A DYNAMICALLY TUNED FINITE DIFFERENCE METHOD FOR RECONFIGURABLE SYSTEMS

Xinyu Niu and Wayne Luk

Dept. of Computing, School of Engineering
Imperial College London, UK
Email: {nx210, wl}@doc.ic.ac.uk

ABSTRACT

This paper introduces a novel adaptive method applicable to all algorithms based on finite difference method. The algorithm is tuned adaptively and incrementally in terms of data presentation and computation structure. Computational accuracy is dynamically predicted and controlled, and hardware resource consumption is analysed to explore run-time potential of target applications. The design flow involves algorithm update, precision estimation, hardware analysis, runtime scheduling and dynamic learning.

1. INTRODUCTION

Run-time reconfigurability of reconfigurable systems is capable is tuning applications while running. By optimising hardware implementations according to specific requirements during run-time, the performance of reconfigurable systems can be pushed forward. Effective as the technique is, existing algorithms and applications tend to be static, and current runtime reconfiguration methods are limited to applications with varying properties [1, 2].

Bit-level optimisation works as an important design technique in the field of reconfigurable computing. By customising the data presentation to algorithm characteristics and user requirements, significant area saving can be achieved, which in turn improves system concurrency and thus increases system throughput. Existing tools to perform bit-level optimisation are limited to either static precision analyses [3, 4, 5] or Monte-Carlo methods [6]. The concept of run-time reconfiguration is missing from current tools.

In this paper, we aim to introduce an approach to dynamically tune algorithms based on finite difference method. Our approach actively generates runtime design space from target algorithms, and the error propagation is dynamically controlled.

The major contributions of this work include:

- An adaptive approach to control computational error due to reduced data presentation. Instead of passively estimated, the error propagation is actively controlled by tuning the parameters.
- A novel methodology to turn static algorithms into dynamic implementations. The runtime properties of algorithms and reconfigurability of systems are explored to adaptively improve system performance.

2. BACKGROUND

2.1. Precision Analysis

As a FPGA-exclusive optimisation technique, bit-width optimisation has been widely used in the field of custom computing. Various algorithm presentations and precision analysis methods were proposed to generate circuits with guaranteed accuracy [3, 4, 5]. These work depend on passive analyse of the target algorithms, the capacity of tuning data presentation during runtime is not explored. A runtime compensation method was proposed in [6]. However, this method is limited to Monte-Carlo methods, where error is bounded within one path and only the final result matters.

2.2. Runtime Reconfiguration

Runtime reconfiguration is an emerging area to improve system performance during runtime. Given runtime information is properly utilised, the implemented operators can be further optimised during specific time slots. The slowly varying properties of input data were captured in [1] to implement arithmetic operators with constant input. The algorithm parameters were dynamically approximated to optimal constants for the operators in [2], to further reduce the upper bound of implemented operators. Besides customising implementations, the tuning process also impacts the computational precision. However, the interactions have not been explored.

2.3. Finite Difference Method

Finite difference method is a widely used numerical method to approximate solutions to differential equations. The approximation error depends on the step size and the approximation order.

$$\frac{\partial u^2(a)}{\partial x^2} \approx \frac{\alpha \cdot u(a-x) + \beta \cdot u(a) + \gamma \cdot u(a+x)}{x^2} \quad (1)$$

This static approximation process can be dynamically accomplished, with coefficients actively varied. The potential benefits include reduced computational effort, as well as opportunity to dynamically tune the algorithm.

3. MOTIVATION

The run-time potential of application depends the varieties of the application in time dimension. Previous work are limited to applications [7, 8, 9, 10] with varying properties. The proposed approach exploit the run-time potential of applications by actively tuning application configurations. Our aim is to show that, with proper run-time design methods, applications with static properties can explore reconfigurability to improve system

The explored properties in current approach include data presentation and constant coefficients. The data presentation involves achieving optimal bit-width optimisation for arithmetic operations, while varied coefficients impact resource usage and error propagation. The interaction between the updated properties and system performance is formulated to incrementally tune the target applications.

In reconfigurable computing, data presentation refers to bit-width optimisation. The data are presented in customised formats to reduce resource consumption. As a consequence, generated results differ from the results of original presentation. If the original results are assumed as accurate, inaccuracy is introduced at the time when data presentation is varied, and propagates through the computational space, as shown in Figure 1. By dynamically introducing and compensating variable inaccuracy in different time slots, an optimal run-time data presentation can be achieved.

