Praktikum Sistem Digital Lanjut
Percobaan 4: Desain Rangkaian Kombinasional

1 Tujuan dan Sasaran

Kegiatan praktikum ini bertujuan untuk mendesain suatu rangkaian kombinasional. Rangkaian kombinasional ini mengintegrasikan pencacah up-down 4-bit (komponen sekuensial, asinkron) dan dekoder hexa-to-7Segmen 1 digit. Dekoder ini akan mendekodekan bilangan desimal 1 digit (0-9) dan bilangan hexadesimal 1 digit (0x0-0xF) serta menampilkannya di LED. Nilai bilangan masukan ditentukan oleh keluaran pencacah up-down (asinkron) yang diatur oleh tombol up/down/reset/set. Pemilihan mode pencacah bilangan (desimal atau hexa) ditentukan oleh sinyal Dec/Hexa.

Sasaran kegiatan praktikum adalah:

1. Praktikan dapat membuat modul pencacah hexa/desimal up-down asinkron dengan set-reset dan konverter hexa-ke-7 segmen;
2. Praktikan dapat memahami tentang *reusable module* (perancangan berbasis komponen) dengan menyusun rangkaian kombinasional yang terdiri dari komponen-komponen modul pencacah dan dekoder hexa-to-7 segmen;
3. Praktikan dapat mengimplementasikan rangkaian kombinasional tersebut ke FPGA. Praktikan dapat menganalisis hasil implementasi rangkaian tersebut, yaitu skematik RTL dan utilisasi/penggunaan device untuk desain tersebut;
4. Praktikan dapat menganalisis perilaku masukan-keluaran desain di board Starter Kit;

2 Alat dan Bahan

Alat dan bahan yang digunakan adalah:

1. Board Starter Kit Spartan-3E berbasis Xilinx FPGA XC3S500E-4FG320C;

   Modul I/O yang akan digunakan dalam praktikum:

   a) 4 buah tombol push-button (BTN North, BTN East, BTN South, BTN West);

   b) 1 buah saklar geser (SW0);

   c) 8 buah LED (LD0-7);
2. Kabel USB dengan konektor tipe-B;
3. Power supplai DC 5 Volt;
4. Software Xilinx ISE Webpack 11.1;

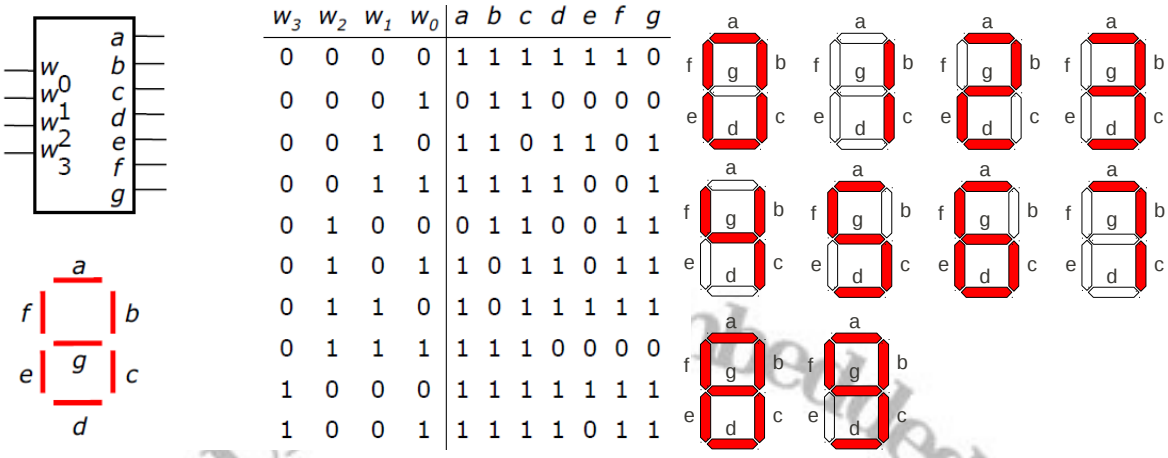
3 Dasar Teori

Di bab ini akan dijelaskan tentang dekoder hexadesimal-ke-7 segmen dan pencacah up-down hexadesimal/desimal.

