Accelerating Fixed-Point Simulations Using Width Reconfigurable Hardware Architectures

Keyvan Shahin
Chair of Computer Engineering
Brandenburg University of Technology
Cottbus, Germany
keyvan.shahin@b-tu.de

Michael Huebner
Chair of Computer Engineering
Brandenburg University of Technology
Cottbus, Germany
michael.huebner@b-tu.de

***Abstract*— Digital Signal Processing (DSP) systems can be described using either fixed-point or floating-point for their numeric representations. As the general computers use floating-point representation, software-based simulation tools used for modeling and simulation of the arithmetic operations in DSP systems, use floating-point representation as well. However, synthesizing customized hardware for fixed-point arithmetic operations for FPGAs or ASICs is more efficient compared to their floating-point counterparts. Thus it is necessary to convert the representation of a floating-point simulated algorithm on MATLAB for example, to a fixed-point representation which is more suitable for hardware implementation. While former approaches for this conversion step have always been software-based, like on MATLAB itself, a new approach has been introduced in this work to show the possibility of accelerating it by using hardware width re-configurable designs.**

***Index Terms*—Fixed-point Simulation, DSP Systems, FPGA Acceleration, Reconfigurable Hardware Design**

I. Introduction

The procedure for implementing a DSP functionality on hardware consists of description of the functionality, designing the functionality in an environment with simulation capabilities, conversion of the numeric representations to a format which is suitable for hardware implementation, description of the functionality using a hardware-aware language and using appropriate tools for compiling and synthesizing the design based on the target platform [1]. Fig. 1 demonstrates this general flow.

Most DSP algorithms use floating-point arithmetic because of its development simplicity and its good numerical properties [2]. However, for numerous embedded systems, fixed-point arithmetic is preferred because of its benefits in terms of power consumption, area and architecture latency. Thus, a conversion from floating-point to fixed-point is required before the hardware implementation of the algorithm [1].

When using MATLAB, there are certain toolboxes available for generating the fixed-point representation of the code and running the simulation. For a complicated project with a lot of different inputs, intermediate variables, and complex arithmetic functions, running the fixed-point simulation in cases, can take several weeks to complete.

This article is a milestone for an approach to accelerate the fixed-point simulation of a general DSP algorithm, using an FPGA-based width reconfigurable hardware implementation of the algorithm.

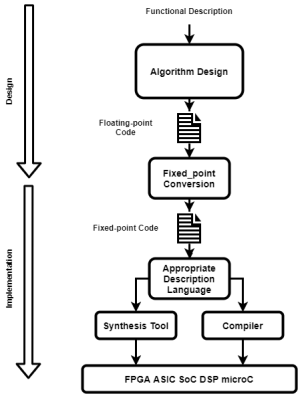


Fig. 1. Digital Signal Processing Hardware Development Flow

II. Floating-Point to Fixed-Point Conversion

The floating-point to fixed-point conversion process is an optimization problem [3] to find the data word-lengths. Searching for a solution to this problem, is a trade-off between the cost (area, power and speed) and the application quality degradation caused by the limited bit-widths of the fixed-point data types. In [4], [5] it is shown that the conversion process can take up to 30% of the total development time which is a motivation of this work, in accelerating this process by using width-reconfigurable hardware implementation of the floating-point to fixed-point conversion.

Generally there are two methods for finding the optimal bitwidths of the intermediate variables, namely analytical and bit-true simulation. Due to the shortcomings of analytical methods specially in dealing with non-LTI systems [6], most recent approaches are based on bit-true simulations.

Our work is a hardware-based conversion, based on a simplified version of the method which was initially introduced in [7]. In this method, the conversion is divided into two main steps, the integer width calculation and the fractional width calculation. The first step tries to find the ranges of the intermediate variables and the second step finds the smallest number of bits which is required for the fractional parts of each

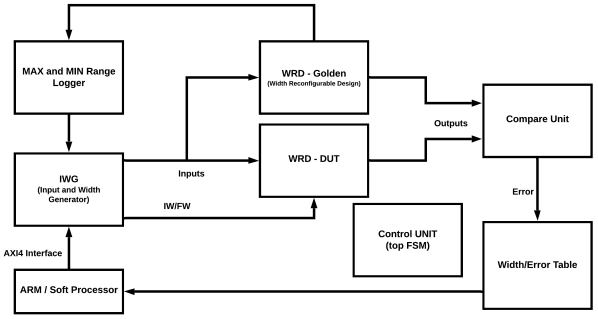


Fig. 2. Hardware-based Fixed-point Simulation Architecture

variable for keeping the resulted error within an acceptable range.

