Generic High Performance Multimode Floating Point Unit for FPGAS

Arunkumar.P¹, Bharanidharan.K², Sampurna.K³, Sharmila Devi.K⁴

¹ME Scholar, ECE Department, CEG Campus, Anna University, Chennai, Tamilnadu, India.
²Lecturer, ECE Department, CEG Campus, Anna University, Chennai, Tamilnadu, India.
³ME Scholar, ECE Department, Arasu Engineering College, Kumbakonam, Tamilnadu, India.
⁴ME Scholar, ECE Department, Arasu Engineering College, Kumbakonam, Tamilnadu, India.

Abstract - Floating point unit form an important unit in the digital circuit for the accuracy of the computations. The advent function in the very large scale integration (VLSI) circuit technology in many complex circuits, which is unbelievable in the past, but now become easy to realize. In this thesis the floating point unit based on IEEE standard for floating point number has been implemented on the FPGA board. Here the generic high performance embedded floating point for FPGAs can be configured to perform a wide range of operations. The embedded floating point unit implemented has a 32-bit processing unit which can allow various floating point operations. The floating point adder and multiplier in the embedded floating point unit can be configured to perform one double precision operation or two single precision operations in parallel.

Keyword - Floating point, Floating point unit(FPU), Embedded floating point unit, Precision.

I. INTRODUCTION

Floating point units are very important in the digital systems for the scientific computations for their numerical stability. This floating point unit is also known as the math coprocessor which is the part of the digital circuits, specially designed to carry out operations on the floating point numbers. With the remarkable progress in the very large scale integration (VLSI) circuit technology, many complex circuits which is unbelievable in the past days have become easily realizable in the present days. Algorithms that are seemed to implement now have attractive implement possibilities for the future. This means that these floating point units are suitable for both conventional and unconventional new enhancement designs. The notation of real numbers in math is convenient for the hand computation and formula manipulations. However, real numbers are not well suited for general purpose computation, because their numeric representation are string of digits which is floating can be very long or even infinitely long. Examples are π, e, and square root(√).

In practice, computers store numbers with finite precision. Numbers and arithmetic used in the scientific computation should meet a few general criteria:

1. Numbers should have modest storage requirements
2. Arithmetic operations should be efficient to carry out
3. A level of standardization, or probability, is desirable-results obtained on one computer should closely match the result of the same computation on other computers.

To implement this floating point unit in VLSI, FPGAs are choose since it is used to selected for high clocking frequency and with the unprecedented logic density increases and a host to other features such as embedded processors, DSP blocks and high speed serial at ever lower price points. FPGAs are programmable which is ideal fit for advanced software, configurable and ready to use IP cores. In this thesis the floating point unit based on IEEE standard for floating point number has been implemented on the Virtex-4 FPGA board. Here the generic high performance embedded floating point for FPGAs can be configured to perform a wide range of operations. The embedded floating point unit implemented has a 32-bit processing unit which can allow various floating point operations.

The floating point adder and multiplier in the embedded floating point unit can be configured to perform one double precision operation or two single
Precision operations in parallel. Each can be selected by a particular operation code. Synthesis of the FPU for the FPGA has been done using Xilinx ISE.

Implementation of the multimode floating point unit is the part of digital circuits designed to carry out operations of more floating numbers in parallel. Some systems (particularly older, microcode-based architectures) can also perform various transcendental functions such as exponential or trigonometric calculations, though in modern processors these are done with software library routines. In most modern general purpose architectures, one or more FPUs are integrated with the CPU; however many embedded processors, especially older designs, do not have hardware support for floating point operations.

II. FPGA DESCRIPTION

2.1 FPGA DESCRIPTION

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.

The term field programmable represent that can be program in the field (i.e) modifying the device function in the laboratory or at the sites where the device is installed. FPGA is well known for inexpensive, easy realization of logic networks in hardware as well as software in the VLSI circuits.

2.2 FPGA FLOATING POINT

The FPGA can be thought of as a reconfigurable co-processor. The chip consists of an array of look up tables (LUT), flip-flops (FF), and digital signal processing (DSP) block that all can be reprogrammed on the order of milliseconds. To use FPGAs to accelerate an application, the programmer must first implement a design for the chip. The microprocessor can then call the FPGA loaded with this design to accelerate the application.

For the best performance of the FPGA there is the need of math library and pass pointers. Generally the function would be performed on the microprocessor, reading and writing to the microprocessor’s memory. The microprocessor would initiate a direct memory access (DMA) transfer, and move the data to the memory associated with the attached FPGA, or directly to the memory located within the FPGA. Obviously the transfer times between the microprocessor and the FPGA can greatly affect the performance, but the microprocessor comparison considers the FPGA’s capabilities itself.

When a microprocessor’s peak performance is quoted, it is usually calculated by the number of 32-bit floating point operations it can perform clock, multiplied by the clock frequency of the chip. In the new world of the multi-cores processors, this calculation has been expanded by multiplying that result by the number of cores of the chip. An FPGA has neither floating-point adders nor multipliers, only generic logic that can be configured any way the user would like. So to get an equivalent type of 32-bit floating point performance, there is a need to figure out how many add and multiply function units will fit on an FPGA and at what clock frequency that design might run.