Besides data presentation, algorithm optimisation during run-time involves reconstructing the target algorithms according to dynamic requirements of applications. Figure 2 demonstrates the structure of a one-dimensional finite difference method. Propagation of generated results between time steps can be dynamically controlled, by varying the mapping constant. For algorithms based on finite difference method, coefficients are decided by approximation order O, stencil size S, step size in time dt and step size in space ds. The computational process for a one-dimensional finite difference method is shown in Figure 2.

$$(\alpha, \beta, \gamma ...) \Leftarrow f(O, S, dt, ds) \quad (2)$$

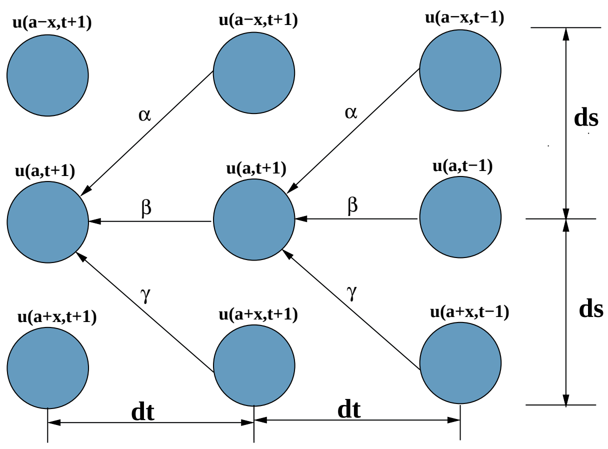

Fig. 2. Structure of one-dimensional finite difference method in time.

With dynamic data presentation and algorithm structures in different time steps, the target algorithms can be dynamically tuned.

4. ADAPTIVE APPROACH

The proposed approach is presented in Figure 3. It works as an iterative method to adapt the target algorithm into dynamically optimised operators. The algorithm is constantly tuned as computation goes through involved grid space. Tuned coefficients are fed into a precision estimator to predict accumulated errors. The precision model analyses accuracy of the specific data presentation and constant values, while the hardware model calculates resource usage of the configuration. Two analytical models cooperate to estimate the benefit of possible reconfiguration opportunities. Target accuracy and available hardware resource work as constraints for possible configurations.

The error for a specific point accumulates from neighbouring points in space and previous calculations in time. Affine Arithmetic [11] (AA) was proposed to estimate the computation range and precision. It can be used here to estimate the dynamic precision.

$$x_{re} = x + 2^{-MA-1} \cdot \lambda, \lambda \in [-1, 1] \quad (3)$$

$$E_i = 2^{-MA_i-1} \cdot \lambda_i \quad (4)$$

$$E_{j,t} = \sum_{i=0}^{n} \lambda_{<i,t-1>} \cdot E_{<i,t-1>} \quad (5)$$

where x_{re} denotes the data presented with reduced precision, and MA is the mantissa size. Computation error introduced in a specific time step is expressed as E_i, while the error propagation is formulated as Eq 5. E_i is estimated with data presentation, and the propagation is controlled with dynamic constants. Therefore, the impacts of varied algorithm

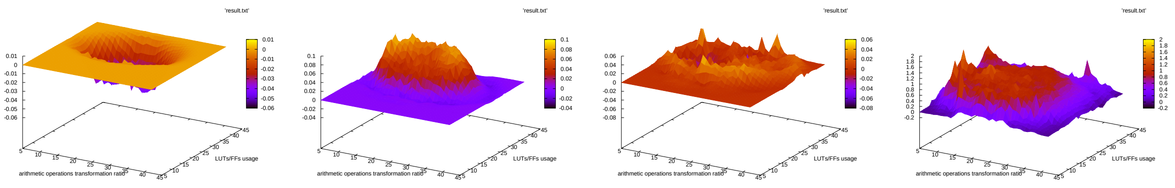

Fig. 1. Error propagation in computation space and time.

on computation accuracy can be adaptively predicted.

