Automatic Conversion of Floating Point MATLAB Programs into Fixed Point FPGA Based Hardware Design

P. Banerjee
Northwestern University
2145 Sheridan Road, Evanston, IL-60208
banerjee@ece.northwestern.edu

D. Bagchi, M. Haldar, A. Nayak, V. Kim, R. Uribe
AccelChip, Inc.
350 E. Hubbard, Suite 400, Chicago, IL-60600
www.accelchip.com

1. Introduction

Most practical FPGA designs are limited to finite precision signal processing using fixed-point arithmetic because of the cost and complexity of floating point hardware. While mapping DSP applications onto FPGAs, a DSP algorithm designer, who often develops his applications in MATLAB, must determine the dynamic range and desired precision of input, intermediate and output signals in a design implementation to ensure that the algorithm fidelity criteria are met. The first step in a flow to map MATLAB applications into hardware is the conversion of the floating point MATLAB algorithm into a fixed point version using "quantizers" from the Filter Design and Analysis (FDA) Toolbox for MATLAB.

This paper describes how the floating point computations in MATLAB can be automatically converted to a fixed point MATLAB version of specific precision for hardware design. The techniques have been incorporated in the AccelFPGA behavioral synthesis tool [7] that reads in high-level descriptions of DSP applications written in MATLAB, and automatically generates synthesizable RTL models in VHDL or Verilog. Experimental results are reported with the AccelFPGA version 1.5 compiler on a set of five MATLAB benchmarks that are mapped onto the Xilinx Virtex II FPGAs.

2. Related Work

The strategies for solving floating-point computations to fixed-point conversion can be roughly categorized into two groups [6]. The first one is basically an analytical approach coming from those algorithm designers who analyze the finite word length effects due to fixed-point arithmetic [1,2]. The other approach is based on bit-true simulation originating from the hardware designers [3,4,5]. There have been some recent work on automated compiler techniques for conversion of floating point representations to fixed point representations [8,9,10]. This paper describes an efficient approach for converting floating point MATLAB programs into fixed point MATLAB programs automatically within the framework of a commercial behavioral synthesis system called AccelFPGA [7].

3. Algorithms for Auto-quantization

We now describe the algorithm for auto-quantization of MATLAB programs. The overall algorithm consists of the following passes:

1. Scalarization
2. Levelization
3. Computation of Partial Value Ranges of Variables
4. Forward Propagation of Value Ranges of Variables
5. Backward Propagation of Value Ranges of Variables
6. Auto-quantization

Scalarization Pass
The scalarization pass takes a vectorized MATLAB statement and converts it into scalar form using enclosing FOR loops.

Levelization Pass
The levelization pass takes a MATLAB assignment statement consisting of complex expressions on the right hand side and converts it into a set of statements each of which is in a single operator form. This pass operates on both scalar and array operations.

Computation of Partial Value Ranges of Variables
This algorithm takes in a scalarized and levelized MATLAB program with some of the quantizations of input variables specified, and computes the initial value ranges for each variable in the program.

Forward Propagation of Value Ranges
This step takes the partial value ranges of the variables obtained at the previous step and propagates the value ranges in the forward direction using use-def analysis. The algorithm handles simple blocks of assignment statements, conditionals such as IF-THEN-ELSE, and FOR loops.

Backward Propagation of Value Ranges
This step propagates the value ranges of variables in the backward direction. If the user has only specified the quantizers of the output variables, the back propagation pass will propagate the results to the right-hand side of an assignment statement. The algorithm is similar to Forward propagation.

Auto-Quantization
This step assigns values of quantizers based on the value ranges of various variables.

4. Results on Benchmarks

We now report some experimental results on various benchmark MATLAB programs using the AccelFPGA version 1.5 compiler:

- A 16 tap Finite Impulse Response Filter (fir)
- A Decimation in Time FIR filter (dec)
- An Infinite Impulse Response Filter of type DF1 (iir)
- An Interpolation FIR filter (int)
- A 64 point Fast Fourier Transform (fft)

Table 1 shows the characteristics of the five MATLAB benchmarks. We show the lines of MATLAB code, the number of variables and computations for which automatic quantization has been applied, and the distribution of the precisions of the automatic quantizations. For example, the 16 tap FIR filter needs 20 lines of MATLAB code, and the automatic quantization has been applied to 10 variables. The precisions of the quantized variables are [16,0], [16,2], [6,0], [16,8],[16,9], [5,0].

