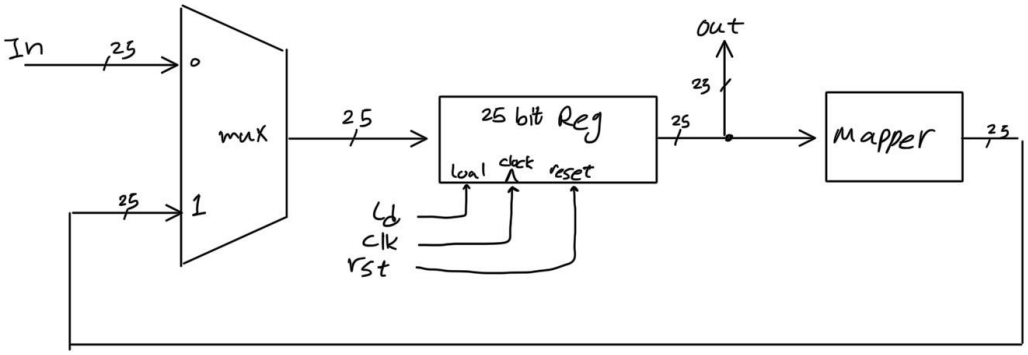
CA1: phase 2

علی هدائی ۸۱۰۱۹۹۵۱۳ پویا صادقی ۸۱۰۱۹۹۴۴۷

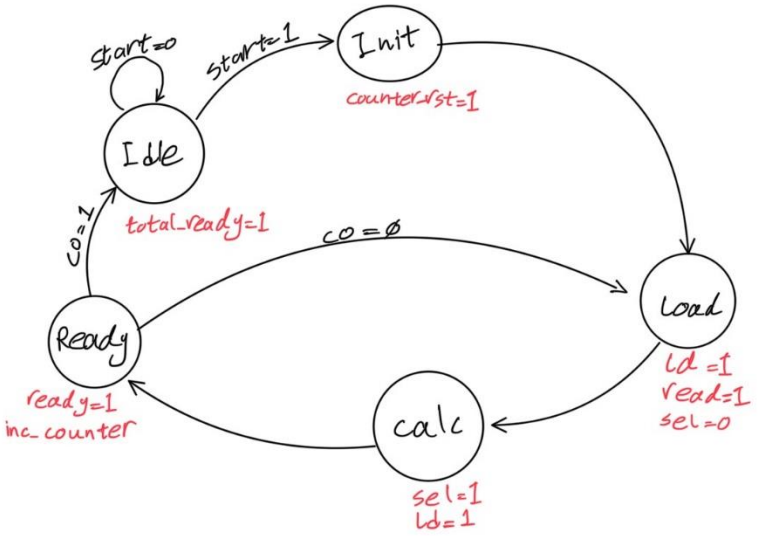
مسیر داده:



همانطور که در مسیر داده مشخص است، مقدار ورودی رحیستر ۲۵ بیتی یا میتواند از ورودی فایل باشد یا حاصل مپ کردن. پس برای مشخص کردن این ورودی از یه مولتی پلکسر استفاده میکنیم به همراه سیگنال مشخص کننده sel. رجیستر ۲۵ بیتی مان هم سیگنال های clk و rst و ld میگیرد که سیگنال ld برای لود کردن مقدار ورودی است.

مپر در واقع می آید طبق فرمول داده شده محاسبه میکند که هر کدام از بیت های رجیستر به کدام بیت باید نظیر شود بدین گونه که به صورت دو بعدی آن را در نظر میگیرد و درایه های آن را مطابق جدول مپ میکند وسپس طبق فرمول مختصات جدید را بدست میآورد و آن را به یک بعدی تبدیل میکند ونظیر میکند. بدین ترتیب در پالس بعد با یکی بودن کلاک مقادیر رجیستر مطابق خواسته مسئله جا به جا میشود.

کنترلر:



کد مسیر داده:

DP.v

```
module Multiplexer25bit2to1 (a0, a1, sel, w);
    input [24:0] a0, a1;
    input sel;
    output [24:0] w;

    assign w = ~sel ? a0 : a1;
endmodule

module Register (clk, rst, ld, in, out);
    input clk, rst, ld;
    input [24:0] in;
    output reg [24:0] out;

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            out <= 24'd0;
        end
        else begin
            if(ld == 1'b1) begin
                out <= in;
            end
        end
    end
endmodule

module Mapper (in, out);
    parameter N = 5;

    input [24:0] in;
    output [24:0] out;

    genvar i, j;

    generate
        for (i = 0; i < N; i = i+1) begin
          for (j = 0; j < N; j = j+1) begin
            assign out[(((j+2)%5))+((((((2*i)+(3*j))%5)+2)%5)*5] = in[((j+2)%5)*5+((i+2)%5)];
          end
        end
    endgenerate
endmodule

module DP (in, clk, rst, sel, load , out);
    input [24:0] in;
    input clk, rst, sel, load;
    output [24:0] out;

    wire[24:0]registerIn, maperOut, registerOut;
    Multiplexer25bit2to1 mux(.a0(in), .a1(maperOut), .sel(sel), .w(registerIn));
    Mapper mapper(.in(registerOut), .out(maperOut));
    Register register(.clk(clk), .rst(rst), .ld(load), .in(registerIn), .out(registerOut));
    assign out = registerOut;
endmodule
```

