RESEARCH PROJECT:
PROBLEM DEFINITION

This document describes the research project topic and goals for 2020 as a whole. Students should use this document in conjunction with the document describing the specific evaluation events that will measure performance at certain milestones. Students are reminded that all assignments related to this project, with the exception of Practical 1, should be done on an *individual basis* and work submitted is the student's own.

1. INTRODUCTION

We have become so accustomed to high level programming languages like Python, Matlab, C, Java, etc. that it is extremely easy to lose track of the tremendous effort that goes into the system layers that translate relatively simple function calls within these languages into instructions performed on very simple, low level fundamental digital logic and the very basic arithmetic functions that digital hardware within a CPU can offer. It is, as an example, so simple to call upon the 'math' package in Python and immediately have higher level mathematical functions like sin, cos, sqrt, etc. available to us. Have you ever wondered what really 'happens behind the scenes'? How are resources (memory, scheduling, processing, etc.) allocated to perform these functions using only very simple digital building blocks?

In this year's research project, we will explore the implementation of some of the most popular functions in hardware, and investigate relevant, recent and applicable (to FPGA implementation) methods to recreate these functions. Higher level mathematical functions exist in every digital area of interest for engineers, ranging from computation for DSP algorithms, scientific calculations, graphics and multimedia processing and artificial intelligence algorithms to name but a few. Although many examples exist, for this research project we will consider

- the sigmoid function,
- the hyperbolic tangent function,
- log and/or antilog function (base 10 or e),
- the dot-product,
- the cross product,
- the error function.

A number of different approaches exist, all with varying success depending on the function we are trying to approximate. In general, approaches include mathematical transformation into an approximate equation that can be more easily implemented in digital logic (say polynomial functions, Taylor series expansion, etc.), searching methods and numerical approximations (binary search algorithms, Newton-Rapson), etc.

The details of this forms part of the investigation. This assignment is aimed at investigating, implementing and analysing efficient function approximators/calculators for one specific function selected from the above list, with particular reference for this course for ease and suitability of implementation with low level digital hardware, particularly in VHDL for FPGA synthesis. Efficiency here can be interpreted as focussing on speed, memory and resource usage and minimisation, and/or a combination and trade-off between these parameters, but reporting on these parameters as part of the resource usage analysis is essential.

Practical groups for this module consist of three members. Each member will be required to pick one function type from the bullet list above and implement that individually. Group members may not pick the same function, i.e. each member's work is unique. The developed FPGA based systems and the Python based experimental setup will be considered a success if:

1. The FPGA based system can generate two versions of mathematical output, one serving as reference or benchmark model, and a second to demonstrate the effect of the implemented research model designed to answer the research question.
2. VHDL and Python setups exist for formal verification of both the implemented FPGA system and the experimental setup to ensure correct functionality and validated data for consideration during the research project.
3. The Python based analysis toolset can perform a comprehensive analysis of the generated experimental data using standardised and industry accepted methods.
4. All final work must be developed and implemented from first principles. Higher level functions are allowed, but only to serve as reference for comparison during verification phases of the model implementation(s) and experimental implementation(s).

Using the information generated by the experimental setup, students should be able to draw conclusions based on the usefulness and success of their function calculator for the specific application for which the student opts to design.

2. REQUIREMENTS OF THE RESEARCH PROJECT IMPLEMENTATION

Due to the investigative nature of this module and the assigned project, the implementation choices for the system described above in the introductory section are not fixed. Some aspects of the problem definition and description remain open ended to encourage students to individualise the projects to their own interest and allow students to demonstrate their ability to make assumptions and limit or restrict the research topic to a well demarcated problem. This means that students should choose the general context within which their implementation can be useful to solve a real world problem; be it signal processing, telecommunications, artificial intelligence, etc. Anything goes, as long as an FPGA solution is justifiable.

Students can further personalize their choice by considering various aspects and the level at which they will aim their 'contribution'. Students can look at overall architectures and investigate differences (say parallel vs pipelined structures), intermediate function alternatives/ improvements (say their function has a square root function, that subcomponent can be considered for improvement), or lower level alternative implementations, say different multiplier configurations as

part of their high level function. Keep in mind that the 'contribution' should be unique within your practical group, as with the function choice.

However, some limitations exist that must be adhered to in your project design. The research assignment is restricted to solutions suited to VHDL implementation and FPGA synthesis. Methods suited to x86, ARM, DSP, etc. type processors are outside the scope of this work. The chosen method should highlight at least some aspect of implementation optimisation and improvement over some base or reference model. Students are offered freedom in their design choices in terms of exactly what this optimisation method is, as well as their choice of implementing a fixed or a floating point number system. Furthermore, all implementations should be from first principles, i.e. demonstrating a number generator using IP libraries of FPGA vendors, freely available open source packages, etc. are not allowed.

