**TUGAS 1 K-MAP & GRAYCODE COUNTER**

**PERANCANGAN KOMPONEN TERPROGRAM**



Disusun oleh:

|  |  |  |
| --- | --- | --- |
| Nama | : | Yohanes Stefanus |
| NRP | : | 5022211089 |
| Github repository | : | [pkt1\_yohanes](https://github.com/yohanesstef/pkt2024_5022211089_Yohanes-Stefanus_tugas1.git) |

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**TEKNIK ELEKTRO**

**2023/2024**

**Exercise 4.5:**

Write an HDL module called minority. It receives three inputs, a, b, and c. It produces one output, y, that is TRUE if at least two of the inputs are FALSE.

**Jawaban:**

Untuk mengerjakan permasalahan di atas saya terlebih dahulu membuat persamaan aljabar booleannya menggunakan metode minimalisasi gerbang k-map. Berikut adalah tabel kebenaran dari sistem di atas.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | | Output |
| A | B | C | Y |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Berikut adalah tabel k-map dari tabel kebenaran di atas.

A screenshot of a graph

Description automatically generated

Dari k-map tersebut, akan didapatkan SOP (sum of product) sebagai berikut.

Persamaan Boolean tersebut dapat digunakan secara langsung pada program Verilog. Berikut adalah program yang telah saya buat.

**Program:**

module minority(A,B,C,Y);

    input A,B,C;

    output Y;

    assign Y = ~A & ~B | ~A & ~C | ~B & ~C;

endmodule

Pada program di atas, saya membuat sebuah module yang saya beri nama minority. Pada modul tersebut terdapat 4 parameter, yakni A, B, C, dan Y. A, B, dan C berperan sebagai variabel input, sedangkan Y berperan sebagai variabel output. Baris program ‘assign Y = ~A & ~B | ~A & ~C | ~B & ~C;‘ serupa dengan persamaan boolean .

**Test Bench:**

`timescale 1ns/1ns

`include "minority.v"

module minority\_tb;

    reg A,B,C;

    wire Y;

    minority uut(A,B,C,Y);

    initial begin

        $dumpfile("minority\_tb.vcd");

        $dumpvars(0,minority\_tb);

        {A,B,C} = 3'd0; #20;

        {A,B,C} = 3'd1; #20;

        {A,B,C} = 3'd2; #20;

        {A,B,C} = 3'd3; #20;

        {A,B,C} = 3'd4; #20;

        {A,B,C} = 3'd5; #20;

        {A,B,C} = 3'd6; #20;

        {A,B,C} = 3'd7; #20;

        $display("Test is complete!");

    end

endmodule

Pada program test bench di atas, saya menggunakan time scale sebesar 1 nano detik. Saya membuat modul test bench yang saya beri nama minority\_tb. Pada modul tersebut terdapat register A, B, dan C serta wire Y. Dalam program tersebut, saya memvariasikan nilai A, B, dan C sebagai satu bilangan biner 3 bit yang nilainya saya count up setiap 20 nS. Hasil dari test bench tersebut akan disimpan dalam file minority\_tb.vcd. Berikut adalah hasil simulasi program diatas menggunakan software gtkwave.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Hasil simulasi di atas sudah memenuhi tabel kebenaran. Dengan demikian, HDL module yang saya buat telah sesuai dengan exercise 4.5.

**Exercise 4.38:**

Write an HDL module for the UP/DOWN Gray code counter from Exercise 3.28.

**Jawaban:**

Gray code counter merupakan salah satu jenis counter yang menghitung ketika terjadi perubahan pada 1 bit. Gray code counter dapat dibuat menggunakan logika XOR yang membandingkan dengan counter binary biasa. Berikut adalah tabel kebenaran dari gray code counter.

|  |  |  |  |
| --- | --- | --- | --- |
| **3 Bit Gray Code** | | | |
| **Decimal** | **Binary** | | |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 1 | 1 | 0 |
| 5 | 1 | 1 | 1 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 0 | 0 |

Untuk mendesain 3 bit gray code counter, terdapat 3 komponen yang harus dibuat, yaitu sumber clock, 3-bit counter, dan konversi gray code. Berikut adalah program verilog gray code counter yang telah saya buat.

