High radix On-line Arithmetic Verification System

Final Year Project 1800478: Interim Report

Zifan Wang, 01077639
Imperial College London

CONTENTS

I Introduction 2

II Background Research 2
- II-A On-line Arithmetic . . . 2
- II-B High radix Arithmetic . . . 2
- II-C High radix On-line Arithmetic . . . 3

III Project Specification 3
- III-A Project Organisation . . . 3
- III-B Project Interfacing . . . 3
- III-C Deliverables . . . 3
- III-D Hardware Choice . . . 3
- III-E Software Choice . . . 3

IV Engineering Background 4
- IV-A Target . . . 4
- IV-B Data Transfer Rate . . . 4
- IV-C Clock Domains . . . 4
- IV-D Testbench Architecture . . . 4

V Implementation Plan 4
- V-A Milestones . . . 4
- V-B Work to Date . . . 4

VI Evaluation Plan 4
- VI-A Metrics . . . 4
- VI-B Risks . . . 4

VII Ethical, Legal, and Safety Plan 5
- VII-A Ethical Considerations . . . 5
- VII-B Legal Considerations . . . 5
- VII-C Safety Considerations . . . 5

VIII Conclusion 5

Appendix A: Formulae 5

Appendix B: Data 5

References 5

I. Introduction

With the right number representation system, it is possible to perform arithmetic operations MSD-first. Consequently, these on-line arithmetic operators are attractive for hardware implementation in both serial and parallel forms. When computing digits serially, they can be chained such that subsequent operations begin before the preceding ones complete. Parallel implementations tend to be most sensitive to failure in their LSD, making them more friendly to overclocking than their LSD-first counterparts, for which the opposite is true.

In the past, online operators have typically been implemented in binary. Along with its sister project, This project will explore high radix on-line operators, investigating their suitability for FPGA implementation and examining the resultant trade-offs between performance, area and power.

II. Background Research

A. On-line Arithmetic

Traditional proposals to achieving a faster and more efficient arithmetic operator have two common characteristics. One, their order operation may be different depending on the operation itself. A traditional adder, parallel or serial, generates its answers from the LSD to the MSD. A traditional divider design on the other hand, generates its answers from the MSD to the LSD [6] [7].

Due to this inconsistency, arithmetic operators may be forced to compute word-by-word, waiting for all digits to finish in the previous operator before moving on to the next. [8] This means if a divider follows an adder of the same width, the divider has to wait until the adder complete its computation before it can start its own.

The other commonality of traditional designs is that their precisions are specified design-time. Once built, a 32-bit adder always adds 32 bits together, as the hardware is fixed at run-time. A possible way of making it more efficient would be using SIMD instructions [9], trying to combine smaller operations into a larger one that fits the hardware. This requires more control circuits in the hardware, or a more complex compiler.

On-line arithmetic does not suffer from the first issue as it performs all arithmetic operations with MSD first [10]. Pipelining can be used with on-line arithmetic operators. This means the output digit of an earlier operation can be fed into the next operator before the earlier operation been fully complete.

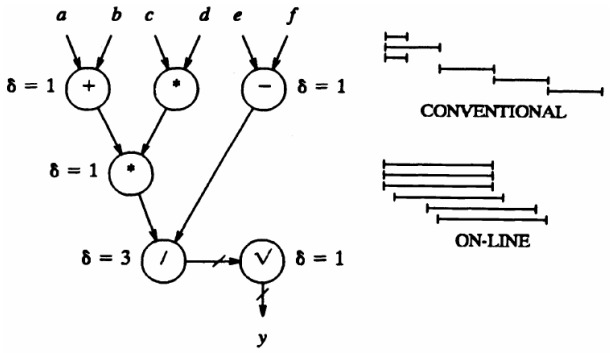


Fig. 1. Computing $y = \sqrt{(a+b)cd/(e-f)}$ [10]

As illustrated in figure 1, while each individual operation may take longer than its conventional counterpart, on-line arithmetic could provide a speed up if the operators are in serial. Individually, on-line arithmetic also sacrifices in terms of memory. As all computation are made MSD to LSD, the use of a redundant number system is compulsory. However, this redundancy also has its advantage in making the operators scalable. The time required per digit can be made independent of the length of the operands. [11]

A recent architecture proposal allows the precision of on-line arithmetic to be controlled at run-time [8]. Traditionally, this run-time control was restricted due to the parallel adders present in the multipliers and dividers. This architecture reuses a fixed-precision adder and stores residues in on-chip RAM. As such, a single piece of hardware can be used to calculate to any precision limited only by the size of the on-chip RAM.

