Floating-Point FPGA: Architecture Performance and Modeling

Shaik Ayesha, B.V. Ramana, K.V. Ramana Rao

Abstract-This paper presents a novel architecture for domain-specific FPGA devices. This architecture can be optimized for both speed and density by exploiting domain-specific information to produce efficient reconfigurable logic with multiple granularity. In the reconfigurable logic, general-purpose fine grained units are used for implementing control logic and bit-oriented operations, while domain-specific coarse-grained units and heterogeneous blocks are used for implementing data paths; the precise amount of each type of resources can be customized to suit specific application domains. Issues and challenges associated with the design flow and the architecture modeling are addressed. Examples of the proposed architecture for speeding up floating point applications are illustrated. We assume that current proposed architecture can achieve 2.5 times improvement in speed and 18 times reduction in area on average, when compared with traditional FPGA devices on selected floating point benchmark circuits.

Keywords: Architecture, field-programmable gate array (FPGA), Floating Point, Application domains, modeling.

I. INTRODUCTION

FPGA technology has been widely adopted to speed up computationally intensive applications. Most current FPGA devices employ an island-style fine-grained architecture, with additional fixed-function heterogeneous blocks such as multipliers and block RAMs; these have been shown to have severe area penalties compared with standard cell ASICs [1]. In this work, we propose domain-specific coarse-grained architectures which can have advantages in speed, density and power over more conventional heterogeneous FPGAs. One key issue associated with such an approach is identifying the correct amount of coarse-grained logic necessary to enhance the performance of an application without adversely affecting area and flexibility. For example, an application that demands high performance floating point computation can potentially achieve better speed and density by introducing dedicated embedded floating point units (FPUs).

However, for those applications which do not have any floating point computations, the FPU resources will be wasted. To address this issue, we advocate domain-specific FPGAs with flexible, parameterized architectures that can be generated to address application sets that are smaller than those targeted by conventional FPGAs, but possibly larger than that of ASICs.

Manuscript received October 30, 2011.

Shaik Ayesha, M.Tech(ECE), Pydah College of Engineering & Technology, India. Email: ayeshashaik2009@gmail.com

B.V. Ramana, M.Tech(ECE), Pydah College of Engineering & Technology, India.

K.V. Ramana Rao, Assoc.Professor & Head, Dept. of ECE, Pydah College of Engineering & Technology, India.

We introduce a hybrid FPGA model in which both fine grained and coarse-grained units are considered important. Given a domain-specific application requirement, a reconfigurable fabric consisting of both types of units is generated, the coarse-grained units being used for the data path and fine-grained units for control and bit-oriented operations. A model is also introduced that allows us to search for the best proportion of each type of fabric, and a method for rapidly evaluating the performance of the architecture is employed.

The key contributions of this paper are:

• A generic hybrid FPGA architecture that supports configurable resources of multiple granularity that can be customized for different applications.

• Use of this architecture to design a domain-specific hybrid FPGA for various floating point computations.

• Demonstration that a single configuration of a floating point specific hybrid FPGA is able to achieve improvements in both speed and area compared with commercial and proposed reconfigurable devices on selected floating point benchmarks.

II. OVERVIEW OF FPGA

Recent years have seen an impressive improvement in the achievable density of integrated circuits. This improvement has led to an increase in the cost and difficulty of designing

and testing a correctly-functioning chip. Stand-alone FPGAs (Field Programmable Gate Arrays) provide one way of reducing the design cost; however, many designs are not suitable for FPGAs because of their speed, density or power requirements. For these types of designs, a fixed-function chip, often designed using standard cells and the System on-Chip (SoC) methodology [2], may be the only option.

Configurability can be provided by embedding one or more programmable logic cores into the fixed-function chip. In such a chip, most of the design is implemented using fixed function ASIC (Application Specific Integrated Circuit) gates, while programmable logic is used sparingly in those parts of the chip that are likely to change. These changes may be due to errors in the design or specification, future upgrades, or to allow for the customization of an integrated circuit for multiple customers. Embedded programmable logic can also provide a mechanism to add debug capability [3].

A programmable logic fabric can either be hard or soft. An ASIC designer using a hard fabric would obtain a layout and embed it directly into the integrated circuit. One challenge with this approach is that design tools that allow seamless integration of fixed and programmable logic are still not mature.

Timing analysis, power distribution, and verification are difficult when the function to be implemented in the core is not known. An alternative technique is recently described which addresses this concern by shifting the burden from the ASIC designer to mature standard-cell synthesis tools [4, 5]. In this technique, an ASIC designer would obtain a synthesizable version of their programmable logic fabric (a soft core) written in a hardware description language, and would synthesize it along with the rest of the ASIC. The primary advantage of this technique is that the task of integrating such cores is far easier than the task of integrating hard cores. The synthesis tools can be the same ones that are used to synthesize the fixed (ASIC) portions of the chip. No modifications to the tools are required, and the flow follows a standard integrated circuit design flow that designers are familiar with. Despite these advantages, there are significant area, speed, and power penalties when using these cores. Previous architectures [14, 15] suffer a 6.4 times overhead compared to a hard programmable logic core. This limits their application to small circuits such as state machines.

In this paper we present a new architecture that is between 6 times and 426 times more area efficient than the best previously reported synthesizable programmable logic core.

Moreover, we show that the new architecture has a density similar to that of a standard full-custom fine-grained FPGA. The density improvement is obtained by using a data path style architecture, optimized for performing computations.

2.1 FPGA Architectures

An FPGA is typically constructed as an array of fine-grained or coarse-grained units. A typical fine-grained unit is a –input LUT, where typically ranges from 4 to 7, and can implement any -input Boolean equation. We call this an LUT-based fabric. Several LUT-based cells can be joined in a hardwired manner to make a cluster. This greatly reduces area and routing resources within the fabric [6]. Heterogeneous functional blocks are found in commercial FPGA devices. For example, a Virtex II device has embedded fixed-function 18-bit multipliers, and a Xilinx Virtex 4 device has embedded DSP units with 18-bit multipliers and 48-bit accumulators. The flexibility of these blocks is limited and it is less common to build a digital system solely using these blocks.

When the blocks are not used, they consume die area without adding to functionality. FPGA fabric can have different levels of granularity. In general, a unit of smaller granularity has more flexibility, but can be less effective in speed, area, and power consumption. Fabrics with different granularity can coexist as evident in many commercial FPGA devices. Most importantly, the aforementioned examples illustrate that FPGA architectures are evolving to be more coarse-grained and application-specific. The proposed architecture in this paper follows this trend, focusing on floating-point computations.

III. MODIFIED FPGA ARCHITECTURE

The first modification embeds floating-point multiply-add units in FPGA, while offering a dramatic reduction in area and improvement in clock rate. The next modifications target a major component of IEEE compliant floating-point computations variable length shifters. The first alternative to lookup tables (LUTs) for implementing the variable length shifters is a coarse-grained approach: embedded variable length shifters in the FPGA fabric. These shifters offer a significant reduction in area with a modest increase in clock rate and are smaller and more general than embedded floating-point units.

3.1 Proposed Method

In order to improve the floating point performance of FPGAs [7], three modifications has to be done in the architecture of FPGA, the three modifications are adding a embedded shifter unit, adding a 4:1 MUX in parallel to the CLB and adding a embedded floating point unit. In this multiplier in the floating point unit is replaced by high speed multiplier. The high speed is achieved through canonic signed digit code (CSD) method. Due to CSD representation the number of non-zero bits is reduced, this results in reduced number of partial products. Due to reduction in partial products accumulation speed will increase, this increases the computation speed and number of adders required for accumulation is reduced since multiplier plays a major role in all computations unit like more area occupation and speed, it is replaced by high speed multiplier to reduce area and to increase the computational speed.

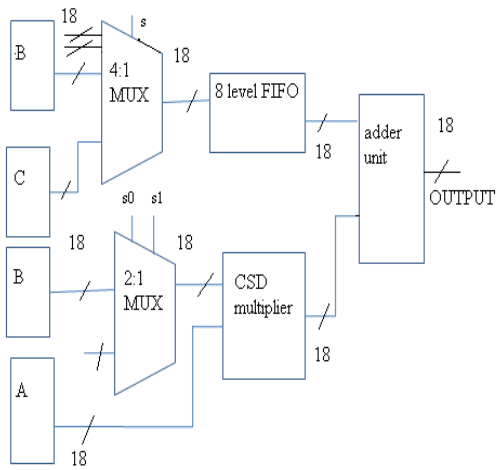

Figure 2.1 floating point unit

IV. MULTIPLICATION OF FLOATING POINT NUMBERS USING CSD

The product of two n-bit binary integers is obtained using CSD multiplication algorithm in n/2-1 steps. This significantly reduces the hardware as compared to other multiplication algorithm. This unit receives two inputs of which one is 24 bits of multiplier the other is 24 bit multiplicand.

1. CSD representation of multiplier is computed.
2. Multiplicand and CSD number of multiplier is given as input.

3. Difference between right most bit having logic value '1' and next bit having value logic value '1'is calculated and based on the resultant value the multiplicand is shifted.
4. If the sign of CSD number is positive it is added with the shifted multiplicand.
5. If the sign of CSD number is negative it is subtracted with the shifted multiplicand.

V. THE BGM ARCHITECTURE

Our design divides the entire MC simulation into three stages: simulation initialization, BGM path generation, and post processing. In the initialization stage, we initialize the volatility vector ~_, reset the Gaussian random number generators and initialize the Brownian motion generator. In the second stage, the BGM paths are generated according to Equation 1. The pseudo code for the main BGM model can be described by:

Step 1: for n = *CurrPeriod* + 1 to N

Step 2: *f actor* = $\_nFn$=(1:0 + $\_nFn$)

Step 3: ~$\_n$ = *f actor* _ ~$\_n$

Step 4: ~$\_n$ = ~$\_n$ + ~$\_n$□1

Step 5: _ = (~$\_n$ _ ~$\_n$)dt + (□□! dW _ ~$\_n$)

Step 6: dFn = _ _ Fn

Step 7: Fn = Fn + dFn

where CurrPeriod is the index of the current standard period, i.e. m(t) = CurrPeriod + 1, and N is the number of standard forward rates.

The for-loop (step 1) is the main loop of the BGM model. The computation consists of one division (step 2), one vector addition (step 4) and three vector product operations (step 3, step 5) in each iteration of the for-loop. We use a Taylor series expansion to implement step 2 and it is discussed in detail in Section.

In order to maximize parallelism, the vector operations are implemented as parallel scalar operations. Finally, we do the post-processing which involves pricing the cap according to Equations 4 and 5 and calculate the mean and standard error of the generated BGM paths on the PowerPC processor.

VI. CONCLUSION

FPGAs provide effective solutions for many coarse-grained applications, such as digital signal processing, encryption, scientific data processing, and others. However, commercial FPGAs are fine-grained, and miss many optimization opportunities.

The use of CSD multiplier reduces the no of adders, no of slices occupied in CLBs and reduced delay. By selecting proper modules of reduced latency, area and delay, significant improvement in speed and area is achieved. Floating point unit is implemented and CSD multiplier is included in it. One modification in FPGAs has to be made, embedding a variable length shifter unit

Current and future research includes further optimization of our design based on techniques such as run-time recon_guration, and extensions of our approach to cover other _nancial models.

REFERENCES

1. I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *Proc. FPGA*. New York, NY, USA: ACM Press, 2006, pp. 21–30.
2. R. Saleh, S. Wilton, S. Mirabbasi, A. Hu, M. Greenstreet, G. Lemieux, P. Pande, C. Grecu, and A. Ivanov. System-on-chip: Reuse and integration. *Proceedings of the IEEE*, 94(6):1050–1069, June 2006.
3. B. Quinton and S. Wilton. Post-silicon debug using programmable logic cores. In *Int. Conf. on Field-Programmable Technology*, pages 241–247, Dec. 2005.
4. S. Wilton, N. Kafafi, J. Wu, K. Bozman, V. Aken'Ova, and R. Saleh. Design considerations for soft embedded programmable logic cores. IEEE Journal of Solid-State Circuits, 40(2):485–497, Feb. 2005.
5. A. Yan and S. Wilton. Product-term based synthesizable embedded programmable logic cores. *IEEE Trans. on VLSI*, 14(5):474–488, May 2006.
6. E. Ahmed and J. Rose, "The effect of LUT and cluster size on deepsubmicron FPGA performance and density," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 3, pp. 288–298, Mar. 2004.
7. Michael J. Beauchamp, Scott Hauck, Keith D.Underwood, and K. Scott Hemmert, (Feb 2008)"Architectural Modifications to Enhance the Floating- Point Performance of FPGAs", IEEE Transactions on Very Large Scale Integration (VLSI)Systems vol. 16. 2.

AUTHORS PROFILE

Shaik Ayesha pursuing her M.Tech(ECE) in Pydah College of Engineering & Technology, under the guidence of K.V Ramana Assoc.Professor & Head, Dept.of ECE at Pydah College of Engineering & Technology, Visakhapatnam, A.P., India. My research Interests are V LSI Design, Digital Signal Processing and Instrumentation.

B. V. Ramana pursuing his M.Tech(ECE) in Pydah College of Engineering & Technology, under the guidence of K.V Ramana Assoc.Professor & Head, Dept.of ECE at Pydah College of Engineering & Technology, Visakhapatnam, A.P., India. My research Interests are VLSI Design and Digital Signal Processing.

Sri K.V. Ramana Rao working as Assoc. Professor & Head, Department of ECE at Pydah College of Engineering & Technology, Visakhapatnam, A.P., India. His research interests are Digital Signal Processing and VLSI Design.