Precision Modeling and Bit-width Optimization of Floating-Point Applications

Zhihong Zhao
Alternative System Concepts, Inc.
Windham, NH

Miriam Leeser
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA

1. Introduction

We present a floating-point precision modeling methodology that can be used to develop application adaptive arithmetic precision models for variable bit-width floating-point computing. We also developed optimization algorithms that minimize the total bit-width for the application such that the output accuracy meets user-defined requirements. The methodology supports different bit-widths for different variables in the datapath.

Computing using floating-point (FP) representations provides a wide dynamic range of real numbers, freeing programmers from writing the manual scaling code required for fixed-point representation. Nevertheless, floating-point operations have always been considered beyond the capabilities of custom or re-configurable hardware implementation. IEEE standard precision floating-point operations cost too much in power and area to be practical on many devices.

A promising solution to reduce the cost of FP implementation is to reduce the bit-width of the FP representation. Research results show that it is feasible and beneficial to use reduced bit-width FP representation in modern multimedia and streaming application workloads [1]. By taking advantage of bit-width information during architectural synthesis, area is reduced by 15-86%, clock speed improved by 3-249%, and power consumption reduced by 46-73% [2].

The optimal bit-widths are the smallest bit-widths that satisfy the accuracy requirement. They can be obtained through simulation-based searching or model-based optimization. Simulation-based bit-width searching is a process that simulates using all possible bit-widths, and finds the best solution. It is a straight-forward method to determine the minimal bit-width, but it does not provide any intelligent optimization, and it can consume enormous computation time, especially when the target applications are large designs or a large input space is involved. As a better approach, model-based bit-width optimization eliminates the need for exhaustive simulation, and automatically analyzes and adapts the level of precision according to the need of an application.

The FP precision modeling methodology presented in this paper is an application-adaptive arithmetic model in the form of a function between the relative error in the output and the mantissa bit-widths (one for each FP variable) used in the FP datapath. The model constructed using this methodology can estimate the output error range given the custom FP bit-widths used. The optimization algorithm developed to optimize the bit-width is a combination of the popular Steepest Descent method and the unique characteristics of the bit-width optimization problem.

2. A Methodology for FP Precision Modeling

An arithmetic precision model can be built, based on the application's function and data, to represent the relationship between output precision and bit-widths used in the FP application. Our experimental results prove that the precision model constructed via this method gives reasonable estimates of the output accuracy. We have successfully used the precision model in a bit-width optimization program and obtained optimized reduced bit-width. The methodology for developing such a model is presented in this section.

The FP application being analyzed is represented in a graphical intermediate format: the Control and Data Flow Graph (CDFG). The CDFG is commonly used in high-level synthesis and can effectively represent the functional and the structural description of an application. A CDFG representation of a differential equation solver is shown in Figure 1.

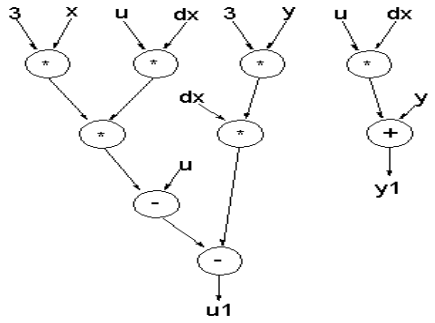

Figure 1. CDFG of a differential equation solver

The first process in the precision modeling is called behavioral profiling. Behavioral profiling is analogous to

software profiling. Given a behavioral specification (CDFG) of an application and a set of input vectors, behavioral profiling involves gathering pertinent profile data such as number of times an operation node is visited, number of times a conditional branch is taken, and number of times a loop or subprogram is executed. The behavioral profiling process involves a one-time simulation prior to constructing the model.

The behavioral profiler does the following. For each CDFG node *n*, determine the number of times the node is executed for the given profiling stimuli, and record each bit of the result of the operation. Bit probabilities (probability of the bit being "1") of the result, *Pn(i)* can be calculated based on this information.

The profile data is used to construct a precision model that best reflects the functional relationship between the bit-width of floating-point operations and the output precision. The model constructed using the methodology is an arithmetic model that describes the function between the output precision and the bit-widths used in the FP application. The overall error at the output of an operation is composed of propagation error, which is determined by the errors of input data and the operation type only, and rounding error, which is caused by the rounding of the operation result.