Each practical group should set a date and time for their members to meet and formally discuss the research project in general. During the meeting members should decide on the allocation of the function type to each individual student, ensuring that each group member will be working on a unique research project. These choices will be communicated to the teaching staff in Assignment 1. It is highly recommended that each student complete an initial/overview literature survey to familiarise themselves with the given options. Once the research projects have been decided, each student is required to continue their own individual literature survey into their specific function choice, as well as architectures and implementations for their choice, optimisation methods, etc.. Consider implementations that are suitable for different architectural choices (parallelism, pipelining, etc.). Do not be distressed; your literature review will reveal a number of interesting approaches.

Also note that students that wish to *propose a new / novel approach are encouraged to do so*. This is exactly the environment that supports and nourishes the exploration of the unknown and to assess the quality of the idea compared to standardised benchmarks. Note that it is compulsory to discuss novel approaches with the lecturer first, before commencing with the assignment. The lecturer will gauge whether the proposal falls in line with the requirements of the module, and very importantly, whether the ambition and scope of the proposal is in check, thus ensuring that the student does not get overloaded during the semester.

3. REQUIREMENTS OF THE STUDENT

A very detailed document describes the research report framework adopted for the module. Refer to the framework document for further details on what is expected in the report. The module evaluation takes place in assignment and practical reports. In summary, the student should complete the research phases/milestones and goals in the respective assignments and reports:

1. *Assignment 1:* Conduct a literature investigation into the theoretical and technical aspects of mathematical function generators using fundamental digital hardware blocks only, and their approaches, techniques and performance rubrics (benchmarks). The literature survey should include at least
    a. an introduction with a problem statement, well demarcated from general problem definition down to a specific knowledge area of function implementation and optimisation.

b. a detailed section with sufficient references discussing well designed standard approaches for function implementation(s) that are proven to be good for the particular implementation.
c. a detailed section covering the investigation and supported by sufficient references (at least 5, but preferably more) focussing on optimisation and efficient architectures of your particular function specifically tailored towards FPGA implementation. It should be very clear which one of these approaches you will implement.
d. a detailed section with at least three references pointing the reader to good experimental procedures aimed towards characterising the properties of the function data. The focus here should be on standardised methods.

2. *Assignment 2:* Write a research proposal detailing the research plan. Following on the literature you reviewed in the first assignment, the proposal should include at least
    a. a section with the research statement(s), even if just one suitable/acceptable research question of hypothesis.
    b. a detailed section on your implementation methodology, specifically your approach to modelling the research problem by i) discussing the base/reference model, ii) highlighting your particular research model with optimisation and FPGA limitations considered, iii) a discussion on variables of the model, and iv) a discussion on the limitations imposed on the project.
    c. a detailed section on your experimental methodology, specifically your envisioned/proposed approach to experiments, data collection and data analysis.
    d. a projected timeline in the form of a Gantt diagram for the research project and critical milestones, considering the module calendar.
3. *Practical 1:* Design a generic research platform that can communicate between your FPGA development board and a PC, that will enable you to execute your function on FPGA, and communicate the generated data to the PC for analysis in Python. Students should ensure that the platform is generic, i.e. it is designed to such an extent that they can easily 'plug in' their own, unique function implementation. This will require demonstration of a well thought out structural model and structural coding approach. The practical will be considered a success if each student can demonstrate a different function being easily swappable, synthesised on the FPGA and communicated to Python. This function *is not* the selected research function. *Any simple arithmetic computation* can be shown (multiply, addition, etc.), just to demonstrate that the platform is indeed functioning as intended.
4. *Practical 2:* Design and implement in simulation a fully working VHDL based function calculation prototype system consisting of two function calculators, i) a base/reference implementation, as well as ii) your specific approach described by your research model. A 16 bit implementation should be considered. You need to verify your system functionality by inspecting the waveforms, but more importantly, by comparing your VHDL simulation results with an equivalent function implementation (algorithmic level is sufficient) in another programming language (Python) that will allow for a formal verification between the implementations. The formal verification typically considers many thousands of data points. Algorithmic design is much simpler in a high level language than in an HDL. Your typical design flow will therefore be to start at the highest level, i.e. an already implemented Python version, which you then redesign using low level digital logic functions (still in Python) for easy comparison which, when successfully verified, is translated into VHDL, and again verified by importing the data into Python. This practical is intended to answer the question

"Does my implementation work as intended, i.e. does it give the correct output?'. The practical is *not intended* to analyse the results, nor the FGPA synthesis to conduct a hardware resource analysis, i.e. *do not* answer the question 'How well does my system perform it's functionality?' in this practical. As an example, a typical output (let's say the topic was to implement an addition system) here would be to verify that a + b = c with a = 3 and b = 4 gives as an output c = 7. That demonstrates functionality. It is not expected here to investigate number accuracy, precision, FPGA resource utilization, etc. That happens next...

