

Mini-Project Lab Assignment

(CAO Theory & CAD of EC Lab [KEC – 554])

A Project report

to be Submitted by –

Suraj

18456

(Electronics Engineering)

Submitted to -

Mr. Rahul Ratnakumar



Department of Electronics Engineering

Kamla Nehru Institute of Technology Sultanpur, 228118

2019-2020

Objective:

Given an input of 100 sorted data points, divide them into 4 groups, labelling from 1 to 4.

Easy approach(But not sufficient):

- I can use simply a for loop that iterate over all data points, and display them into given number of groups.
- So, here in this approach were are just printing data points in a row and labelling their group number.
- But this is not sufficient because we can't use those groups if future requirement.

(*just printing the data points in different group not a solution, we should divide the whole memory allocated into different groups to use those groups in efficient manner.)

My Approach to the problem:

- This problem is simply based on dividing the memory allocated to the points into different groups to perform different – different tasks on individual group.
- So, I'm going to create four different memories to store each group. (*I can also use that single memory, but right now the idea behind it just to make the project and idea clear to all and learning the concepts)
- Total number of data points in each group can be calculated by dividing the total number of data points by total number of groups.

- So, now according to the problem we are given total 100 data points, and we've to divide them into four groups, hence each group contains total 25 data points. (*Groups are labelling from 1 to 4).

An advanced idea to use different groups:

- Here I am using 4x1 MUX that contain 2 selection lines.
- By using the selection lines one can use the desired group to perform in any task.

Hardware Designing:

- Verilog HDL modelling language supports three kinds of modelling styles:
- gate-level, dataflow, and behavioral. The gate-level and dataflow modelling are
- used to model combinatorial circuits whereas the behavioral modelling is used for
- both combinatorial and sequential circuits.

Here I am using behavioral modelling in most of the cases to make it more simple.

So let's start step-wise discussion for the designing.

1. Identifying the required inputs:

- To get the input as 100 sorted data points I am reading an input file "**datazen.txt**" that contain 100 sorted data points in **Hexadecimal** number system.

2. Constraints (digit size):

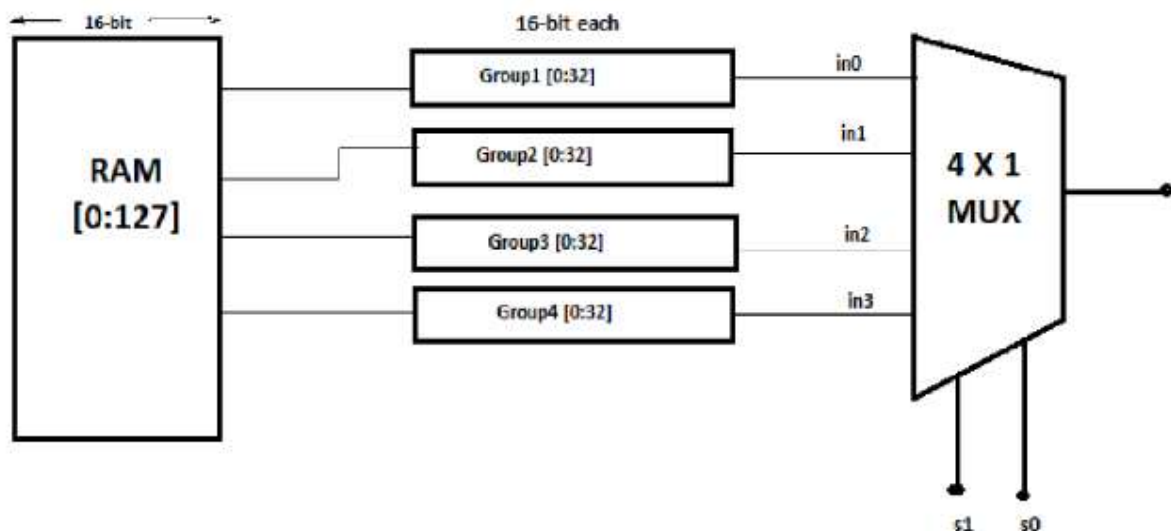
- Here I am creating **reg [15:0]** **RAM[0:127]** that has **16-bit** in each row to store a single digit in each row.
- So, we can store 128 digits total, and the digit can be size of maximum **$(6 \times 10^4)_{10}$ or $(FFFF)_{Hex}$** (*input data points must be sorted).

