000001;
54           4'h1: display = 7'b1001111;
55           4'h2: display = 7'b0010010;
56           4'h3: display = 7'b0000110;
57           4'h4: display = 7'b1001100;
58           4'h5: display = 7'b0100100;
59           4'h6: display = 7'b0100000;
60           4'h7: display = 7'b0001111;
61           4'h8: display = 7'b0000000;
62           4'h9: display = 7'b0000100;
63           default: display = 7'bx;
64        endcase
65  endmodule
```






Shift
Ctrl
Fn
Alt
intel
Download DE10-Lite CD
http://de10-lite.terasic.com/cd
terasIC
DE10-Lite


Shift
Ctrl
Fn
Alt
intel
Download DE10-Lite CD
http://de10-lite.terasic.com/cd
terasIC
DE10-Lite

Modul Praktikum

Bagian IV

Rancang dan terapkan sirkuit yang menampilkan sebuah kata pada empat tampilan 7-segmen HEX3 ÿ 0. Kata yang akan ditampilkan untuk papan seri DE Anda diberikan pada Tabel 1. Buatlah huruf-huruf berputar dari kanan ke kiri dalam interval sekitar satu detik . Pola putaran DE10-Lite diberikan pada Tabel 2. Jika Anda menggunakan DE0-CV, DE1-SoC, atau DE2-115, gunakan kata yang diberikan pada Tabel 1. Ada banyak cara untuk merancang rangkaian yang diperlukan. Salah satu solusinya adalah dengan menggunakan kembali kode Verilog yang dirancang pada Latihan Laboratorium 1, Bagian V. Dengan menggunakan kode tersebut, perubahan utama yang diperlukan adalah mengganti dua saklar yang digunakan untuk memilih karakter yang diputar pada tampilan dengan 2-bit penghitung yang bertambah pada interval satu detik.

Papan	Kata
DE10-Lite	dE10
DE0-CV	dE0
DE1-SoC	dE1
DE2-115	dE2

Tabel 1: Papan seri DE dan kata terkait untuk ditampilkan

Menghitung	Karakter
00	hari E 1 0
01	E 1 0 d
10	1 0 dE
11	0 d E 1

Tabel 2: Memutar kata dE10 pada empat tampilan.

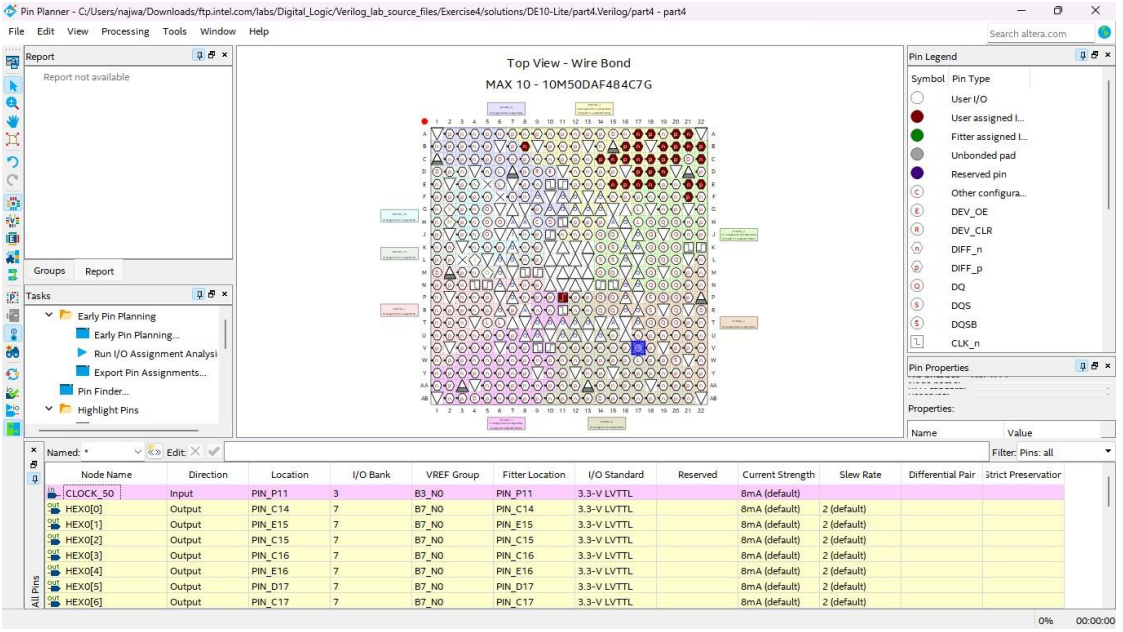
```
// Implements a circuit that rotates dE10 across the 7-segment displays HEX3 - HEX0
module part4 (KEY, CLOCK_50, HEX3, HEX2, HEX1, HEX0);
   input [0:0] KEY;
   input CLOCK_50;
   output [0:6] HEX3, HEX2, HEX1, HEX0;  // 7-seg displays

   wire [1:0] Ch0, Ch1, Ch2, Ch3;
   reg [1:0] Ch_Sel;
   wire [1:0] H3_Ch, H2_Ch, H1_Ch, H0_Ch;
   assign Ch1 = 2'b00;     // d
   assign Ch2 = 2'b01;     // E
   assign Ch3 = 2'b10;     // 1
   assign Ch0 = 2'b11;     // 0

   parameter m = 25;
   reg [m-1:0] slow_count;

   // Create a 1Hz 2-bit counter

   // A large counter to produce a 1 second (approx) enable from the 50 MHz Clock
   always @(posedge CLOCK_50)
      slow_count <= slow_count + 1'b1;

   // two-bit counter that uses a slow enable for selecting characters
   always @ (posedge CLOCK_50)
      if (KEY[0] == 1'b0)
         Ch_Sel <= 2'b00;
      else if (slow_count == 0)
         Ch_Sel <= Ch_Sel + 1'b1;

