Praktikum Sistem Digital Lanjut

Percobaan 2: Multiplekser 4 Masukan dan Enkoder Prioritas 4-ke-2

1 Tujuan dan Sasaran

Kegiatan praktikum ini bertujuan untuk mengimplementasikan blok rangkaian kombinasional di board Xilinx Spartan-3E Starter Kit. Blok rangkaian kombinasional yang akan dibuat adalah multiplekser 4-masukan (MUX-4) dan enkoder 4-ke-2 dengan prioritas.

Sasaran kegiatan praktikum adalah:

1. Praktikan dapat memahami blok rangkaian kombinasional, yaitu multiplekser 4-masukan, dan enkoder prioritas 4-ke-2 (*priority encoder*);
2. Praktikan dapat mengimplementasikan blok rangkaian tersebut sebagai modul menggunakan desain secara prosedural maupun struktural;
3. Praktikan dapat mengaplikasikan modul (*reusable*) tersebut untuk membuat suatu rangkaian berbasis komponen;
4. Praktikan dapat memahami (dan membedakan) hasil sintesis rangkaian RTL (register transfer level) kedua bentuk desain tersebut serta utilisasi/penggunaan device untuk desain tersebut;
5. Praktikan dapat menganalisis perilaku masukan-keluaran desain di board Starter Kit;

Sumber referensi yang bisa digunakan:

1. UG230: Spartan-3E FPGA Starter Kit Board User Guide, Xilinx, June 2008
2. Spartan-3E Starter Board Schematic, Digilent, Feb 2006
3. Verilog Tutorial (online): http://www.asic-world.com/verilog/veritut.html

2 Alat dan Bahan

Alat dan bahan yang digunakan adalah:

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

   Modul (komponen) I/O yang akan digunakan dalam praktikum:

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

   b) 4 buah saklar geser (SW0, SW1, SW2, SW3);

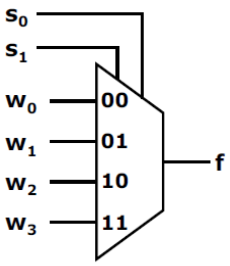
   c) 8 buah LED (LD0-7);
2. Kabel USB dengan konektor tipe-B;
3. Adaptor sumber daya DC 5 Volt;
4. Software Xilinx ISE Webpack 11.1;

3 Dasar Teori

Board Starter Kit beserta komponennya dan Xilinx ISE Webpack telah dijelaskan dalam modul praktikum 1. Di bab ini akan dijelaskan tentang blok rangkaian kombinasional yang akan digunakan dalam praktikum ini. Blok rangkaian kombinasional yang akan dibahas adalah multiplekser 4-masukan dan enkoder 4-ke-2 dengan prioritas.

3.1 Multiplekser 4-masukan

Multiplekser 4-masukan digunakan untuk memilih satu dari empat data masukan yang akan menentukan nilai keluaran. Pemilihan data masukan dilakukan dengan menggunakan kontrol 2-jalur (Gambar 1).

