به نام خدا

محمد محجل صادقی 810199483

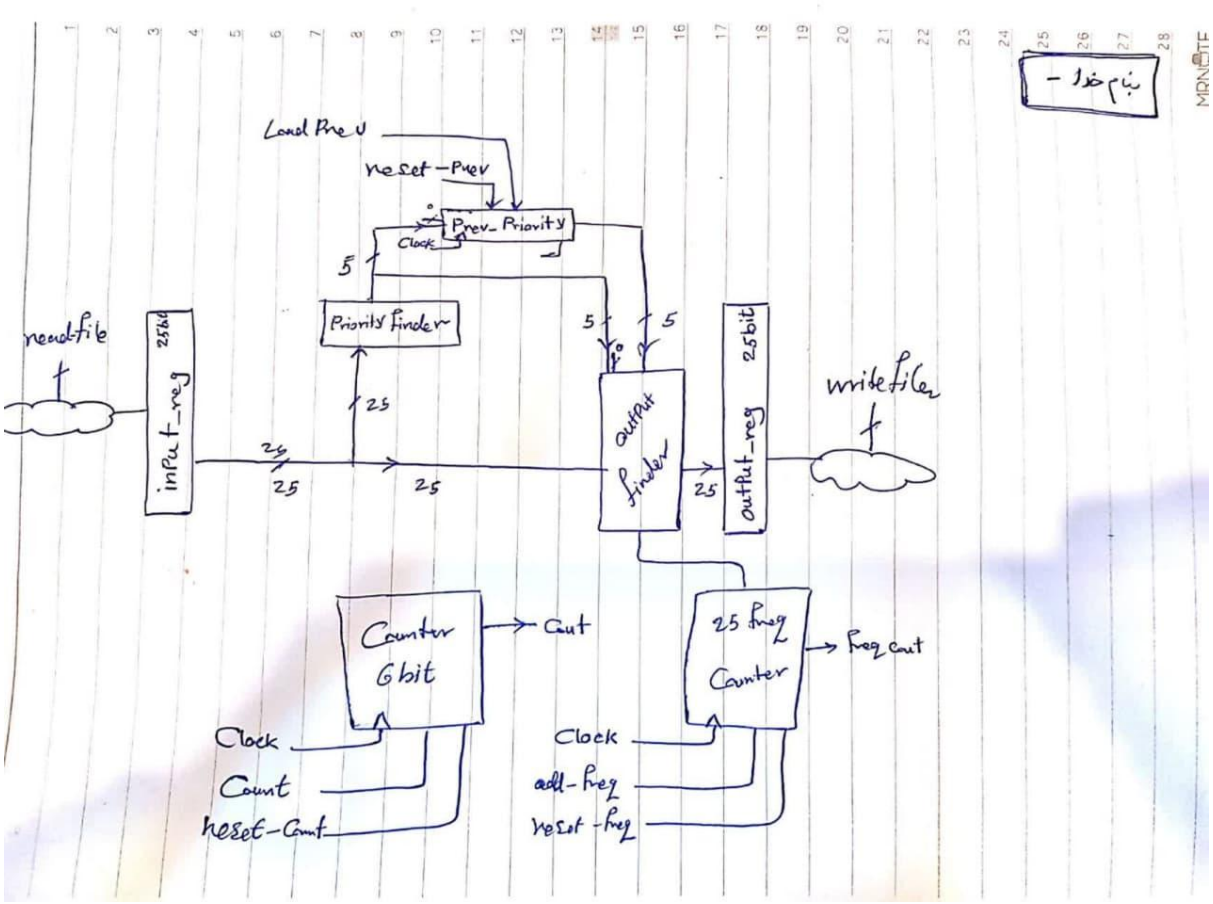
علی عطاءاللهی 810199461

میان ترم طراحی سیستم های دیجیتال (CAD)

استاد مدرسی

datapath:

مسیر داده مطابق شکل زیر است:



توضیح مسیر داده:

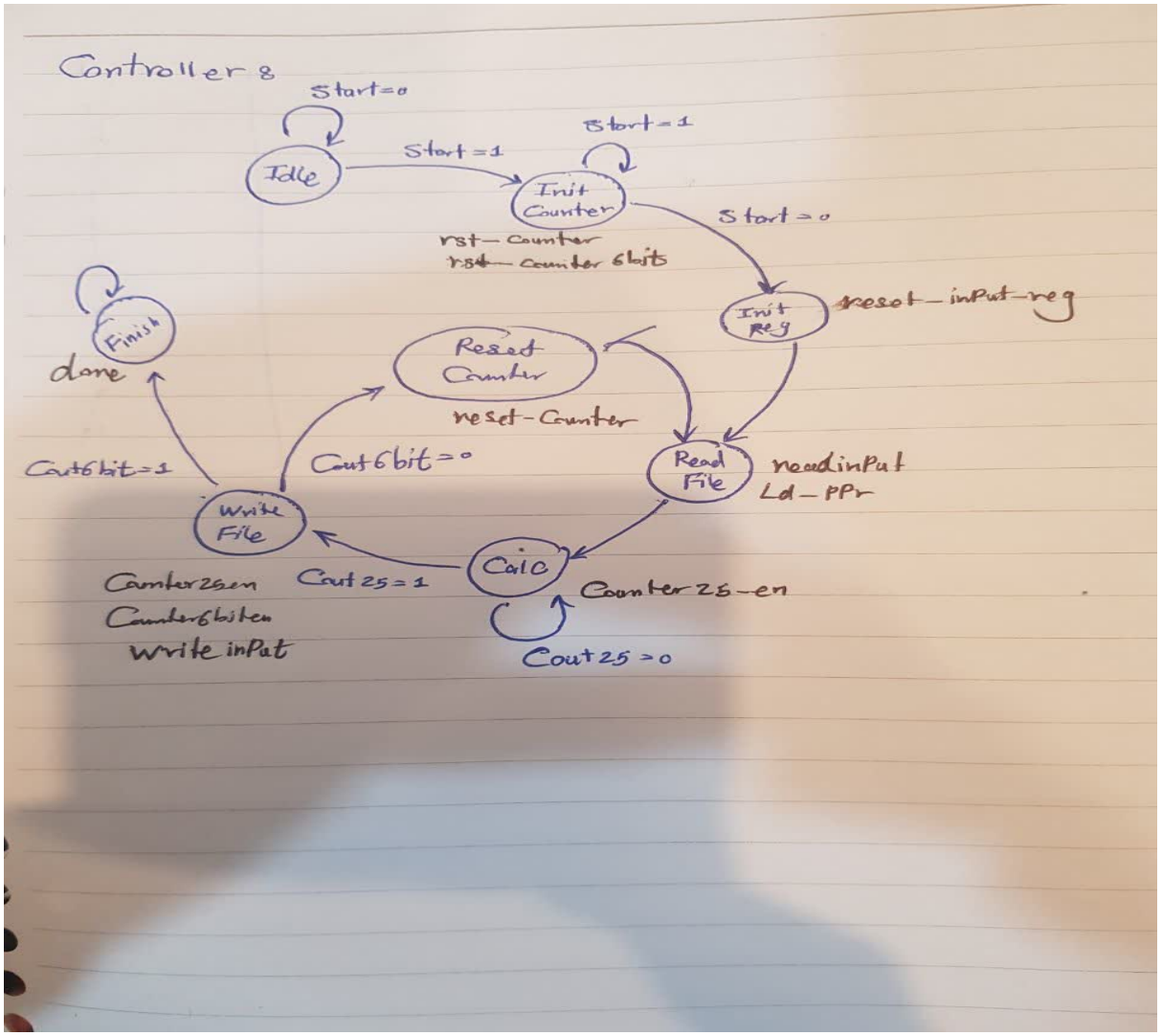
برای خواندن ورودی از یک input-reg 25 bit استفاده می کنیم.
ماژول parity finder با گرفتن ورودی، parity ی ۵ ستون آن را محاسبه می کند تا ماژول های بعدی از آن استفاده کنند.
برای محاسبه ی هر بیت، به parity ی ۵ ستون قبلی و ۵ ستون حال نیازمندیم.
به همین سبب برای ذخیره ی ۵ ستون قبلی از یک رجیستر ۵ بیتی priv parity استفاده می کنیم.

دقت کنید که طبق توضیحات داخل امتحان، از هیچ for loop در داخل طراحی استفاده نشده است.

