Challenges in Verifying and Optimizing Fixed-Point Arithmetic-intensive Designs

Yu Pang, O. Sarbishei, K. Radecka, Zeljko Zilic
Dept. of Electrical and Computer Engineering, McGill University, Montreal, Canada
Email: {yu.pang@mail.mcgill.ca, omid.sarbishei@mail.mcgill.ca, katarzyna.radecka@mcgill.ca, zeljko.zilic@mcgill.ca}

Abstract *– Arithmetic circuit plays a key role in digital signal processing (DSP). A datapath is used to implement the specification usually represented as a polynomial. The two most important problems are verification and optimization of the arithmetic circuits. Circuit verification confirms whether the implementation can realize the specification with correct behavior or two implementations match well, and optimization generates suitable bit-widths according to constraints. This paper depicts specification of arithmetic circuits, explains the techniques of verification and optimization, and describes current challenges in arithmetic circuit designs.*

I. Introduction

In digital signal processor and Application Specific Integrated Circuits (ASICs), arithmetic units are key components which mainly impact system performance. Fig. 1 illustrates a generic processor including an arithmetic datapath.

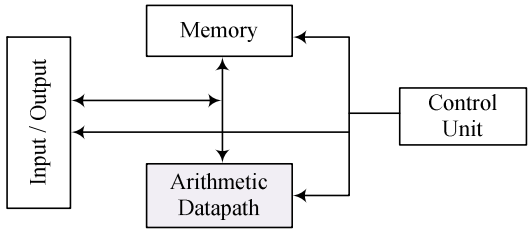

Fig. 1. A general digital processor

Because fixed-point arithmetic circuits are very important in low-power design and provide more flexibility than floating-point arithmetic circuits, we mainly discuss fixed-point circuits which include integer bit-widths (IBs) and fractional bit-widths (FBs). The whole procedure to design an arithmetic datapath includes steps of verification and optimization.

The specification is often represented as a polynomial. The authors in [1] propose an algorithm to convert a polynomial specification into a datapath by the constraint of the minimum area. The algorithm has capability to compensate coefficients and has been proved better than Horner transform and common subexpression elimination. Such an approach can be enhanced with scheduling techniques like the method in [16] to efficiently allocate the available arithmetic resources, i.e. multipliers and adders, to realize the datapath. After the datapath structure is obtained, verification can be performed to check whether it has correct behavior.

Verification is a necessary procedure aiming to check whether the design is correct or two designs have the same behavior. A popular verification process for arithmetic circuits is formal methods. Usual formal verifier adopts techniques of decision diagrams which can perform equivalence to formally proves that two representations of a circuit design exhibit exactly the same behavior. However, they can only yield information on whether or not an implementation matches a specification exactly, but offer no path for quantifying the degree to which the two offer. Therefore, if two functions are similar but not exactly equal, decision diagrams may implement drastically different arithmetic functions, while two very different diagrams may implement the same mathematical operation with different degrees of precision. So a great challenge is finding a new method to check whether the difference of two similar implementations is limited in a given error bound.

Optimization refers to a problem of bit-width allocation. Generally, finding the smallest suitable bit-widths is significant in order to obtain the smallest area and limit the error. Many numerical techniques have been developed for the purpose of the goal.

Dynamic analysis has been applied in most precision optimization schemes – works in [2] - [3] present the straightforward simulation-based techniques to assign bit-widths. The simplicity of this approach makes it prevalent; however, one has to enumerate and simulate all possible input values to provably complete the job.

On the other hand, static analysis attempts to determine, often by approximate methods, the range without resorting to simulations. *Interval arithmetic* (IA) is a usual approximate method to calculate the value bound, but it unavoidably leads to coarse results. The *affine arithmetic* model (AA) is a derivation of IA, in which the quantities of interest are represented as linear combinations (affine forms) of certain primitive variables standing for sources of uncertainty in the data or for approximations made during the computation. Work in [4] adopts AA to investigate integer bit-widths and implement FPGA by different bit-widths. Authors in [5] extend [4] to provide a method for optimizing word-lengths of hardware designs with fixed-point arithmetic based on analytical error models. However, the coarse results by AA generate overestimation leading to additional hardware cost. Therefore, verification and optimization by AA are limited due to such overestimation.

Other static methods include these based on *saturation arithmetic* [6] and the *SAT-Modulo Theory* (SMT) [7]. Authors in [8] [11] adopt a spectral technique, *Arithmetic Transform* (AT), to explore precision in the imprecise representation of Taylor series.