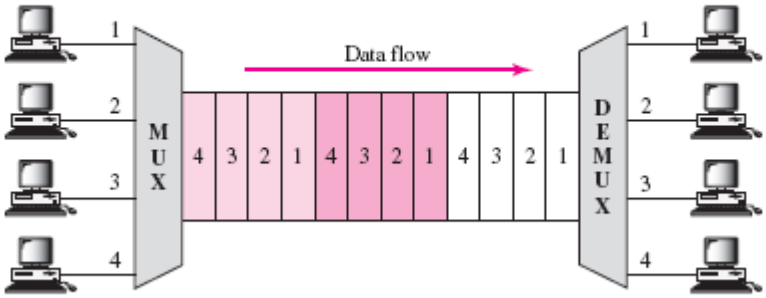


s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

$$f = s_1's_0'w_0 + s_1's_0w_1 + s_1s_0'w_2 + s_1s_0w_3$$

Gambar 1: Diagram blok multiplekser 4-masukan dan fungsi logikanya

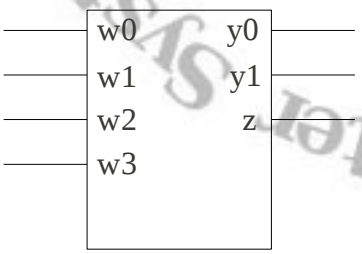
Salah satu aplikasi dari multiplekser di sistem transmisi data adalah multiplekser TDM (time division multiplexing) yang mengkombinasikan sejumlah data dengan bitrate konstan ke dalam satu aliran data serial (frame data) untuk mendapatkan rate yang lebih tinggi (Gambar 2).



Gambar 2: Aplikasi multiplekser-demultiplekser (sinkron) di sistem TDM

3.2 Enkoder 4-ke-2 dengan Prioritas

Enkoder 4-ke-2 mengkodekan data 4-masukan ke dalam kode keluaran 2-bit. Keluaran merepresentasikan bilangan biner yang mengidentifikasikan masukan mana yang bernilai '1'. Salah satu aplikasi enkoder adalah untuk mentransmisikan informasi di sistem digital. Salah satu tipe enkoder adalah *priority encoder*. Tiap input mempunyai level prioritas (Gambar 3).



w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	D	D	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Gambar 3: Enkoder 4-ke-2 dengan w3 mempunyai prioritas tertinggi. Keluaran z mengindikasikan semua masukan bernilai '0'

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

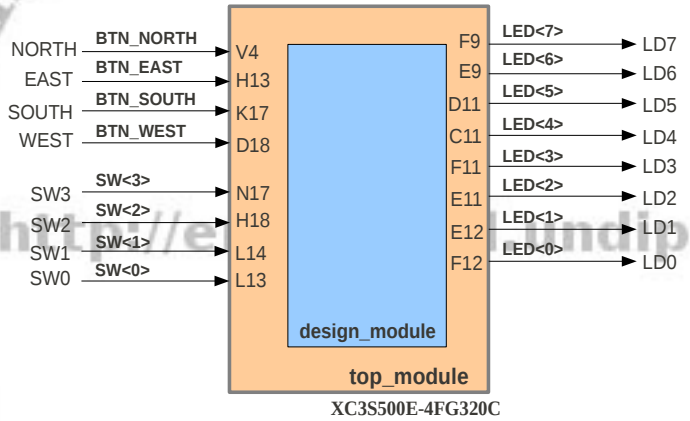
Desain yang akan diimplementasikan dalam praktikum ini adalah:

1. Multiplekser 4-masukan (MUX_4)
2. Enkoder 4-ke-2 dengan prioritas (ENC_4TO2)

Tiap desain akan dibuat secara prosedural dan struktural untuk dibedakan bagaimana rangkaian logikanya setelah sintesis.

Jelaskan tentang desain secara prosedural dan struktural sehingga terlihat perbedaan keduanya. Apa kelebihan dan kekurangannya masing-masing

Semua desain dibuat dalam 1 projek yang tersimpan di lokasi direktori masing-masing kelompok (lihat subbab 4.2.1) . Desain/fungsi akan dibuat sebagai modul, yang akan kemudian dipanggil oleh top_module (Gambar 4). Diagram blok top_module (yang menggambarkan port masukan dan keluaran) untuk semua fungsi/desain **sama**, sehingga bisa disalin untuk menghemat waktu praktikum. Perilaku tiap modul mengikuti tabel kebenaran fungsi yang dijabarkan dalam bab 3. Desain-desain tersebut di atas menggunakan file konstrain *rangkaian_kombinasional.ucf* yang sama, karena ada dalam satu proyek. Koneksi sinyal dari tiap modul dengan top_module diuraikan dalam Tabel 1.