**Program:**

`timescale 1ns/1ns

module graycode\_counter(

    input clk,

    input reset,

    input up\_down,

    output [2:0] o\_bin,

    output [2:0] o\_gray\_code

);

reg [2:0] bin\_counter = 0;

always @(posedge clk or posedge reset)

    if(reset)

        bin\_counter <= 0;

    else if(up\_down)

        bin\_counter <= bin\_counter + 1;

    else

        bin\_counter <= bin\_counter - 1;

assign o\_bin = bin\_counter;

*// reg [2:0] r\_gc = 0;*

*// always @(bin\_counter)*

*//     case (bin\_counter)*

*//         0 : r\_gc = 3'b000;*

*//         1 : r\_gc = 3'b001;*

*//         2 : r\_gc = 3'b011;*

*//         3 : r\_gc = 3'b010;*

*//         4 : r\_gc = 3'b110;*

*//         5 : r\_gc = 3'b111;*

*//         6 : r\_gc = 3'b101;*

*//         7 : r\_gc = 3'b100;*

*//     endcase*

*// assign o\_gray\_code = r\_gc;*

assign o\_gray\_code[2] = bin\_counter[2];

assign o\_gray\_code[1] = bin\_counter[2] ^ bin\_counter[1];

assign o\_gray\_code[0] = bin\_counter[1] ^ bin\_counter[0];

endmodule

Pada program tersebut terdapat modul graycode\_counter yang memiliki variabel input clk, reset, dan up\_down, dan variabel output o\_bin (3 bit), dan o\_gray\_code (3 bit). Selain itu juga terdapat register bin\_counter (3 bit). Counter akan mulai menghitung setiap perubahan clock atau reset pada sisi rising edge. Untuk menciptakan sebuah counter up/down, diperlukan variabel up\_down. Ketika up\_down bernilai 1, counter akan berhitung maju (counter up), sedangkan ketika bernilai 0, counter akan berhitung mundur (counter down). Baris program

assign o\_gray\_code[2] = bin\_counter[2];

assign o\_gray\_code[1] = bin\_counter[2] ^ bin\_counter[1];

assign o\_gray\_code[0] = bin\_counter[1] ^ bin\_counter[0];

merupakan algoritma yang digunakan untuk mengkonversi 3-bit counter ke 3-bit up/down gray code counter.

**Test bench:**

`timescale 1ns/1ns

`include "graycode\_counter.v"

module graycode\_counter\_tb;

    reg clk;

    reg reset;

    reg up\_down;

    wire [2:0] gray\_code;

    wire [2:0] bin\_count;

    graycode\_counter gc(.clk(clk), .reset(reset), .up\_down(up\_down), .o\_bin(bin\_count), .o\_gray\_code(gray\_code));

    always #1 clk = ~clk;

    always #10 up\_down = ~up\_down;

    initial begin

        $dumpfile("graycode\_counter\_tb.vcd");

        $dumpvars(0,graycode\_counter\_tb);

        clk = 0;

        up\_down = 1;

        reset = 1;

        #2;

        reset = 0;

        #40;

        $display("Test complete");

        $finish;

    end

endmodule

Pada program test bench di atas, saya menggunakan skala waktu sebesar 1 nano detik. Modul test bench tersebut saya beri nama graycode\_counter\_tb. Terdapat 3 register, clk, resetm dan up\_down, dan 2 wire 3 bit, gray\_code dan bin\_count. Nilai clock selalu berubah dari 0 ke 1 atau sebaliknya setiap 1 nS dan nilai up\_down selalu berubah setiap 10 nS. Simulasi tersebut saya mulai selama 40 nS. Hasil dari test bench tersebut saya simpan dalam file graycode\_counter\_tb.vcd. Berikut adalah grafik simulasi menggunakan software gtkwave.

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

Pada grafik tersebut, nilai bin\_count dan gray\_code semula-mula 0 karena reset = 1. Terlihat bahwa counter gray code sudah sesuai dengan tabel yang saya sertakan di awal. Ketika up\_down HIGH, counter akan menghitung maju (counter up), sedangkan ketika up\_down LOW, counter akan menghitung mundur (counter down).

**Repository:**

[**https://github.com/yohanesstef/pkt2024\_5022211089\_Yohanes-Stefanus\_tugas1.git**](https://github.com/yohanesstef/pkt2024_5022211089_Yohanes-Stefanus_tugas1.git)