3.1 Dekoder BCD-ke-7 Segmen

Dekoder BCD-ke-7 segmen ini mengkonversikan satu digit BCD (*binary-coded decimal*, kode biner untuk bilangan desimal) untuk mendrive display berorientasi digit, yaitu 7-segmen. Lebih

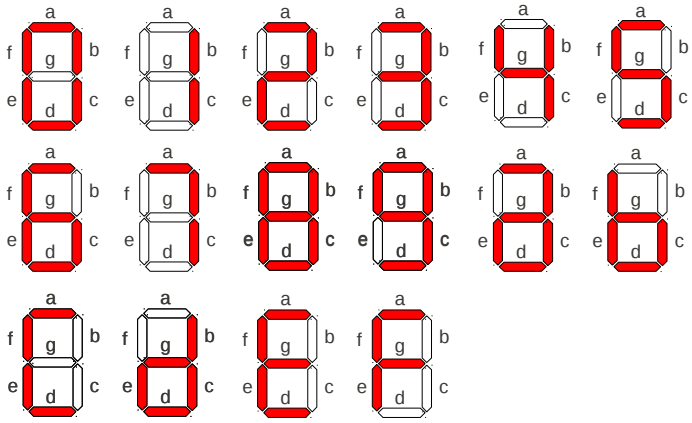
luasnya adalah berupa hexadesimal-ke-7segmen. Display ini mempunyai 7 segmen. Tiap segmen diimplementasikan dengan 1 LED yang akan menyala saat didrive dengan sinyal bernilai '1' (lihat Gambar 1).



w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Gambar 1: Dekoder BCD ke 7-segmen beserta tabel kebenaran fungsi logikanya

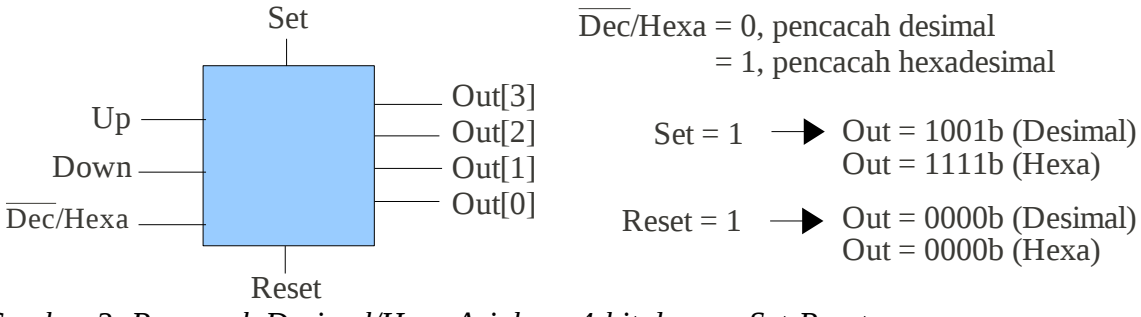
Dekoder biner-ke -7segmen menambahkan masukan 1010 sampai 1111 untuk menghasilkan keluaran tampilan A, B, C, D, E dan F.



Gambar 2: Tampilan keluaran dekoder hexa-ke-7 segmen

3.2 Pencacah Desimal/Hexa Asinkron 4-Bit (Up-Down)

Pencacah asinkron ini mempunyai 5 jalur masukan kontrol up dan down, set, reset dan selektor $\overline{\text{Dec}}$/Hexa serta 4 jalur keluaran (Gambar 3). Kontrol Up menambah isi register keluaran dengan 1, sedangkan kontrol Down mengurangkan isi register keluaran dengan 1. Kontrol Set mengeset isi register dengan 9 (desimal) atau 0xFF (hexa), sedangkan kontrol Reset mereset isi register ke 0.