به همین سبب از ۲ کانتر یکی ۶ بیت برای شمردن ۶۴ بار ورودی و یک freq 25 برای محاسبه ی یکی یکی (و بدون for loop) تمام ۲۵ بیت ورودی در هر بار خواندن استفاده می کنیم.

پس از نوشن تمام بیت ها در رجیستر پایانی، آن سطر را در فایل خروجی چاپ می کنیم.

controller:



توضیح کنترلر :

همانگونه که مشاهده می کنید، پس از گرفتن یک پالس کامل **start** شروع به خواندن یک سطر از فایل ورودی می کنیم.

توجه کنید برای محاسبه ی پاسخ های ۲۵ بیت اول، به parity ی ۲۵ بیت آخر نیازمندیم.
به همین جهت قبل از افتادن در loop و توسط ۲۵ parity، reset_input_register بیت آخر را در آن ذخیره می کنیم.
در هر لوپ در استیت calc ، سیگنال counter25_rn را فعال می کنیم.
با این کار برای ۲۵ بار در اینجا توقف می کنیم و ماژول output finder در ۲۵ سری و هر بار یکی از بیت ها را محاسبه می کند.

دقت کنید هر بار در این لوپ، parity های این ۲۵ بیت کنونی را در prev parity ذخیره می کنیم تا ۲۵ بیت بعدی بتوانند از آن استفاده کنند.

در آخر هم پس از پایان تمام سطر ها به استیت نهایی می رویم و سیگنال doneرا فعال می کنیم.

تفاوت مسیر داده و کنترل در اینجا و امتحان:

مسیر داده و کنترلر در برخی اجزای کوچک باهم تفاوت دارند.
یکی از تفاوت ها در داشتن freq-25-counter است چراکه در اواخر امتحان تاکید شد که استفاده از حلقه ی for مجاز نمی باشد. به همین سبب برای ویزیت کردن ۲۵ بیت از آن استفاده کردیم.
در کنترلر هم بخش های مربوط به کنترل کردن freq-25-counter اضافه شده است.

CODE:

modules used:

prev_parity_reg_5bit:

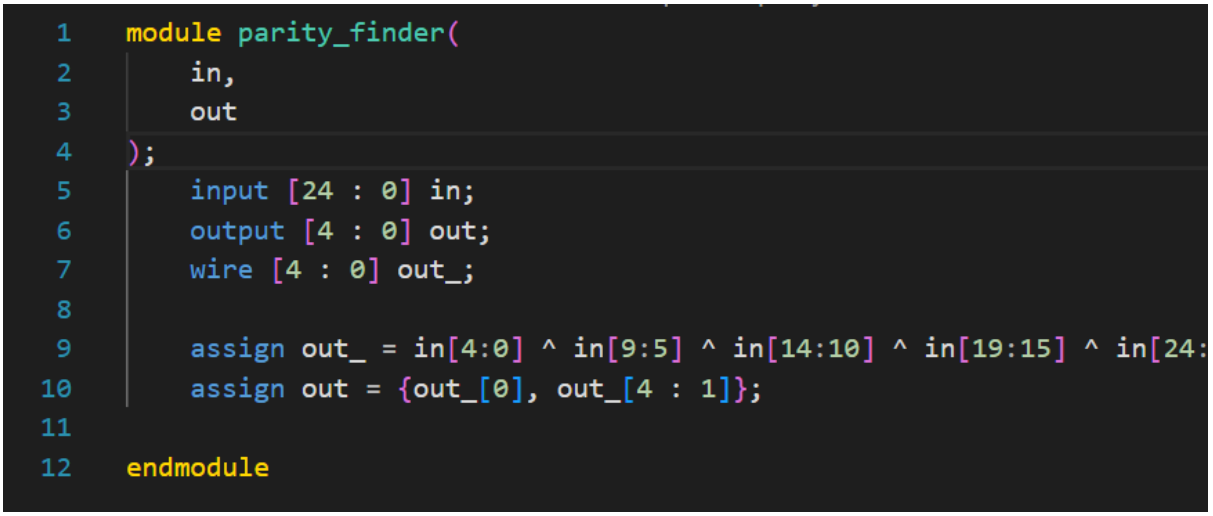
stores 5 bit parity of the previous 25 bit for use in future.

```
module prev_parity_reg5bit(
    in,
    clk,
    rst,
    ld,
    out
);
    input [4 : 0] in;
    reg [4 : 0] out_;
    output [4 : 0] out;
    input clk, rst;
    input ld;

    always @(posedge clk, posedge rst, ld) begin
        if (rst)
            out_ <= 5'd0;
        else
            if (ld) begin
                out_[0] <= in[3];
                out_[1] <= in[4];
                out_[2] <= in[0];
                out_[3] <= in[1];
                out_[4] <= in[2];
            end
    end
    assign out = out_;

endmodule
```

parity_finder:

finds parity of 5 columns using xor.