Gambar 4: Diagram blok rancangan sistem masukan-keluaran (top_modul)

4.2 Langkah Kerja

Kegiatan praktikum meliputi hal-hal sebagai berikut:

1. Membuat proyek baru, dengan format nama <nama_kelompok>-Modul2-RangkaianKombinasional;
2. Menulis kode HDL untuk kedua blok sistem di atas sebagai modul, masing-masing dengan desain prosedural dan struktural, sehingga total ada 4 modul;
3. Mengaplikasikan modul-modul tersebut dalam top_modulenya membentuk satu desain;
4. Menambah satu file konstrain yang digunakan oleh semua desain;
5. Mensintesis dan mengimplementasikan untuk tiap desain;
   a) Melihat skematik RTL dari tiap desain;
   b) Melihat utilisasi device dari tiap desain;
6. Membangkitkan file programming *.bit untuk tiap desain;
7. Memprogram file *.bit tersebut dan mengamati perilaku sistem;

Sinyal	Desain Modul (dan sinyalnya)			
	MUX_4	DEC_3TO8	DEMUX_8	ENC_4TO2
BTN_NORTH	w_in<0>	NC	DATA_IN	NC
BTN_EAST	w_in<1>	NC	NC	NC
BTN_SOUTH	w_in<2>	NC	NC	NC
BTN_WEST	w_in<3>	NC	NC	NC
SW<3>	NC	ENABLE	NC	w_in<3>
SW<2>	NC	W<2>	SEL<2>	w_in<2>
SW<1>	sel<1>	W<1>	SEL<1>	w_in<1>
SW<0>	sel<0>	W<0>	SEL<0>	w_in<0>
LED<7>	NC	Y<7>	Y<7>	zero
LED<6>	NC	Y<6>	Y<6>	NC
LED<5>	sel_stat<1>	Y<5>	Y<5>	NC
LED<4>	sel_stat<0>	Y<4>	Y<4>	NC
LED<3>	NC	Y<3>	Y<3>	NC
LED<2>	NC	Y<2>	Y<2>	NC
LED<1>	NC	Y<1>	Y<1>	y_out<1>
LED<0>	fout	Y<0>	Y<0>	y_out<0>

Tabel 1: Koneksi sinyal tiap modul dengan top_modulnya. NC menandakan no-connect. Baris yang diarsir merupakan keluaran

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>-Modul2-RangkaianKombinasional). 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 4 buah. Langkah-langkahnya adalah sebagai berikut:

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

No	Nama Modul	Masukan	Keluaran	Keterangan
1.	mux_4_proc	w_in[3:0] sel[1:0]	sel_stat[1:0] fout	Prosedural
2.	mux_4_struc			struktural
3.	enc_4to2_proc	w_in[3:0]	zero y_out[1:0]	prosedural
4.	enc_4to2_struc			struktural

Tabel 2: Nama modul-modul desain yang akan dibuat

2. Edit file mux_4_proc.v, mux_4_struc.v, enc_4to2_proc.v dan enc_4to2_struc.v Listing kode sumber HDL untuk semua desain modul ada dalam Lampiran;

   Catatan: seringkali untuk membuat desain struktural diperlukan bantuan K-map untuk mensintesis fungsi menjadi ekspresi paling sederhana

   
   Tuliskan persamaan fungsi untuk mux_4 dan enc_4to2 dalam lembar kegiatan

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

4.2.3 Mengaplikasikan Modul dalam Desain (Top Module)

Tiap modul/komponen akan diaplikasikan dalam top_modulenya masing-masing (desain berbasis komponen). Langkah yang diperlukan adalah:

1. Buat top_module untuk tiap-tiap modul. Berikan nama filenya <nama_modul>_top.v, sehingga akan ada 4 buah file *_top.v. **Ingat**: semua *_top.v mempunyai port masukan dan keluaran yang sama (Gambar 4), sehingga bisa disalin dari satu *_top.v ke lainnya;

No	Nama Modul	Masukan	Keluaran	Keterangan
1.	mux_4_proc_top	BTN_NORTH, BTN_EAST, BTN_SOUTH, BTN_WEST, SW[3:0]	LED[7:0]	prosedural
2.	mux_4_struc_top			struktural
3.	enc_4to2_proc_top			prosedural
4.	enc_4to2_struc_top			struktural

