Programming Project with HWU machine (20% of the final mark)

From the lectures slides on vision and tutorial problems you can see that this machine is fully programmable allowing unconditional and conditional jumps and IO control. Using instruction set described in the document **HWU_machine_instructions.pdf** you are required to write a program to execute this equation:

```
result = A
result = ((result + B) – C) xor D iterated N times
```

Download the Verilog scripts describing the HWU machine, the program and data files, and the vector waveform file:

a) cpu32_working_xor.v,

b) alu_xor.v

c) cpu32.vwf (timing simulator file)

d) instructTest.txt (program instructions)

e) loadData.txt (program data)

f) cpu34_disp_xor_clocked.v

g) display_drv.v

h) slowClock.v

The file in d) is a txt file which will hold your program written as a series of hexadecimal numbers(machine code) along with comments.

For example, Red is the 24 bit machine code, // is the comment delimiter, black is the comment

```
319060 //SUB R1-R1 (to make it 0) r1 = counter
100060 //load register 0 with data 0 D0=0
308060  //just in case make it 0 R0 (subtract R0-R0)
// loop1
120260 //load R2 D2 (D2=12) (and refresh it when we loop it)
Note the above code does not function, this an example only
/*
You may also put long comment like this SUB R1, R1 // zero R1
LD R0, D0 // load R0, with D0 SUB R0, R0 //zero R0
// begin loop 1
LD R2, D2 // load loop terminator
*/
```

The file in e) the data file ‘loadData.txt’ is organized similarly to the program file the numbers are data in hexadecimal and on compilation loaded into data register in the order given here. As before the delimiter // is the comment for that line

```
2 // D0
3 // D1
```

4 // D2

5 // D3

It is important to keep the same file names as the Verilog code ‘cpu32_working_xor.v’ and cpu34_disp_xor_clocked.v expects a file called instructTest.txt for the program and ‘loadData.txt’ for the data. If you change the program and or the data in each of these files, then you must recompile the Verilog file to incorporate the changes.

Debugging your program using cpu32_working_xor.v

load cpu32_working_xor.v, alu_xor.v

load the vector waveform file cpu32.vwf

in the main menu in Quartus, select assignments, settings, simulator settings and browse for the file cpu32.vwf.