in the end it shifts cols for logic reasons and get valid output.



```
module parity_finder(
    in,
    out
);
    input [24 : 0] in;
    output [4 : 0] out;
    wire [4 : 0] out_;

    assign out_ = in[4:0] ^ in[9:5] ^ in[14:10] ^ in[19:15] ^ in[24:
    assign out = {out_[0], out_[4 : 1]};

endmodule
```

counter6bit:

simple counter to count 64 rows of input.

```
module counter6bit
(
    count_en,
    clk,
    rst,
    pout,
    cout
);
    input count_en;
    input clk;
    input rst;
    output reg [4:0] pout;
    output reg cout;
    always @(posedge clk, posedge rst) begin

        if (rst == 1) begin
            pout <= 5'd0;
        end
        else if (count_en) begin
            pout <= pout + 1;
        end

        assign cout  = (count_en == 1) ? &{pout} : 0;
    end
endmodule
```

reg25bit:

25 bit register to store 25 bit of input.

```
1   module reg25bit(
2       in,
3       out,
4       clk,
5       rst,
6       ld
7   );
8       input [24 : 0] in;
9       output [24 : 0] out;
10      reg [24 : 0] out_;
11      input clk, rst;
12      input ld;
13
14      always @(posedge clk, posedge rst) begin
15          if (rst)
16              out_ <= 25'd0;
17          else
18              if (ld)
19                  out_ <= in;
20      end
21      assign out = out_;
22
23  endmodule
```

Controller:

In the controller implementation, we used 3 always statements as discussed in class.

huffman module has been taken into account and separation of combinational and sequential part is shown.

```
input start, clk, cout25, cout6bit;
output reg rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, write_input, done, counter6bit_en, rst_counter6bit;

reg [2:0] ps, ns;
parameter Idle = 0,InitCounter = 1,InitReg = 2, ReadFile = 3,Calc = 4, WriteFile = 5, ResetCounter = 6, Finish = 7;

always @(posedge clk) begin
    ps <= ns;
end

always @(ps, start, cout25, cout6bit) begin
    ns = Idle;
    case (ps)
        Idle: ns = start ? InitCounter : Idle;
        InitCounter: ns = start ? InitCounter : InitReg;
        InitReg: ns = ReadFile;
        ReadFile: ns = Calc;
        Calc: ns = cout25 ? WriteFile : Calc;
        WriteFile: ns = cout6bit ? Finish : ResetCounter;
        ResetCounter: ns = ReadFile;
        Finish: ns = Finish;
        default: ns = Idle;
    endcase
end

always @(ps, start, cout25, cout6bit) begin
    case (ps)
        Idle:begin
            {rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, counter6bit_en, write_input, rst_counter6bit, done} <= 9'b000000000;
        end
        InitCounter:begin
            {rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, counter6bit_en, write_input, rst_counter6bit, done} <= 9'b100000010;
        end
        InitReg:begin
            {rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, counter6bit_en, write_input, rst_counter6bit, done} <= 9'b010000000;
        end
        ReadFile:begin
            {rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, counter6bit_en, write_input, rst_counter6bit, done} <= 9'b001100000;
        end
        Calc:begin
            {rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, counter6bit_en, write_input, rst_counter6bit, done} <= 9'b000010000;
        end
        WriteFile:begin
            {rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, counter6bit_en, write_input, rst_counter6bit, done} <= 9'b000011100;
        end
        ResetCounter:begin
            {rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, counter6bit_en, write_input, rst_counter6bit, done} <= 9'b100000000;
        end
        Finish:begin
            {rst_counter, rst_in_reg, read_input, ld_ppr, counter25_en, counter6bit_en, write_input, rst_counter6bit, done} <= 9'b000000001;
        end
    endcase
```