3. Memory for Different groups:

- Here I am creating 4 different groups **Reg [16:0] group[0:32]** to store data points.

4. Selection of groups:

- Using **4x1 MUX** to select different groups.



Designing of Code:

```
C:/Modeltech_pe_edu_10.4a/examples/group_divider.v (/group_divider) - Default

Ln#
1  module group_divider;
2      reg [15:0] RAM [0:127];           //memory for storing all data points (16-bit)
3      reg [15:0] group1[0:31];          //memory for group 1
4      reg [15:0] group2[0:31];          //memory for group 2
5      reg [15:0] group3[0:31];          //memory for group 3
6      reg [15:0] group4[0:31];          //memory for group 4
7      integer i;                        //initialization iterator i for loop
8      initial
9      begin
10         $readmemh("datasen.txt",RAM); //reading data file
11     end
12     initial
13     begin
14         for(i=0;i<=99;i=i+1)
15         begin
16
17             if(i<=24)group1[i]=RAM[i]; //dividing into group 1
18             if(i>24 && i<=49)group2[i-25]=RAM[i]; //dividing into group 2
19             if(i>49 && i<=74)group3[i-50]=RAM[i]; //dividing into group 3
20             if(i>74 && i<=99)group4[i-75]=RAM[i]; //dividing into group 4
21
22         end
23     end
24     initial
25     begin
26         for(i=0;i<=99;i=i+1)
27         begin
28
29             if(i==0)$display("Here Group: 1\n");
30             if(i>=0 && i<=24)$display("point: %d\n",group1[i]); //displaying group 1
31             if(i==24)$display("Here Group: 2\n");
32             if(i>24 && i<=49)$display("point: %d\n",group2[i-25]); //displaying group 2
33             if(i==49)$display("Here Group: 3\n");
34             if(i>49 && i<=74)$display("point: %d\n",group3[i-50]); //displaying group 3
35             if(i==74)$display("Here Group: 4\n");
36             if(i>74 && i<=99)$display("point: %d\n",group4[i-75]); //displaying group 4
37
38         end
39     end
40 endmodule
41
```

Transcript x groups.v x group_divider.v x Project x Memory List x sim x

Ln: 23 Col: 3 Project : T_flipflop Now: 100 ns Delta: 0 sim:/group_divider

Output: Compilation Result – compile successfully

```
# Compile of group_divider.v was successful.
VSIM 15> vsim -gui work.group_divider
# vsim
# Start time: 16:51:26 on Dec 20,2020
# Loading work.group_divider
```

Groupwise Distribution:

```
# Loading work.group_divider
VSIM 16> run
# Here Group: 1      # Here Group: 2      # Here Group: 3      # Here Group: 4
#                   #                   #                   #
# point:      1      # point:      26      # point:      51      # point:      76
# point:      2      # point:      27      # point:      52      # point:      77
# point:      3      # point:      28      # point:      53      # point:      78
# point:      4      # point:      29      # point:      54      # point:      79
# point:      5      # point:      30      # point:      55      # point:      80
# point:      6      # point:      31      # point:      56      # point:      81
# point:      7      # point:      32      # point:      57      # point:      82
# point:      8      # point:      33      # point:      58      # point:      83
# point:      9      # point:      34      # point:      59      # point:      84
# point:     10      # point:      35      # point:      60      # point:      85
# point:     11      # point:      36      # point:      61      # point:      86
# point:     12      # point:      37      # point:      62      # point:      87
# point:     13      # point:      38      # point:      63      # point:      88
# point:     14      # point:      39      # point:      64      # point:      89
# point:     15      # point:      40      # point:      65      # point:      90
# point:     16      # point:      41      # point:      66      # point:      91
# point:     17      # point:      42      # point:      67      # point:      92
# point:     18      # point:      43      # point:      68      # point:      93
# point:     19      # point:      44      # point:      69      # point:      94
# point:     20      # point:      45      # point:      70      # point:      95
# point:     21      # point:      46      # point:      71      # point:      96
# point:     22      # point:      47      # point:      72      # point:      97
# point:     23      # point:      48      # point:      73      # point:      98
# point:     24      # point:      49      # point:      74      # point:      99
# point:     25      # point:      50      # point:      75      # point:     100
#                   #                   #                   #
VSIM 17>
```