   // instantiate module mux_2bit_4to1 (S, U, V, W, X, M);
   mux_2bit_4to1 M3 (Ch_Sel, Ch0, Ch1, Ch2, Ch3, H3_Ch);
   mux_2bit_4to1 M2 (Ch_Sel, Ch1, Ch2, Ch3, Ch0, H2_Ch);
   mux_2bit_4to1 M1 (Ch_Sel, Ch2, Ch3, Ch0, Ch1, H1_Ch);
```

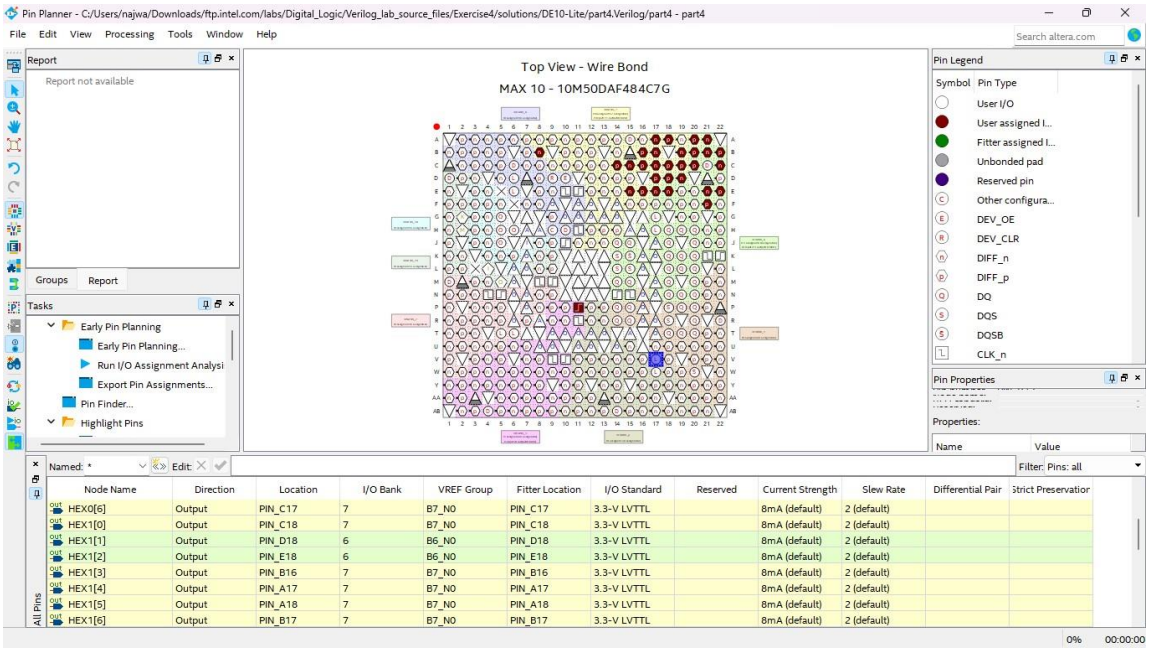
```
   // instantiate module mux_2bit_4to1 (S, U, V, W, X, M);
   mux_2bit_4to1 M3 (Ch_Sel, Ch0, Ch1, Ch2, Ch3, H3_Ch);
   mux_2bit_4to1 M2 (Ch_Sel, Ch1, Ch2, Ch3, Ch0, H2_Ch);
   mux_2bit_4to1 M1 (Ch_Sel, Ch2, Ch3, Ch0, Ch1, H1_Ch);
   mux_2bit_4to1 M0 (Ch_Sel, Ch3, Ch0, Ch1, Ch2, H0_Ch);

   // instantiate module char_7seg (C, Display);
   char_7seg H3 (H3_Ch, HEX3);
   char_7seg H2 (H2_Ch, HEX2);
   char_7seg H1 (H1_Ch, HEX1);
   char_7seg H0 (H0_Ch, HEX0);
endmodule

// Implements a 2-bit wide 4-to-1 multiplexer
module mux_2bit_4to1 (S, U, V, W, X, M);
   input [1:0] S, U, V, W, X;
   output [1:0] M;
   wire [1:0] U_V, W_X;          // used for first multiplexer stage, U_V selects either
                                 // U or V, and W_X selects either W or X

   // 2-bit wide 4-to-1 multiplexer first stage
   assign U_V[0] = (~S[0] & U[0]) | (S[0] & V[0]);
   assign U_V[1] = (~S[0] & U[1]) | (S[0] & V[1]);
   assign W_X[0] = (~S[0] & W[0]) | (S[0] & X[0]);
   assign W_X[1] = (~S[0] & W[1]) | (S[0] & X[1]);

   // 2-bit wide 4-to-1 multiplexer second stage
   assign M[0] = (~S[1] & U_V[0]) | (S[1] & W_X[0]);
   assign M[1] = (~S[1] & U_V[1]) | (S[1] & W_X[1]);

endmodule

// Converts 2-bit input code on C1-0 into 7-bit code that produces
```

```
// Converts 2-bit input code on C1-0 into 7-bit code that produces
// a character on a 7-segment display. The conversion is defined by:
//     C 1 0   Char
//    -----------------
//       0 0   'd'
//       0 1   'E'
//       1 0   '1'
//       1 1   '0'
//
module char_7seg (C, Display);
   input [1:0] C;          // input code
   output [0:6] Display;   // output 7-seg code