Fig. 2 compares the time requirement for each method. Static methods lead to over-allocation of bit-widths and in

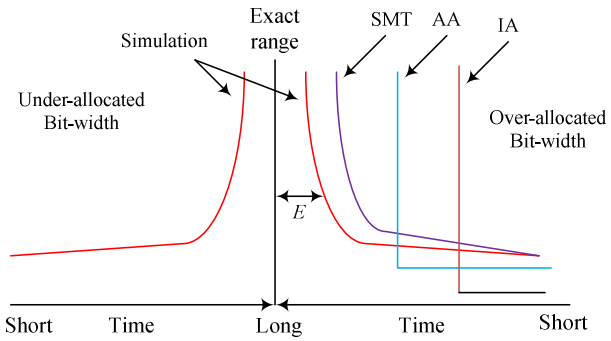

Fig. 2. Tradeoff between ranges and calculation times

consequence, the creation of imprecise results enlarges implementation area. Dynamic methods might err on the side of not providing enough bit-widths by underestimating a range. Therefore, a challenge for optimization is to design a new method which can automatically allocate the suitable bit-widths with few execution time.

Since feedback circuits such as IIR filters always occur in DSP, verification and optimization by analysis of range and precision for IIR filters are also very important. Past methods do not have capability of handling the feedback circuits. Therefore, we outline new algorithms to handle arithmetic circuits including IIR filters for verification and optimization in this paper.

II. Background

The usual static analysis is commonly based on IA and AA. IA defines a set of operations on intervals. An operation <OP> on two intervals with <OP> is defined by:

$[x_1, x_2]$ <OP> $[y_1, y_2] = \{\, x$ <OP> $y \mid x \in [x_1, x_2], y \in [y_1, y_2] \,\}$

For example, the addition and multiplication operations yield:

$$[x_1, x_2] + [y_1, y_2] = [x_1 + y_1, x_2 + y_2]$$

$$[x_1, x_2] * [y_1, y_2]$$

$$= [\min(x_1y_1, x_1y_2, x_2y_1, x_2y_2), \max(x_1y_1, x_1y_2, x_2y_1, x_2y_2)]$$

In AA, an ordinary interval $[x_{min}, x_{max}]$ for an input variable can be converted into an equivalent affine form $x_A = x_0 + x_1\varepsilon$ with:

$$x_0 = \frac{x_{max} + x_{min}}{2}, \quad x_1 = \frac{x_{max} - x_{min}}{2} \qquad (1)$$

The intermediate signal or the output is represented as a first degree polynomial:

$$y_A = y_0 + y_1\varepsilon_1 + y_2\varepsilon_2 \ldots + y_n\varepsilon_n \qquad (2)$$

where $y_0, y_1, \ldots y_n$ are real-valued numbers and $\varepsilon_1, \varepsilon_2, \ldots \varepsilon_n$ are symbolic *uncertain variables* whose values are not exact, but only known to lie in the range [-1, +1].

IA and AA both calculate coarse results to obtain over-estimation of bit-widths. In order to avoid the disadvantage, we propose new methods based on *Arithmetic Transform* (AT) which has been successfully used for precision verification. AT is an instance of spectral representations, where the spectral coefficients contain function information that is global over the domain of definition, and by that allow a number of properties to be more easily deduced than in the Boolean domain. Fig. 3 illustrates the relationship of the Boolean domain and spectral domain connected by AT.

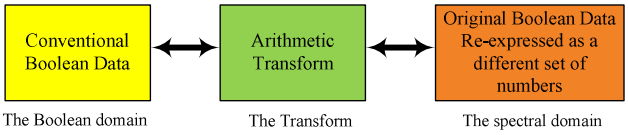

Fig. 3. Relationship of the Boolean domain and spectral domain

Definition 1: The ***Arithmetic Transform (AT)*** [17] is a polynomial representing a pseudo Boolean function $f: B^n \to w$ with an arithmetic operation "+", word-level coefficients c, binary inputs $x_1, x_2 \ldots x_n$ and binary exponents $i_1, i_2 \ldots i_n$:

$$AT(f) = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \ldots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n} \qquad (3)$$

A matrix multiplication is frequently used to determine AT of a given function. In this method, the set of AT coefficients $c = \{c_{i_1 i_2 \ldots i_n}\}$ are obtained by multiplying the $2^n \times 2^n$ matrix T_n by a $2^n \times 1$ vector of function values (truth table of f): $c = T_n \times 2^n$ where the transform matrix T_n is defined recursively:

$$T_n = \begin{bmatrix} T_{n-1} & 0 \\ -T_{n-1} & T_{n-1} \end{bmatrix} \qquad T_0 = 1 \qquad (4)$$

By Def. 1 and Eqn. (4), it is easy to conclude that AT has linear features, that is:

$$\begin{cases} AT(f_1 + f_2) = AT(f_1) + AT(f_2) \\ AT(const * f_1) = const * AT(f_1) \end{cases} \qquad (5)$$