5. *Practical 3:* Design and conduct experiments to evaluate the effectiveness, efficiency and (statistical) accuracy and precision of the function calculation system given the performance metrics you identified from literature in assignments 1 and 2. This includes the design and implementation of a fully working Python based analysis tool compatible with your generated data, using industry standards and accepted methods. The student should focus on the implementation of a standardised method to measure properties and performance, and implement the tests from first principles. There should be a clear methodology to the design and verification of the experimental setup. The intention here is that data from both the base/reference/benchmark system and the research implementation *can be analysed*. This will be a necessary condition to proceed with the examination assignment. Everything described in point 5. above that you should avoid, must be included here. This practical assignment is to ensure that you can, in a quantifiable manner, gauge how well / efficient both your FPGA implementations work, i.e. here you will also be required to synthesize the VHDL of Practical 2 for the FPGA development board, and retrieve the data from FPGA hardware by communicating it with the generic FPGA / PC platform developed in Practical 1.
6. *Examination assignment:* Analyse, interpret and derive information from data, specifically experimentally obtained data, from which conclusions can be developed regarding the parameters of the investigation. Be sure to compare your results of both systems to one another, and to that of at least one reference author, be that with the same or different implementation.

4. ADMINISTRATIVE REQUIREMENTS FOR SUBMISSIONS

Submissions should meet a number of requirements in order to be considered valid submissions for the module, as is the case with any submission at the University of Pretoria that you wish to be evaluated. Before continuing, take note of the following University policy:

The University of Pretoria commits itself to produce academic work of integrity. I affirm that I am aware of and have read the Rules and Policies of the University, more specifically the Disciplinary Procedure and the Tests and Examinations Rules, which prohibit any unethical, dishonest or improper conduct during tests, assignments, examinations and/or any other forms of assessment. I am aware that no student or any other person may assist or attempt to assist another student, or obtain help, or attempt to obtain help from another student or any other person during tests, assessments, assignments, examinations and/or any other forms of assessment.

To ensure students meet the necessary requirements, they should follow the steps below when submitting their reports (assignments, practicals, etc.).

4.1 PHASE 1: CHECKLIST

1. The following documents should be prepared for submission:
    a. A copy of your report with suitable front matter (A signed plagiarism declaration using the student's own signature, Cover page, Index, optionally List of abbreviations, etc) and the content you wish to be evaluated on, i.e. a *complete/full submission document*.
    b. A copy of your report without any front matter content, and only the body of work you wish to have evaluated. This means that no generic forms, templates or text (like plagiarism statements used by everyone) should be included. Such forms will count against your submissions originality score. This is your *TurnItIn submission document*.
    c. A *TurnItIn (TII) receipt*, required only for assignments and reports that are submitted via TII and checked for plagiarism. Note that this is emailed to you if you successfully complete Phase 2 below.

4.2 PHASE 2: TurnItIn (TII) SUBMISSION

2. Familiarise yourself with the University guideline for students that make TII submissions. The document (Turnitin factsheet for students) can be found at the Library Plagiarism page: http://www.library.up.ac.za/plagiarism/index.htm
3. Once again, take special note of especially point 2 on page 1, and the point on "front matter" on the final page.
4. Follow the steps in the document to make your *TII submission of document 1 b)* above.
5. Be sure to check that you received a receipt confirming your submission.

4.3 PHASE 3: Judicator AMS SUBMISSION

You will notice at this stage that your TII submission does not meet the University requirements of including a signed plagiarism statement. That is easily remedied.

6. Log into the EPS system and follow the 'Submissions' link available in the top navigation bar.
7. Click on the applicable assessment ID, say as example 2020EES424A01.
8. Upload your complete submission with the signed plagiarism statement (document 1 a) above and confirm the originality of your work by ticking the box indicating your declaration of originality.
9. Upload your TII receipt from step 5.

5. READING MATERIAL TO GET YOU STARTED

Note that the intention here is not to give students a comprehensive list of reading material, but rather just a point of departure. Due to the large scope of function implementations, and also the individualization options offered to students, it would be impossible to find a common reference. The references here deviate somewhat from the research project by focusing on function option(s) not listed for students, yet they demonstrate how one would go about finding work related to a student's function choice.

[1] Sutikno, Tole; Jidin, Aiman Zakwan; Jidin, Auzani; Idris, Nik Rumzi Nik, *'Simplified VHDL Coding of Modified Non-Restoring Square Root Calculator*', International Journal of Reconfigurable and Embedded Systems; Yogyakarta Vol. 1, Iss. 1, (Mar 2012): 37. DOI:10.11591/ijres.v1i1.440

[2] Jaafar Alghazo, '*Modeling and Realization of the Floating Point Inverse Square Root, Square Root, and Division unit (fP ISD) Using VHDL and FPGAs*', Proceedings of the 2006 International Conference on Computer Design & Conference on Computing in Nanotechnology, CDES 2006, Las Vegas, Nevada, USA, June 26-29, 2006