Variable Precision Floating Point Division and Square Root

Miriam Leeser, Xiaojun Wang
Department of Electrical and Computer Engineering
Northeastern University, Boston, MA 02115
mel,xjwang@ece.neu.edu

Division and square root are important operations in many high performance signal processing applications including matrix inversion, vector normalization, least squares lattice filters and Cholesky decomposition. We have implemented floating point division and square root designs for our VHDL variable precision floating point library. These designs are implemented in VHDL and are designed to make efficient use of FPGA hardware.

Both the division [1] and square root [2] algorithms are based on table lookup and Taylor series expansion. These algorithms are particularly well-suited for implementation on an FPGA with embedded RAM and embedded multipliers such as the Altera Stratic and Xilinx Virtex2 devices. The division and square root components have been incorporated into the framework of our variable precision floating-point library.

1 Variable Precision Floating-Point Library

Our parameterized floating-point library is composed of three parts: format control, arithmetic operations, and format conversion. Format control includes modules `denorm` and `rnd_norm`. The first is used for denormalizing (introduction of the implied one bit) and the second is used for rounding and normalizing. Format conversion includes modules `fix2float` and `float2fix`. The first is used for converting from fixed-point representation (both unsigned and signed) to floating-point representation and the second converts in the other direction. Arithmetic operations include modules `fp_add`, `fp_sub` and `fp_mul` for floating-point addition, subtraction and multiplication respectively. We recently added floating-point division (`fp_div`) and floating-point square root (`fp_sqrt`). For both floating-point division and square root, we use the small table-lookup method with small multipliers [1, 2]. These algorithms are both small and elegant. Our result shows that these algorithms are very well suited to FPGA implementations, and lead to a good tradeoff of area and latency. Some features of our library are:

- Our parameterized floating-point library is a superset of all the previously published floating-point formats including IEEE standard format.
- Our library is flexible. It supports the creation of custom format floating-point datapaths, as well as hybrid fixed and floating-point implementations.
- Our library is more complete than all other earlier work with a separate normalization unit, rounding with support for both "round to zero" and "round to nearest", and some error handling features.
- Each component in our library has synchronization signals to aid in the creation of pipelines.

2 Division and Square Root

The division and square root we built are based on previously published algorithms [1, 2]. Both of these algorithms are based on Taylor Series and use both small table-lookups and small multipliers to obtain the first few terms of the Taylor Series. These algorithms are both simple and elegant, and very well suited to FPGA implementations. They are also non-iterative algorithms, unlike other implementations of division and square root based on Newton-Raphson. This allows these components to be easily integrated into a larger pipelined design built with other library modules without decreasing the throughput of the whole design.

Table 1: Cost and Performance for Floating-Point Division

Floating Point Format	8(2,5)	16(4,11)	24(6,17)	32(8,23)
number of slices	69 (1%)	110 (1%)	254 (1%)	335 (2%)
number of BlockRAM	1 (1%)	1 (1%)	1 (1%)	7 (7%)
number of 18x18 embedded multiplier	2 (2%)	2 (2%)	8 (8%)	8 (8%)
clock period (ns)	8	10	9	9
maximum frequency (MHz)	124	96	108	110
number of clock cycles to generate final results	10	10	14	14
latency(ns) = clock $\times$ number of clock cycles	80	105	129	127
throughput (million results per second)	124	96	108	110

Table 1 shows the cost and performance of four different floating-point formats (including IEEE single precision format) for division. Results for square root are similar. All our designs are specified in VHDL and mapped to Xilinx Virtex-II XC2v3000-4 FPGA. All area and timing results in the above tables are those reported by the Xilinx tools. Our results show that both the area and the latency of our floating-point division and square root implementations are small. For IEEE single precision format division, it takes 14 clock cycles to generate final results with a 9ns clock period, so the latency is only 127ns. Since it can be fully pipelined, the throughput is high at 110 million results per second. This design takes only 2% of the slices, 7% of the BlockRAMs, and 8% of the 18x18 embedded multipliers on the FPGA chip, which is a very small design. Our floating-point square root shows the similar good tradeoff of area, latency and throughput.

To demonstrate the division implementation, we are incorporating it into our implementation of the K-means clustering algorithm applied to multispectral satellite images [3] K-means clustering is an iterative algorithm where the total number of clusters is known in advance. The algorithm works as follows. First means are initialized using a hierarchical method. During each iteration, each pixel of the image is assigned to the closest cluster based on the distance between each pixel and each of the K cluster centers. At the end of one iteration, the new mean of each cluster is calculated based on the new pixel assignments and is used for the next iteration as the center of each cluster. To obtain the new mean of each cluster, an accumulator and a counter are associated with each cluster. Once a pixel is assigned to a cluster, the value of the pixel is added to the accumulator and the counter is incremented. The new mean is obtained by dividing the accumulator value by the counter value. In our previous design [3] this mean updating step is done on the host because it requires floating-point division. With our new `fp_div` module, we are able to implement the mean updating in FPGA hardware. This greatly reduces the communication between host and FPGA board and further accelerates the runtime.

References

[1] P. Hung, H. Fahmy, O. Mencer, and M. J. Flynn, "Fast division algorithm with a small lookup table," in *Asilomar Conference on Signals,Systems and Computers*, vol. 2, pp. 1465–1468, November 1999.

[2] M. D. Ercegovac, T. Lang, J.-M. Muller, and A. Tisserand, "Reciprocation, square root, inverse square root, and some elementary functions using small multipliers," *IEEE Transactions on Computers*, vol. 49, pp. 628–637, July 2000.

[3] P. Belanovic and M. Leeser, "A library of parameterized modules for floating-point arithmetic and their use," in *High Performance Embedded Computing*, September 2002.