We now report on some results of comparisons of the hardware designs produced by the AccelFPGA 1.5 compiler using the automatic quantizations. We report results of resources and the frequency of the design for a default case of 32 bit quantizations that is specified for all variables using manual quantizations Table 2 shows the results of the manual and automatic quantization for 5 benchmark examples for the Xilinx XC2V250 Virtex2 device.

5. References

1. L. B. Jackson, "On the interaction of roundoff noise and dynamic range in digital filters," *Bell Syst. Tech. J.,* pp. 159-183, Feb. 1970.
2. R. M. Gray, D. L. Neuhoff,, **"Quantization"**, *Information Theory, IEEE Transactions on* , Volume: 44 Issue: 6 , Oct. 1998 pp. 2325 –2383.
3. W. Sung, and K.I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Processing,* vol. 43, no. 12, Dec. 1995.
4. S. Kim, K. I. Kum, and W. Sung, " Fixed-point optimization utility for C and C++ Based digital signal processing programs," *IEEE Trans. Cir. Sys.-II: analog and digital signal* processing, vol. 45, no. 11, Nov, 1998.
5. H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: a fixed-point design and simulation environment" *Design, Automation and Test in Europe, 1998., Proceedings* , pp. 429 –435,1998.
6. C. Shi, "Statistical Method for Floating-point to Fixed point Conversion," M.S. Thesis, Univ. California, Berkeley, EECS Department, 2002.
7. P. Banerjee, M. Haldar, A. Nayak, V. Kim, V. Saxena, R. Anderson, J. Uribe, "AccelFPGA: A DSP Design Tool for Making Area Delay Tradeoffs While Mapping MATLAB Programs onto FPGAs," Proc. International Signal Processing Conference,Apr. 2003.
8. J. Babb, M. Rinard, C. A. Moritz, W. Lee, M. Frank, R. Barua, A. Amarasinghe, "Parallelizing Applications into Silicon," Proc. FCCM Apr. 1999.
9. D. Brooks and M. Martonosi, "Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance, " Proc. HPCA, Jan. 1999.
10. A. Nayak, M. Haldar, A. Choudhary, P. Banerjee, "Precision And Error Analysis Of MATLAB Applications During Automated Hardware Synthesis for FPGAs," Proc. Design Automation and Test in Europe (DATE 2001), Mar. 2001, Berlin, Germany.

Table 1. Characteristics of the MATLAB Benchmarks.

Benchmark	fir	fft	dec	iir	int
MATLAB lines	20	98	38	33	38
Number of auto quantizers	10	42	11	17	11
Auto-Quantizer Values	[16,0], [16,2], [6,0], [16,8],[16,9], [5,0]	[16,15], [16,14], [16,0] [8,0], [7,0], 7,1], [5,0], [4,0] , [3,0], [1,0]	[16,11], [17,15], [8,0], [7,0], [3,0]	[17,15], [17,16], [16,14], [3,0]	[16,11], [17,15], [7,0], [4,0]
AccelFPGA 1.5 Compile Time (sec)	4.0	10.2	8.9	2.7	2.5

Table 2. Comparison of resource usage and performance with the auto-quantization feature.

	Resources				Freq
fir	**LUTS**	**MUX**	**Mult**	**ROMS**	**(MHz)**
32 bit	2938	2837	0	0	78.1
Auto	1185	1108	0	0	83.8
dec					
32 bit	1837	696	4	32	44.3
Auto	975	362	1	17	57.1
iir					
32 bit	1085	835	8	0	58.7
Auto	321	197	2	0	83.5
int					
32 bit	832	318	4	32	49.5
Auto	407	97	1	0	48.8
fft					
32 bit	23227	8770	0	32	29.6
Auto	5700	2628	4	16	39.2