Gambar 3: Pencacah Desimal/Hexa Asinkron 4-bit dengan Set-Reset

4 Cara Kerja

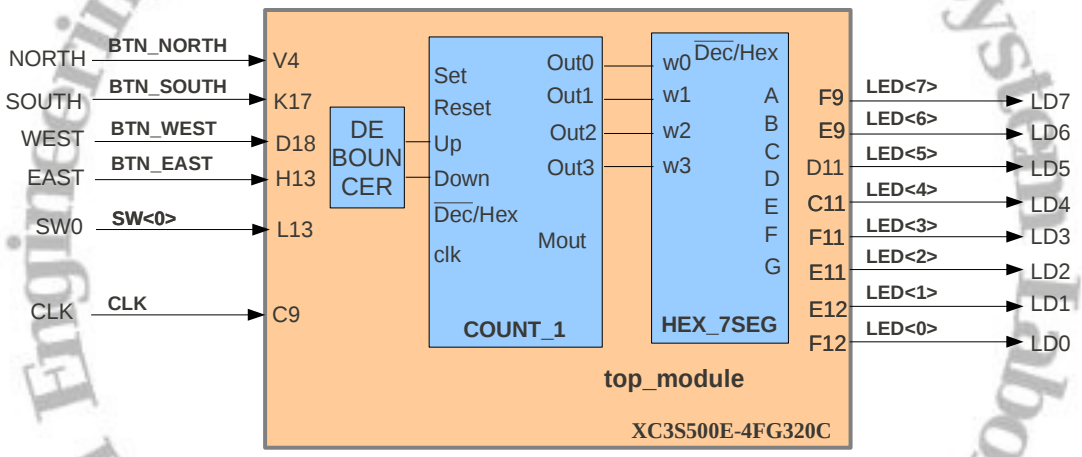
Kegiatan praktikum dilakukan untuk memenuhi kebutuhan desain yang diinginkan. Setiap tahap dilakukan berdasarkan cara kerja yang diuraikan dalam Subbab 4.2. Hasil kegiatan praktikum yang ditandai dengan ikon dituliskan dalam Lembar Isian Kegiatan. Laporan akhir disusun dengan menyertakan hasil kegiatan praktikum.

4.1 Kebutuhan Desain

Desain yang akan diimplementasikan dalam praktikum ini adalah:

1. Dekoder hexadesimal-ke-7 segment (HEX_7SEG)
2. Pencacah asinkron up-down desimal/hexa 1 digit (COUNT_1)
3. 2 Buah modul debouncer (DEBOUNCER_UP, DEBOUNCER_DOWN)

Desain tersebut dibuat sebagai modul dan akan diintegrasikan oleh satu top_module. Diagram blok top_module (port masukan dan keluaran) beserta koneksinya diperlihatkan dalam Gambar 4 dan Tabel 1. Detail tentang desain modul pencacah (rangkaian sekuensial) akan dijabarkan di praktikum #5. Modul debouncer digunakan untuk menstabilkan penekanan tombol BTN_WEST (Down) dan BTN_EAST (Up).



Gambar 4: Diagram blok sistem yang dirancang

Nama sinyal (top)	top_module	module
BTN_NORTH	Set	COUNT_1
BTN_WEST	Down_in	DEBOUNCER_DOWN
BTN_EAST	Up_in	DEBOUNCER_UP
BTN_SOUTH	Reset	COUNT_1
SW<0>	$\overline{\text{Dec/Hex}}$	COUNT_1
CLK	clk	Semua, kecuali HEX_7SEG
LED<7>	$\overline{\text{Dec/Hex}}$ Status atau Mout	HEX_7SEG
LED<6>	SEG_A atau A	HEX_7SEG
LED<5>	SEG_B atau B	HEX_7SEG
LED<4>	SEG_C atau C	HEX_7SEG
LED<3>	SEG_D atau D	HEX_7SEG
LED<2>	SEG_E atau E	HEX_7SEG
LED<1>	SEG_F atau F	HEX_7SEG
LED<0>	SEG_G atau G	HEX_7SEG

Tabel 1: Koneksi sinyal di top_module dengan FPGA

4.2 Langkah Kerja

Kegiatan praktikum meliputi hal-hal sebagai berikut:

1. Menuliskan kode HDL untuk kedua blok sistem (COUNT_1 dan HEX_7SEG) sebagai modul. Masukkan modul tersebut dalam proyek *desain_kombinasional*;
2. Mengaplikasikan modul-modul tersebut dalam satu top_module (kombinasional_top);
3. Menambah file konstrain desain_kombinasional.ucf ;
4. Mensintesis dan mengimplementasikan kombinasional_top;
    1. Melihat skematik RTL atau skematik teknologi ;
    2. Melihat utilisasi device yang digunakan oleh desain;
5. Membangkitkan file programming kombinasional_top.bit;
6. Memprogram file *.bit tersebut dan mengamati perilaku sistem;