Given a real-valued polynomial with one or multiple word-level variables composed of binary vectors ($x_{N-1}, x_{N-2}, \ldots x_0, y_{M-1}, y_{M-2}, \ldots y_0, \ldots$), the AT can be constructed by replacing variables by its defining polynomial. For instance, if X and Y are unsigned input integers represented by 2 and 3 bits, and the polynomials is $f(X,Y) = 2X^3 + Y^2$, then its corresponding AT polynomial form is:

$$AT[f(X,Y)] = AT(2X^3 + Y^2) = AT(2X^3) + AT(Y^2)$$

$$= 2AT(X)^3 + AT(Y)^2 = 2 * (\textstyle\sum_{i=0}^{1} 2^i x_i)^3 + (\sum_{k=0}^{2} 2^k y_k)^2$$

$$= 16x_1 + 36x_1x_0 + 2x_0 + 16y_2 + 16y_2y_1 + 8y_2y_0 + 4y_1 + 4y_1y_0 + y_0$$

After the polynomial is converted to AT, its exact implementation can be verified using equivalence checking, while an imprecise one is checked through the search for maximal imprecision. In fact AT has been proved very efficient for imprecise circuit verification [10] [17]. In order to achieve this goal, the AT generation algorithm and a branch-and-bound searching method have been fine tuned in [12] to efficiently handle the intermediate expression swell during the calculation of the extreme values of the AT polynomial.

III. Verification Of Two Implementations

For fixed-point circuits, the presence of fractional bit-widths causes the precision problem leading to the implementation cannot exactly realize the specification, or two implementations do not have the same behavior which is referred to component matching. Fig. 4 illustrates two implementations, *Imp1* and *Imp2*, differing from the specification Spec by errors $e1$ and $e2$,

respectively. If the error is within the allowed error bound E, then any such implementations is deemed acceptable, but the least costly one is preferable.

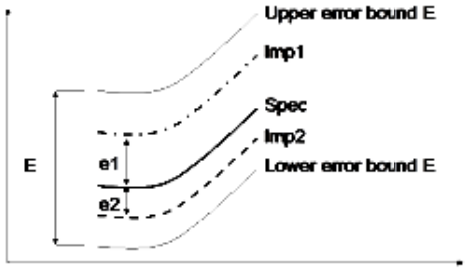

Fig. 4. Comparison of two implementations

The use of the algorithm within a realistic tool for comparing precision among two different implementations of real-valued functions is shown in Fig. 5. The interface file describes two implementations of Taylor series or real-valued polynomials and the bit-widths of corresponding variables. The front-end and the parser hide the details needed to deal with AT. The conversion algorithm converts the two implementations of real-valued functions into two AT polynomials, while the error AT is obtained by subtracting the two polynomials. Then the imprecision is obtained by the searching algorithm. The two key subroutines of the conversion algorithm which can convert a polynomial into an AT representation and the branch searching algorithm which can find the absolute maximum value for an AT representation have been proposed in [12].

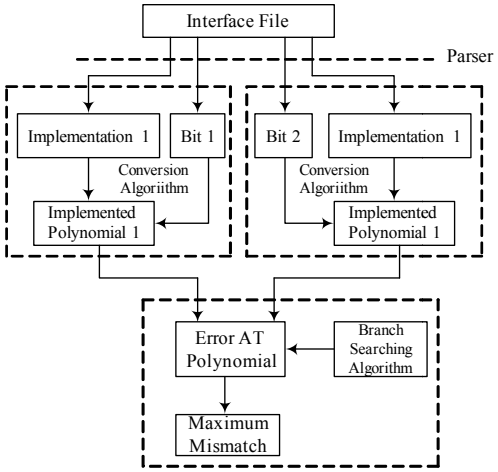

Fig. 5: Algorithm of computing imprecision between two implementations

IV. Optimization Of Bit-width Allocation

Allocating bit-widths in a datapath is a necessary step in synthesis because of its direct impact on resources and delay. The fixed-point representation includes the precision problem and the range problem. Manual or sub-optimal methods might over- or under-allocate bit-widths. Too few bits will cause overflow or beyond the given error bound, while too many are costly. Therefore, finding optimization algorithms for the most appropriate bit-widths is a significant contribution in the high-level synthesis of datapaths.

4.1 Allocation of IBs

In order to avoid disadvantages of low efficiency and coarse results, we propose a hybrid method to calculate ranges for each signal in a given datapath.