Another way that on-line arithmetic alleviates the problem of fixed precision falls out directly from its MSD-first nature. Suppose the output of a conventional ripple adder is sampled before it has completed its operation. In this case, the lower digits would have been completed, but the carry would not have reached the higher ones. This means the error on the result would be significant, as the top bits were still undetermined [12].

However, if the output of a serial on-line adder is sampled before its completion, the lower bits would be the undetermined ones. This means the error of the operation would be small. With overclocking, on-line arithmetic would fail gracefully, losing its precision gradually from the lowest bits first. Thus, it allows for a run-time trade-off between precision and frequency [13].

B. High radix Arithmetic

Conventional designs of arithmetic operators use binary representations. This was chosen four decades ago to maximise numerical accuracy per bit of data. However, using a high radix representation system could yield better numerical accuracy while reducing area cost of FPGAs. For example, a hexadecimal floating-point adder has a 30% smaller area-time product than its binary counterpart, while still delivering equal worst-case and better average-case numerical accuracy [15].

However, the savings are not without trade-offs. This trade-off can become unfavourable if the specification requires much I/O and little computation [14]. This is because the overhead

of radix conversion would be significant. It is also unwise to use high radix representations when the numbers are unusually small, thus making the savings offered by the high radix negligible [15].

C. High radix On-line Arithmetic

Using high radix number representations for on-line arithmetic is a relatively novel concept. While there have been some research in this field [16] [17], this project takes a more direct approach by implementing custom operators made for high radix on-line arithmetic on a FPGA. This would allow for empirical results to be obtained, hopefully revealing practical insights to the method.

As an potential extension to the project, optimising this exotic arithmetic on a system level for popular FPGA accelerations such as neural networks would be innovative as well.

III. Project Specification

A. Project Organisation

This project is a part of a larger project investigating the effect of using high radix number systems with on-line arithmetic operators. The overarching aim would involve implementing such a system on FPGA and quantifying its performance improvements. This aim would be achieved through two vertically split individual projects. One would design and verify the arithmetic operator modules, while the other would design a system from the top-level to test and evaluate these operators. This project deals with the system-level issues.

As this project progresses in parallel with the designing of the operator modules, it is necessary to decouple the two project so that as individual projects, they are individually evaluated. The success of one project should not be restricted by the status of the other. To this end, the goal of the system-level design is more focussed on its functionalities and robustness. This relationship and its effect on the evaluation will be examined further in the evaluation chapter of this report.

B. Project Interfacing

A number representation system should be decided before detailed design starts.

The modules would be compiled hardware in

C. Deliverables

At the end of the project, the system should be able to perform the following:

1) Takes in the arithmetic modules designed by the sister project as its input;
2) Generate and run tests on these modules;
3) Vary the frequency and voltage of the FPGA;
4) Evaluate its performance.

D. Hardware Choice

The system itself will be built on a Cyclone V SX SoC Development Board from Intel [1].