4.2.1 Membuat Proyek Baru

Langkah-langkah membuat proyek baru:

1. Pilih menu **File→New Project** (Alt+F W). Dialog pop-up **New Project Wizard** akan muncul. Ketikkan *field* **Name** dengan nama proyek (Format: <nama_kelompok>-Modul4-DesainKombinasional). Browse lokasi proyek/**Location** (folder $HOME_DIR/<nama_kelompok>). Isi field **Description** dengan penjelasan tentang proyek. Tipe source top-level menggunakan HDL

    Tuliskan field **Name**, **Location** dan **Description** dalam lembar kegiatan;
2. Klik tombol **Next**. Jendela **Device Properties** muncul. Pilih **Family** (Spartan3E), **Device** (XC3S500E), **Package** (FG320) dan **Speed** (-4). Properti ini adalah device-dependent (Starter Kit menggunakan FPGA XC3S500E-4FG320C), jadi harus dimasukkan dengan benar. Set **Preferred Language** dengan Verilog

    Catat field Family, Device, Package dan Speed ini dalam lembar kegiatan;

4.2.2 Menambah Modul Desain dan Konstrain

Modul yang perlu ditambahkan ada 2 buah, ditambah modul debouncer (rangkaian switch debouncer). Langkah-langkahnya adalah sebagai berikut:

1. Buat modul-modul verilog seperti dalam Tabel 2. Masukkan dalam proyek;

No	Nama Modul	Masukan	Keluaran	Keterangan
1.	count_1	set reset up down mode clk	y_out[3:0] mode_out	
2.	hex_7seg	w_in[3:0]	A, B, C, D, E, F, G	
3.	debouncer	clk in_state	out_state	Switch debouncer

Tabel 2: Nama modul-modul desain yang akan dibuat

2. Edit file count_1.v, hex_7seg.v dan debouncer.v. Listing kode sumber HDL untuk

semua desain modul ada dalam Lampiran;

Jelaskan perilaku kode HDL di kedua file tersebut

3. Tambahkan file konstrain desain_kombinasional.ucf. Isi file tersebut ada dalam Lampiran;

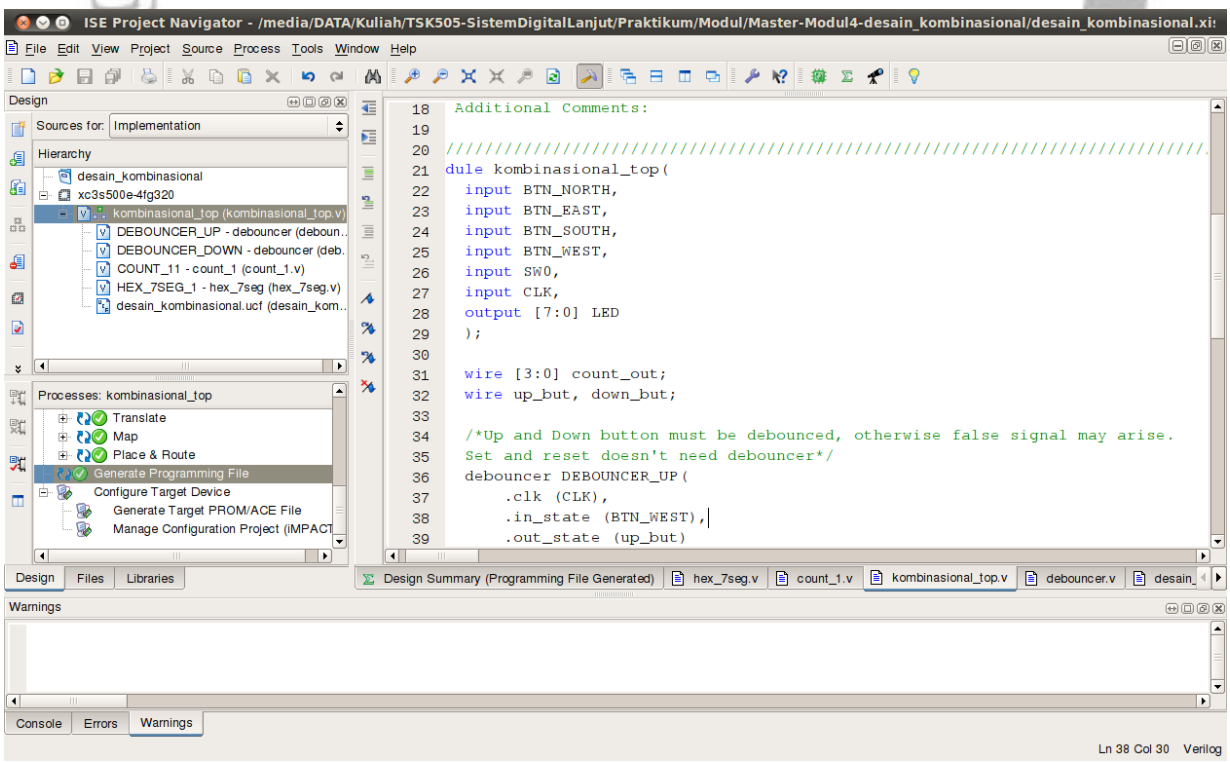
4.2.3 Mengaplikasikan Modul dalam Desain (Top Module)