   /*
    *       0
    *      ---
    *     |   |
    *    5|   |1
    *     | 6 |
    *      ---
    *     |   |
    *    4|   |2
    *     |   |
    *      ---
    *       3
    */
   // the following equations describe HEX0[0-6] in SOP form
   assign Display[0] = ~C[0];
   assign Display[1] = ~C[1] & C[0];
   assign Display[2] = ~C[1] & C[0];
   assign Display[3] = C[1] & ~C[0];
   assign Display[4] = C[1] & ~C[0];
   assign Display[5] = ~C[0];
   assign Display[6] = C[1];
```

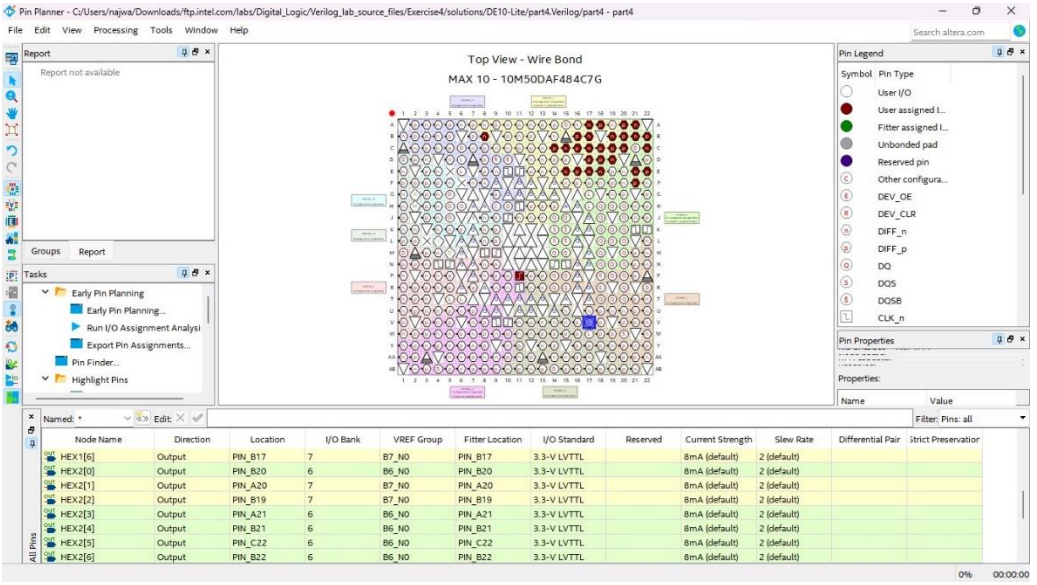
Modul Praktikum

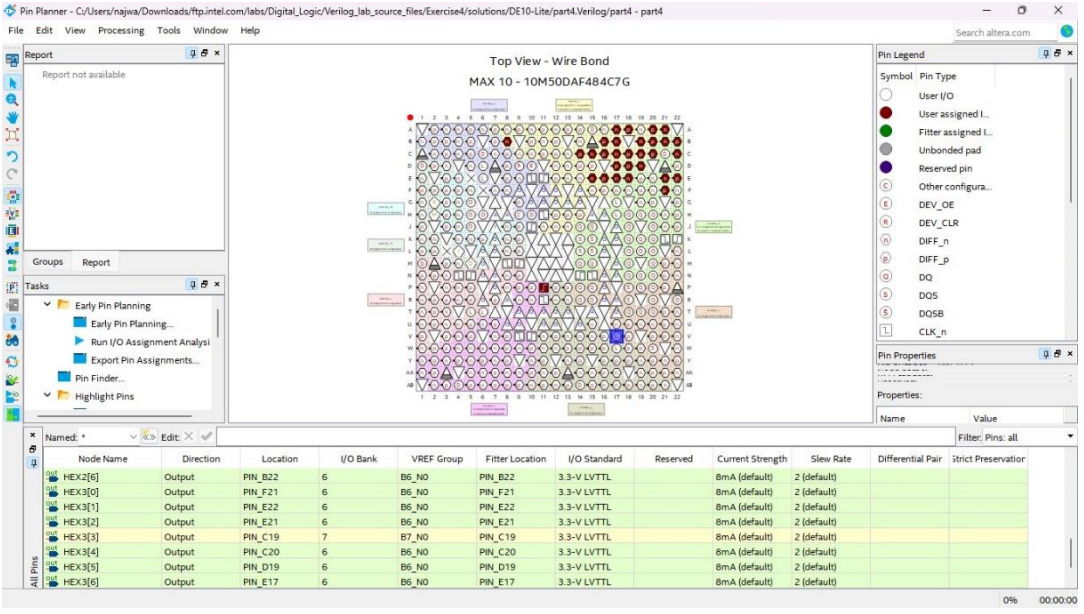
```
//       0 0   'd'
//       0 1   'E'
//       1 0   '1'
//       1 1   '0'
//
module char_7seg (C, Display);
   input [1:0] C;          // input code
   output [0:6] Display;   // output 7-seg code