Tabel 3: Nama top_module dari desain yang akan dibuat

2. Panggil modul yang bersesuaian dari modul *_top ini dan interkoneksikan port-portnya. Listing kode sumber HDL untuk semua *_top.v ada dalam Lampiran;

   Catatan: mux_4_proc_top dan mux_4_struc_top hampir mirip, yang membedakan adalah pemanggilan modul di dalamnya. Modul mux_4_proc_top memanggil modul mux_4_proc, sedangkan mux_4_struc_top memanggil modul mux_4_struc

4.2.4 Sintesis dan Implementasikan Top Modul

Tiap modul *_top (4 modul) akan disintesis dan diimplementasikan. Sebaiknya melakukan langkah di 4.2.4 dan 4.2.5 secara berurutan untuk tiap desain (modul *_top). Ini dapat menghemat waktu Anda. Langkah-langkahnya adalah sebagai berikut:

1. Set *_top modul yang akan disintesis dan diimplementasikan sebagai top_module. Klik kanan *_top modul tersebut dan pilih 'Set as Top Module'. Misalnya: untuk mensintesis modul mux_4_proc_top, klik kanan modul tersebut dan pilih 'Set as Top Module'. Modul mux_4_proc_top akan menjadi top_module dan siap untuk disintesis;

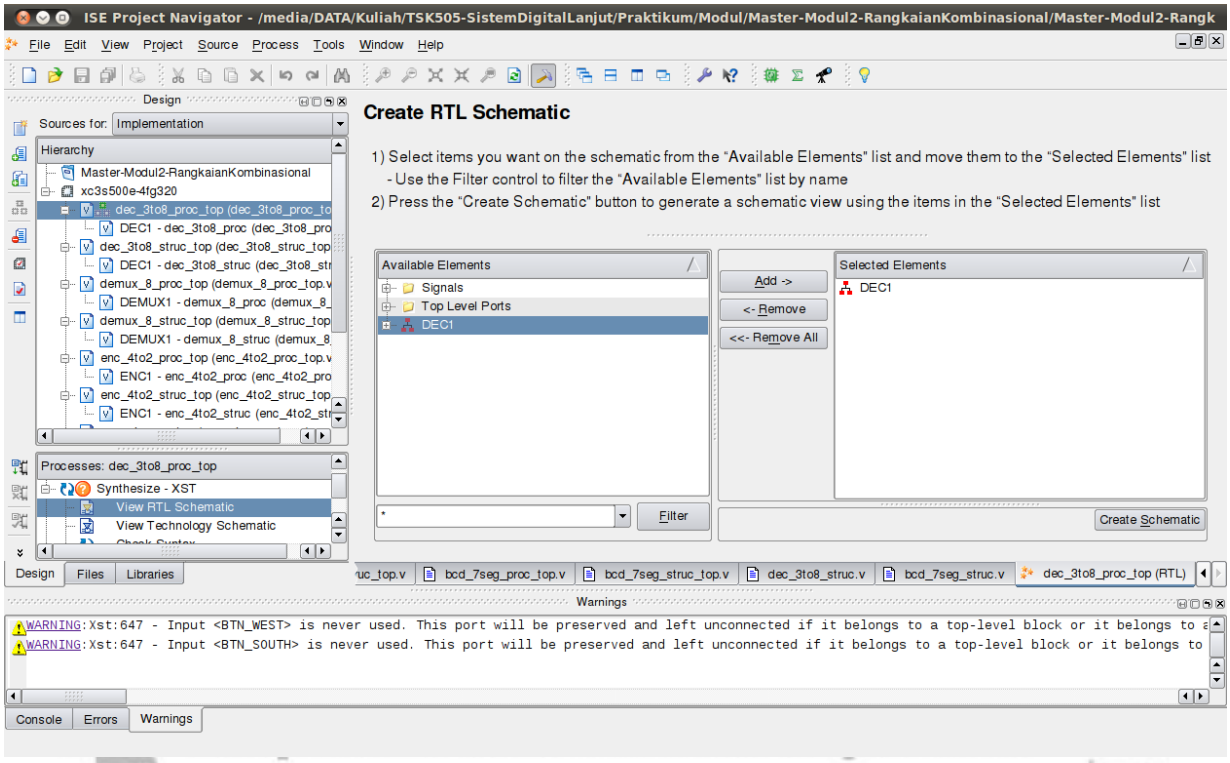
2. Bangkitkan skematik RTL dari top_module. Klik “View RTL Schematic” dari proses Synthesize. Tambahkan elemen yang tersedia dan klik tombol “Create Schematic” . Lihat blok dari elemen dan bagaimana isinya (rangkaian logikanya). Misalnya: untuk melihat skematik RTL dari DEC1 (Gambar 5);