Definition 2: The error bound labeled as E in Fig. 2 is the largest difference between the exact and the obtained ranges, while the error ratio e_r calculated as $\left(1-\frac{\text{exact range}}{\text{obtained range}}\right)*100\%$ represents the effective size of the obtained range. ■

The objective is to find the smallest value of E and e_r whilst maintaining the one-sided error, i.e., not underestimating the bit-width.

Definition 3: Correlation is present if at least two monomials in a polynomial represented by sum-of-product have in their support the same variable. ■

The correlation in two monomials means that if the value of one monomial changes, the other will follow the change. Clearly, the case may lead to overestimation of range because the two monomials might not reach their maximum or minimum values at the same time, so handling the correlation becomes a key task in range analysis.

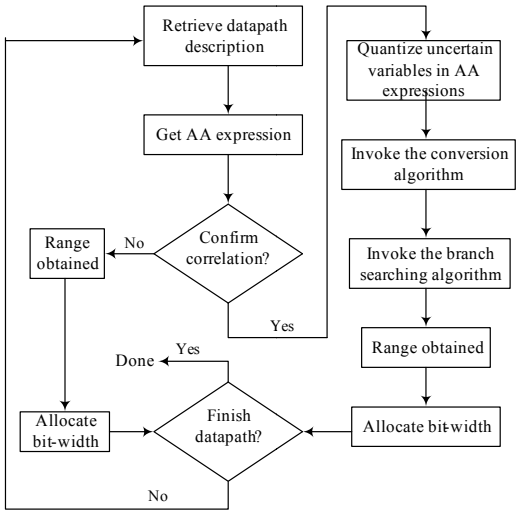

Fig. 6. Algorithm for allocating IBs for a datapath

Fig. 6 summarizes our algorithm for allocating integer bit-widths in a datapath. It first retrieves the polynomial description representing a datapath, and generates the AA expression for future utilization. If the polynomial has no correlation, IA is used to compute the exact range so the bit-widths will be determined; otherwise, the uncertain variables are quantized in the AA expression, and the conversion algorithm is invoked to convert the expression to its AT form. Then the branch-and-bound searching algorithm is applied to find the upper and the lower bounds, and estimate the bound intervals. Finally, the IBs of the datapath are allocated. The novel algorithm can obtain both better indicators of the error bound and error ratio compared to past methods.

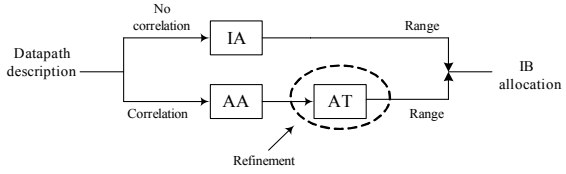

Fig. 7. An abstract model of the proposed algorithm

Fig. 7 illustrates an abstract model of the algorithm. It invokes different methods to handle a datapath, and can distribute correlation to AA and AT for two-step processing. First, AA partly handles correlation, and then AT continues to refine the results. In contrast to our method, other, commonly encountered approach, i.e., SMT is very time consuming, as it only has one-step processing, Fig. 8.

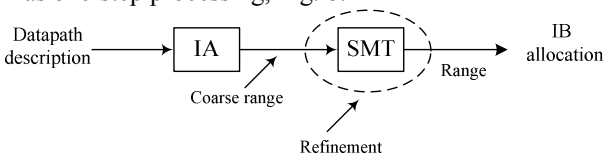

Fig. 8. SMT model to perform range analysis

4.2 Allocation of FBs

Most fixed-point circuits include fractional bit-widths leading to imprecise results. Infinite-length Taylor series are a typical real-valued function representation of transcendental and other functions of importance in engineering. Their implementations are inherently approximated, as only a finite number of terms can be realized in hardware. Therefore, its result is imprecise and cannot be exactly equal to the specified value.

Definition 4: A real and differentiable function *f(X)* can be represented as a **Taylor series** over an interval *I* in the neighborhood of the initial value X_0:

$$f(X) = \sum_{n=0}^{\infty} \frac{1}{n!}(X - X_0)^n f^{(n)}(X_0)$$

The **finite Taylor expansion** restricted to the first *n+1* terms is:

$$f(X_0) + Xf'(X_0) + \frac{X^2}{2!}f''(X_0) + \cdots + \frac{(X - X_0)^n}{n!}f^{(n)}(X_0)$$

and hence the **approximation error** given in terms of a point φ in the interval *I* is:

$$R_n(X) = \frac{f^{(n+1)}(\varphi)}{(n+1)!}(X - X_0)^{n+1} \qquad (6)$$

Determining the set of parameters needed to achieve a sufficiently precise circuit is a challenge requiring a well structured precision analysis. We perform the analysis of an arithmetic imprecision caused by all approximations and finite bit-widths in the implementations of real-valued specifications such as Taylor series example in Fig. 9.

