Non-Newtonian Accelerators: Low-Compromise Design for Fault Tolerant Accelerators

Abstract

To achieve longevity in the face of increasingly fault-prone datapaths, fault tolerance is needed, especially in accelerator datapaths. We propose a novel architecture for accelerator fault tolerance, Non-Newtonian Accelerators, which leverages modular acceleration to enable fault tolerance without burdensome area requirements.

In order to streamline the development and enforce modular conventions, we introduce the Viscosity language, an actor based approach to hardware-software co-design.

To show the feasibility of Non-Newtonian Accelerators, we show three case-studies, FFT, AES, and DCT accelerators each demonstrating how Non-Newtonian Accelerators perform under faults.

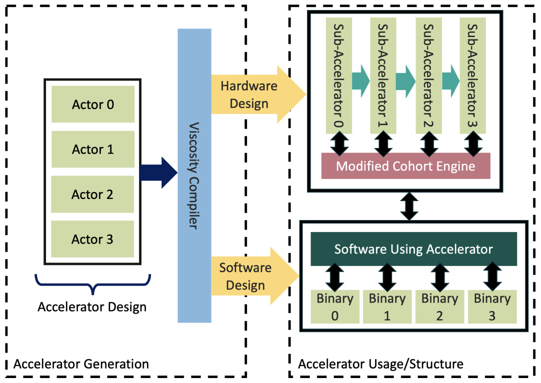

Figure 1. The structure of the proposed accelerator generation and structure

1 Introduction

As Moore's Law comes to an end [16], the steady performance improvements which came from transistor miniaturisation will no longer be as readily accessible.

Additionally, longer hardware refresh cycles mean that the hardware itself will have to stay functional longer. However, in data center contexts, various device errors can occur, cutting the life of the processors short [3, 8, 13, 20].

These two trends are seemingly in competition. The required life-cycle of hardware is increasing but the practical lifetime of hardware is staying the same or decreasing [7]. We choose to focus on hardware accelerators, which have been the key to improvements in performance for various domain specific applications [1, 6, 14]. Moreover, they have become ubiquitous in modern data center processors [10, 16].

Without fault tolerance, when a fault does occur inside an accelerator, in the worst case, an accelerator can be replaced by a fully software implementation of the same algorithm. In contrast to this method, we propose **Non-Newtonian Accelerators**, an architectural design principle leveraging modular design to increase the fault tolerance of accelerators.

2 Background

2.1 Motivation

To assess the effect of fault tolerance on data centers, we model data centers employing two types of accelerators. The first are accelerators which are not fault tolerant, which we call "single fault accelerators" (SFA). These need to be replaced after a single fault has been detected. The second type of accelerators we call "variable fault accelerators" (VFA) which is a generalisation of the Non-Newtonian Accelerator architecture we propose. These can handle multiple faults up to some threshold. Once this threshold has been reached, the accelerator is unusable and has to be replaced.

We found that as the likelihood of faults approaches zero, the difference between the two types of accelerators increases. By using VFAs, data centers could reduce the number of processors replaced to less than one on average, even at fault likelihoods where using SFAs would result in over 50 replacements.

3 Non-Newtonian Accelerators

For an accelerator implementing a function f, we model a single interface for the input and a single interface for the output of the accelerator. Suppose that a non-transient fault occurs in some arbitrary location inside the accelerator. This means that for some input into the accelerator, the accelerator will produce the incorrect output. This is despite the fact that the fault itself may make up an extremely small portion of the total accelerator design. Ideally, there should be a way to utilise the unbroken logic within the accelerator, processing the broken step using some other method of computation.

To build a Non-Newtonian Accelerator using Viscosity (or another HDL), create sub-accelerators capturing functions $f_1, f_2, \ldots, f_n$ such that $f_n \circ \ldots \circ f_2 \circ f_1 \equiv f$. Each sub-accelerator has two sets of interfaces, one set of interfaces with the software thread and one set of interfaces with the previous and following sub-accelerators.

Under no faults, the accelerator acts as one cohesive accelerator (such that there is no latency as opposed to a normal accelerator). However, after a fault, the system can adapt to avoid it. There is no singular method of fault recognition that must be used; any method which can send the signal to the sub-accelerators can be used with Non-Newtonian Accelerators. Suppose you have a three stage accelerator and the second stage breaks. The accelerator adapts to bypass the faulty logic. After the first stage, the output of f_1 is moved to the software thread via the software interface. The software then processes the data using an executable version of f_2 before moving the data onto f_3 where the rest of the accelerator can run normally using the interface between sub-accelerators.

Our implementation of Non-Newtonian Accelerators runs on a modified version of the Cohort Engine [21]. Cohort eases the burden of adding new accelerators while keeping system-level guarantees by providing FIFO queue endpoints for communication between software threads and accelerators built on top of cache-coherent memory queues. Whereas the Cohort Engine supports a single queue per tile, our modified version supports multiple queues interfacing with multiple sub-accelerators. Additionally, we introduce queue-bypassing to enable sub-accelerators to communicate with each other directly. The Cohort consumers and producers interact with the software thread while the bypass interfaces communicate directly with the previous and following sub-accelerators.