Ketiga modul tersebut di atas akan diaplikasikan dalam satu top_module (desain berbasis komponen). Langkah yang diperlukan adalah:

1. Buat top_module yang mengintegrasikan kedua modul tersebut dengan nama kombinasional_top. Definisi port masukan dan keluaran dari kombinasional_top ini ditunjukkan dalam Tabel 3;
2. Panggil ketiga modul dari kombinasional_top ini dan interkoneksikan port-portnya. Listing kode sumber HDL untuk semua kombinasional_top.v ada dalam Lampiran. Tampilan ISE setelah integrasi ditunjukkan dalam Gambar 5;

No	Nama Modul	Masukan	Keluaran	Keterangan
1.	kombinasional_top	BTN_NORTH, BTN_EAST, BTN_SOUTH, BTN_WEST, SW0, CLK	LED[7:0]	

Tabel 3: Nama port masukan dan keluaran di modul kombinasional_top.v



Gambar 5: Modul-modul yang diperlukan diintegrasikan ke kombinasional_top

4.2.4 Sintesis dan Implementasikan Top Modul

Modul kombinasional_top akan disintesis dan diimplementasikan. Langkah-langkahnya adalah sebagai berikut:

1. Cek apakah modul kombinasional_top sudah menjadi top_module. Hanya top_module yang bisa disintesis dan diimplementasikan. Jika belum, klik kanan modul tersebut dan pilih 'Set as Top Module';
2. Bangkitkan skematik RTL dari kombinasional_top. Klik “View RTL Schematic” dari proses Synthesize dan juga “View Technology Schematic”. Tambahkan elemen yang tersedia dan klik tombol “Create Schematic”. Lihat blok dari elemen dan bagaimana isinya (rangkaian logikanya);

   Simpan gambar blok elemen beserta rangkaian skematiknya dalam file gambar (dengan snapshot). Skematik ini beserta penjelasannya harus dilampirkan di laporan

   Jelaskan perbedaan antara skematik RTL dan skematik teknologi (secara terminologi dan konseptual)
3. Sintesis dan implemen kombinasional_top dengan mengklik kanan proses “Implement Design” dan pilih “Rerun All”;

   Lihat ringkasan laporan desain dari implementasi kombinasional_top tersebut dengan melihat tab Design Summary. Jika belum ada, klik menu **Project→Design Summary/Report**. Catat hasilnya untuk mengisi tabel utilisasi device di lembar kegiatan;

4.2.5 Memprogram FPGA dan Pengamatan Perilaku Sistem

Langkah percobaan:

1. Bangkitkan file programming kombinasional_top.bit;
2. Jalankan Xilinx iMPACT jika belum dijalankan. Pastikan kabel USB telah terpasang di starter kit dan komputer;
3. Pasang board ekstension 7-segmen ke J1 dan J2 (minta petunjuk asisten)
4. Dari Xilinx iMPACT, klik ganda “Boundary Scan”. Kemudian klik kanan dan pilih Initialize Chain (Ctrl+I) untuk menginisialisasi Boundary Scan;
5. Yang diperlukan adalah mengkonfigurasi FPGA, sedangkan flash XCF04S dan CPLD XC2C64 dibiarkan bypass. Klik kanan device FPGA dan pilih “Assign New Configuration File”. Pilih file kombinasional_top.bit untuk diprogram ke FPGA. Nama file konfigurasi akan tampil di bawah device FPGA;
6. Klik kanan device FPGA dan pilih “Program” untuk mengkonfigurasi FPGA;

