ENGR xD52: MP b010

Due Nov 27th, but you really ought to consider getting it in before Thanksgiving break.

This lab stresses time and expectation management. You will need to make decisive tradeoffs in order to finish the lab within the time allocated.

One representative from each team must submit all deliverables electronically to comparch2012@gmail.com as a single compressed archive. Please format the email's Header as "[CA] [MP1] Name1 Name2 Name3"

The Situation

You are the engineering team of a small silicon startup aiming to take over the low end of the embedded motion control market with your new processor. The Babbies in marketing have just returned from a "wildly successful" lunch with a potential major investor, which is code for "they have wildly over-promised the state of development and expect you to make up the difference".

You are now on the hook to create a plausible demo of your as of yet non-existent hardware… Tomorrow. The fate of your company depends on it.

The Timing

In order to simulate the time constraint of what marketing has done to you, no member of your team is allowed to work more than 12 hours on the lab. Individually track the hours spent by each team member, and what portion of the project those hours were spent on.

These hours do not have to occur at the same time, and can be spread out as appropriate. You may "pause" your individual clock at your discretion.

The planning phase and the final report are not included in this timing.

The Planning

Before beginning your individual clocks, spend two hours as a team to create a plan of attack. This should include a breakdown of tasks, their predicted duration, who they are assigned to, and when they are "due". **Look at the dependencies** of the tasks, and break them to an appropriate level to do a realistic planning phase. Your final report should include a comparison of the expected timings vs actual timings. **Points for honesty, not accuracy.**

The Customer Need

The customer intends to use your core in their new Turboencabulator. Their control engineer wants to know how many iterations per second you can handle of an algorithm that includes the following:

1) The capacitive duractance is modeled with an 8 tap FIR filter. The tap constants will change only when the logarithmic casing is re-spurved. Inputs and outputs are I16Q16.
2) Each of the six hydracoptic marzul vanes have a rotary sensor for which they need to calculate arcsine and arccosine. These come in as unit-less I1Q31 readings and need to be converted to angles with 16 bit precision to reduce sinusoidal deplenaration.
3) The Unilateral Phase Detractors require the square-root of the prefabulated amulite's temperature. The measured subsystem has a thermal time constant of a half second. The resulting precision needs to be at least 12 bit and be updated every control loop iteration.

When pressed for additional details about "the algorithm" he guffaws loudly, sending a wave a crumbs from his neck-beard and explains that it is highly confidential and that you couldn't possibly understand it anyway. It looks like you will need to make some assumptions…

The Core

Your company is to produce the core of a microcontroller. It may be MIPs compatible or one of your own design. It may be single cycle, multicycle, pipelined, or some combination thereof. Remember, your customer is focused on execution speed of their algorithm, but their ability to buy your product is predicated on your execution speed of getting it ready to be sold. A fast promise of a product will lose to a slow actual product.

The absolute minimum instruction set is (the equivalent of) **J, BEQ, ADD, SLLV**. Your understanding of the customer's requirements will inform the rest of your instruction set.

The Testing

When presented with this type of schedule, the first thing a bad engineer will do is cut out testing. Proactive testing is well worth the time you put in to it, especially with teams. It is much more time efficient for an author to find a bug in their own module with a test than for a team mate to find that same bug while attempting integration.

Agree on a test strategy during your plan phase. Include a post mortem analysis of your test plan in your final report. Was it too much? Too little? What bugs did it catch? What bugs should it have caught? Most importantly, how will you test next time?

The Marketing Pitch

Create a 1 page brief of your core to be handed out by the Marketing team at the upcoming trade show. Maybe this will keep them honest…

The Final Writeup

The final writeup should read as an internally documented post-mortem of the "sprint". It is not covered by the time limit. It needs to include the following:

Deviations between the plan and the execution. What went wrong? What went right? How would you improve this for next time? Be honest. If everything went horribly, write a well thought out analysis of why.

Test results and test coverage confidence. Prove that the core is working as expected, or call out specifically what portions still need work. Write this clearly so that grading is easier. Happy graders give better grades.

3 specific ways to improve the core with respect to the customer's wishes. These should be about a paragraph each.

Hints / Notes

Design Reuse

You may freely reuse code created for previous labs. You may "purchase IP" from other groups' previous labs if necessary by asking them politely and then crediting them in your writeup.

Behavioral Verilog

It is highly recommended that you use behavioral Verilog for this lab. Behavioral Verilog is to Structural Verilog as Python is to Assembly: It favors engineer productivity over silicon size and speed.

English Verilog

It may become appropriate to implement some features in "design spec", rather than implement it in "it actually works now". This decision should be reflected in your planning.

For example, you may choose to mark a particular opCode as a yellow feature (to be completed after this design iteration). Clearly describe in english what it will do when it is implemented, and then refer to it in your performance analysis.

Square Root

You may wish to look in to the Newton Raphson approach for calculating the square root of a number. Pay special attention to the growth characteristic of the accuracy of its result.

Accuracy: Polynomial Fit

When modeling the accuracy of a polynomial fit, it can be helpful to run your algorithm in e.g. Matlab for a first pass approximation. This approximation can be 'good enough' so long as you avoid quantization / round off errors. That is, as long as you are calculating with sufficient extra bits.