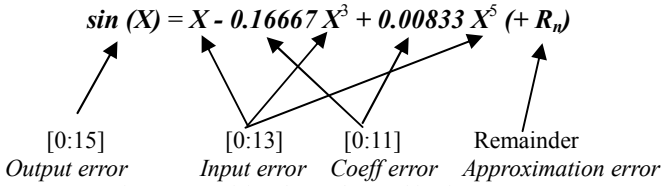

Fig. 9. Imprecision due to the combined sources

We demonstrate the imprecise circuit optimization for real-valued specifications that simultaneously considers multiple bit-widths as well as the function approximation by finite Taylor series since these two factors impact precision hugely than other factors.

In realizing real-valued functions by arithmetic circuits, an algorithm might be employed to *approximate*, rather than *exactly* implement the function. For instance, when using the first *n* terms of a Taylor series to represent a transcendental function, the approximation error is provably bounded by a remainder $R_n(X)$, Eqn. (6). For a function defined in interval *I*, this truncation error bound e_t translates into:

$$e_t = \max_{X \in I} |R_{n-1}(X)| \qquad (7)$$

In our scheme, we present the interval analysis in terms of AT. For simplicity, the original interval *I* is normalized to [0,1] causing the input *X* to be scaled down. In consequence, instead of dealing with the input bit-width, we consider the number of *Fractional Bits (FB)*. The input range is divided into uniform 2^{FB} intervals of the size 2^{-FB}. Hence, the value X_{th} lies between two consecutively quantized numbers and the relation between X_{th} and *X* is then:

$$|X_{th} - X| \le 2^{-(FB+1)}$$
$$\Rightarrow X - 2^{-(FB+1)} \le X_{th} \le X + 2^{-(FB+1)} \qquad (8)$$

By replacing X_{th} with *m* fractional bits of *X* in accordance with Eqn. (8), we obtain the ATs for the theoretical f_{th} and quantized *f* Taylor functions (given $X_0 = 0$):

$$f_{th} = \sum_{i=0}^{n-1} C_i X_{th}^i = \sum_{i=0}^{n-1} C_i [(\sum_{k=0}^{m-1} 2^{-(k+1)} x_k) \pm 2^{-m-1}]^i \quad (9)$$

$$f = \sum_{i=0}^{n-1} C_i X^i = \sum_{i=0}^{n-1} C_i (\sum_{k=0}^{m-1} 2^{-(k+1)} x_k)^i \qquad (10)$$

where C_i is a Taylor coefficient equal to $\frac{f_{th}^i(X_0)}{i!}$.

For functions f_{th} and *f*, the AT polynomials $AT(f_{\text{th}})$ and $AT(f)$ are on the right-hand side of Eqn. (9) and (10), respectively. The bound e_{i} on the effects of input quantization of half a *ulp* (*unit in the last place*) to the output precision is obtained by the AT formulation as follows. The *error polynomial* $AT(f_{ei})$ is determined as a difference between $AT(f_{th})$ and $AT(f)$:

$$AT(f_{ei}) = AT(f_{th}) - AT(f) =$$
$$= AT(\sum_{i=0}^{n} C_i [(\sum_{k=0}^{m-1} 2^{-(k+1)} x_k) \pm 2^{-m-1}]^i) - AT(\sum_{i=0}^{n} C_i (\sum_{k=0}^{m-1} 2^{-(k+1)} x_k)^i) \qquad (11)$$

The maximum value of $AT(f_{ei})$ in Eqn. (11) gives the error bound e_i. In practice, e_i can be obtained by an efficient branch-and-bound search algorithm tuned for this application.

As the number of AT polynomial terms exhibits the same tendency as the one just described, we use |terms(*AT*(*f*))| as the cost function to be minimized. The size of AT is obtained by directly expanding the *n*-term Taylor polynomial over the *m*-bit input. One can show that the number of AT terms is:

$$|terms(AT(f_{n,m})| = \sum_{i=1}^{n} \binom{m}{i} \qquad (12)$$

It is hard to know in advance which factor has more effect on area since the Taylor terms and the input bit-width are involved in the interplay and determine AT size jointly by Eqn. (12).

The paper [11] proposes an algorithm optimizing the number of Taylor terms and the input bit-width described in Fig. 10. In the first iteration, the algorithm gets the smallest number of Taylor terms for given error bound, and obtains initial input bit-width. To explore the search space, it suffices to consecutively increase the set of Taylor terms, while simultaneously exploring alternative the input bit-widths. If the new node can satisfy the error bound *E*, the newly computed number of Taylor terms is assumed, and the algorithm continues to decrease input bit-width until the current node breaks the bound. When it happens, the algorithm backtracks to

the previous node and stores it. The procedure is repeated until the change of bit-widths are exhausted. Since Taylor series cannot be compared directly, it is necessary to use AT for comparison because of easy computation of Eqn. (12). A subroutine is called to compare AT size of each stored node and select the one with the smallest AT representation.