   /*
    *       0
    *      ---
    *     |   |
    *    5|   |1
    *     | 6 |
    *      ---
    *     |   |
    *    4|   |2
    *     |   |
    *      ---
    *       3
    */
   // the following equations describe HEX0[0-6] in SOP form
   assign Display[0] = ~C[0];
   assign Display[1] = ~C[1] & C[0];
   assign Display[2] = ~C[1] & C[0];
   assign Display[3] = C[1] & ~C[0];
   assign Display[4] = C[1] & ~C[0];
   assign Display[5] = ~C[0];
   assign Display[6] = C[1];
endmodule
```

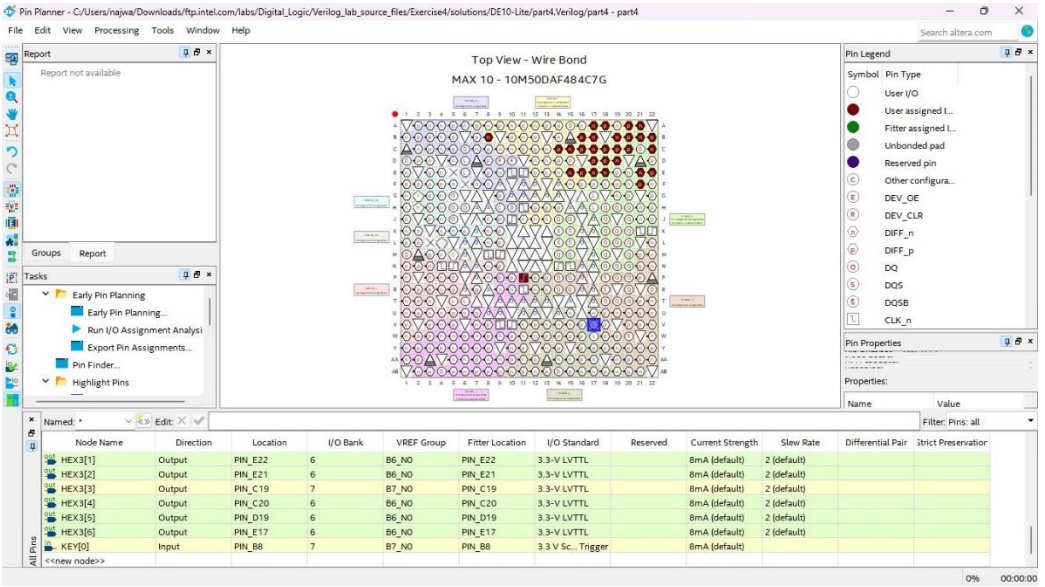


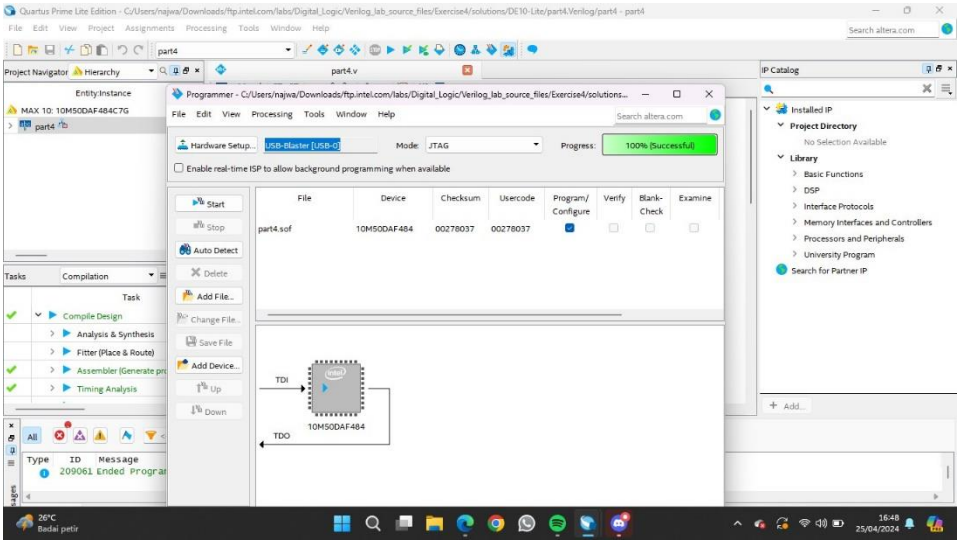


Modul Praktikum









Bagian V

Tingkatkan sirkuit Anda dari Bagian IV sehingga dapat memutar kata di seluruh tampilan 7-segmen di komputer Anda Papan seri DE. Pola perpindahan DE10-Lite ditunjukkan pada Tabel 3.

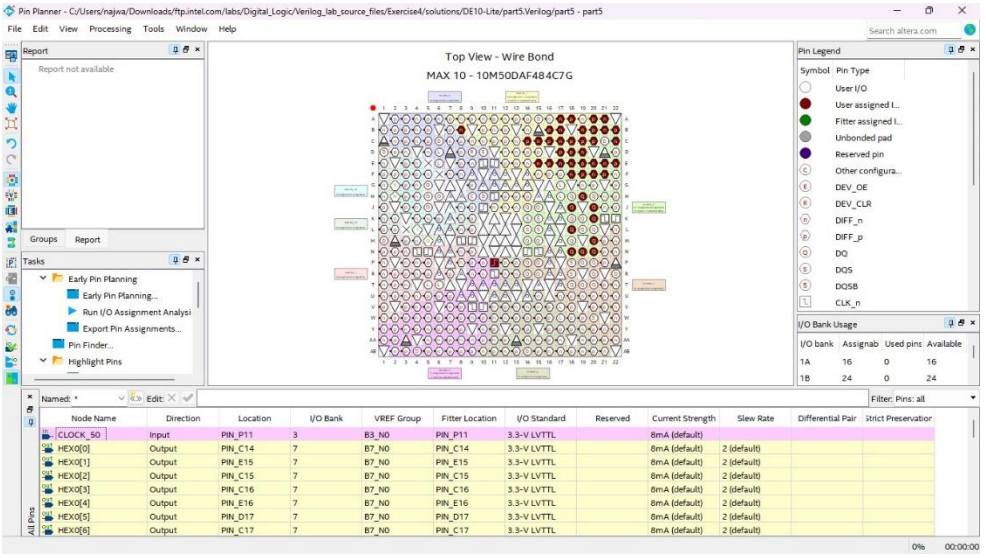
```
// Implements a circuit that rotates dE10 across the 7-segment displays HEX5 - HEX0
module part5 (KEY, CLOCK_50, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0);
   input [0:0] KEY;
   input CLOCK_50;
   output [0:6] HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;   // 7-seg displays

   reg [2:0] Ch_Sel; // mux select lines
   wire [2:0] Ch0, Ch1, Ch2, Ch3, Blank;
   wire [2:0] H5_Ch, H4_Ch, H3_Ch, H2_Ch, H1_Ch, H0_Ch;

   assign Ch0 = 3'b000;    // 'd'
   assign Ch1 = 3'b001;    // 'E'
   assign Ch2 = 3'b010;    // '1'
   assign Ch3 = 3'b011;    // '0'
   assign Blank = 3'b100;  // used to blank a 7-seg display (see module char_7seg)

   parameter m = 25;
   reg [m-1:0] slow_count;

   // Create a 1Hz 2-bit counter

   // A large counter to produce a 1 second (approx) enable from the 50 MHz Clock
   always @(posedge CLOCK_50)
      slow_count <= slow_count + 1'b1;

