Microprocessors and Microsystems 48 (2017) 48–55

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Approximate computing of two-variable numeric functions using multiplier-less gradients

Jochen Rust*, Nils Heidmann, Steffen Paul

Institute of Electrodynamics and Microelectronics (ITEM.me), University of Bremen, Building NW 1, Bremen, Germany

A R T I C L E I N F O

Article history:
Received 22 February 2016
Revised 22 June 2016
Accepted 16 September 2016
Available online 23 September 2016

Keywords:
Approximate computing
Numeric function approximation
Two-variable
Least-squares
Computer arithmetic

A B S T R A C T

Approximate computing of non-trivial numeric functions is a well-known design technique and, therefore, used in several different application areas. Its main idea is the relaxation of conventional correctness constraints to achieve high performance results in terms of throughput and/or energy consumption. In this paper we propose an automated approximate design method for the fast hardware generation of two-dimensional numeric functions. By using multiplier-less gradients in combination with an advanced non-uniform segmentation scheme, high hardware performance is achieved. To qualify our approach, exhaustive evaluation is carried out, considering six different two-variable numeric functions. In a first step, a global complexity estimation is performed with varying design constraints on the algorithmic level. Out of this, most suitable candidates are selected for logic and physical CMOS synthesis. The results are compared to actual references highlighting our work as a powerful approach for the hardware-based calculation of two-variable numeric functions, especially in terms of throughput and energy consumption.

1. Introduction and related work

Efficient digital signal processing of mathematical terms and expressions has been a fundamental research topic in electrical engineering and computer science for decades [1]. Within this scope, the calculation of elementary functions[1] by digital means is one of the most essential tasks that has been exhaustively studied in a considerable number of different applications like mobile communications [3,4], medical devices [5,6], video, image and 3D graphics processing [7–9], artificial intelligence [10,11], hardware-accelerated simulation [12], etc.. Moreover, additional research has been conducted for numeric functions with a higher number of variables. In this case, hand-optimized solutions which are highly adapted to a specific application are mostly taken into account [13–15].

For the signal processing of (one-variable) elementary functions, three different techniques can be distinguished in general: iterative methods, polynomial approximations and ROM-methods [16]. For further hardware performance[2] improvement, a combination of two or more techniques has turned out to be a highly feasible approach. For instance, the utilization of ROM-methods in conjunction with polynomial approximations leads to piecewise signal processing with a set of sub-functions of reduced complexity [2]. Further approaches consider iterative methods associated with simple polynomial approximations, providing a rough but useful start point estimation. Another promising approach is to enhance the signal processing performance by quantizing multiplicands which results in low-complexity multiplier-less coefficients [17]. In all cases, one or more performance criteria are always traded off against others.

Recently, the exploration of efficient numeric function computation has been revitalized by the arised paradigm of approximate computing [18]. Its objective is to exploit faulty signal processing elements, such as transistors, gates or even more complex hardware units, for an increase of hardware performance, mostly energy efficiency [19]. Due to several reasons, e.g., inherent algorithmic resilience or "good-enough" results from the users perspective, this effect can be tolerated by various applications, whereas the degradation of the resulting accuracy is only marginal or can even be neglected. In the scope of approximate computing for numeric functions, most research work has focused on elemen-

* Corresponding author.
E-mail addresses: rust@me.uni-bremen.de (J. Rust), heidmann@me.uni-bremen.de (N. Heidmann), steffen.paul@me.uni-bremen.de (S. Paul).

[1] According to [2] the term "'elementary function" comprises in this paper most commonly used functions, e.g., trigonometric functions, root- and exponential-functions, etc.

[2] In this paper the term "hardware performance" summarizes various different performace criterias, e.g., timing (delay/latency), complexity or energy

http://dx.doi.org/10.1016/j.micpro.2016.09.005
0141-9331/© 2016 Elsevier B.V. All rights reserved.

tary arithmetic. For example, multi-speculative-adders [20] and -multipliers [21] were explored, recently. For the approximate computing of two-variable (two-dimensional, 2D) numeric functions ($f(x_1, x_2)$) only a few different approaches are available, though, these functions appear in a large number of algorithms, e.g., QR-decomposition [22] or synchronization [14]. First approaches for fault-affected high-throughput hardware realizations for 2D numeric function approximations (NFAs) have been presented by Nagayama et al. [23]. A bilinear interpolation scheme is exploited to achieve high-accuracy NFAs. However, in comparison to one-dimensional function approximation techniques, this method requires comparatively high signal processing effort. Another drawback occurs for symmetric functions ($f(x_1, x_2) = f(x_2, x_1)$): As this type of function normally allows to swap the input operands, the function range can be evidently decreased. Hence, in the scope of piecewise NFAs, this is a welcome feature to reduce the segmentation effort. However, considering bilinear interpolation polynomials, additional control effort is required to take advantage of this technique, as the coefficients must be reassigned [23]. In [24] a first approach for 2D NFA with multiplier-less gradients and a piecewise approximation scheme has been introduced with very promising results for low-accuracy NFAs. However, considering more demanding approximations, the coarse coefficient estimation technique causes a large number of segments and, consequently, weakens the overall signal processing performance excessively.