III. Hardware-based conversion

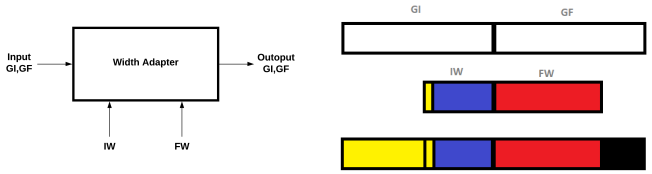
Even thought it is possible to execute the fixed-point conversion in software, in practice, it can be a very time consuming task to run all the possible inputs and observe the output validity and accuracy for each set of widths for intermediate variables. This motivates this work to accelerate the process using hardware-based conversion.

A. Architecture and Flow

Fig. 2 demonstrates the architecture used in our approach which is implemented on an FPGA to preform the fixed-point simulation. This architecture uses two instances of the implemented design. one of these instances has constant maximum width values and is serving as the accurate instance with golden output and the other instance's variables widths can be changed at run time. we call these instances Golden and Design Under Test (DUT). The Input and Width Generation module (IWG) has Finite State Machines (FSM) to configure the DUT with a set of widths, and for each set of widths, it feeds all the inputs to both instances, so that their outputs can be compared with each other. This comparison is done by the Comparison Unit and the results of this comparison is written to a table with the corresponding widths sets and sent to the ARM core on the Zynq FPGA for logging and further analysis in the future. The Min and Max Range Logger keeps the maximum and minimum values of all the variables in the first phase for finding out the dynamic range of each of them. The Control Unit is the FSM responsible for the flow of the simulation. It starts the simulation, controls when the IWG needs to generate the next set of widths or the next set of inputs, when the outputs of WRD instances are valid for error comparison and the flow of the simulation phases to find the variable integer widths or fractional widths.

B. Width Reconfigurable Design

In this work, piece-wise polynomial approximation of the functions were used, to reduce every function to adders and multipliers. A combination of an adder or multiplier with maximum widths and something we call a Width Adaptor



(a) Width Adaptor

(b) Width Manipulation Inside The Width Adaptor

Fig. 3. Width Adaptor and Width Manipulation

(WA) was used to make a width-reconfigurable adder or multiplier. The WA (Fig. 3a) has a data input, a width input and data output. It uses maximum widths for the physical implementation of data input and data output but manipulates the data input, so that the data output has the effective width which is dictated by the width input.

The WA manipulates the data input, by using sign extension and changing the least significant bits to zero. How this trimming is done is depicted in Fig. 3b. By using WAs on the inputs of each adder and multiplier in a piece-wise polynomial implementation of a DSP system, it will be transformed to a WRD, ready for the execution of the fixed-point simulation.

IV. Conclusion and Future Work

In this article we had an overview of the flow in DSP hardware design and the fixed-point simulation step. Considering that previous approaches used software-based fixed-point simulation and its long run-times, the main contribution of this work can be summarized as the introduction of a hardware-based approach, to use a width reconfigurable hardware architecture for accelerating the fixed-point simulation.

While this article describes the flow and the architecture of the modules in this approach, we are now working to implement various examples of mathematical and DSP algorithms and use this approach and generalizing this approach to do the fixed-point simulation on them.

References

[1] Nehmeh, Riham. (2017). Quality Evaluation in Fixed-point Systems with Selective Simulation.

[2] Nehmeh, Riham and Banciu, Andrei and Michel, Thierry and Rocher, Romuald. (2014). A Fast Method for Overflow Effect Analysis in Fixed-point Systems. Conference on Design and Architectures for Signal and Image Processing, DASIP. 2015. 10.1109/DASIP.2014.7115608.

[3] Shi, Changchun and Brodersen, Robert. (2003). An automated floating-point to fixed-point conversion methodology. 2. II - 529. 10.1109/ICASSP.2003.1202420.

[4] Ren, S.-J and An, J.-P and Bu, X.-Y and Xu, Z. and Zhang, X.. (2016). Design of MB-OFDM UWB system frequency synchronization. 36. 1283-1288. 10.15918/j.tbit1001-0645.2016.12.014.

[5] T.Hill. AccelDSP Synthesis Tool Floating-Point to Fixed-Point Conversion of MATLAB Algorithms Targeting FPGAs. White papers, Xilinx, april 2006.

[6] Roy, Sanghamitra and Banerjee, Prith. (2005). An algorithm for trading off quantization error with hardware resources for MATLAB-based FPGA design. Computers, IEEE Transactions on. 54. 886- 896. 10.1109/TC.2005.106.

[7] Roy, Sanghamitra and Banerjee, Prithviraj. (2004). An algorithm for converting floating-point computations to fixed-point in MATLAB based FPGA design. 484-487. 10.1109/DAC.2004.240395.