GROUP WRITING PROJECT

Project Name: Review on Logic Design

Due Date: 2022/12/28

This is the tracking document for our group writing project. We'll use this document to share details, resources, and schedules for our project. Please keep this document up-to-date as we work.

Participants

Name	Role	Student ID
Ali Keramati	Author	810198519
Ali Pakdel Samadi	Author	810198368

Milestones

Milestone name	Target date	Status
Datapath	2022/12/28	done
Controller	2022/12/28	done
Testbench	2022/12/28	done

Controller

As shown in the picture below, our Controller has 3 inputs(clk, rst, start) and 1 output(lineRegLoad).

Our controller has 4 states called:

1.Start = 0

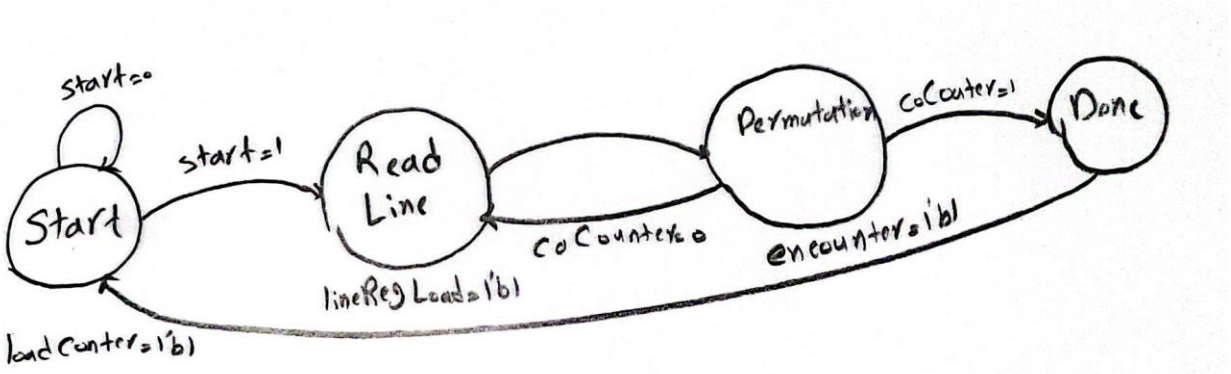
2.ReadLine = 1

3.Permutation = 2

4.Done = 3

We also have a counter in our controller design which count up to 64 and handles the depth of matrices.

```
`timescale 1ns/1ns
module Controller (
    clk,
    rst,
    start,

    lineRegLoad
);

    input clk, rst, start;
    output reg lineRegLoad;


    parameter [2:0]
        Start = 3'b000,
        ReadLine = 3'b001,
        Permutation = 3'b010,
        Done = 3'b011;


    reg [3:0] ps, ns;

    reg [5:0] counterInit = 6'b000000;
    reg loadCounter = 1'b0;
    reg enCounter = 1'b0;
    wire [5:0] count;
    wire coCounter;

    Counter #6 counter(.clk(clk), .rst(rst), .loadEn(loadCounter), .initCount(counterInit), .en(enCounter), .count(count), .cout(coCounter));

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

    always @(ps, start, coCounter) begin
        case (ps)
            Start:          ns = start ? ReadLine : Start;
    always @(ps, start, coCounter) begin
        case (ps)
            Start:          ns = start ? ReadLine : Start;
            ReadLine:       ns = Permutation;
            Permutation:    ns = coCounter ? Done : ReadLine;
            Done:           ns = Start;
            default:        ns = Start;
        endcase
    end

    always @(ps) begin
        {lineRegLoad, loadCounter, enCounter} = 3'b000;
        case (ps)
            Start:
                loadCounter = 1'b1;
            ReadLine:
                lineRegLoad = 1'b1;
            Permutation:
                enCounter = 1'b1;
        endcase
    end


endmodule
```



Datapath

```
1   `timescale 1ns/1ns
2   module Datapath #(parameter lineSize)(
3       clk,
4       rst,
5       lineRegLoad,
6       line,
7
8       permutationOutput
9   );
10
11      input clk, rst, lineRegLoad;
12      input [lineSize-1:0] line;
13
14      output [lineSize-1:0]permutationOutput;
15
16
17      wire [lineSize-1:0] lineRegOutput;
18
19      Register #(lineSize) lineRegister(.clk(clk), .rst(rst), .load(lineRegLoad), .reg_input(line), .reg_output(lineRegOutput));
20
21      Permutation #(lineSize) permutation(.inp(lineRegOutput), .permutationOutput(permutationOutput));
22
23  endmodule
```

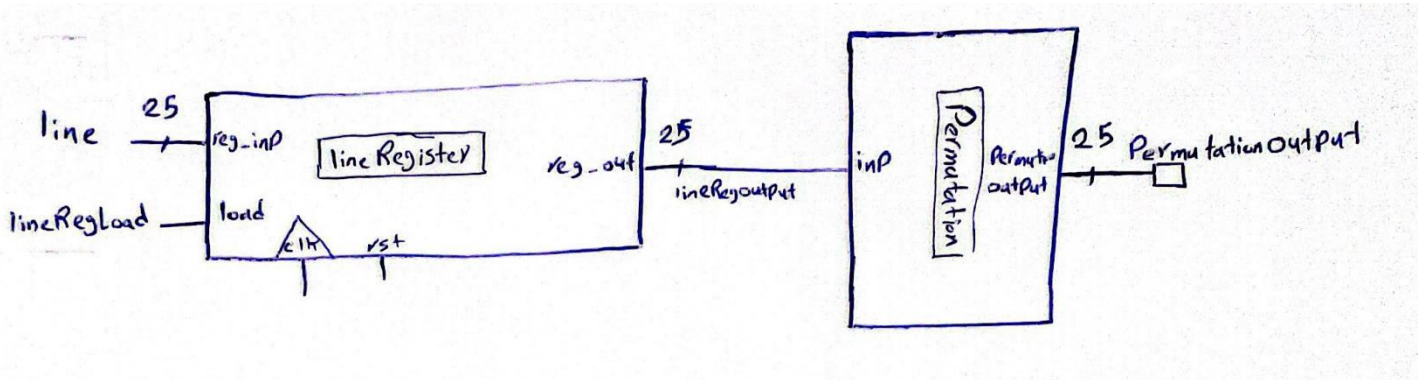
As shown in the picture above, our datapath has 4 inputs(clk, rst, lineRegLoad,line) and 1 output(permutationOutput).

We have parameterized the Datapath module with lineSize, in order to have flexible module with different matrices as input.

In our module there is a Register with size #25 called lineRegister. This register is synced with the clock pulse and has asynchronous reset. Whenever its load signal is issued the reg_input wire would be stored in the registered and it can be derived with its output wire called lineRegOutput.

Next to out register module, there is another module called Permutation. This module is responsible of calculation of the permutation function on matrix. It has an input wire called inp which is deriven from the register output. And also the output of this module is the final permuted matrix.

So we only have one register for both input and output and the result of each matrix will be written to file directly.



Test cases


input_0 - Notepad
File Edit View
Ln 6, Col 13 100% Unix (LF)
output_0 - Notepad
File Edit View
Ln 12, Col 2 100% Windows
input_1 - Notepad
File Edit View
Ln 1, Col 1 100% Unix (LF)
output_1 - Notepad
File Edit View
Ln 1, Col 1 100% Windows
input_2 - Notepad
File Edit View
Ln 40, Col 2 100% Unix (LF)
output_2 - Notep
File Edit Vie
Ln 1, Col 1 100%