   
   Simpan gambar blok elemen beserta rangkaiannya dalam file gambar (misalnya dengan snapshot). Skematik RTL ini beserta penjelasannya harus dimasukkan ke laporan

3. Sintesis dan implemen top_module dengan mengklik kanan proses “Implement

Design” dan pilih “Rerun All”;

Lihat ringkasan laporan desain dari implementasi top_module 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;



Gambar 5: Membangkitkan skematik RTL

4.2.5 Memprogram FPGA dan Pengamatan Perilaku Sistem

Langkah percobaan:

1. Bangkitkan file programming *.bit untuk top_module;
2. Jalankan Xilinx iMPACT jika belum dijalankan. Pastikan kabel USB telah terpasang di starter kit dan komputer;
3. Dari Xilinx iMPACT, klik ganda “Boundary Scan”. Kemudian klik kanan dan pilih Initialize Chain (Ctrl+I) untuk menginisialisasi Boundary Scan. Langkah inisialisasi chain hanya dilakukan sekali untuk semua top_module;
4. 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 *.bit yang ingin diprogram ke FPGA. Nama file konfigurasi akan tampil di bawah device FPGA;
5. Klik kanan device FPGA dan pilih “Program” untuk mengkonfigurasi FPGA;
6. Perilaku keluaran sistem ditunjukkan dengan memberikan nilai masukan dari BTN_* dan SW[3:0] serta mengamati keluarannya berupa penyalaan LED;

Amati perilaku sistem dan isi tabel di lembar kegiatan;

4.3 Lampiran Kode HDL

Nama file modul bersesuaian dengan nama modulnya. Misalnya: modul mux_4_proc berada dalam file mux_4_proc.v. Modul dec_3to8_proc_top akan berada dalam file dec_3to8_proc_top.v.

4.3.1 MUX_4: mux_4_proc dan mux_0,08"4_struc

```
module mux_4_proc(
    input [3:0] w_in,
    input [1:0] sel,
    output [1:0] sel_stat,
    output fout
    );

      reg fout_reg; /*storage to store fout in procedural design*/

      //Procedural block. Sensitivity: sel, w_in
      //Changes in sel and w_in will execute this block
      always @(sel, w_in) begin
            case (sel)
                  2'b00: fout_reg <= w_in[0];
                  2'b01: fout_reg <= w_in[1];
                  2'b10: fout_reg <= w_in[2];
                  2'b11: fout_reg <= w_in[3];
                  default: fout_reg <= w_in[0];
            endcase
      end

      assign fout = fout_reg;
      assign sel_stat = sel;

endmodule
```

```
module mux_4_proc_top(
    input BTN_NORTH,
    input BTN_EAST,
    input BTN_SOUTH,
    input BTN_WEST,
    input [3:0] SW,
    output [7:0] LED
    );

      wire [1:0] sel_stat;
      wire fout;

      /*Module instantation*/
      mux_4_proc MUX1 (
            .w_in ({BTN_WEST, BTN_SOUTH, BTN_EAST, BTN_NORTH}),
            .sel (SW[1:0]),
            .sel_stat (sel_stat),
            .fout (fout)
      );

      assign LED = {fout,1'b0,sel_stat,4'b0};

endmodule
```

```
module mux_4_struc(
    input [3:0] w_in,
    input [1:0] sel,
    output [1:0] sel_stat,
    output fout
    );

      /*structural design: doesn't need a storage*/
      /*fout = (~sel1 & ~sel0 & w_in[0]) | (~sel1 & sel0 & w_in[1]) |
                  (sel1 & ~sel0 & w_in[2]) | (sel1 & sel0 & w_in[3]) */
      assign fout = (~sel[1] & ~sel[0] & w_in[0]) |
            (~sel[1] & sel[0] & w_in[1]) |
            (sel[1] & ~sel[0] & w_in[2]) |
            (sel[1] & sel[0] & w_in[3]) ;
      assign sel_stat = sel;

endmodule
```

```
module mux_4_struc_top(
    input BTN_NORTH,
    input BTN_EAST,
    input BTN_SOUTH,
    input BTN_WEST,
    input [3:0] SW,
    output [7:0] LED
    );

      wire [1:0] sel_stat;
      wire fout;

      /*Module instantation*/
      mux_4_struc MUX1 (
            .w_in ({BTN_WEST, BTN_SOUTH, BTN_EAST, BTN_NORTH}),
            .sel (SW[1:0]),
            .sel_stat (sel_stat),
            .fout (fout)
      );

      assign LED = {fout,1'b0,sel_stat,4'b0};

endmodule
```