   Amati perilaku sistem dan isi tabel di lembar kegiatan. Tekan tombol set, reset, up, down dan amati LED yang menyala dan angka yang ditunjukkan di 7-segmen. Geser saklar SW0 dan ulangi kegiatan sebelumnya;

4.3 Lampiran Kode HDL

Nama file modul bersesuaian dengan nama modulnya. Misalnya: modul hex_7seg akan berada dalam file hex_7seg.v. File desain_kombinasional.ucf juga disertakan.

4.3.1 hex_7seg: hex_7seg.v

```
module hex_7seg(
    input [3:0] w_in,
    output A, output A_buf,
      output B, output B_buf,
      output C, output C_buf,
      output D, output D_buf,
      output E, output E_buf,
      output F, output F_buf,
      output G, output G_buf
    );

      reg [6:0] seg_value; /*A,B,C,D,E,F,G value*/

      always @(w_in) begin
            case (w_in)
                /*See module text*/
                4'b0000: seg_value <= 7'b1111110; /*0*/
                4'b0001: seg_value <= 7'b0110000; /*1*/
                4'b0010: seg_value <= 7'b1101101; /*2*/
                4'b0011: seg_value <= 7'b1111001; /*3*/
                4'b0100: seg_value <= 7'b0110011; /*4*/
                4'b0101: seg_value <= 7'b1011011; /*5*/
                4'b0110: seg_value <= 7'b1011111; /*6*/
                4'b0111: seg_value <= 7'b1110000; /*7*/
                4'b1000: seg_value <= 7'b1111111; /*8*/
                4'b1001: seg_value <= 7'b1111011; /*9*/
                4'b1010: seg_value <= 7'b1111101; /*a*/
                4'b1011: seg_value <= 7'b0011111; /*b*/
                4'b1100: seg_value <= 7'b1001110; /*C*/
                4'b1101: seg_value <= 7'b0111101; /*d*/
                4'b1110: seg_value <= 7'b1001111; /*E*/
                4'b1111: seg_value <= 7'b1000111; /*F*/
                default: seg_value <= 7'b0000001; /*-,but never happen*/
            endcase
      end
      assign {A,B,C,D,E,F,G} = seg_value;
      assign {A_buf,B_buf,C_buf,D_buf,E_buf,F_buf,G_buf} = ~seg_value;

endmodule
```

4.3.2 count_1: count_1.v

```
module count_1(
    input set_in,
    input reset_in,
    input up,
    input down,
    input mode,
    input clk,
    output [3:0] y_out,
    output mode_out,
    output mode_out_buf
    );
```

```
	reg delay_up;
	reg delay_down;
	reg [3:0] nmax; /*max value of counter: dec=9, hex=0xf*/

	reg [3:0] cur_state, next_state;
	reg [3:0] out_reg;

	initial begin
		cur_state <= 4'b0000;
		next_state <= 4'b0000;
	end

	/*Mode: dec or hexa*/
	always @(mode)
	begin
		nmax <= (mode==0)?4'b1001:4'b1111;
	end

	/*Current State Logic : Sequential*/
	always @(posedge clk)
	begin
		delay_down <= down;
		delay_up <= up;
		if (reset_in==1) cur_state <= 0; /*Reset*/
		else if (set_in==1) cur_state <= nmax; /*Set*/
		else cur_state <= next_state;
	end

	/*Next state logic: combinational*/
	always @*
	begin
		if ((up==1) & (delay_up==0))
			next_state = (cur_state==nmax)?nmax:cur_state+1;
		else if ((down==1) & (delay_down==0))
			next_state = (cur_state==0)?0:cur_state-1;
		else next_state = cur_state;
	end

	/*Cur State & Output*/
	always @*
	begin
		out_reg <= cur_state;
	end

	assign y_out = out_reg;
	assign mode_out = mode;
	assign mode_out_buf = !mode; /*dot in 7-segment*/

endmodule
```