In this paper, a novel approach for the automated generation of 2D linear function approximations is proposed. A hardware design method is presented realizing high-performance signal processing at runtime. For the coefficient estimation, a linear regression technique that bases on the well-known least-squares (LS) methodology is taken as reference. The use of a restricted non-uniform segmentation scheme enables fast access to existing sub-functions and, consequently, realizes a high throughput. For a further performance increase, several optimization steps are performed on both the algorithmic and the hardware level. The hardware mapping is performed by transferring the necessary data (parameters) on customizable VHDL template files. As an additional feature, a Matlab-based simulation model is generated enabling rapid prototyping in an early state of the digital design flow.

The content of this paper is organized as follows. In Section 2 the basic notions and LS fundamentals are introduced. Section 3 provides a detailed explanation on the proposed design method for 2D NFAs. After that, performance results of selected functions are presented and discussed in Section 4 before our work is concluded in Section 5.

2. Preliminaries

2.1. Definitions

Definition 1. A segment denotes a specified (sub-)range of a given 2D function. In this paper the size of a segment is determined by start and end point arguments.

Definition 2. The maximum error $\tilde{\varepsilon}_{\max}$ denotes the maximum difference between the (quantized) absolute values of an original function and its NFA inside a segment.

Definition 3. The specified error ε is the user-defined constraint determining the maximum tolerable error.

Definition 4. The resolution denotes the smallest considerable value that occurs in the chosen fixed-point format due to digital quantization effects

Definition 5. The quantization factor (QF) denotes the number of nonzero digits of multiplier-less gradient coefficients.

Definition 6. The quantization function of multiplier-less gradients $Q_{QF}(\cdot)$ calculates the (floored) quantized fixed point value of a real number which can be realized by the accumulation of q partial products (QF=q).

2.2. Least-squares-based linear regression

Linear (multiple) regression is a method to model the relationship among independent variables $x_1, \ldots, x_D$ and a response variable y [25]. The linear relationship can be approximated by the regression model

$$y = f(x_1, \ldots, x_D) + \tilde{\varepsilon}_R = \beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D + \tilde{\varepsilon}_R \ , \tag{1}$$

where $\beta_0, \ldots, \beta_D$ are called regression coefficients and $\tilde{\varepsilon}_R$ is a random error term. A data set of n units with D uncorrelated input variables can be summarized in a design matrix $\mathbf{X}$

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \ldots & x_{1D} \\ 1 & x_{21} & \ldots & x_{2D} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \ldots & x_{nD} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix} \ , \tag{2}$$

where the first column, filled by ones, refers to the constant β_0. According to the design matrix, Eq. (1) can be written in vector form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \tilde{\varepsilon}_R \ . \tag{3}$$

In this paper a linear LS approach is taken into account for the generation of the 2D NFAs (see Section 3). Mathematically speaking, the sum of squared differences between modeled and acquired data values is minimized. As long as the design matrix $\mathbf{X}$ has linearly independent columns, the LS-solution of the approach is unique. In order to estimate the coefficients $\boldsymbol{\beta}$, Eq. (3) can be transposed to

$$\tilde{\boldsymbol{\varepsilon}}_R = \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \ . \tag{4}$$

The vector of the LS estimator $\boldsymbol{\beta}_{LS}$ is obtained by minimizing the sum of squares

$$L = \sum_{i=1}^{n} \tilde{\boldsymbol{\varepsilon}}_{R,i}^2 = \tilde{\boldsymbol{\varepsilon}}_R^T \tilde{\boldsymbol{\varepsilon}}_R = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_{LS})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_{LS}) \ . \tag{5}$$

The estimator must satisfy $\delta L / \delta \boldsymbol{\beta}|_{\boldsymbol{\beta}_{LS}} = 0$, resulting in

$$\boldsymbol{\beta}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \ . \tag{6}$$

Based on the fitted regression model the output value for a vector of input variables can be estimated by

$$y_{LS} = \beta_{LS,0} + \beta_{LS,1} \cdot x_1 + \ldots \beta_{LS,d} \cdot x_D \ . \tag{7}$$

For the proposed 2D function approximation method, the number of input variables is set to the number of operands ($D = 2$).

3. Approximate synthesis of 2D functions

As briefly mentioned in Section 1, the main idea of our work is the automated transfer of a given original 2D function $f(x_1, x_2)$ to a corresponding NFA hardware architecture. In order to provide accuracy-driven hardware design, the resulting approximation quality has to be specified in advance considering the error constraint ε. This method can be interpreted as a synthesis step for digital hardware design of 2D numeric functions with an user-defined accuracy.