Design a circuit that calculates average of the smallest data and the largest data from 8-Bytes data

First and Second Test and Testability Homeworks

Mohammad Hashemi
Univercity of Tehran
Tehran,Islamic Republic of Iran
md.hashemi@ut.ac.ir

***Abstract*—This document describe how to design a circuit that calculates average of the smallest and largest data from 8-Bytes data from a BUS and include the result of Verilog simulations and schematics of process.**

8-bit Controller, Data path, Huffman model,

Magnitude comparator

I. Introduction

This document, modified for the first homework of test and testability design. This paper is going to describe: (1) how to design a circuit that calculates our operations, (2) show us the schematics of our circuit process, and (3) give us the vision by simulation result of Verilog Sources.

II. Design Phase

A. Schematic circuit view

First, we describe our circuit function in a schematic view for this we use Microsoft Visio. Here is the functional process:

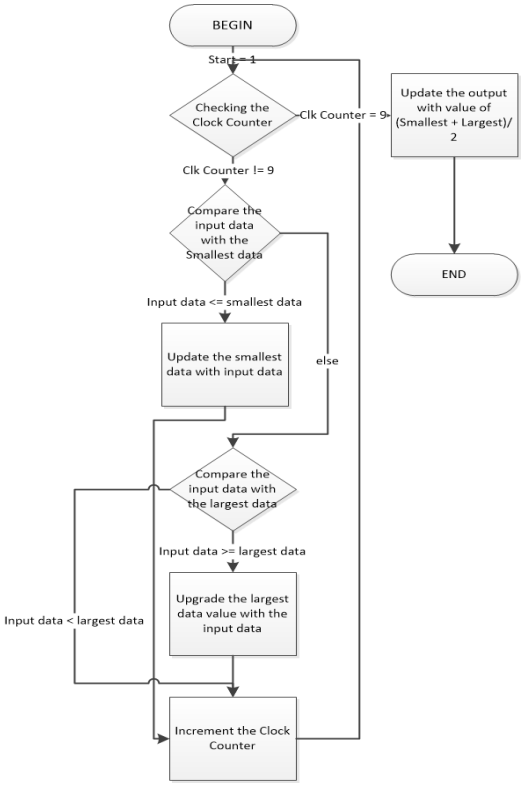

Fig.1 circuit process schematics

B. Elements of Circuit

Our circuit includes an 8-bit comparator and an 8-bit BUS and an Arithmetic Logic Unit that calculates our summation. All of these elements are described in Verilog format and they simulated with Modalism synthesizer and the results will show in the next categories.

III. Test Phase

After all these simulations and Verilog descriptions we are going to test our circuit with three kinds of vectors:

- Deterministic vectors
- Random vectors
- TextIO

A. Deterministic Vectors

As deterministic inputs we trigger the input with these vectors in binary form: (first input set)

- 185 = 8’b10111001
- 75 = 8’b01001011
- 170 = 8'b10101010
- 215 = 8'b11010111
- 77 = 8'b01001101
- 222 = 8'b11011110
- 225 = 8'b11100001
- 57 = 8'b00111001

The second input set is:

- 15
- 76
- 218
- 38
- 41
- 241
- 112
- 72

The result of first input set must be: 75

The result of second input set must be: 128

This controller phase include **Huffman Method**.

B. *Random Vectors*

As random inputs we trigger the input as random vector. For this we use $random syntax in Verilog test bench (showed in Fig.2).

```
initial repeat(8) data = $random;
```

Fig.2 $random syntax

C. *TextIO*

For triggering the inputs as TextIO method we use $read and $scanf to read information from an external file.

The vectors in inputs.txt are in fig.3:

```
10001101
11110011
10001101
00001101
10101010
01001010
11101110
10010001
```

Fig.3 inside inputs.txt information

IV. Simulation Results

After all these assignments and provide input vectors we are going to see output result.

A. *Results of Deterministic Input Vectors*

As we applied our input vectors in III.A we expect our circuit picks the 75 input vector as smallest data and 225 as the largest data and also the output must be:

(First input set)

(225 + 75) / 2 = 141

And the binary result of output is “8’b10010110”.

(Second input set)

(241 + 15) / 2 = 128

For display our output we use $display or $monitor syntax in Verilog (showed in fig.4).

```
begin
  ALU (wlargest,wsmallest,ww);
  $display("\nLargest is : %a", wlargest);
  $display("\nSmallest is : %a", wsmallest);
  $display("\nThe average output is : %a", ww);
  donew = 1;
end
```

Fig.4 $display syntax

The results of simulation as we expected showed in waveform in fig.5 and fig.7 the $display results showed in fig.6 and fig.8:

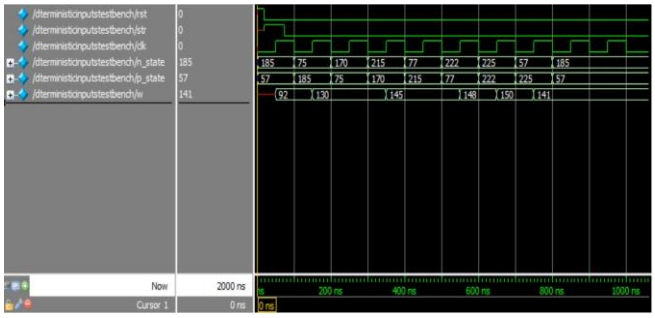

Fig.5 first deterministic inputs waveform results

```
Transcript
# vsim -voptargs="+acc" work.dterministicinputstestbench
# Start time: 00:46:54 on Oct 17,2018
# ** Note: (vsim-3813) Design is being optimized due to module recompilation.
#
# Loading work.dterministicinputstestbench(fast)
# Loading work.my_main(fast)
# Loading work.mycomparator(fast)
add wave -position insertpoint  \
sim:/dterministicinputstestbench/rst \
sim:/dterministicinputstestbench/str \
sim:/dterministicinputstestbench/clk \
sim:/dterministicinputstestbench/n_state \
sim:/dterministicinputstestbench/p_state \
sim:/dterministicinputstestbench/w
VSIM 20> run
# our average is : 92
#
# our average is :130
#
# our average is :145
#
# our average is :148
#
# our average is :150
#
# our average is :141
#
VSIM 21> quit -sim
```

Fig.6 first deterministic inputs $display results

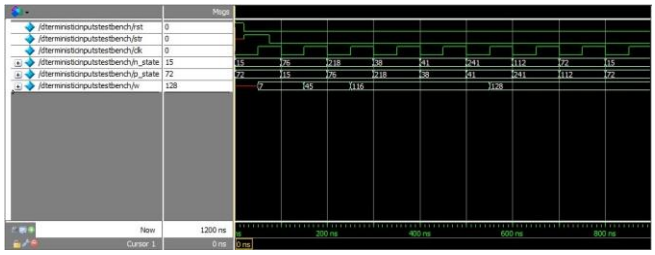

Fig.7 second deterministic inputs waveform results

```
# Compile of my_main.v was successful.
# Compile of random_inputs_testbench.v was successful.
# Compile of my_comparator.v was successful.
# Compile of deterministic_inputs_testbench.v was successful.
# Compile of textio_inputs_testbench.v was successful.
# 5 compiles, 0 failed with no errors.
ModelSim> vsim -voptargs=+acc work.dterministicinputstestbench
# vsim -voptargs=+acc work.dterministicinputstestbench
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
#
# Loading work.dterministicinputstestbench(fast)
# Loading work.my_main(fast)
# Loading work.mycomparator(fast)
VSIM 234> run
# our average is :  7
#
# our average is : 45
#
# our average is :116
#
# our average is :128
#
# our average is :128
#
# our average is :128
#
VSIM 235>
```

Fig.8 second deterministic inputs $display results

B. *Random Vectors Input Simulation Result*

As inputs are random we do not expect specific output so start the simulation and we will see the result. For example our circuit in one situation displays this output (waveform fig.9 and display form fig.10):

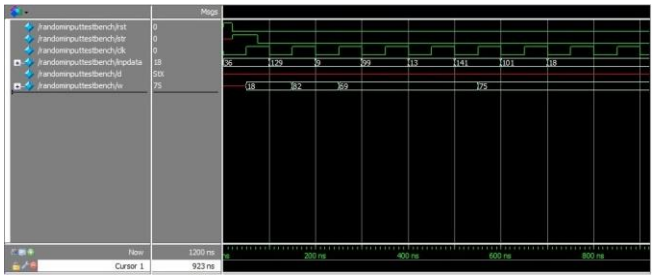

Fig.9 random inputs result waveform

```
VSIM 235> quit -sim
ModelSim> vsim -voptargs=+acc work.randominputtestbench
# vsim -voptargs=+acc work.randominputtestbench
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
#
# Loading work.randominputtestbench(fast)
# Loading work.my_main(fast)
# Loading work.mycomparator(fast)
VSIM 237> run
# our average is : 18
#
# our average is : 82
#
# our average is : 69
#
# our average is : 75
#

VSIM 238>
```

Fig.10 random inputs result display with $display

C. *TextIO Input Simulation Result*

This method is exactly like the deterministic method. The only deference is that we read deterministic inputs from an external file like a (.txt) file or an eternal network or USART etc. Except we apply the inputs as internal parameters.

We can see the outputs.txt file filled with the results step by step (fig.11).

```
10111011
01111011
01111101
```

Fig.11 outputs.txt

REFERENCES

[1] Professor Zain Navabi Kordestan University Verilog Lectures University of Kordestan, 2013.

[2] Professor Zain Navabi Test and Testability Lectures Univercity of Tehran, 2018.