Hi All,

Please follow these mini project report guidelines.

1. Introduction

What project are you doing? Practical usage of your project and why you choose it. Very high-level description of your whole project (in few lines). I should have an idea about what I am going to find in the report from your introduction.

2. Specification of the project.

Add here the pictures (proj#1), state machine (proj#2), and operation table and schematic symbold (proj#3) from the manual. For proj#1 and proj#2 add a table (like in proj#3) showing the selections/buttons and corresponding operation.

3. Design and architecture

a) Block diagram of your project (different from state machine). You need to show the block diagram following your Verilog implementation. For example, the state machine block decides next state, the update state block updates the next state and takes input from clock block to make transition slower from one state to another.

b)  State machines for proj#1 and proj#2. You need to draw the state machine that matches your Verilog implementation. I will cross check with your codes. No truth table for proj#1 and proj#2.

c) Truth table showing three example for each case in proj#3. No state machine required.

4. Implementation

a) Problems and hurdles faced during design and implementation.

b) Does the implementation work as specified/designed?

c) Possible ways to improve it.

5. Verilog codes

Use small fonts, if possible use double column format to fit your codes nicely in the report. Just adding codes at the end of the report will cause point deduction for formatting.

5. Conclusion

What have you learned from it and how the project (the manual, specification, timeline etc.) may be improved?

 Formatting instruction

1. You need to have captions for all diagrams and tables.

2. You need to refer to all the figures, tables, codes and diagrams in your text. Just adding something and not mentioning it anywhere in the text is not good.

3. Bad formatting like one small diagram in one page will cause point deduction for formatting.

4. Do not use very large fonts. For example, use 10 pts for body and 12 points for headings.

Grading criteria.

Total 10 points.

Submission on time (Mar 26 6:00 pm sharp) - 2 pts

Specification -  1 pts

Design and architecture - 3 pts

Implementation - 1 pts

Verilog codes - 1 pts

Formatting and conclusion - 2 pts

Contract me if you have any questions.

Good luck with your finals.

Mohammad

Project Presentation:

-We made an ALU that can perform 4 operations on up to two 3-bit inputs.

-Unsigned addition, unsigned subtraction, bit equality test, division by 2

-unsigned addition: 3-bit + 3-bit → 4-bit output

-unsigned subtraction: input a > input b only as per specifications, 3-bit – 3-bit → 3-bit output, 7 is max value and 0 is min value, the carry out is ignored

-bit equality test: compare input a to input b and light up LED0 if equal

-division by 2: shift input a 1-bit to the right and output on LED1LED0, since output is at most 3 and at least 0

-General approach involved creating the 4 separate data paths with a mux that selects the final output of the operations according to the user input

-2 select switches to pick operation, output is displayed on 4 LED’s.

Individual Defense:

Design choices:

-Picked the multiple data path and mux approach because it is conceptually easier

-Reused the adder for unsigned subtraction because it saved on creating a new module specifically for unsigned addition.

Problems encountered:

-Unsigned subtraction wasn’t working because there was an overflow bit from doing two’s complement and the 3-bit adder was not narrow enough to ignore the carry out bit, so we applied a bit mask to the first bit.

-Problems with the test bench – we copied over the verilog modules and ucf into an empty project and it just worked.

-General verilog code issues, such as assigning a wire vs register, instantiating instances of modules, etc.

Improvements:

-Taking the ALU extender and adder approach would improve the scalability of the design if we were to add more components.

-Using wider inputs/outputs, more operations like gt/lt, n-bit shift, wider adder, signed addition and subtraction, etc.