   // three-bit counter that uses a slow enable, and counts from 0 to 5
   always @ (posedge CLOCK_50)
      if (KEY[0] == 1'b0)
         Ch_Sel <= 3'b000;
      else if (slow_count == 0)
         if (Ch_Sel == 3'b101)
            Ch_Sel <= 3'b000;
         else
            Ch_Sel <= Ch_Sel + 1'b1;
```

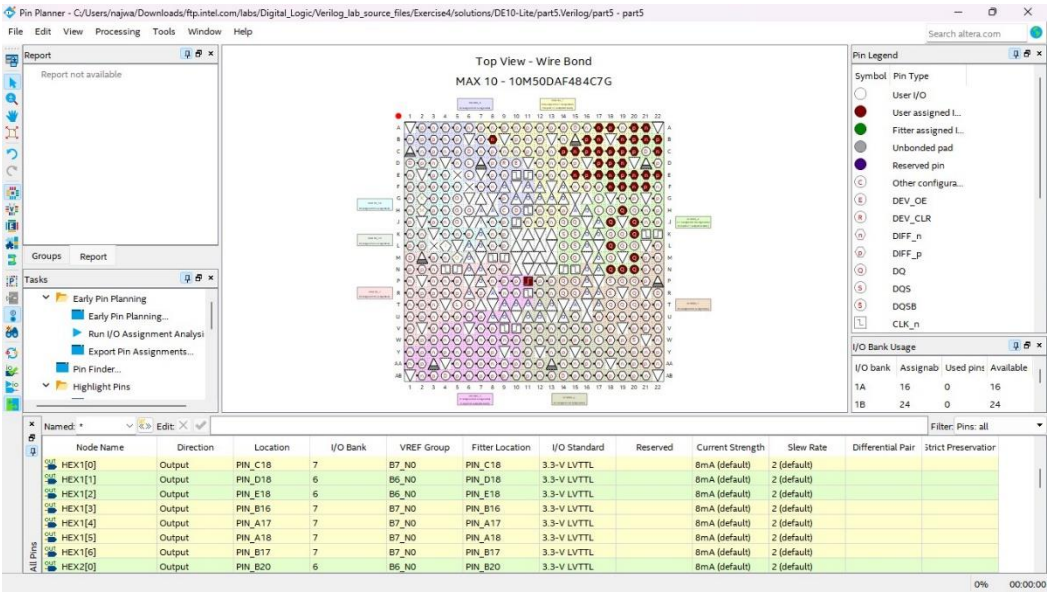
```
33            else
34               Ch_Sel <= Ch_Sel + 1'b1;
35
36        // instantiate module mux_3bit_6to1 (S, U, V, W, X, Y, Z, M) to
37        // create the multiplexer for each hex display
38
39        mux_3bit_6to1 M5 (Ch_Sel, Blank, Blank, Ch0, Ch1, Ch2, Ch3, H5_Ch);
40        mux_3bit_6to1 M4 (Ch_Sel, Blank, Ch0, Ch1, Ch2, Ch3, Blank, H4_Ch);
41        mux_3bit_6to1 M3 (Ch_Sel, Ch0, Ch1, Ch2, Ch3, Blank, Blank, H3_Ch);
42        mux_3bit_6to1 M2 (Ch_Sel, Ch1, Ch2, Ch3, Blank, Blank, Ch0, H2_Ch);
43        mux_3bit_6to1 M1 (Ch_Sel, Ch2, Ch3, Blank, Blank, Ch0, Ch1, H1_Ch);
44        mux_3bit_6to1 M0 (Ch_Sel, Ch3, Blank, Blank, Ch0, Ch1, Ch2, H0_Ch);
45
46        // iinstantiate module char_7seg (C, Display) to drive the hex displays
47        char_7seg H5 (H5_Ch, HEX5);
48        char_7seg H4 (H4_Ch, HEX4);
49        char_7seg H3 (H3_Ch, HEX3);
50        char_7seg H2 (H2_Ch, HEX2);
51        char_7seg H1 (H1_Ch, HEX1);
52        char_7seg H0 (H0_Ch, HEX0);
53    endmodule
54
55    // implements a 3-bit wide 6-to-1 multiplexer
56    module mux_3bit_6to1 (S, U, V, W, X, Y, Z, M);
57        input [2:0] S;
58        input [2:0] U, V, W, X, Y, Z;
59        output [2:0] M;
60        wire [1:4] m_0, m_1, m_2;  // four intermediate multiplexers in the mux tree for each
61
62        // 6-to-1 multiplexer for bit 0
63        assign m_0[1] = (~S[0] & U[0]) | (S[0] & V[0]);
64        assign m_0[2] = (~S[0] & W[0]) | (S[0] & X[0]);
65        assign m_0[3] = (~S[0] & Y[0]) | (S[0] & Z[0]);
