# TUGAS 1 K-MAP & GRAYCODE COUNTER

## PERANCANGAN KOMPONEN TERPROGRAM


Disusun oleh:

| | | |
|---|---|---|
| Nama | : | Yohanes Stefanus |
| NRP | : | 5022211089 |
| Github repository | : | pkt1_yohanes |


**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.

| F(A,B,C) = ∑(0,1,2,4) | | | | |
|---|---|---|---|---|
| K-map | | | | |
| A | BC | | | |
| | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |

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

$$F = \bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{C} + \bar{B} \cdot \bar{C}$$

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
```

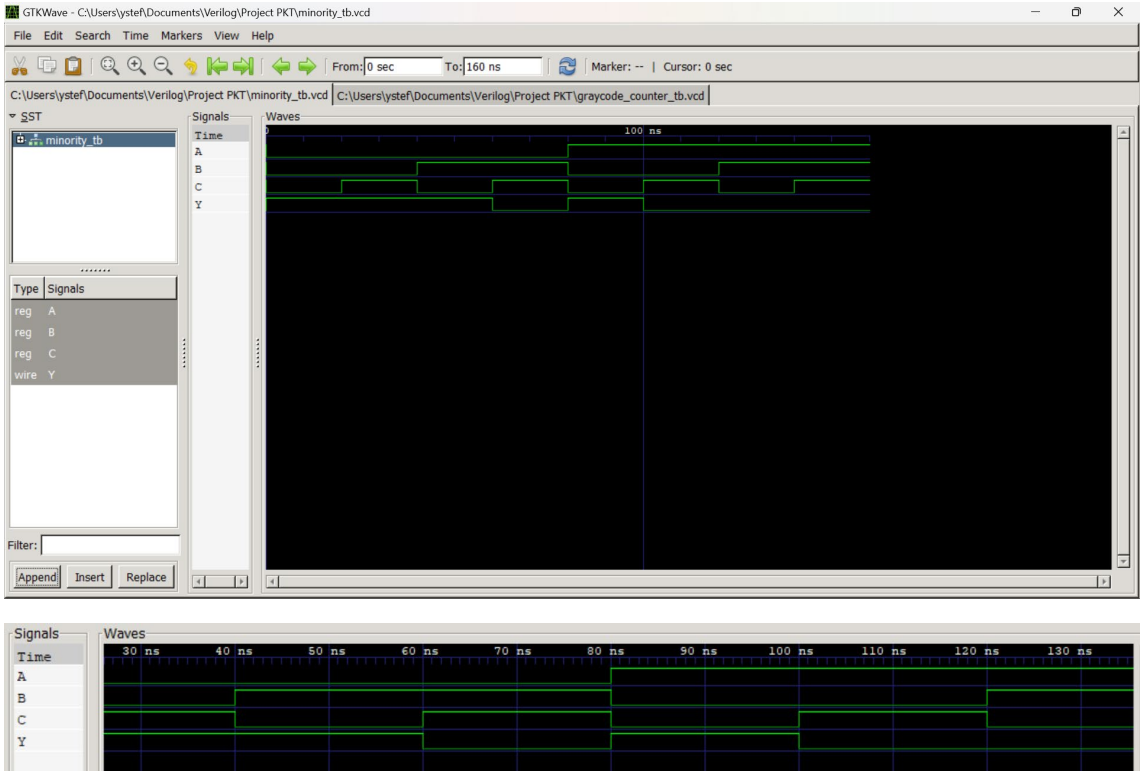
Pada program di atas, saya membuat sebuah module yang saya beri nama minority. Pada modul tersebut terdapat 4 parameter, yakni A, B, C, dan Y. A, B, dan C berperan sebagai variabel input, sedangkan Y berperan sebagai variabel output. Baris program ‘assign Y =

~A & ~B | ~A & ~C | ~B & ~C;‘ serupa dengan persamaan boolean $F = \bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{C} + \bar{B} \cdot \bar{C}$.

**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.




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];
```

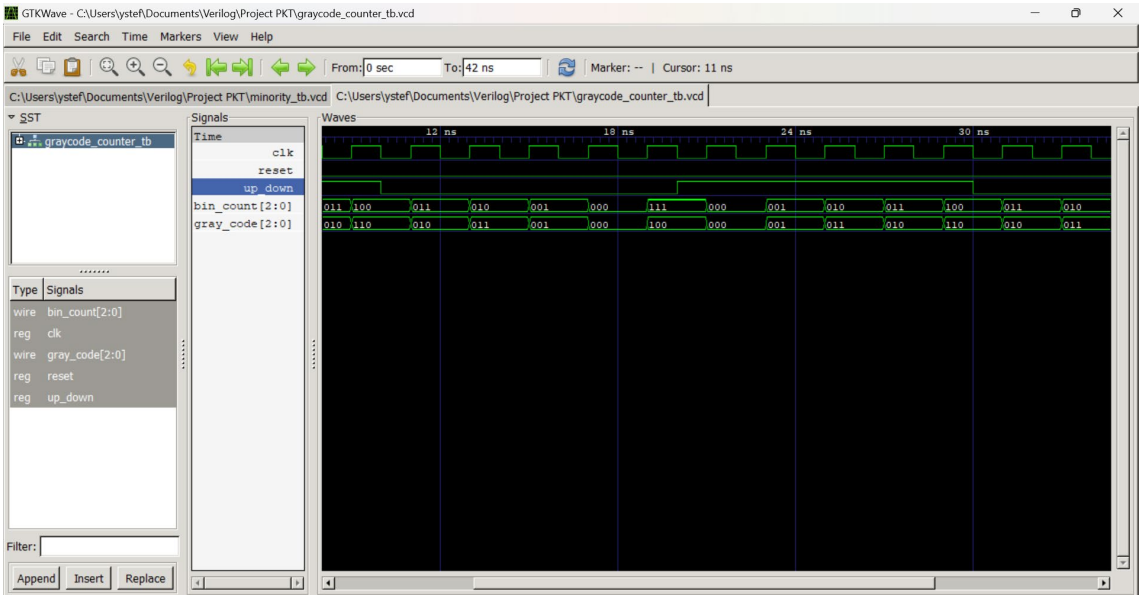
merupakan algoritma yang digunakan untuk mengkonversi 3-bit counter ke 3-bit up/down gray code counter.

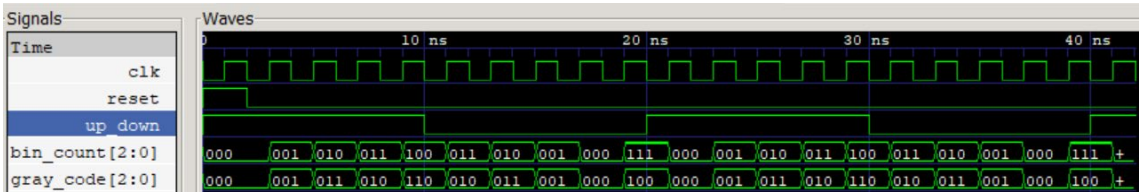
**Test bench:**

```
`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.




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