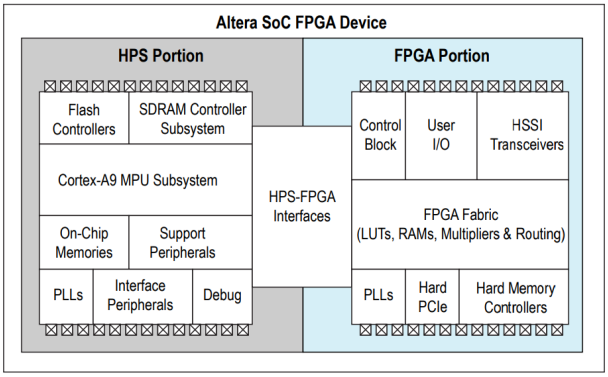


Fig. 2. Structure of the System-on-Chip

The 5CSXFC6D6F31C6N SoC has a Am Cortex-A9 MP-Core accompanied by Intel's 28nm FPGA fabric [2]. The FPGA is necessary for implementing the hardware design and obtaining empirical results for the project. Having an HPS on the same chip is useful as the test software can run on it, allowing a better user interface to be constructed with more detailed, on-the-fly control to the FPGA.

It should be noted that Xilinx offers similar boards as well. Its Zynq SoC family has a very comparable structure as they too integrate the software programmability of an Arm processor with the hardware possibility of an FPGA. For example, like the Cyclone V SX, Zynq-7000S also features a Arm Cortex-A9 coupled with a Xilinx 28nm FPGA [5].

As there are very few significant functional differences between the two brands, this project will initially explore with the Intel board, simply for its availability and the personal familiarity with their development tools.

Once the project has progressed to a point where the system design is mature and tested, the Xilinx alternative could be explored as an extension.

E. Software Choice

The software choice follows closely with the hardware choice in this project. To develop for Intel FPGA, Quartus needs to be used. The version picked is arbitrary as there is not much functional difference between the versions that would be critical to the project. As Quartus Prime 16.0 is the version installed in the computers in the department. The project will be using the same version for the convenience. This naturally means the hardware system will be build with the system integration tool that comes with Quartus – Qsys.

Other than the hardware design tools, there are some freedom of choice in the HPS side of the project. The test will be build with Python, which will be running on an Ubuntu system that is installed in the HPS. This choice is made for there are previous unrelated projects on the same develop board that uses the same configuration, which means a lot of time could be saved on getting an operating system booting.

IV. Engineering Background

A. Target

The design of the verification system is the major engineering challenge of this project. In order to stress the DUT, the verification system must perform at a much higher frequency than the expected frequency of the DUT. Assuming the DUT is to run at 300MHz, to fully explore the effect of overclocking, the testbench must be able to run at double the frequency or higher. This gives a target frequency of 800MHz. Assuming data width of 32-bits, the target data transfer rate is then This required data transfer rate is estimated to be 25.6Gbps.

B. Data Transfer Rate

As the test is to be run on the HPS, the HPS-FPGA bridge will be the immediate bottleneck if the test data is to flow from HPS to FPGA. While HPS is able to easily generate test data, there is a large amount of overhead as data crosses from one architecture to another. This overhead exists in terms of both a lowered bandwidth and a high delay. Thus, it would not be sensible for HPS to send out data during run-time.

Another thought may be to first populate the off-chip DDR SDRAM on FPGA side, then feed the data from there to the DUT during test. This is already much faster than passing the data from HPS. The 1GB, 32-bits wide DDR3 on FPGA side is rated at 400MHz. With double rate transfer, this would gives a maximum transfer rate of 25.6Gbps.

While using the off-chip RAM may theoretically achieve the targets, it still has its disadvantages. First, the process of filling up the memory and then using them for the tests takes time. This means the test would be broken up into bursts with time in between for checking the results and filling up with new data. The complexity of the SDRAM interface also requires a SDRAM controller to be used to manage SDRAM refresh cycles, address multiplexing and interface timing. These all adds up to a significant access latency. While this could be overcame with burst accesses and piplined accesses, it would further complicate the SDRAM controller. While this controller is provided by Altera [4], it is consumes a non-negligible amount of the limited FPGA resources, while adding unnecessary complexity to the design. Customising the controller to fit this project may also be time-consuming.

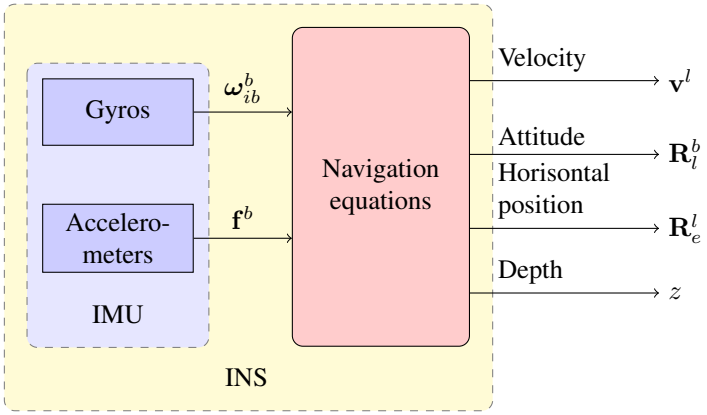
The on-chip memory is much faster and simpler to handle. This memory is implemented on the FPGA itself, and thus there is no external connections for access to this memory. It has the highest possible throughput, with the lowest possible latency in an FPGA-based system. The memory transactions can also be piplined, giving one transaction per clock cycle. With an on-chip FIFO accessed in dual-port mode, the write at one end and the reads at the other end can happen simultaneously. This effective doubling of the bandwidth is useful as tests are prepared and fed into the DUT, or when test results are collected and fed to a checker.

On-chip memory is not without its drawbacks. It is volatile and very limited in capacity. While the off-chip can have its storage reaching 1GB, that of the on-chip memory could only reach a few MB [3]. Volatility is not exactly of concern in this project, but its small capacity means not much test data can be held before it needs more fed in.

Looking at the options listed above, with a way of generating test data at run-time on the FPGA, using on-chip memory would be the most sensible option.

C. Clock Domains

D. Testbench Architecture



Chapter 4 of Shi's paper [12] provides a similar structure for this verification system.

V. Implementation Plan

A. Milestones

The initial deliverable for the engineering side involves running a simple program on the FPGA through the HPS with the FPGA frequency being controllable. After this, the next critical step would be making sure the modules under test will be the point of failure and not the testbench. This would include some research on ways in improving the speed of feeding inputs to the arithmetic units, and checking its outputs. Once this could be confirmed, we can start adding a selection of different functionalities.

1) Running standard benchmarks;
2) Running key algorithms or their components;
3) Experiment with other power efficiency improving techniques, such as undervolting;
4) Add support for configurable radix arithmetic;
5) Allow graceful failures for the testbench in case of unintended behaviour for the arithmetic modules;
6) Add an interactive UI to control the voltage and frequency at run time and examine the DUT's behaviour;

Depending on the time situation, more or less items on this list may be fulfilled. The method of evaluation will be discussed later in the evaluation chapter.

B. Work to Date

VI. Evaluation Plan

A. Metrics

B. Risks

A major risk of this project is related to its schedule and the existence of an initial blocking task. While most of

the later sections of the project can be selectively added or removed from the scope relatively easily, the initial setup of the testbench will always remain critical to any further improvements. It is thus vital that the bare minimum system gets done early. To ensure this happens, it will be placed in the highest priority before its completion, and any blocking issue should be discussed with the supervisor if it could not be resolved after significant effort.

The other major risk has to do with the progress of the sister project. The purpose of the testbench is to verify and stress the arithmetic designs. If these designs would not be available near the end of this project, it would be difficult to empirically prove the capabilities of the testbench and its surrounding system. It is not impossible, as there are still substitutions for them. For functional purposes, standard off-the-shelf adders and multipliers could be used in-lieu. For other purposes, it is possible to have a model done before the actual design starts in the paired project. While this would allow this project to progress easier, it would be extra work for the other project, which is ultimately up to the decision of the other student. In all, it would be nice to have a solid arithmetic module completely to run in this testbench, but without one, the system can still be built and completed, albeit generating less useful data towards the overall aim of the project.

VII. Ethical, Legal, and Safety Plan

A. Ethical Considerations

Checking against the ethical issue list provided by Imperial College Research Ethics Committee [18], this project

- does not damage participants' mental or physical health;
- does not jeopardise the safety and liberty of the researchers;
- does not use any private information;
- does not involve sensitive subject matter or methods;
- does not risk any conflict of interest between the researchers and the College.