66        assign m_0[4] = (~S[1] & m_0[1]) | (S[1] & m_0[2]);
```

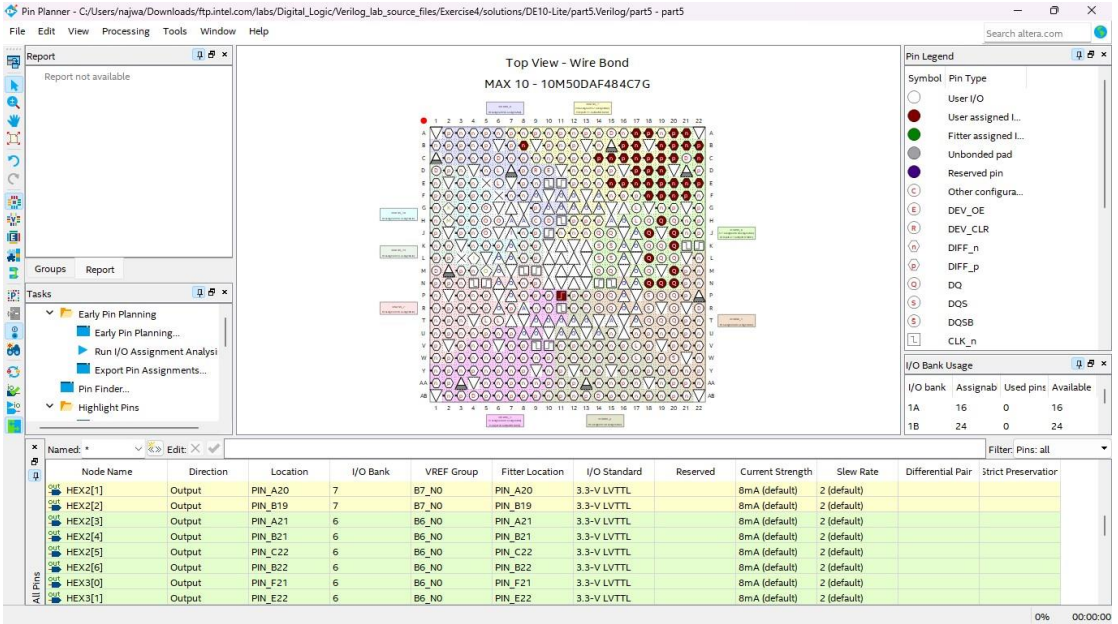
```
62        // 6-to-1 multiplexer for bit 0
63        assign m_0[1] = (~S[0] & U[0]) | (S[0] & V[0]);
64        assign m_0[2] = (~S[0] & W[0]) | (S[0] & X[0]);
65        assign m_0[3] = (~S[0] & Y[0]) | (S[0] & Z[0]);
66        assign m_0[4] = (~S[1] & m_0[1]) | (S[1] & m_0[2]);
67
68        assign M[0] = (~S[2] & m_0[4]) | (S[2] & m_0[3]);
69
70        // 6-to-1 multiplexer for bit 1
71        assign m_1[1] = (~S[0] & U[1]) | (S[0] & V[1]);
72        assign m_1[2] = (~S[0] & W[1]) | (S[0] & X[1]);
73        assign m_1[3] = (~S[0] & Y[1]) | (S[0] & Z[1]);
74        assign m_1[4] = (~S[1] & m_1[1]) | (S[1] & m_1[2]);
75
76        assign M[1] = (~S[2] & m_1[4]) | (S[2] & m_1[3]);
77
78        // 6-to-1 multiplexer for bit 2
79        assign m_2[1] = (~S[0] & U[2]) | (S[0] & V[2]);
80        assign m_2[2] = (~S[0] & W[2]) | (S[0] & X[2]);
81        assign m_2[3] = (~S[0] & Y[2]) | (S[0] & Z[2]);
82        assign m_2[4] = (~S[1] & m_2[1]) | (S[1] & m_2[2]);
83
84        assign M[2] = (~S[2] & m_2[4]) | (S[2] & m_2[3]);
85
86    endmodule
87
88    // Converts 3-bit input code on C2-0 into 7-bit code that produces
89    // a character on a 7-segment display. The conversion is defined by:
90    //     C 2 1 0 Char
91    //    ----------------
92    //       0 0 0 'd'
93    //       0 0 1 'E'
94    //       0 1 0 '1'
95    //       0 1 1 '0'
```

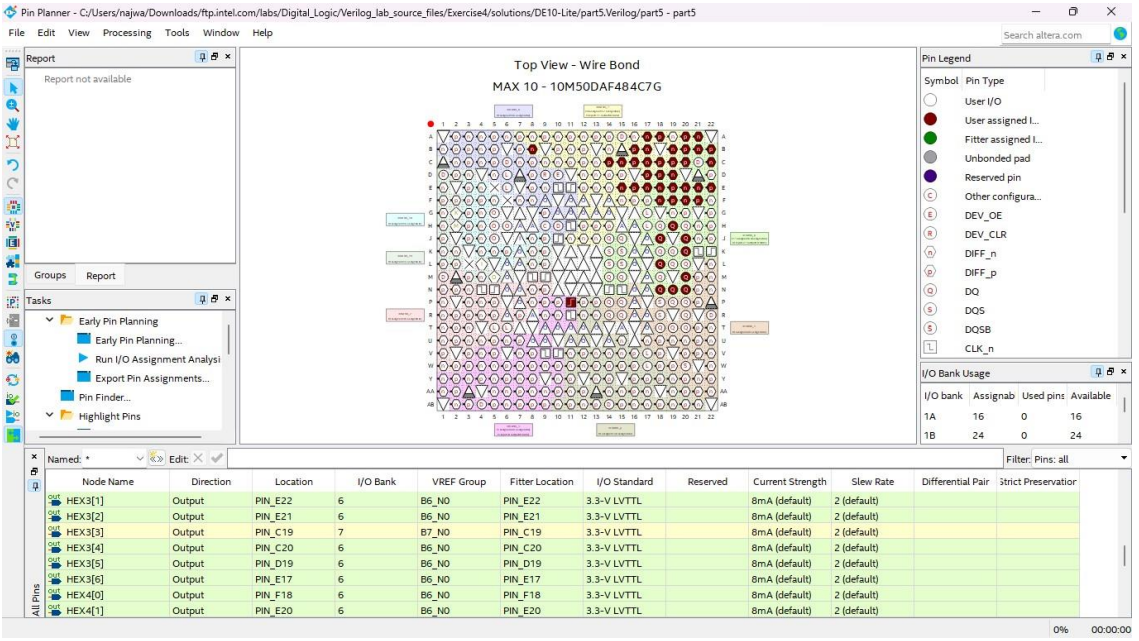
```
90    //     C 2 1 0 Char
91    //    ----------------
92    //       0 0 0 'd'
93    //       0 0 1 'E'
94    //       0 1 0 '1'
95    //       0 1 1 '0'
96    //       1 0 0 'Blank'
97    module char_7seg (C, Display);
98        input [2:0] C;     // input code
99        output [0:6] Display;   // output 7-seg code
100
101       /*
102        *       0