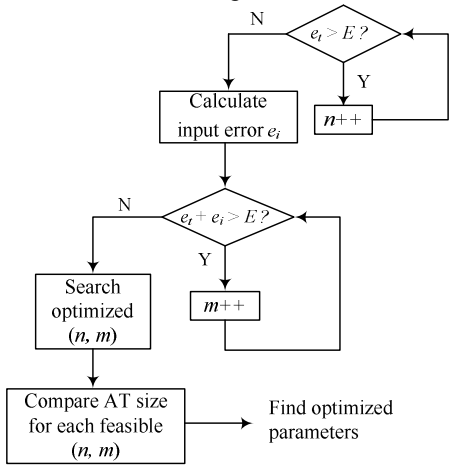

Fig. 10. Algorithm to find optimized parameters for Taylor series

V. Optimization of Linear Circuits With Feedbacks

A major deficiency of the explained approaches is that they only handle direct datapaths while they are invalid if the circuits have feedbacks, which constitute infinite loops. As the high-level representation of several fixed-point Digital Signal Processing (DSP) circuits is based on arithmetic expressions with possible feedbacks, the range and precision analysis of such circuits in general remains challenging. *Linear* fixed-point arithmetic circuits can be modeled as infinite impulse response (IIR) filters, which have important applications. For instance, authors in [18] use an IIR filter to realize echo cancellation in an IP network.

Some previous works have been dedicated to the problem. *Milic et al.* [13] conceive a new design for multiplierless IIR filters in the form of a parallel connection of two all-pass networks. The technique implements a smaller number of shifters and adders and guarantees precision. *Carletta et al.* [14] present an analytical framework to determine coefficient bits. The technique estimates the bound output and analyzes the truncation error. By the results, IBs and FBs can be selected to avoid overflow and guarantee accuracy in the fixed-point hardware. The major drawback of this method is that not only it may result in overestimations of range and precision, but also the sensitivity analysis it utilizes for tracking the impact of coefficient quantization errors on the position of zeros and poles, is only valid for 2nd order IIR filters.

We solve the above problems and propose two robust heuristics to compute range and precision for an arbitrary order IIR filter. The algorithms are utilized for allocating efficient values of IBs and FBs to all the variables based on the given input and error bounds.

Definition 5: Linear Circuits with Feedbacks (LCF). Using a polynomial to represent a circuit with feedbacks, if the summation of variable degrees for each monomial including feedback ones, is equal to "1", the circuit is a LCF. ■

IIR systems, which have an impulse response function that is non-zero over an infinite length of time, can represent a LCF in general. A condensed form of the difference equation for an IIR filter is:

$$z[n] = \sum_{i=0}^{p} b_i x[n-i] + \sum_{j=1}^{Q} a_j z[n-j] \quad (13)$$

where P is the feedforward filter order, b_i are the feedforward filter coefficients, Q is the feedback filter order, and a_i are the feedback filter coefficients.

Based on Eqn. (13), IIR filters are typical LCFs as the feedforward/feedback variables are multiplied by constants and the corresponding monomial degrees are all equal to "1".

Lemma 1: The linear IIR filter in Eqn. (13) can be flattened as:

$$z[n] = c_n x[n] + c_{n-1} x[n-1] + \ldots c_1 x[1] \quad (14)$$

where c_i $(i = 1,2,\ldots,n)$ is a constant value. Computing the values of c_i has the complexity of $O(n)$.

Definition 6: BIBO Stability. A system is Bounded-Input, Bounded-Output (BIBO) stable when the output bound is always finite for an arbitrary bounded input.

Definition 7: The symbol $B_o[n] = (B_{olow}[n], B_{oupp}[n])$ representing the output bound after n iterations is defined as:

$B_{oupp}[n] = max\{max(z[n]), max(z[n-1]), \ldots, max(z[1])\}$

$B_{olow}[n] = min\{min(z[n]), min(z[n-1]), \ldots, min(z[1])\}$

where $max(z[n])$ / $min(z[n])$ is the maximum/minimum value of z in the n^{th} iteration and z is given by Eqn. (14). ■

Lemma 2: $B_{oupp}[n]$ and $B_{olow}[n]$ are monotonically increasing and decreasing functions w.r.t the number of iterations "n" respectively and when n approaches to infinity, they reach the exact upper and lower bounds.

Definition 8: The bound difference between two iterations n and $n-W$, where W is a constant value and $n - W > 0$, is defined as $\Delta[n,W] = |B_{oupp/low}[n] - B_{oupp/low}[n-W]|$.

Corollary 1: The bound difference $\Delta[n,W]$ is a monotonically decreasing function w.r.t number of iterations n and we have $\{\Delta[n,W] \to 0 | n \to \infty\}$.

Fig. 11 illustrates the idea for B_{oupp}. Clearly, $B_{oupp}[n]$ is monotonically increasing as n increases, and the difference of $\Delta[n,W] = |B_{oupp}[n] - B_{oupp}[n-W]|$ is monotonically decreasing w.r.t n. The other parameter W plays an important role for finding B_{oupp} by numerical methods especially in high order filters, where we have several delay lines (registers) that must be filled with data so that new coefficients can impact the final result. W has to be bigger than the order of the filter. Moreover, W represents a window of iterations (from $n-W$ to n), in which B_{oupp} has settled with the maximum change of $\Delta[n,W]$ and as a result higher values of W constitute more accurate results.

The paper [15] gives two heuristic algorithms to perform analysis of range and precision for IIR filters. Here we introduce the algorithm for range analysis described in Fig. 12. The inputs to the algorithm include the feedforward coefficients b, feedback coefficients a, and input bound $B_i = [x_{low}, x_{upp}]$. The W value is set to "100" due to the

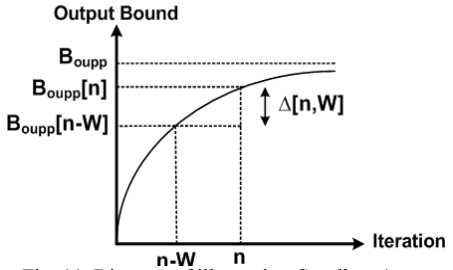

Fig. 11. Diagram of illustrating Corollary 1

explained reasons. Then a loop begins to calculate the individual bound in each iteration (Step 3). A subroutine *Flatten* expands $z[n]$ according to Lemma 1 and computes a new coefficient array $C_n = [c_n, c_{n-1}, \dots, c_1]$ relevant to Eqn. (14), which relies on the Q previous flattened arrays as well as a and b. In Steps 5 and 6, the maximum and minimum values of z at the n^{th} iteration labeled as Z_{max} and Z_{min} are computed. These computations are exact because $z[n]$ is a linear function of the input variables due to Eqn. (14). Step 7 and Step 8 compare the obtained bound to the previous iteration bound, if the current bound is larger, it will be reserved, or else it will be replaced by the previous bound. If the bounds difference defined by $|\Delta[n, W]|$ is limited to "$\varepsilon = 1$", the loop terminates and the range is returned.

IIR_Range (b, a, B_i)
{ /* *Inputs: Feedforward Coefficients* $b = [b_0, b_1, b_2, \dots\ b_P]$
Feedback Coefficients $a = [a_0, a_1, a_2, \dots\ a_Q]$
Input bound $B_i = [x_{low},\ x_{upp}]$ */
1. W=100; ε=1; *//Convergence parameters*
2. n=1;
3. while $(|B_{oupp/low}[n] - B_{oupp/low}[n-W]| \geq \varepsilon)$
4. { C_n = Flatten $(b, a,\ C_{n-1}, \dots, C_{n-Q})$;
// $C_n = [c_n, c_{n-1}, \dots, c_1]$
5. $Z_{max} = \sum_{i=1}^{n} \max(c_i x_{low}, c_i x_{upp})$;
6. $Z_{min} = \sum_{i=1}^{n} \min(c_i x_{low}, c_i x_{upp})$;
// $Z_{max/min}$=max/min value of z at the n^{th} iteration
7. $B_{olow}[n] = \min\{B_{olow}[n-1], Z_{min}\}$;
8. $B_{oupp}[n] = \max\{B_{oupp}[n-1], Z_{max}\}$;
// $B_{oupp/low}[n]$=upper/lower bound after n iterations
9. n++;
}
10. return R_o = ceiling ($B_o[n]$);
}

Fig. 12. A proposed algorithm for computing ranges for IIR filters

VI. Conclusion

Verifying and optimizing arithmetic circuits is always a significant job in high-level synthesis, which impacts area and delay. Past explorations have obvious disadvantages of low efficiency and coarse results. In order to solve the problems, we propose new algorithms based on Arithmetic Transform (AT) which is defined in the spectral domain.