کد کنترلر:

CU.v

```
module CU (clk, rst, start, sel, ld, total_ready, ready, read);
    parameter [2:0] Idle = 0, Init = 1, Load = 2, Calc = 3, Ready = 4;

    input clk, rst, start;
    output reg sel, ld, total_ready, ready, read;

    wire co;
    reg [2:0] ps, ns;
    reg [5:0] count;
    reg counter_rst, inc_counter;

    always @(ps or co or start) begin
        ns = Idle;
        case(ps)
            Idle:  ns = (start) ? Init : Idle;
            Init:  ns = Load;
            Load:  ns = Calc;
            Calc:  ns = Ready;
            Ready: ns = (co) ? Idle : Load;
        endcase
    end

    always @(ps or co or start)begin
        {sel, ld, total_ready, ready, read, counter_rst, inc_counter} = 7'd0;
        case (ps)
            Idle: begin
                total_ready = 1'b1;
            end
            Init: begin
                counter_rst = 1'b1;
            end
            Load: begin
                sel = 1'b0;
                ld = 1'b1;
                read = 1'b1;
            end
            Calc: begin
                sel = 1'b1;
                ld = 1'b1;
            end
            Ready: begin
                ready = 1'b1;
                inc_counter = 1'b1;
            end
        endcase
    end

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            ps <= Idle;
        end else begin
            ps <= ns;
        end
    end

    always @(posedge clk or posedge rst) begin
        if (rst) begin
```

```
        end
    end

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            count <= 6'd0;
        end
        else begin
            if (counter_rst) begin
                count <= 6'd0;
            end
            else if (inc_counter) begin
                count <= count+1;
            end
        end
    end

    assign co = &count;
endmodule
```

ماژول اصلی:

Permutation.v

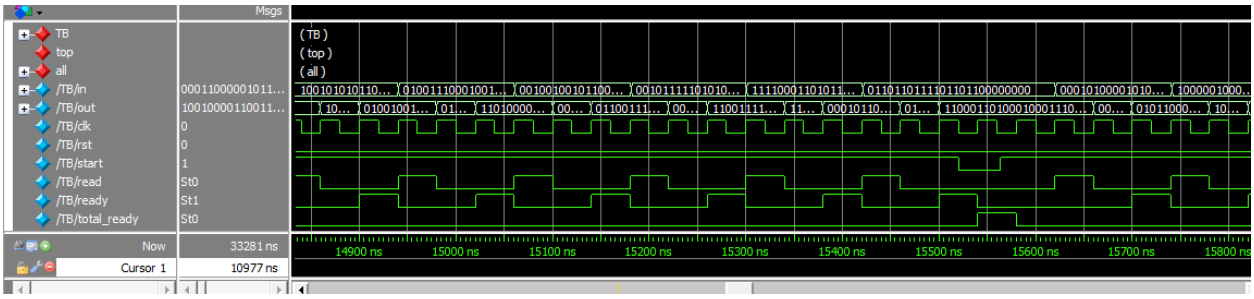
```
module Permutation (in, clk, rst, start, total_ready, ready, read, out);
    input [24:0] in;
    input clk, rst, start;
    output total_ready, ready, read;
    output [24:0] out;

    wire load,sel;
    CU cu(.clk(clk), .rst(rst), .start(start), .sel(sel), .ld(load),
          .total_ready(total_ready), .ready(ready), .read(read));
    DP dp(.in(in), .clk(clk), .rst(rst), .sel(sel), .load(load) , .out(out));
endmodule
```

اجرا کردن trunk :

```
# It appears vopt was not run with -debugdb option on this
# Advanced Causality and Schematic debug features will not
# ** UI-Msg: (vish-4014) No objects found matching '/TB/uut/
# Executing ONERROR command at macro ./sim_top.tcl line 32
# invoked "break" outside of a loop
# ** Note: $stop    : ./tb/TB.sv(38)
#    Time: 33281 ns  Iteration: 0  Instance: /TB
# Break in Module TB at ./tb/TB.sv line 38
add wave -position end  sim:/TB/in
add wave -position end  sim:/TB/out
add wave -position end  sim:/TB/clk
add wave -position end  sim:/TB/rst
add wave -position end  sim:/TB/start
add wave -position end  sim:/TB/read
add wave -position end  sim:/TB/ready
add wave -position end  sim:/TB/total_ready
add wave -position end  sim:/TB/inputFile
add wave -position end  sim:/TB/i
add wave -position end  sim:/TB/outFile
add wave -position end  sim:/TB/inputFileCounts
add wave -position end  sim:/TB/k
add wave -position end  sim:/TB/inputFileName
add wave -position end  sim:/TB/outputFileName
```

ویو فورم:



فایل های خروجی: