International Journal on Intelligent Electronic Systems, Vol.3, No.1, January 2009

DESIGN AND IMPLEMENTATION OF PARTIALLY RECONFIGURABLE FP-AU

P.S.Surekha[1], R.C.Joshi[2], A.K.Saxena[3]
[1,2,3] Department of Electronics and Computer Engineering
Indian Institute of Technology Roorkee, India.
E-mail:[1]kumarfec@iitr.ernet.in

Abstract

In this paper we present the partial reconfiguration of floating point arithmetic unit that improves the area occupied by floating point arithmetic unit and also makes this unit flexible to operate in a rapidly changing environment. The hardware resources occupied by this unit have been reduced through time-sharing them between modules. Since the FP-AU occupies a significant amount of silicon area in any application due to wide dynamic range, our proposed design shows a very efficient area reduction technique for FP-AU. Partial reconfiguration is the ability of certain Field Programmable Gate Arrays (FPGAs) to reconfigure only selected portions of their programmable hardware while other portions continue to operate undisturbed. A FPGA can be partially reconfigured using a partial bitstream. We can use such a partial bitstream to change the structure of one part of an FPGA design as the rest of the device continues to operate and this reduces the reconfiguration time. The floating point arithmetic unit is modeled in VHDL and synthesized with Xilinx ISE tools. The floating point arithmetic modules are designed for Virtex-2 Pro XC2VP50 FPGA.

Key words: Floating Point Arithmetic Unit (FP-AU), FPGA, Reconfiguration, Partial Reconfiguration.

I. INTRODUCTION

In recent years computer applications have increased in their computational complexity. The industry wide usage of performance benchmarks such as SPECmarks forces processor designers to pay particular attention to implementation of the floating point unit or FPU. Special purpose applications such as digital signal processing, audio processing and many real time applications placed further demands on processors with floating point unit. Unfortunately, the sheer size of these floating-point units makes it difficult to house a large number of units in a single FPGA. So, there is a need for a technique which reduces the resources requirement of FP-AU so that it can be implemented with fewer and/or smaller number of resources. This paper presents the design of partially reconfigurable floating point arithmetic modules which uses less hardware resources.

These partially reconfigurable modules are implemented on FPGA. There FPGAs can be configured to implement complex hardware flexible systems. FPGA reconfiguration [1] typically requires the whole chip to be reprogrammed even for the slightest circuit change and also some systems reconfiguration time adds delay to the application. Dynamic reconfiguration modifies the functionality of the system when it is under operation thus reducing each FPGA configuration to only required circuit elements, a greater amount of circuitry is available for the implementation of parallel structures which can lead to an increased optimal performance. By using partial dynamic reconfiguration, reconfiguration time and hardware resources occupied by floating point arithmetic unit can be reduced through time sharing the hardware resources between modules.

IEEE 754 floating point standard has been chosen for doing partial reconfiguration because of its wide range of real time applications. The IEEE floating point standard makes floating point unit implementation [2] portable and the precision of the results predictable. A variety of different circuit structures can be applied to the same number representations, offering flexibility. The floating point unit algorithm, architecture, and bit-width adaptation offer significant potential for optimization. In this work Virtex-2pro (XC2VP50) FPGA device has been used to design partially reconfigurable floating point arithmetic modules.

This paper presents the partial reconfiguration of floating point modules and is organized in six sections. Section I introduces about the FPGA and its reconfiguration Section II briefly discusses about floating point number system, Section III briefly discusses the Xilinx Virtex II Pro FPGA architecture, section IV discusses analysis of dynamic reconfigurable systems using floating point modules section V gives brief description of implemented FP-AUs using partial reconfiguration, and Section VI gives results and conclusions.

II. FLOATING POINT NUMBER SYSTEM

The FP-AU that we designed handles the operations of IEEE-754 standard single precision format. In this work we have used single precision IEEE 754 floating point format for representing floating point numbers. The bit representation of a single precision floating point format [3] is as follows

Sign-Bit	Exponent = Base $^{\text{exponent + bias}}$	mantissa
1-bit	8-bits	23-bits

This arithmetic operation includes alignment, rounding, normalization of floating point numbers. Alignment includes comparison and shifting operations. The exponents of operands are compared and the significand with smaller exponent is shifted right in order to align two exponents. Rounding is made in order to restrict the resultant significant to 23 bits. If the result exceeds its limits then it is indicated by using overflow bit. Normalization of mantissa of the result is done by shifting it to the left until the high order bit is a one. Adjusting exponent of the result , is done by subtracting it by the number of positions that mantissa was shifted left.

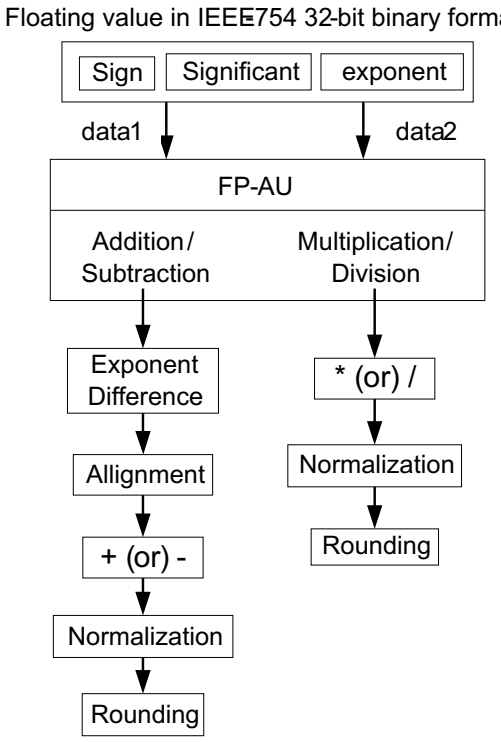

Fig. 1. FP-AU Design Flow

Following are the design steps to perform arithmetic operations on floating point numbers and those are as shown in Fig .1

Addition/Subtraction:

1. Exponent difference
2. Pre-shift for mantissa alignment
3. Mantissa addition/subtraction
4. Post-shift for result normalization
5. Rounding

Multiplication & Division:

1. Mantissa multiplication or division
2. Shifting the result for normalization.
3. Rounding the result

The complete description of the functionality of these modules can be referred from [4,5]. Once the high-level description (VHDL [6]) of these modules is completed, the code goes through simulation for verifying the functionality and synthesis for generating the net-list and optimizes in area, performance or both fits it into the target device.

III. XILINX VIRTEX II PRO FPGA

The basic architecture of any FPGA device consists of an array of logic blocks surrounded by programmable I/O blocks, and connected with a programmable interconnects. The general architecture of Xilinx Virtex-II Pro is shown in Fig. 2. The basic cell of any Virtex FPGA is configurable logic block (CLB). Each CLB contains 4 slices and each slice contains 2 Look Up Tables (LUTs) and 2 D flip-flops. A LUT is used to store in memory the result to every possible input and does not require actual computation, only fetching of the data. In an FPGA, the LUT stores the outcome of any logic operation on 4 inputs (some vendors are migrating to a 6-input LUT, but the concept is the same.) In logic, 4 bits can represent 16 values; any logic operation on these 4 bits will result in one of these values. The D-flip flop inputs can be driven by the LUTs with in a slice or directly from slice inputs, bypassing the LUTs.The Virtex FPGA uses Block select RAM memory blocks that are organized into columns. It contains programmable input/output blocks (IOBs) interconnected to the CLBs by fast, versatile routing resources. The availability of these routing resources permits the Virtex family to accommodate large, complex designs. The complete Xilinx Virtex-II pro resources list and its functionality can be found from [7].

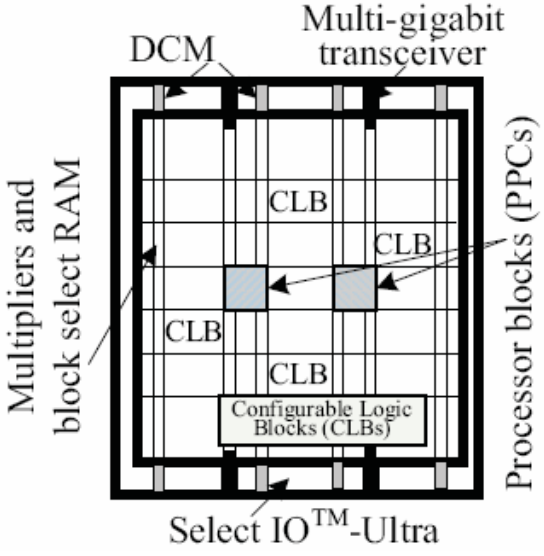

Fig. 2. Xilinx Virtex-II Pro Device Architecture

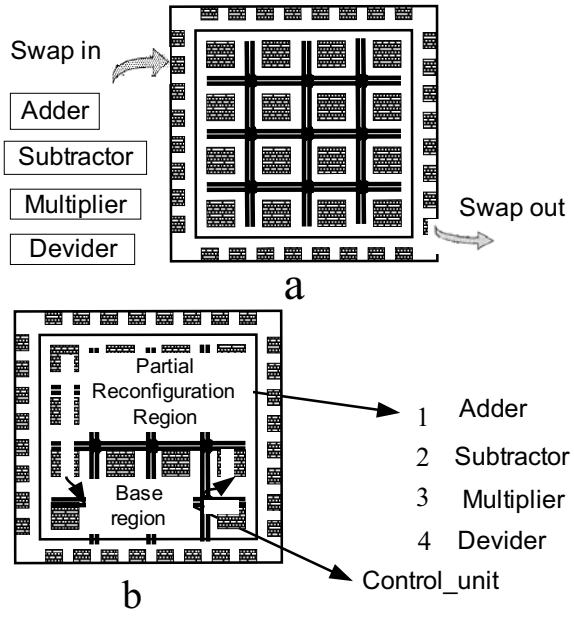

Fig. 3. (a, b): Run-time and Partial reconfigurations of floating point modules

IV. DYNAMIC RECONFIGURABLE SYSTEM ANALYSIS

One way to further exploit the reconfigurable resources of FPGAs and increase functional density is to reconfigure them during system operation. This process is referred to as Run-Time Reconfiguration (RTR). RTR is an approach to system implementation that divides an application or algorithm into time-exclusive operations that are implemented as separate configurations. One type of RTR is Partial Reconfiguration. Partial reconfiguration [8] is useful for systems with multiple functions that can time-share the same FPGA device resources. In such systems, one section of the FPGA continues to operate, while other sections of the FPGA are disabled and partially reconfigured to provide new functionality. This is analogous to the situation where a microprocessor manages context switching between software processes. Except in the case of partial reconfiguration of an FPGA, it is the hardware not the software that is being switched. Partial reconfiguration provides an advantage over multiple full bitstreams in applications that require continuous operation not otherwise accessible during full reconfiguration. By using partial reconfiguration, designers can dramatically increase the functionality of a single FPGA, allowing a system to be implemented with fewer and smaller devices than is otherwise required. The analysis of run-time reconfiguration and partial run-time reconfigurations applied to floating point modules is as shown in Fig.3.

In run-time reconfiguration individual designed modules are downloaded on needed basis by swap out the existing modules configuration data and swap in with needed module configuration data. It means in some applications it is necessary to run one particular module by stoping the functionality of already running module. In our application we have designed floating point modules which are partially reconfigurable. In partial reconfiguration the base module always runs and partially reconfigurable modules are interchanged depending on requirement. So in our design PCI-interface will always run and partial reconfigurable modules adder/subtractor, multiplier and divider are loaded on to the FPGA depending on requirement. The PCI-interface logic works as a control unit for these modules. The hardware modules and assigned area constraints are shown in Table-I.

V. PARTIAL RECONFIGURATION METHODOLOGY

There are two ways of partial reconfiguration one is module based and another is difference based partial reconfiguration. Further details refer [9]. Floating point modules are partially reconfigured using Command Line. In this we have used module based partial reconfiguration approach i.e., if we need to change functionality of a PR module then we have to replace the entire running module bitstream with another module bitstream. In our approach we have divided entire design into three parts Top level Design, Base design and partially reconfigurable modules. Top level design includes all modules, bus macros and global signals black box instantiations. Base design includes static logic that is PCI interface logic to give inputs to reconfigurable modules from files and to receive output from reconfigurable modules and write them on files. Partial reconfigurable modules adder/subtractor, multiplier and divider are swapped in or swapped out depending on application into the partial reconfigurable region without stopping the functionality of base region i.e. PCI interface. We have implemented all these modules separately and then finally combined them using merging process. It is important to note that we have to mention area group ranges with area constraints [10] and modes with reconfigurable regions are assigned with proper values in the UCF file.

VI. RESULTS & CONCLUSIONS

The floating point arithmetic modules are designed in VHDL and pre-synthesis simulation has been done with simulation tool ModelSim 6.0d. After confirming the functional verification, hardware resources estimation has been carried out with XILINX 9.2i (ISE) and modules has been designed for command line partial reconfiguration for XUP development kit based on Virtex-II Pro (XC2VP-50) FPGA. The post-synthesis simulation also carried out to ensure correct operation. The resource utilization summary generated from synthesis tool has been given in Table-2. The results we obtained after implementing the floating

point modules on the Virtex II-Pro Device are given in Table-III. Input is given in hex format through files and output is extracted into file in hex format.

Table-I Hardware Modules

Name of the module	Input1	Input2	Output
Addition-Subtraction	12121231 98571234 091a0470	31310016 02091a04 abcdef10	3158800b 986b8919 abe6f787
Multiplication	12121231 98571234 091a0470	31310016 02091a04 abcdef10	03c9fd40 ffe65d32 fff7cac2
Division	12121231 98571234 091a0470	31310016 02091a04 abcdef10	6c534426 e980000 e8bf7630

The presimulation results for partially reconfigurable modules are also given in Fig. 4, Fig. 5 and Fig. 6 respectively. From Table-II we can say that for implementing FP-AU on FPGA instead of using approximately 2500 slices we can get the same performance with approximately 1600 slices added with this is the reduced reconfiguration time. From this we concluded that by optimizing the designed modules in area and performance through automated tools and through runtime reconfiguration and partial reconfiguration methods to reuse hardware and by evaluating partial reconfiguration [11] techniques one can achieve high level of flexibility in hardware without compromising in high performance.

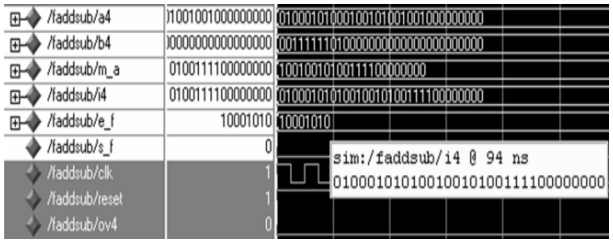

Fig. 4. Addition and Subtraction

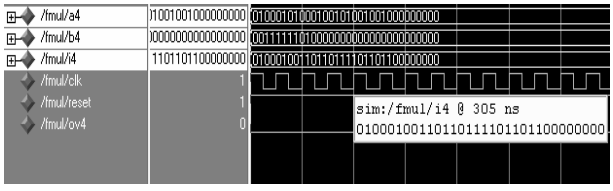

Fig. 5. Multiplication

Fig. 6. Division

Table-II Hardware Modules

Module	Function	Area Constraint
Adder-Subtractor	Performs 32-bit addition/subtraction	Reconfigurable
Multiplier	Performs 32-bit Multiplication	Reconfigurable
Divider	Performs 32-bit division operation	Reconfigurable
Base	Contains Base design in this we have included PCI interface logic	Static

Table-III: Resource utilization Summary

Name of the module	No. of slices (23616)	No. of LUTS (47232)	No. of bonded IOBs (692)	No. of GCLKs	No. of Partitioned Slices
Addition-Subtraction	98	172	50	1	1300
Multiplication	975	610	50	1	
Division	1119	1815	52	1	
PCI-interface	237	133	34	1	300

REFERENCES

[1] Katherine Compton, Scott Hauck, "Reconfigurable computing: a survey of systems and software," ACM computing Surveys, vol. 34,pp. 171 - 210, June 2002.

[2] Gerardo Leyva, Gabriel Caffarena, Carlos Carreras, Octavio Nieto-Taladriz Dpto, "A Generator of High-speed Floating-point Modules," IEEE Symposium on Field-Programmable Custom Computing Machines pp.306-307,2004.

[3] Jian Liang; Tessier, R.; Mencer, O, "Floating point unit generation and evaluation for FPGAs ," in Proceedings of 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines,9-11 April 2003, pp:185 – 194.

[4] Jean-Pierre, Deschamps, Gery, Jean Antoine Bioul, Gustavo D.Sutter, "Synthesis of arithmetic circuits FPGA ASIC and Embedded Systems," John Wiley and Sons publishers 2006.

[5] S. F. Anderson, J. G. Earle, R. E. Goldschmidt, D. M. Powers, "The IBM System/360 Model 91: Floating-point Execution Unit," IBM tech International Journals [online] http://domino.research.ibm.com/tchjr/.

[6] Volnei A.Pedroni, “Circuit Design with VHDL," MIT Press, 2004.

[7] Xilinx Virtex II Pro Complete data sheet http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf

[8] Xilinx website [online] http://www.xilinx.com/publications/xcellonline/xcell_55/xc_prmethod55.htm.

[9] Two Flows for Partial Reconfiguration: Module Based or Difference Based, XAPP290 (v1.2), Xilinx Inc., Sept 9,2004.

[10] Xilinx Early Access Partial Reconfiguration User Guide [online] http://www.xilinx.com/support/documentation/user-guides/ug208.pdf.

[11] Ross Hymel, Alan D.George and Herman Lam, “Evaluating Partial Reconfiguration for Embedded FPGA Applications,” NSF Center for High-Performance Reconfigurable Computing, University of Florida, 27 July 2007.

Dr. Ramesh C. Joshi received the B.E. degree in electrical engineering from Allahabad University in 1967, M.E. and Ph.D. degrees in Electronics and Computer Engineering from University of Roorkee (Now, IIT Roorkee) in 1970 and 1980, respectively. He is currently working as professor in Department of Electronics and Computer Engineering at Indian Institute of Technology Roorkee, India. He has received a Gold Medal by Institute of Engineers in 1978 for best research paper. He has published about 150 research papers in National/International International Journal/Conferences and delivered about 20 special lectures in various US and Indian Universities and Organizations. His main research interests are in Parallel & Distributed Processing, Databases and VLSI Design.