4 Viscosity

The design of Non-Newtonian Accelerators requires both a software version and a hardware version of each sub-accelerator. There are three reasons we would want to generate both the hardware and software from a single description: Firstly, it makes Non-Newtonian Accelerators simpler to design since the operation need only be described once. Secondly, it ensures that the software and hardware versions of the operation are logically equivalent which is especially important with the complexity of subdividing hardware modules. Lastly, it lets the language enforce the sub-accelerator modularity convention.

A naive version of generating both a hardware and software description from a single description could be done by running hardware simulation of the modules. Given the high overhead of hardware simulators, they would not be suitable for the software descriptions needed for Non-Newtonian Accelerators. We opted to create our own language, Viscosity, which compiles to C and to Verilog (via Shakeflow [12]). Shakeflow is a Rust-based DSL which improves on previous functional HDLs by adding latency-insensitive interface combinators which are perfect for modelling the accelerators we target. Other HDLs have used the actor paradigm but do not support native software and hardware generation [18].

5 Evaluation Methodology

We implemented Non-Newtonian Accelerators with a modified Cohort Engine on the OpenPiton+Ariane RISC-V Research Platform [4, 21]. We then booted Linux (v6.2, built via Buildroot) on a Digilent Genesys2 FPGA (Kintex-7 XC7K32T5-2FFG900C) running at a clock speed of 67 MHz. We show the potential benefit of three different accelerators built as Non-Newtonian Accelerators: an FFT, AES and DCT accelerator.

6 Results

When there is no fault present, the **FFT** Accelerator performs with only 7.4% of the cycles when compared to the software implementation, a speedup of 13.5×. When there is a single fault present, the FFT Accelerator runs in approximately 19.3% of the cycles of its purely software counterpart, a speedup of 5.181×.

We designed an **AES** accelerator with two different configurations: an 11-stage AES accelerator and a 3-stage AES accelerator.

When a fault does occur in the hardware and the software fallback takes over, the efficiency of the system drops more than for the FFT or DCT accelerators with the AES accelerator taking 58% of software's execution time under a fault rather than 7% it would otherwise without a fault.

We also analysed the performance of computing a **2-D Discrete Cosine Transform** using a Non-Newtonian Accelerator. Compared to the other accelerators we evaluated, the DCT accelerator has the lowest software to hardware cycle ratio, which is due to the design already leveraging the fastest known DCT algorithm.

With one faulty stage, the DCT accelerator records a speedup of 2.87×, an encouraging performance as the software implementation is already heavily optimised.

7 Fault Detection

Our goal with this tolerance mechanism was that it would be compatible with nearly any fault detection mechanism that is available, software or hardware. There has previously been much work regarding how to detect the presence of non-transient faults in hardware [2, 5, 19].

Software can directly set the registers inside the accelerator datapath to bypass fault ridden components. If hardware based detection is used, then the detection mechanism can be hardwired to the bypass signals.

8 Related Work

Most work on fault tolerance in accelerators has come from exploring hardware generated from HLS tools [22]. Karri and Orailoglu proposed building fault tolerant ASICs using HLS [15, 17]. They reduced the area trade-off of N-Module redundancies by sharing functional units between modules. The work has been continuously iterated on by mixing time and space redundancies [9] and by introducing various methods to better explore the design space such as genetic algorithms [11, 22]. Other works have focused on only building redundancies for only the most critical paths [9].

9 Conclusion

This work presented the Non-Newtonian Accelerator methodology as well as the Viscosity language, a tool for building Non-Newtonian Accelerators, which helps enforce this modular paradigm. Our evaluations show how this accelerator structure can reduce slowdown under faults in the FFT, AES, and DCT accelerators.

References

[1] Tim Ansell. Google investment in open source custom hardware development including no-cost shuttle program. In *Proceedings of the 2023 International Symposium on Physical Design*, ISPD '23, page 207, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399784. doi: 10.1145/3569052.3580028. URL https://doi.org/10.1145/3569052.3580028.

[2] Anna Antola, Vincenzo Piuri, and Mariagiovanna Sami. High-level synthesis of data paths with concurrent error detection. In *13th International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT '98), 2-4 November 1998, Austin, TX, USA, Proceedings*, pages 292–300. IEEE Computer Society, 1998. doi: 10.1109/DFTVS.1998.732178. URL https://doi.org/10.1109/DFTVS.1998.732178.

[3] David F. Bacon. Detection and prevention of silent data corruption in an exabyte-scale database system. In *The 18th IEEE Workshop on Silicon Errors in Logic – System Effects*, 2022. URL http://research.google/pubs/pub51477/.

[4] Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen, Yanqi Zhou, Alexey Lavrov, Mohammad Shahrad, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff. OpenPiton: An open source manycore research framework. *SIGARCH Comput. Archit. News*, 44(2):217–232, mar 2016. ISSN 0163-5964. doi: 10.1145/2980024.2872414. URL https://doi.org/10.1145/2980024.2872414.