103        *      ---
104        *     |   |
105        *    5|   |1
106        *     | 6 |
107        *      ---
108        *     |   |
109        *    4|   |2
110        *     |   |
111        *      ---
112        *       3
113        */
114       // the following equations describe display functions in cannonical SOP form
115       assign Display[0] = ~C[0];
116       assign Display[1] = C[2] | (~C[1] & C[0]);
117       assign Display[2] = C[2] | (~C[1] & C[0]);
118       assign Display[3] = C[2] | (C[1] & ~C[0]);
119       assign Display[4] = C[2] | (C[1] & ~C[0]);
120       assign Display[5] = ~C[0];
121       assign Display[6] = C[2] | C[1];
122   endmodule
123
```

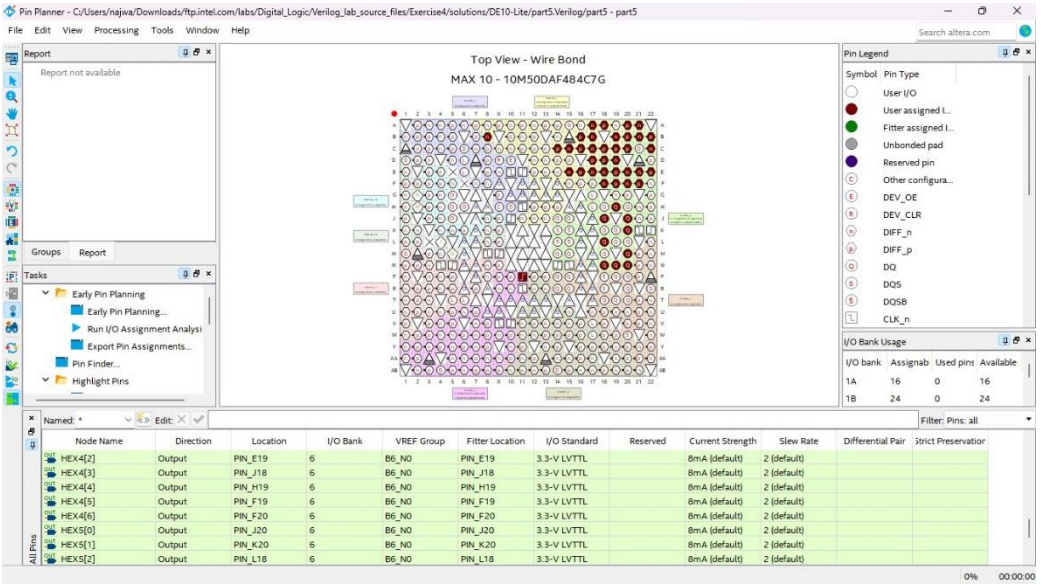
Modul Praktikum

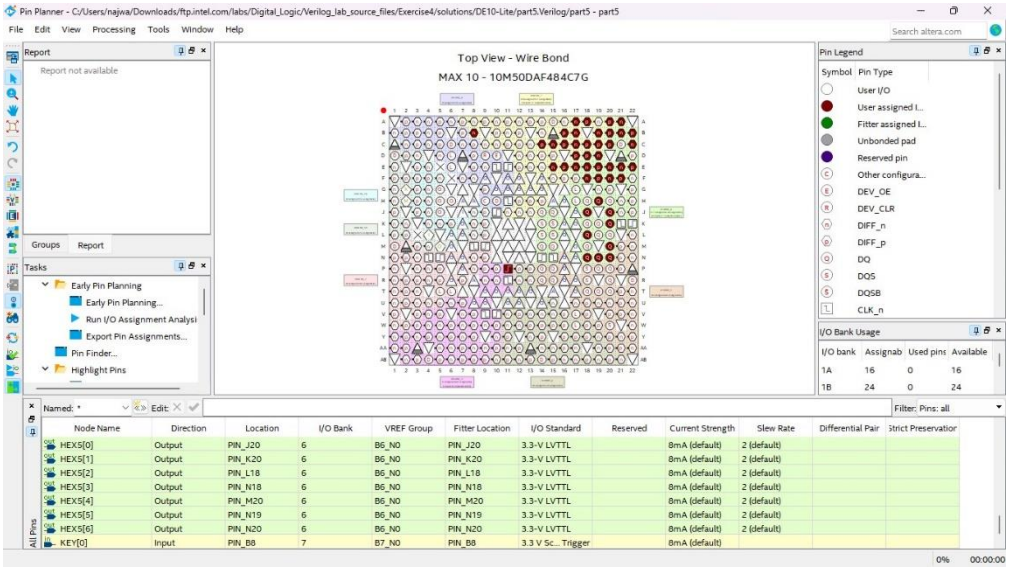




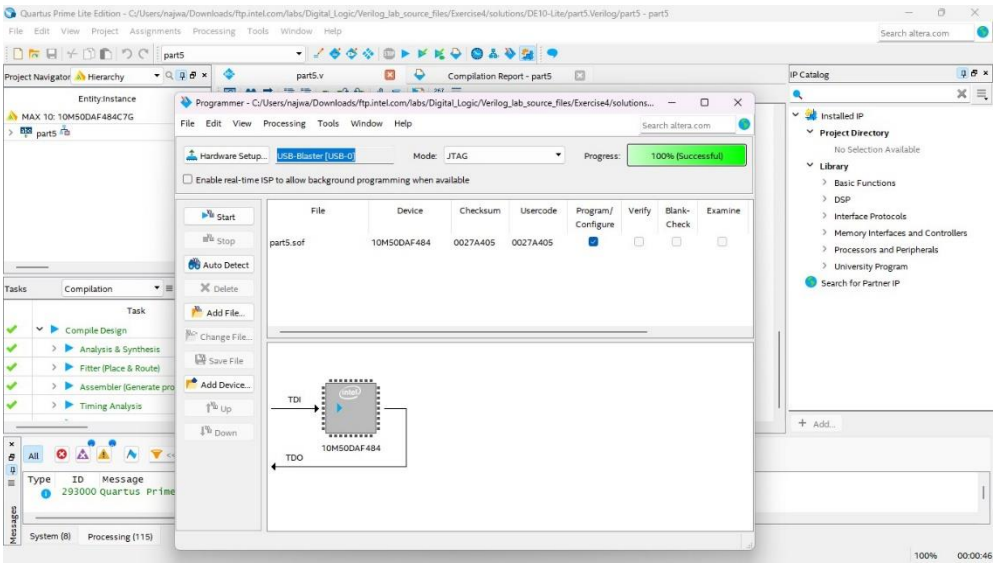


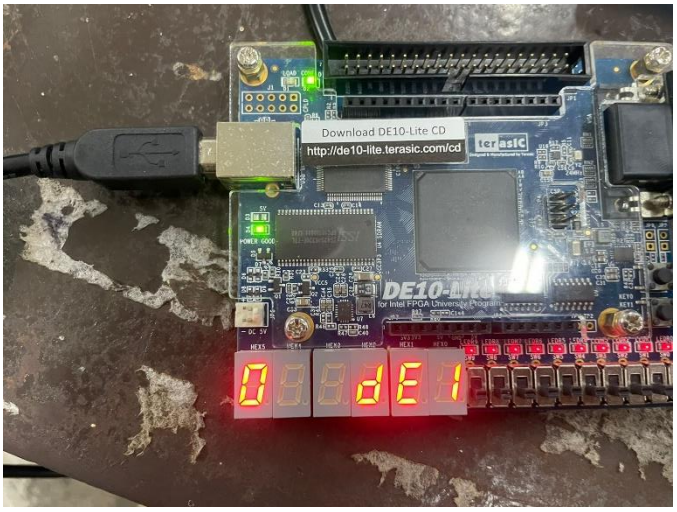

Pin Planner - C:/Users/najwa/Downloads/ftp.intel.com/labs/Digital_Logic/Verilog_lab_source_files/Exercise4/solutions/DE10-Lite/part5.Verilog/part5 - part5
File Edit View Processing Tools Window Help
Top View - Wire Bond
MAX 10 - 10M50DAF484C7G
Pin Legend
I/O Bank Usage
Node Name Direction Location I/O Bank VREF Group Fitter Location I/O Standard Reserved Current Strength Slew Rate Differential Pair Strict Preservation
HEX3[1] Output PIN_E22 6 B6_N0 PIN_E22 3.3-V LVTTL 8mA (default) 2 (default)
HEX3[2] Output PIN_E21 6 B6_N0 PIN_E21 3.3-V LVTTL 8mA (default) 2 (default)
HEX3[3] Output PIN_C19 7 B7_N0 PIN_C19 3.3-V LVTTL 8mA (default) 2 (default)
HEX3[4] Output PIN_C20 6 B6_N0 PIN_C20 3.3-V LVTTL 8mA (default) 2 (default)
HEX3[5] Output PIN_D19 6 B6_N0 PIN_D19 3.3-V LVTTL 8mA (default) 2 (default)
HEX3[6] Output PIN_E17 6 B6_N0 PIN_E17 3.3-V LVTTL 8mA (default) 2 (default)
HEX4[0] Output PIN_F18 6 B6_N0 PIN_F18 3.3-V LVTTL 8mA (default) 2 (default)
HEX4[1] Output PIN_E20 6 B6_N0 PIN_E20 3.3-V LVTTL 8mA (default) 2 (default)