4.3.2 ENC_4TO2: enc_4to2_proc dan enc_4to2_proc

```
module enc_4to2_proc(
    input [3:0] w_in,
    output zero,
    output [1:0] y_out
    );

      reg [1:0] y_out_reg;
      reg zero_reg;

      always @(w_in) begin
            casex (w_in)
                  4'b0000: begin
                        /*incompletely function, input never happen*/
```

```
                        y_out_reg <= 2'b00; zero_reg = 0;
                    end
                4'b0001: begin
                        y_out_reg <= 2'b00; zero_reg = 1;
                    end
                4'b001X: begin
                        y_out_reg <= 2'b01; zero_reg = 1;
                    end
                4'b01XX: begin
                        y_out_reg <= 2'b10; zero_reg = 1;
                    end
                4'b1XXX: begin
                        y_out_reg <= 2'b11; zero_reg = 1;
                    end
                default: begin
                        y_out_reg <= 2'b11; zero_reg = 1;
                    end
            endcase
      end

      assign y_out = y_out_reg;
      assign zero = ~zero_reg; /*zero indicator*/

endmodule
```

```
module enc_4to2_proc_top(
    input BTN_NORTH,
    input BTN_EAST,
    input BTN_SOUTH,
    input BTN_WEST,
    input [3:0] SW,
    output [7:0] LED
    );

      wire zero;
      wire [1:0] y_out;

      enc_4to2_proc ENC1 (
            .w_in (SW),
            .zero (zero),
            .y_out (y_out)
      );

      assign LED = {zero,5'b0,y_out};

endmodule
```

```
module enc_4to2_struc(
    input [3:0] w_in,
    output zero,
    output [1:0] y_out
    );

      /*Default value of y_out is 0*/
      assign y_out[1] = w_in[3] | w_in[2];
      assign y_out[0] = w_in[3] | w_in[1];
      assign zero = ~w_in[3] & ~w_in[2] & ~w_in[1] & ~w_in[0];

endmodule
```

```
module enc_4to2_struc_top(
    input BTN_NORTH,
    input BTN_EAST,
    input BTN_SOUTH,
    input BTN_WEST,
    input [3:0] SW,
    output [7:0] LED
    );

      wire zero;
      wire [1:0] y_out;

      enc_4to2_struc ENC1 (
          .w_in (SW),
            .zero (zero),
            .y_out (y_out)
      );

      assign LED = {zero,5'b0,y_out};


endmodule
```

4.3.3 File rangkaian_kombinasional.ucf

```
# 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 ;
# Koneksi SW
NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<3>" LOC = "N17" | 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 ;
```