DataPath:

```
    initial begin
        depth = 0;
        read_file_buffer = $fopen(read_file_name, "r");
        write_file_buffer = $fopen(write_file_name, "w");
        $fclose(read_file_name);
        $fclose(write_file_name);
    end

    always @(posedge write_input) begin
        if (!(^out2 === 1'bx))
            $fdisplay(write_file_buffer, "%b", out2);
    end

    always @(posedge read_input) begin
        scan_file = $fscanf(read_file_buffer, "%b\n", in_arr_temp);
    end

    always @(posedge read_input) begin : computation
        in1 <= in_arr_temp;
        if (depth < 66)
            depth = depth + 1;
    end

    wire [4 : 0] out3, out4;
    wire [4 : 0] count_;
    wire [5 : 0] count6bit;
    integer count;
    assign count = count_;

    always @(count) begin
        in2_[count - 2] <= ( out1[(count - 2)] ^ out3[(count - 2) % 5] ^ out4[(count - 2) % 5] );
    end

    assign in2 = in2_;

    reg25bit #(read_file_name) in_reg(in1, out1, clk, rst_in_reg, ld);
    reg25bit #(read_file_name) out_reg(in2, out2, clk, rst, ld);
    parity_finder pf(out1, out3);
    prev_parity_reg5bit ppr(out3, clk, rst, ld_ppr, out4);
    counter25 c25(counter25_en, clk, rst_counter25, count_, cout25);
    counter6bit c6b(counter6bit_en, clk, rst_counter6bit, count6bit, cout6bit);
```

در این دیتا پث ابتدا ورودی را در یک رجیستر میریزیم. سپس ستون ها را xor کرده و در prev_parity_reg5bit نگه‌داری می‌کنیم. در مرحله بعد از این ستون ها دوباره xor کرده و با استفاده از رجیستر prev_parity_reg5bit و هم‌چنین ورودی هر سطر در 25 کلاک (از کانتر برای شمردن کلاک ها استفاده می‌کنیم) در out_reg ریخته و سپس در کلاک آن را در فایل می‌ریزیم.

در نظر بگیرید که out2 همان خروجی رجیستر out_reg خواهد بود. همچنین out3 خروجی xor ستون ها و out4 خروجی prev_parity_reg5bit می‌باشد.

در لود کردن prev_parity_reg5bit کمی احتیاط می‌کنیم و با استفاده از کنترلر زمان هایی که آن باید لود بکند را ملاحظه می‌کنیم. چون اگر در زمان های درستی این کار انجام نشود، خروجی ما غلط خواهد بود.

TesteBench:

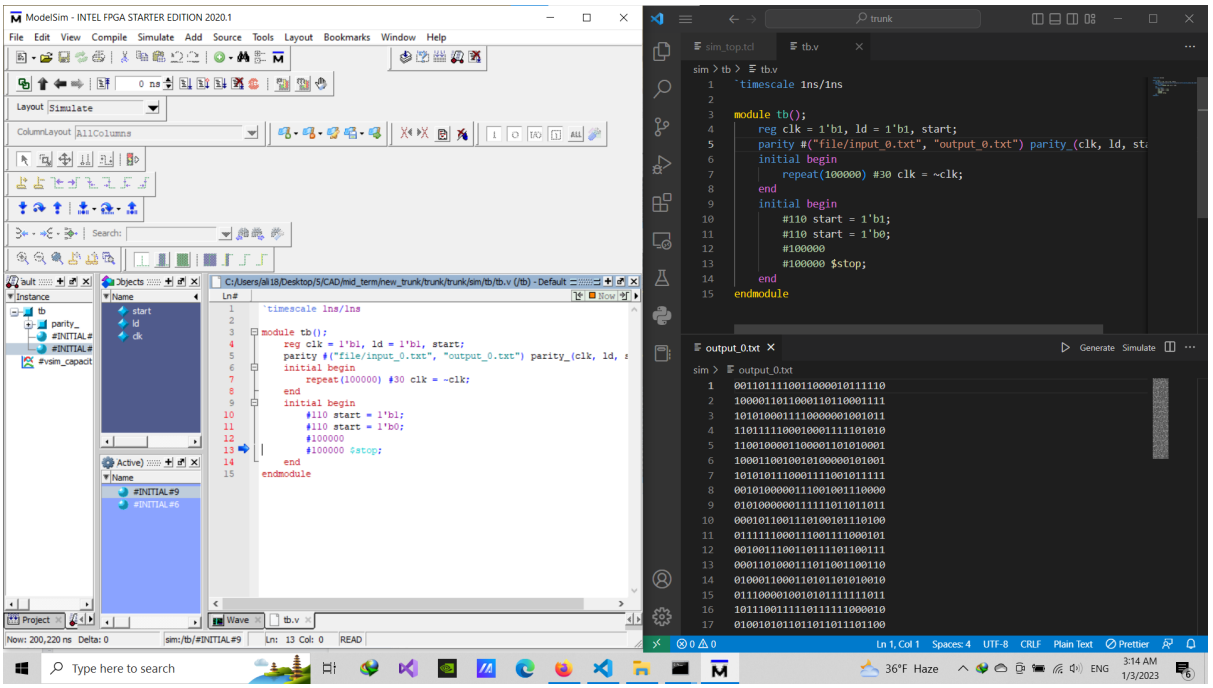
this is our simple test bench.