This project is thus free from significant ethical concerns.

B. Legal Considerations

Intel Quartus Prime software offers a variety of IP cores. These are encrypted module designs that would be integrated into the verification system of the project [19]. Intel FPGA Evaluation Mode allows

C. Safety Considerations

As the project is done mainly on a computer with minimum physical aspects, there is no major safety concern. For the minor concerns associated with the project, the physical development board will be handled with care, and the desk works will be interleaved with breaks.

VIII. Conclusion

Appendix A
Formulae

$$1 + 1 = 2$$

Appendix B
Data

Sum	1	2	3	4
1	2	3	4	5
2	3	4	5	6

References

[1] Intel Corporation, "*Cyclone V SoC Development Kit and Intel SoC FPGA Embedded Development Suite*".

[2] Altera Corporation, "*Cyclone V SoC Development Board Reference Manual*", 2015.

[3] Altera Corporation, "*Memory System Design*", *Embedded Design Handbook*, 2010.

[4] Altera Corporation, "*Introduction to Altmemphy IP*," *External Memory Interface Handbook: Reference Material*, vol. 3, 2012.

[5] Xilinx, Inc, "*Zynq-7000 All Programmable SoC*", 2018.

[6] R.P. Brent, "*A Regular Layout for Parallel Adders*", *IEEE Trans. Comput.*, vol. C-31, pp. 260-264, 1982.

[7] H.R. Srinivas, K.K. Parhi, "*High-Speed VLSI Arithmetic Processor Architectures Using Hybrid Number Representation*", *J. of VLSI Sign. Process.*, vol. 4. pp. 177-198, 1992.

[8] Y. Zhao, J. Wickerson, G.A. Constantinides, "*An Efficient Implementation of Online Arithmetic*", *Int. Conf. on Field-Programmable Technology*, 2016.

[9] R. Duncan, "*A Survey of Parallel Computer Architectures*", *Computer*, vol. 23, pp. 5-16, 1990.

[10] M.D. Ercegovac, "*On-line Arithmetic: An Overview*", *28th Annual Technical Symposium*, pp. 86-93, Internaltional Society for Optics and Photonics, 1984.

[11] K.S. Trivedi, M.D. Ercegovac, "*On-line Algorithms for Division and Multiplication*", *IEEE Trans. Comput.*, vol. C-26, no. 7, pp. 667-680, 1977.

[12] K. Shi, D. Boland, G.A. Constantinides, "*Accuracy-Performance Trade-offs on an FPGA through Overclocking*", *Proc. Int. Symp. Field-Programmable Custom Computing Machines*, pp. 29-36, 2013.

[13] K. Shi, D. Boland, E. Stott, S. Bayliss, G.A. Constantinides, "*Datapath Synthesis for Overclocking: Online Arithmetic for Latency-Accuracy Trade-offs*", *Proceedings of the 13th Symposium on Field-Programmable Custom Computing Machines*, pp. 1-6, ACM, 2014.

[14] P. Whyte, "*Design and Implementation of High-radix Arithmetic Systems Based on the SDNR/RNS Data Representation*" *Edith Cowan University*, 1997.

[15] B. Catanzaro, B. Nelson, "*Higher Radix Floating-Point Representations for FPGA-Based Arithmetic*", *Proceedings of the 51st Annual Design Automation Conference*, 2005.

[16] T. Lynch, M.J. Schulte, "*A High Radix On-line Arithmetic for Credible and Accurate Computing*", *Journal of Universal Computer Science*, vol. 1, no. 7, pp. 439-453, 1995.

[17] T. Lynch, M.J. Schulte, "*Software for High Radix On-line Arithmetic*", *Reliable Computing*, vol. 2, no. 2, pp. 133-138, 1996.

[18] Imperial College "*An Ethics Code*", *Imperial College Research Ethics Committee*, 2013.

[19] Intel Corporation, "*Introduction to Intel® FPGA IP Cores*", 2018.