Recompile your program

Run the timing simulation and you should see something like Figure 1

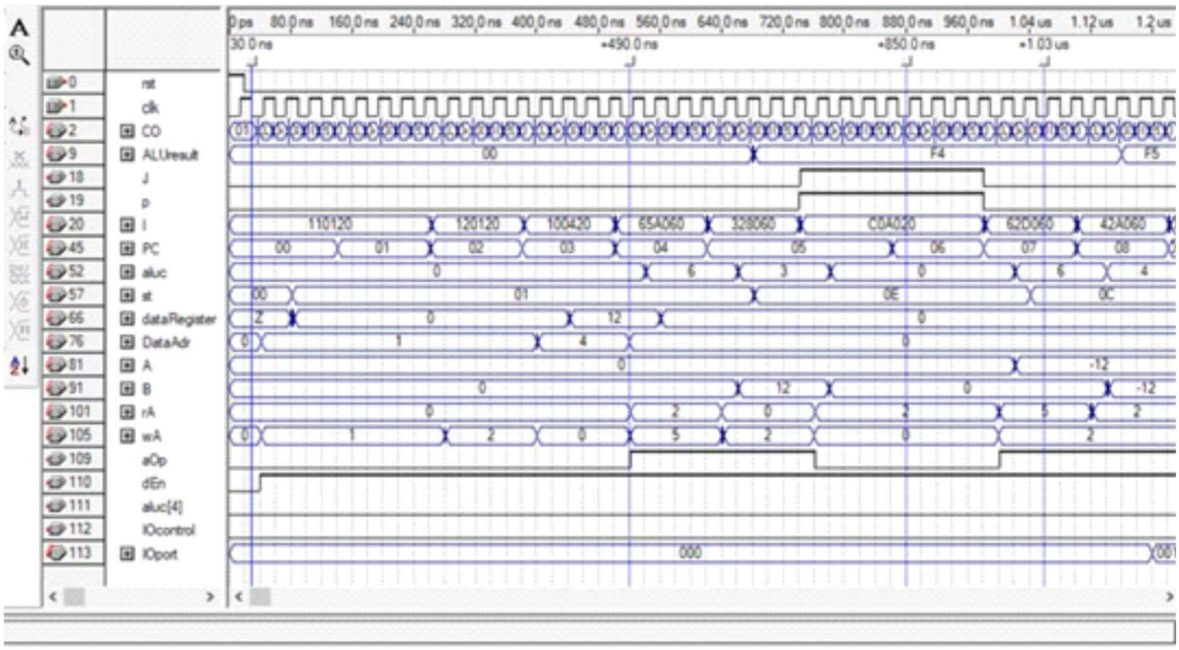


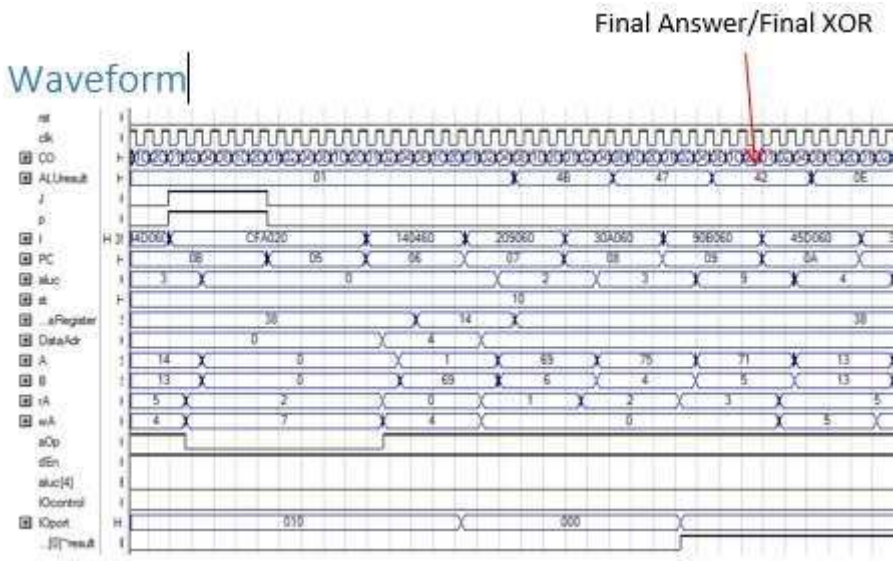
Figure 1 timing diagram after simulation

In Figure 1 the top line is general reset, next line clock and so on. After N iterations, after which theprogram should terminate, you should see the final result of the iterated equation in ALUresult in hexadecimal. Lines below PC (program counter) are shown in decimal number format. More information about each timeline can be found by examining the Verilog file cpu32_working_xor.v.

This is a groupwork of two (2) students. When you are ready to submit your work, it should be a single document (pdf) containing your program, comments, and result. The submission deadline is **Monday 1st May 2023, 23:59.**

You will be assessed on

Logically correct program and an image of your timing simulation (waveforms) with a clear indication of the answer (8 marks)



Proper comments (see note above on commenting) showing logic of your code (4 marks)

A correct answer and explanation on the results (8 marks)

Let *d* be the last digit of your ID (use ID of any member in the group).

If *d* = 0 use A = 38, B = 6, C = 4, D = 5, N = 14.

If *d* = 1 use A = -56, B = -12, C = 4, D = -27, N = 14.

If *d* = 2 use A = -10, B = -14, C = 4, D = 25, N = 14.

If *d* = 3 use A = -64, B = 3, C = 4, D = -8, N = 19.

If *d* = 4 use A = -33, B = 12, C = 4, D = 25, N= 4.

If *d* = 5 use A = -5, B = 14, C = 4, D = -5, N = 5.

If *d* = 6 use A = 15, B = 9, C = 4, D = -15, N = 13.

If *d* = 7 use A = -37, B = 7, C = 4, D = 16, N = 16.

If *d* = 8 use A = 33, B = 9, C = 4, D = 21, N = 11.

If *d* = 9 use A = -11, B = 8, C = 4, D = 5, N = 16.

Good Luck!