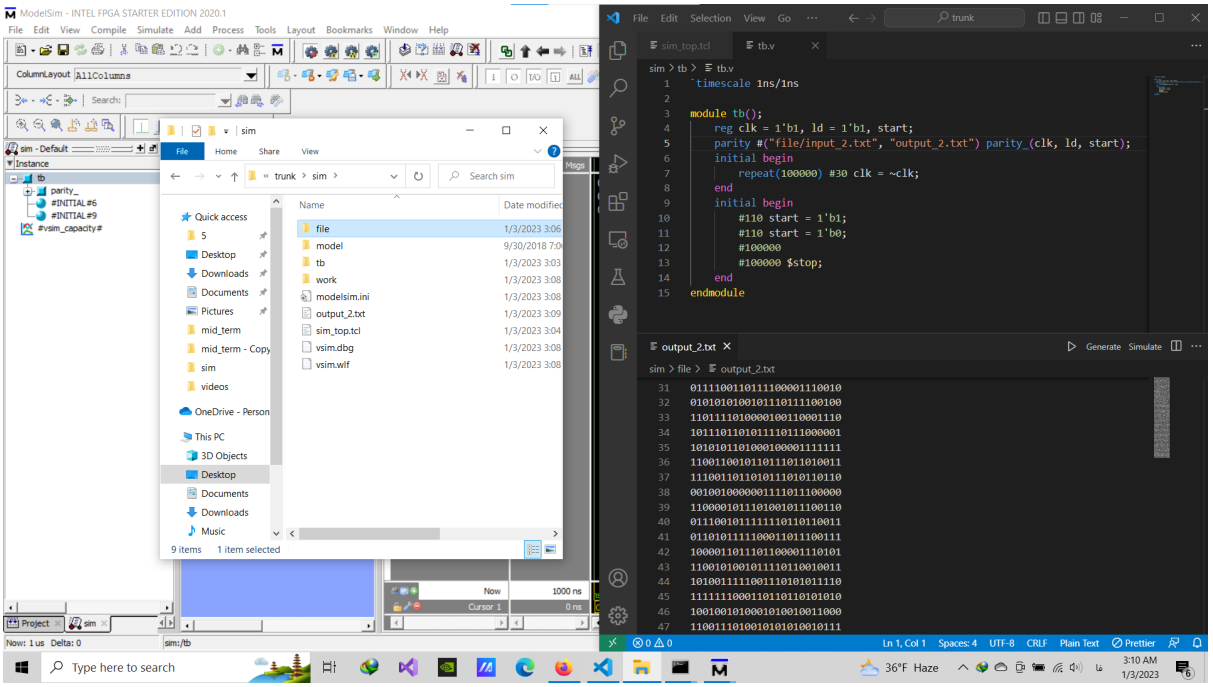
note that our permutate function gets input and output file name as arguments here,

