SiFT: Signature Based Fault Tolerance for High Level Synthesis

Nicholas V. Giamblanco
nicholas.giamblanco@mail.utoronto.ca

Lino Carrillo
lino.carrillo@mail.utoronto.ca

Ishita Ray
ishita.ray@mail.utoronto.ca

I. Introduction

With today's heavy reliance on electronic devices, quick and efficient performance is desirable. However, the amount of data available today has negatively impacted performance. Hence, it is worthwhile to investigate acceleration techniques where some proportion of an algorithm can be *offloaded* to hardware. A common technique is to utilize High-Level Synthesis (HLS). HLS tools can produce a hardware based implementation of an algorithm from a high-level description. One such tool is LegUp [1], where a high-level description written in C code is *translated* to Verilog (a hardware description language). However, there is concern with the input to this HLS tool. HLS tools imply that the high-level description to be translated was checked for correctness against implementation and logical bugs. There is limited prevention of faults due to physical disturbances upon the circuit. Additionally, there lacks a correctness-checking mechanism for the translated hardware description. Herein lies the motivation for this project; an automated fault tolerance tool, SiFT[1], which aims to (1) identify latent bugs within an ANSI C description and (2) insert fault tolerance through signature based correctness.

A. Signatures

Signatures, which are unique identification mechanisms, permit verification of the origin/integrity of some data [2]. However, signatures need not be limited to individuals, or computers. It has been demonstrated that signatures can be produced for logical descriptions and mathematical functions ([3], [4], [5]). Utilizing signatures as a method of correctness may entail the following: suppose that we have some function $f(x) = x^2$. By quickly inspecting this relationship, we can see that $f(x)$ shall always be positive. Hence this is a *signature* for this particular function. We will extrapolate on this ideology to produce signatures for high-level descriptions.

II. Problems & Challenges

Hardware faults differ from their software counterpart in the several ways: (1) hardware bugs are usually caused through physical disturbances [6] and (2) the hardware may have latent logic defects [7]. However, due to the usage of HLS, software is *transformed* into this hardware description. A significant challenge faced with our method is an efficient implementation of signature based fault tolerance. With any additional logic to be synthesized with a hardware description, considerations of latency, power, and area in the produced circuit should be investigated.

III. Expected Stages for Project Development

Our team has identified the following stages to complete this project:

- Review techniques that can identify correctness within High Level Languages (i.e [8])
- Researching different signature-based fault detection systems
- Identifying signatures of high level descriptions automatically
- Adding fault tolerance from *discover* signatures
- Researching and providing evaluation techniques for this novel fault tolerance method
- Perform a review of the LegUp system (i.e. understand LLVM's infrastructure [9])
- Implementation & Evaluation of SiFT with LegUp

[1] **Si**gnature based **F**ault **T**olerance

References

[1] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J. H. Anderson, S. Brown, and T. Czajkowski, "Legup: high-level synthesis for fpga-based processor/accelerator systems," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pp. 33–36, ACM, 2011.

[2] M. Atreya and B. Hammond, *Digital signatures*. Osborne/McGraw-Hill, 2002.

[3] J. E. Smith, "Measures of the effectiveness of fault signature analysis," *IEEE Transactions on Computers*, vol. 29, no. 6, pp. 510–514, 1980.

[4] F. Corsi, M. Chiarantoni, R. Lorusso, and C. Marzocca, "A fault signature approach to analog devices testing," in *Proceedings ETC 93 Third European Test Conference*, pp. 116–121, Apr 1993.

[5] N. Oh, P. P. Shirvani, and E. J. McCluskey, "Control-flow checking by software signatures," *IEEE transactions on Reliability*, vol. 51, no. 1, pp. 111–122, 2002.

[6] E. Normand, "Single event upset at ground level," *IEEE transactions on Nuclear Science*, vol. 43, no. 6, pp. 2742–2750, 1996.

[7] M. L. Li, P. Ramachandran, S. K. Sahoo, S. V. Adve, V. S. Adve, and Y. Zhou, "Trace-based microarchitecture-level diagnosis of permanent hardware faults," in *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pp. 22–31, June 2008.

[8] D. Engler, D. Y. Chen, S. Hallem, A. Chou, and B. Chelf, "Bugs as deviant behavior: A general approach to inferring errors in systems code," in *ACM SIGOPS Operating Systems Review*, vol. 35, pp. 57–72, ACM, 2001.

[9] C. Lattner and V. Adve, "Llvm: A compilation framework for lifelong program analysis & transformation," in *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*, p. 75, IEEE Computer Society, 2004.