In this paper, we describe the AT definition and features, then a verification algorithm of comparing two similar implementations is proposed. A hybrid method mixing with IA, AA and AT for calculating ranges and allocating IBs is explained, while an algorithm of finding optimized parameters for precision is given. Because some arithmetic circuits include feedbacks such as IIR filters, we propose heuristics-based algorithms to calculate the range and precision. Therefore, our algorithms have capability to handle not only arithmetic circuits represented by polynomials but also any arbitrary arithmetic operations. In the future, we will extend the precision analysis to cover mean square error as well as Signal to Noise Ratio (SNR) for IIR filters.

Reference

[1] O. Sarbishei, B. Alizadeh and M. Fujita, "Polynomial Datapath Optimization Using Partitioning and Compensation Heuristics", *International Design Automation Conference (DAC),* 2009, pp. 931-936.

[2] M. Willems, V. Bürgens, H. Keding, T. Grötker and H. Meyr, "System level fixed-point design based on an interpolative approach," *Proc. Design Autom. Conf.* 1997, pp. 293–298.

[3] A. Gaffar, O. Mencer, W. Luk, and P. Cheung, "Unifying bit-width optimisation for fixed-point and floating-point designs," in *Proc. IEEE Symp. Field-Programmable Custom Comput. Mach*, FCCM 2004, pp. 79–88.

[4] D.-U. Lee, A. Gaffar, R. C. C. Cheung, O. Mencer, W. Luk, G. Constatinides, "Accuracy-Guaranteed Bit-Width Optimization", *IEEE Trans. CAD,* Vol. 25, No. 10, Oct. 2006, pp. 1990 –2000.

[5] W.G. Osborne, J. Coutinho, R. Cheung, W. Luk, O. Mencer, "Instrumented Multi-Stage Word-Length Optimization", *Proc. Field-Programmable Technology*, Dec. 2007, pp. 89 – 96

[6] G. Constantinides, P. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans. on CAD* vol. 22, no. 10, pp. 1432–1442, Oct. 2003.

[7] A. Kinsman and N. Nicolici, "Finite Precision bit-width allocation using SAT-Modulo Theory", *IEEE DATE09*, April 2009 Page(s):1106 – 1111.

[8] Y. Pang, K. Radecka, "Optimizing imprecise fixed-point arithmetic circuits specified by Taylor Series through Arithmetic Transform", *45th ACM/IEEE Design Automation Conference, DAC 2008.E*, June 2008, pp. 397 – 402

[9] K. Radecka and Z. Zilic, "Arithmetic Transforms for Compositions of Sequential and Imprecise Datapaths", *IEEE Trans. CAD*, Vol. 25, No. 7, Jul. 2006, pp. 1382-1391.

[10] Y. Pang, K. Radecka and Z. Zilic, "Verification of Fixed-point Circuits Specified by Taylor Series using Arithmetic Transform", *Proceedings of joint 6th International NEWCAS-TAISA Conference*, Jun. 2008.

[11] Y. Pang, K. Radecka and Z. Zilic, "Optimization of Imprecise Circuits Represented by Taylor Series and Real-Valued Polynomials" *IEEE Transactions on Computer-Aided Design*, 2010, to appear, 15 pages.

[12] P. Yu, K. Radecka and Z. Zilic, "Arithmetic Transforms of Imprecise Datapaths by Taylor Series Conversion", *Proceedings of IEEE Intl. Conference on Electronics, Circuits and Systems, ICECS 2006*, pp. 696 – 699

[13] Milic, L.D.; Lutovac, M.D.; "Design of Multiplierless Elliptic IIR Filters with a Small Quantization Error", Signal Processing, IEEE Transactions on, Volume: 47 , Issue 2, 1999 , Page(s): 469 – 479.

[14] Carletta J., Veillette R., Krach F., Fang Z., "Determining appropriate precisions for signals in fixed-point IIR filters", Design Automation Conference, 2-6 June, 2003. Proceedings, page: 656 – 661.

[15] O. Sarbishei, Y. Pang and K. Radecka; "Analysis of Range and Precision for Fixed-Point Linear Arithmetic Circuits with Feedbacks", 2010 HLDVT, to appear, 8 pages

[16] D. Gomez-Prado, Q. Ren, M. Ciesielski, J. Guillot, E. Boutillon, "Optimizing Data Flow Graphs to Minimize Hardware Implementations", *IEEE DATE09,* April 2009, pp. 117-122.

[17] K. Radecka, Z. Zilic; "Verification by Error Modeling: Using Testing Techniques for Hardware Verification", Kluwer Academic Publishers, 2003.

[18] J. Radecki, Z. Zilic and K. Radecka, "Echo Cancellation in IP Networks", invited presentation at *Proceedings of IEEE International Midwest Symposium on Circuits and Systems*, Tulsa, Aug. 2002.