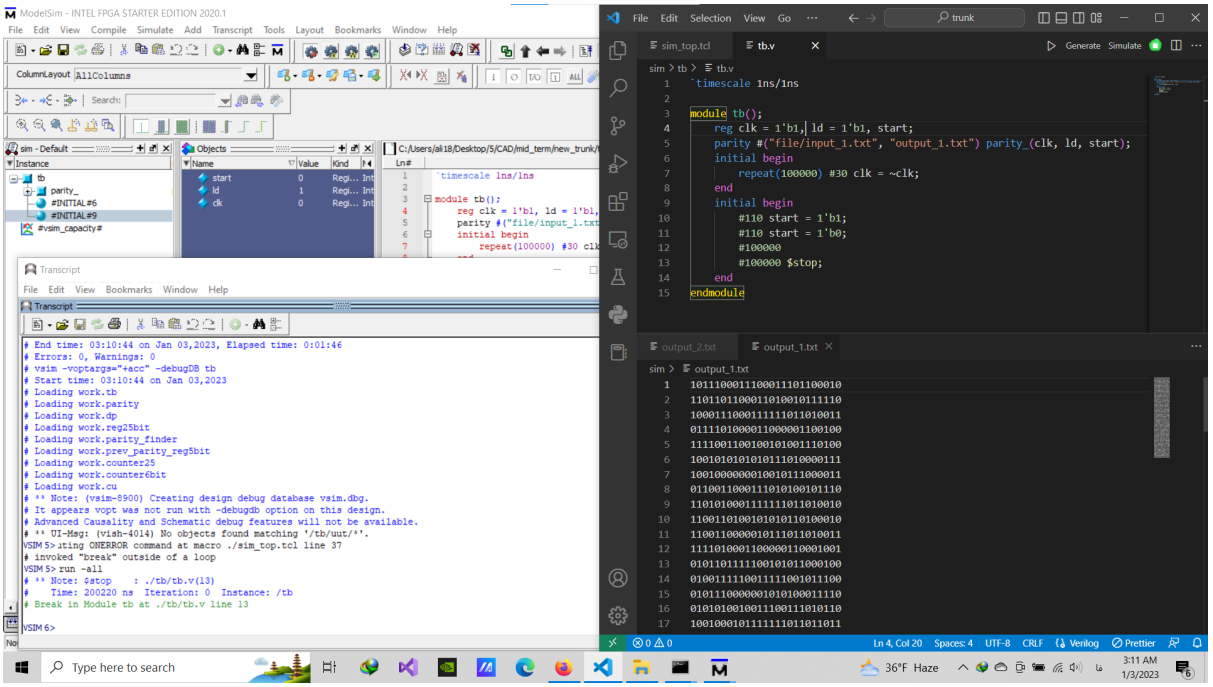
```
sim > tb > tb.v
1   `timescale 1ns/1ns
2
3   module tb();
4       reg clk = 1'b1, ld = 1'b1, start;
5       parity #("file/input_0.txt", "output_0.txt") parity_(clk, ld, start);
6       initial begin
7           repeat(100000) #30 clk = ~clk;
8       end
9       initial begin
10          #110 start = 1'b1;
11          #110 start = 1'b0;
12          #100000
13          #100000 $stop;
14      end
15  endmodule
```

TestsResults:

all three tests passed successsfully.







sim_top.tcl

```
    alias clc ".main clear"

    clc
    exec vlib work
    vmap work work

    set TB                  "tb"
    set hdl_path            "../src/hdl"
    set inc_path            "../src/inc"

    set run_time            "1 us"
#   set run_time            "-all"

#============================ Add verilog files  ==============================
# Pleas add other module here
    vlog    +acc -incr -source  +define+SIM     $hdl_path/counter6bit.v
    vlog    +acc -incr -source  +define+SIM     $hdl_path/counter25.v
    vlog    +acc -incr -source  +define+SIM     $hdl_path/cu.v
    vlog    +acc -incr -source  +define+SIM     $hdl_path/dp.v
    vlog    +acc -incr -source  +define+SIM     $hdl_path/parity_finder.v
    vlog    +acc -incr -source  +define+SIM     $hdl_path/parity.v
    vlog    +acc -incr -source  +define+SIM     $hdl_path/prev_parity_reg5bit.v
    vlog    +acc -incr -source  +define+SIM     $hdl_path/reg25bit.v

    vlog    +acc -incr -source  +incdir+$inc_path +define+SIM   ./tb/$TB.v
    onerror {break}

#================================ simulation ======================================

    vsim    -voptargs=+acc -debugDB $TB
```