There are many ways to estimate the bounds of the rounding error. With the profile data available, the most accurate and convenient method for this research is presented in formula (1).

$$\varepsilon_{z\_r}=\frac{z-\hat{z}}{z}=\frac{(a_{b+1}2^{-b-1}+\cdots+a_{22}2^{-22}+a_{23}2^{-23})}{(1.0+a_1 2^{-1}+a_2 2^{-2}+\cdots+a_b 2^{-23})}\approx\frac{p_{b+1}2^{-b-1}+\cdots+p_{22}2^{-22}+p_{23}2^{-23}}{1.0+p_1 2^{-1}+p_2 2^{-2}+\cdots+p_b 2^{-23}} \quad (1)$$

The propagation error is derived based-on the Mean-value Theorem. The result is presented in formula (2).

$$\varepsilon_f=\frac{|f(x,y)-f(\hat{x},\hat{y})|}{f(x,y)}\approx\frac{f'(\hat{x},\hat{y})\hat{x}}{f(\hat{x},\hat{y})}\varepsilon_x+\frac{f'(\hat{x},\hat{y})}{f(\hat{x},\hat{y})}\varepsilon_y=k_x\varepsilon_x+k_y\varepsilon_y \quad (2)$$

The *k*s depend on the operation type, and also the profile data. For operation MULTIPLY ($z = x * y$),

$$k_x=\frac{f'(\hat{x},\hat{y})\hat{x}}{f(\hat{x},\hat{y})}=\frac{\hat{x}\hat{y}}{\hat{x}\hat{y}}=1.0,\ k_y=\frac{f'(\hat{x},\hat{y})\hat{y}}{f(\hat{x},\hat{y})}=\frac{\hat{x}\hat{y}}{\hat{x}\hat{y}}=1.0 \quad (3)$$

For operation DIVIDE ($z = x / y$),

$$k_x=\frac{f'(\hat{x},\hat{y})\hat{x}}{f(\hat{x},\hat{y})}=\frac{\hat{x}\hat{y}}{\hat{x}\hat{y}}=1.0,\ k_y=\frac{f'(\hat{x},\hat{y})\hat{y}}{f(\hat{x},\hat{y})}=\frac{-\hat{x}\hat{y}}{\hat{x}\hat{y}}=-1.0 \quad (4)$$

The arithmetic model for any FP operation is the sum of these two errors. The precision model for the entire application whose structural information is represented in the CDFG can be easily derived.

3. Experiment Results

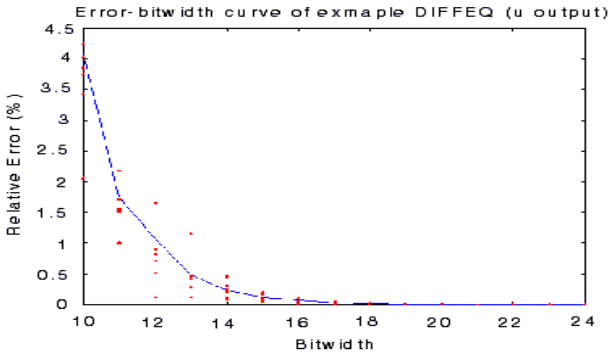

Figure 2. Comparison of estimated error and actual error

Figure 2 shows the comparison of the error estimated using the precision model (dashed line) and the actual error (dots) for the DIFFEQ example (shown in Figure 1). The result proves that the precision models developed using the methodology can effectively estimate the error range.

The bit-width optimization problem is solved by a Grid Steepest Descent (GSD) method derived for this specific precision modeling methodology and this bit-width optimization problem. Optimization results of 2 examples are shown in Table 1 – the DIFFEQ example and a simple three-multiplication-operation example.

Number of Total Bit-Width in Datapath

	IEEE 754	1%	5%	10%
DIFFEQ	**230**	**162**	**132**	**124**
3 MULT	**69**	**43**	**37**	**34**

Table 1. Optimization results for different precision targets

The results demonstrate that the GSD optimization method can be successfully used with the precision models to calculate the minimal bit-widths that satisfy the user-defined precision requirement of the application. The minimal bit-widths can be the same bit-width for all operations, or one for each individual operation.

References

[1] J.Y.F. Tong, D. Nagle, and R. Rutenbar, "Reducing Power by Optimizing the Necessary Precision Range of Floating Point Arithmetic," in *IEEE Transactions on VLSI systems*, Vol. 8, No.3, pp 273-286, June 2000.

[2] M. Stevenson, J. Babb, and S. Armarasinghe, "Bitwidth Analysis with Application to Silicon Compilation," in *ACM SIGPLAN Conference on Programming Language Design and Implementation*, June 2000, pp. 108-120.