The hardware analytical model is built to capture the dynamic optimisation opportunities as design configurations are mapped onto reconfigurable fabrics. Bit-width optimisation impacts resource usage of arithmetic operations linearly. In the meanwhile, the varied constants construct the upper bound of resource usage. As resource consumption goes down with data presentation, system performance can be increased after a reconfiguration operation. For a reconfigurable area with upper bound resource consumption, multiple time steps can be mapped into it with reduced data presentation. As shown in Figure 2, the two time steps can be accomplished with data streamed one time, given the available resource can accommodate the dynamically optimised circuits.

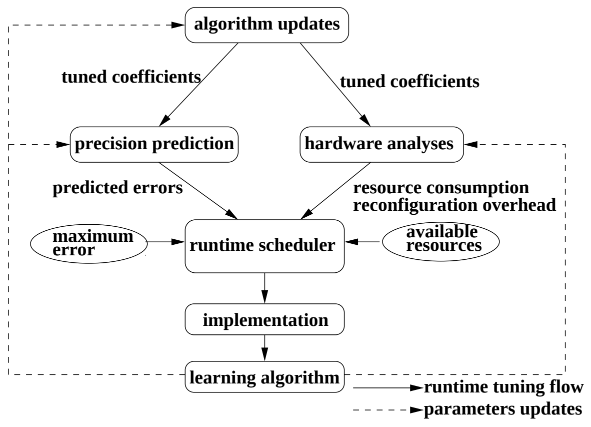

Fig. 3. Tuning process of the proposed approach.

Analysed data are fed into a runtime scheduler to decide whether the current configuration needs to be reconfigured. For the implemented circuits, data are sampled back into a learning algorithm to update module parameters, closing the tuning process.

5. CONCLUSION

In this paper, we present an adaptive approach to explore runtime properties of algorithms based on finite difference method. The computational error is dynamically controlled and the arithmetic operators are adaptively optimised. Work in progress and future work include expanding modules in the turning process, exploring more runtime properties and building various applications to evaluate the proposed approach.

6. ACKNOWLEDGEMENT

This work was supported in part by UK EPSRC, by the European Union Seventh Framework Programme under Grant agreement number 248976, 257906 and 287804, by the HiPEAC NoE, by Maxeler University Program, and by Xilinx.

7. REFERENCES

[1] K. Bruneel, F. Abouelella, and D. Stroobandt, “Automatically mapping applications to a self-reconfiguring platform,” in *DATE*, 2009, pp. 964–969.

[2] Q. Jin, T. Becker, W. Luk, and D. B. Thomas, “Optimising explicit finite difference option pricing for dynamic constant reconfiguration,” in *FPL*, 2012.

[3] J. Zhang, Z. Zhang, S. Zhou, M. Tan, X. Liu, X. Cheng, and J. Cong, “Bit-level optimization for high-level synthesis and fpga-based acceleration,” in *FPGA*, 2010, pp. 59–68.

[4] D.-U. Lee, A. A. Gaffar, R. C. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, “Accuracy-guaranteed bit-width optimization,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[5] S. Bayliss and G. A. Constantinides, “Optimizing sdram bandwidth for custom fpga loop accelerators,” in *FPGA*, 2012, pp. 195–204.

[6] G. C. T. Chow, A. H. T. Tse, Q. Jin, W. Luk, P. H. W. Leong, and D. B. Thomas, “A mixed precision monte carlo methodology for reconfigurable accelerator systems,” in *FPGA*, 2012, pp. 57–66.

[7] L. Singhal and E. Bozorgzadeh, “Multi-layer floorplanning on a sequence of reconfigurable designs,” in *Proc. FPL*, 2006.

[8] K. Bruneel, F. Abouelella, and D. Stroobandt, “Automatically mapping applications to a self-reconfiguring platform,” in *Proc. DATE*, 2009.

[9] F. Nava *et al.*, “Applying dynamic reconfiguration in the mobile robotics domain: A case study on computer vision algorithms,” *ACM Trans. on Reconfigurable Technology and Systems*, vol. 4, no. 29, 2010.

[10] D. Koch and J. Torresen, “FPGASort: A high performance sorting architecture exploiting run-time reconfiguration on FPGAs for large problem sorting,” in *Proc. FPGA*, 2011.

[11] J. Stolfi and L. Figueiredo, “Self-validated numerical methods and applications,” *Brazilian Mathematics Colloquium Monograph*, 1997.