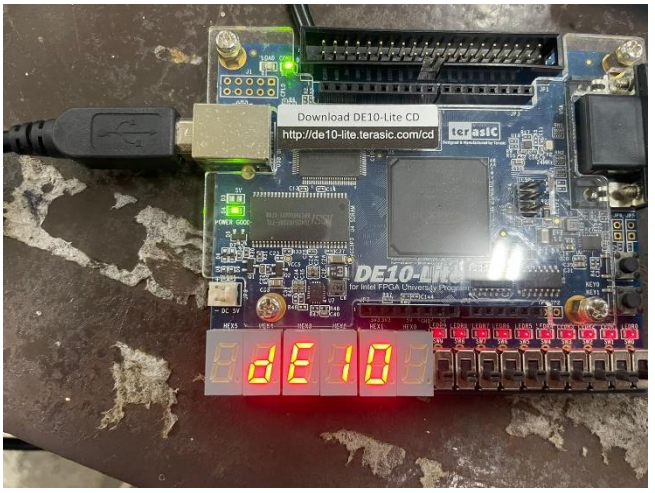

Pin Planner - C:/Users/najwa/Downloads/ftp.intel.com/labs/Digital_Logic/Verilog_lab_source_files/Exercise4/solutions/DE10-Lite/part5.Verilog/part5 - part5
Top View - Wire Bond
MAX 10 - 10M50DAF484C7G
HEX4[2] Output PIN_E19 6 B6_N0 PIN_E19 3.3-V LVTTL 8mA (default) 2 (default)
HEX4[3] Output PIN_J18 6 B6_N0 PIN_J18 3.3-V LVTTL 8mA (default) 2 (default)
HEX4[4] Output PIN_H19 6 B6_N0 PIN_H19 3.3-V LVTTL 8mA (default) 2 (default)
HEX4[5] Output PIN_F19 6 B6_N0 PIN_F19 3.3-V LVTTL 8mA (default) 2 (default)
HEX4[6] Output PIN_F20 6 B6_N0 PIN_F20 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[0] Output PIN_J20 6 B6_N0 PIN_J20 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[1] Output PIN_K20 6 B6_N0 PIN_K20 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[2] Output PIN_L18 6 B6_N0 PIN_L18 3.3-V LVTTL 8mA (default) 2 (default)

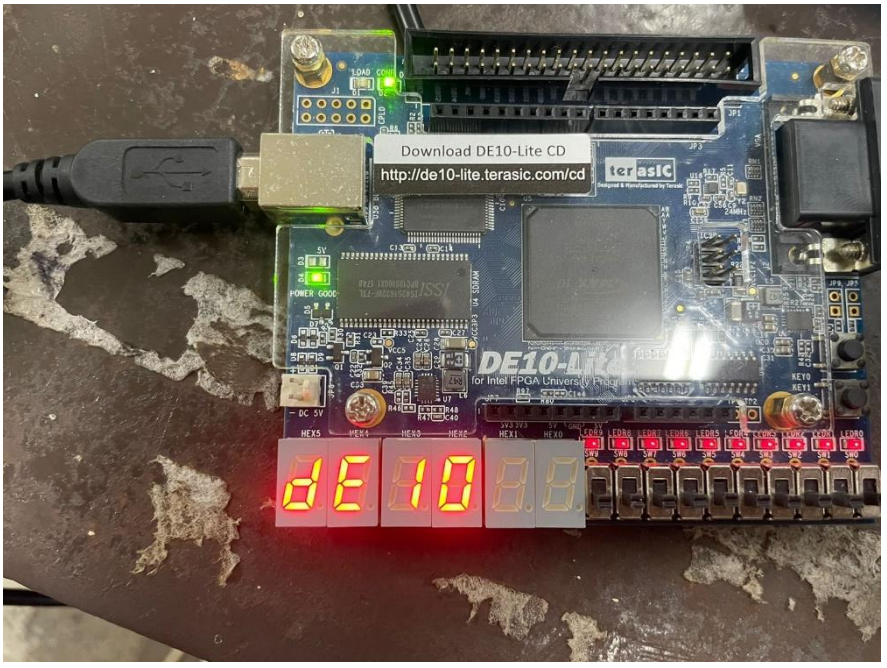

Pin Planner - C:/Users/najwa/Downloads/ftp.intel.com/labs/Digital_Logic/Verilog_lab_source_files/Exercise4/solutions/DE10-Lite/part5.Verilog/part5 - part5
Top View - Wire Bond
MAX 10 - 10M50DAF484C7G
HEX5[0] Output PIN_J20 6 B6_N0 PIN_J20 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[1] Output PIN_K20 6 B6_N0 PIN_K20 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[2] Output PIN_L18 6 B6_N0 PIN_L18 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[3] Output PIN_N18 6 B6_N0 PIN_N18 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[4] Output PIN_M20 6 B6_N0 PIN_M20 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[5] Output PIN_N19 6 B6_N0 PIN_N19 3.3-V LVTTL 8mA (default) 2 (default)
HEX5[6] Output PIN_N20 6 B6_N0 PIN_N20 3.3-V LVTTL 8mA (default) 2 (default)
KEY[0] Input PIN_B8 7 B7_N0 PIN_B8 3.3 V Sc... Trigger 8mA (default)

Modul Praktikum









SELAMAT MENGERJAKAN