4.3.3 debouncer: debouncer.v

```
module debouncer(
    input clk,
    input in_state,
    output out_state
    );

	parameter NBITS = 16;

	reg [NBITS-1:0] COUNT;
	reg PB_sync_0;
```

```
	reg PB_sync_1;
	reg PB_state;

	always @(posedge clk) PB_sync_0 <= ~in_state;
	always @(posedge clk) PB_sync_1 <= PB_sync_0;

	wire PB_idle = (PB_state==PB_sync_1);
	wire max_COUNT = &COUNT;

	always @(posedge clk)
	begin
		if (PB_idle) COUNT <= 0;
		else begin
			COUNT <= COUNT + 1;
			if (max_COUNT) PB_state  <= ~PB_state;
		end
	end

	assign out_state = PB_state;

endmodule
```

4.3.4 kombinasional_top: kombinasional_top.v

```
module kombinasional_top(
    input BTN_NORTH,
    input BTN_EAST,
    input BTN_SOUTH,
    input BTN_WEST,
    input SW0,
    input CLK,
    output [7:0] LED, /*to 7 LEDs*/
    output [6:0] LED_BUF, /*Led mirror: to real 7-segment*/
    output MOD_BUF /*Dot in 7-segment*/
    );

    wire [3:0] count_out;
    wire up_but, down_but;

    /*Up and Down button must be debounced, otherwise false signal
may arise.
    Set and reset doesn't need debouncer*/
    debouncer DEBOUNCER_UP(
        .clk (CLK),
        .in_state (BTN_EAST),
        .out_state (up_but)
    );

    debouncer DEBOUNCER_DOWN(
        .clk (CLK),
        .in_state (BTN_WEST),
        .out_state (down_but)
    );

    count_1 COUNT_11(
        .set_in (BTN_NORTH), /*Set button*/
        .reset_in (BTN_SOUTH), /*Reset button*/
        .up (up_but),
        .down (down_but),
        .mode (SW0),
        .clk (CLK),
        .y_out (count_out),
```

```
            .mode_out (LED[7]),
            .mode_out_buf (MOD_BUF)
      );

      hex_7seg HEX_7SEG_1(
            .w_in (count_out),
            .A (LED[0]), .A_buf (LED_BUF[0]),
            .B (LED[1]), .B_buf (LED_BUF[1]),
            .C (LED[2]), .C_buf (LED_BUF[2]),
            .D (LED[3]), .D_buf (LED_BUF[3]),
            .E (LED[4]), .E_buf (LED_BUF[4]),
            .F (LED[5]), .F_buf (LED_BUF[5]),
            .G (LED[6]), .G_buf (LED_BUF[6])
      );

endmodule
```

4.3.5 desain_kombinasional.ucf

```
# Period constraint for 50MHz operation
NET "CLK" PERIOD = 20.0ns HIGH 50%;
# soldered 50MHz Clock.
NET "CLK" LOC = "C9" | IOSTANDARD = LVTTL;
#
# Koneksi tombol tekan
NET "BTN_EAST" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_NORTH" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_SOUTH" LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_WEST" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;
# Switch SW<0>
NET "SW0" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
# Koneksi LED
NET "LED<7>" LOC = "F9"| IOSTANDARD= LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<6>" LOC = "E9"| IOSTANDARD= LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<5>" LOC = "D11"| IOSTANDARD= LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<4>" LOC = "C11"| IOSTANDARD= LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<3>" LOC = "F11"| IOSTANDARD= LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<2>" LOC = "E11"| IOSTANDARD= LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<1>" LOC = "E12"| IOSTANDARD= LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<0>" LOC = "F12"| IOSTANDARD= LVTTL | SLEW = SLOW | DRIVE = 8 ;
# Koneksi 7 Segmen
NET "MOD_BUF" LOC = "F7" |IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; #Dot
NET "LED_BUF<6>" LOC = "C5" |IOSTANDARD = LVTTL |SLEW = SLOW | DRIVE = 8 ;
#seg_G
NET "LED_BUF<5>" LOC = "D5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#seg_F
NET "LED_BUF<4>" LOC = "B6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#seg_E
NET "LED_BUF<3>" LOC = "E7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#seg_D
NET "LED_BUF<2>" LOC = "A6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#seg_C
NET "LED_BUF<1>" LOC = "B4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#seg_B
NET "LED_BUF<0>" LOC = "A4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#seg_A
```

5 Lembar Kegiatan