[5] M. A. Breuer. Hardware fault detection. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part II*, AFIPS '68 (Fall, part II), page 1502–1503, New York, NY, USA, 1968. Association for Computing Machinery. ISBN 9781450379007. doi: 10.1145/1476706.1476792. URL https://doi.org/10.1145/1476706.1476792.

[6] Jared Casper and Kunle Olukotun. Hardware acceleration of database operations. In *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '14, page 151–160, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326711. doi: 10.1145/2554688.2554787. URL https://doi.org/10.1145/2554688.2554787.

[7] Jacqueline Davis, Daniel Bizo, Andy Lawrence, Owen Rogers, and Max Smolaks. Uptime institute global data center survey 2022. https://uptimeinstitute.com/resources/research-and-reports/uptime-institute-global-data-center-survey-results-2022, 2022.

[8] Harish Dattatraya Dixit, Sneha Pendharkar, Matt Beadon, Chris Mason, Tejasvi Chakravarthy, Bharath Muthiah, and Sriram Sankar. Silent data corruptions at scale. *CoRR*, abs/2102.11245, 2021. URL https://arxiv.org/abs/2102.11245.

[9] Shane T. Fleming and David B. Thomas. Stitchup: Automatic control flow protection for high level synthesis circuits. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2016. doi: 10.1145/2897937.2898097.

[10] Adi Fuchs and David Wentzlaff. The accelerator wall: Limits of chip specialization. 02 2019. doi: 10.1109/HPCA.2019.00023.

[11] Michael Glass, Martin Lukasiewycz, Thilo Streichert, Christian Haubelt, and Jurgen Teich. Reliability-aware system synthesis. In *2007 Design, Automation and Test in Europe Conference and Exhibition*, pages 1–6, 2007. doi: 10.1109/DATE.2007.364626.

[12] Sungsoo Han, Minseong Jang, and Jeehoon Kang. Shakeflow: Functional hardware description with latency-insensitive interface combinators. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS 2023, page 702–717, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399166. doi: 10.1145/3575693.3575701. URL https://doi.org/10.1145/3575693.3575701.

[13] Peter H. Hochschild, Paul Turner, Jeffrey C. Mogul, Rama Govindaraju, Parthasarathy Ranganathan, David E. Culler, and Amin Vahdat. Cores that don't count. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '21, page 9–16, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384384. doi: 10.1145/3458336.3465297. URL https://doi.org/10.1145/3458336.3465297.

[14] Tom Hogervorst, Răzvan Nane, Giacomo Marchiori, Tong Dong Qiu, Markus Blatt, and Alf Birger Rustad. Hardware acceleration of high-performance computational flow dynamics using high-bandwidth memory-enabled field-programmable gate arrays. *ACM Trans. Reconfigurable Technol. Syst.*, 15(2), dec 2021. ISSN 1936-7406. doi: 10.1145/3476229. URL https://doi.org/10.1145/3476229.

[15] R. Karri and A. Orailoglu. High-level synthesis of fault-tolerant asics. In *1992 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 1, pages 419–422 vol.1, 1992. doi: 10.1109/ISCAS.1992.229924.

[16] Charles E. Leiserson, Neil C. Thompson, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, and Tao B. Schardl. There's plenty of room at the top: What will drive computer performance after moore's law? *Science*, 368(6495):eaam9744, 2020. doi: 10.1126/science.aam9744. URL https://www.science.org/doi/abs/10.1126/science.aam9744.

[17] A. Orailoglu and R. Karri. A design methodology for the high-level synthesis of fault-tolerant asics. In *Workshop on VLSI Signal Processing*, pages 417–426, 1992. doi: 10.1109/VLSISP.1992.641073.

[18] Haven Skinner, Rafael Trapani Possignolo, and Jose Renau. Liam: An actor based programming model for hdls. In *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*, MEMOCODE '17, page 185–188, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350938. doi: 10.1145/3127041.3127060. URL https://doi.org/10.1145/3127041.3127060.

[19] F.N. Taher. *Fault Tolerance in Hardware Accelerators: Detection and Mitigation*. University of Texas at Dallas, 2019. URL https://books.google.com/books?id=HnohzgEACAAJ.

[20] Shaobu Wang, Guangyan Zhang, Junyu Wei, Yang Wang, Jiesheng Wu, and Qingchao Luo. Understanding silent data corruptions in a large production cpu population. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 216–230, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613149. URL https://doi.org/10.1145/3600006.3613149.

[21] Tianrui Wei, Nazerke Turtayeva, Marcelo Orenes-Vera, Omkar Lonkar, and Jonathan Balkind. Cohort: Software-oriented acceleration for heterogeneous socs. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS 2023, page 105–117, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399180. doi: 10.1145/3582016.3582059. URL https://doi.org/10.1145/3582016.3582059.

[22] Zhiqi Zhu, Farah Naz Taher, and Benjamin Carrion Schafer. Exploring design trade-offs in fault-tolerant behavioral hardware accelerators. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, GLSVLSI '19, page 291–294, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362528. doi: 10.1145/3299874.3318020. URL https://doi.org/10.1145/3299874.3318020.