III. IEEE 754 STANDARD FOR BINARY FLOATING POINT UNIT

The IEEE (Institute of Electrical and Electronics Engineers) has produced a Standard to define floating-point representation and arithmetic. The standard brought out by the IEEE come to be known as IEEE 754. The IEEE 754 Standard for Floating-Point Arithmetic is the most widely-used standard for floating-point computation, and is followed by many hardware (CPU and FPU) and software implementations. Many computer languages allow or require that some or all arithmetic be carried out using IEEE 754 formats and operations.

The standard specifies:

1. Basic and extended floating-point number formats
2. Add, subtract, multiply, divide, square root, remainder, and compare operations
3. Conversions between integer and floating-point formats
4. Conversions between different floating-point formats
5. Conversions between basic format floating-point numbers and decimal strings
6. Floating-point exceptions and their handling, including non numbers

3.1 FORMATS

There are three binary floating-point formats (which can be encoded using 32, 64, or 128 bits) and two decimal floating-point formats (which can be encoded using 64 or 128 bits). The first two binary formats are the ‘Single Precision’ and ‘Double Precision’ formats of IEEE 754-1985, and the third is often called ‘quad’; the decimal formats are similarly often called ‘double’ and ‘quad’.

3.1.1 SINGLE-PRECISION FLOATING POINT FORMAT

The IEEE 754 standard specifies a binary32 as having:

<table>
<thead>
<tr>
<th>Sign bit: 1 bit</th>
<th>Exponent width: 8 bits</th>
<th>Significand precision: 24 bits</th>
</tr>
</thead>
</table>

Exponent Mantissa

IEEE 754 Single Precision Formats

Exponent is either an 8 bit signed integer from –128 to 127 (2’s Complement) or an 8 bit unsigned integer from 0 to 255 which is the accepted biased form in IEEE 754 binary32 definition. For this case an exponent value of 127 represents the actual zero. The true significand includes 23 fraction bits to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1 unless the exponent is stored with all zeros.

3.1.2 DOUBLE-PRECISION FLOATING-POINT FORMAT:

The IEEE 754 standard specifies a binary64 as having:

<table>
<thead>
<tr>
<th>Sign bit: 1 bit</th>
<th>Exponent width: 11 bits</th>
<th>Significand precision: 53 bits</th>
</tr>
</thead>
</table>

IEEE Floating Point Representation

<table>
<thead>
<tr>
<th>S</th>
<th>Exponent</th>
<th>Mantissa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>8 bits</td>
<td>23 bits</td>
</tr>
</tbody>
</table>

IEEE Double Precision Floating Point Representation

<table>
<thead>
<tr>
<th>S</th>
<th>Exponent</th>
<th>Mantissa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>11 bits</td>
<td>52 bits</td>
</tr>
</tbody>
</table>

Fig 2.2: Double precision format

This gives from 15–17 significant decimal digits precision. If a decimal string with at most 15 significant digits is converted to IEEE 754 double precision representation and then converted back to a string with the same number of significant digits, then the final string should match the original; and if an IEEE 754 double precision is converted to a decimal string with at least 17 significant digits and then converted back to double, then the final number must match the original.

IV. FPU GENERATION

With gate counts approaching ten million gates, FPGAs are quickly becoming suitable for major floating point computations. However, to date, few comprehensive tools that allow for floating point unit trade-offs have been developed. Most commercial and academic floating point libraries provide only a small number of floating point modules with fixed parameters of bit-width, area, and speed.

The VLSI design community has developed a variety of floating point algorithms, architectures, and pipelining approaches. With modification, these techniques can be applied to FPGAs. To better evaluate the floating point unit design space on FPGAs, we have developed a floating point unit generator. Three trade-off levels can be explored: the architectural level, the floating point algorithm level, and the floating point representation level.
The above figure shows the structure of the embedded FPU block. Optional registers are available at the inputs and outputs of the FPU block to allow for easy implementation of pipelined or multicycle circuits. To increase the usefulness of the floating-point units, several key integer components within the floating-point units were made accessible. The multimode embedded FPU was designed to include a dual-precision floating-point multiplier and a dual-precision floating-point adder.

VI. SIMULATION RESULT

The software requirement for the multimode floating point unit is Xilinx Virtex 4. Combining Advanced Silicon Modular Block (ASMBL™) architecture with a wide variety of flexible features, the Virtex-4 family from Xilinx greatly enhances programmable logic design capabilities, making it a powerful alternative to ASIC technology. The basic Virtex-4 FPGA building blocks are enhancements of those found in the popular Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X product families, so previous-generation designs are upward compatible. Virtex-4 devices are produced on a state-of-the-art 90 nm copper process using 300 mm (12-inch) wafer technology. The embedded floating point explained in chapter three is coded in VHDL and simulated using Modelsim simulator. The simulation results are shown below.
6.1 HARDWARE IMPLEMENTATION

The proposed design implemented on ALTERA – QUARTUS DE2 based kit. The total power consumption of the proposed design based on this device has been calculated and it can be observed from that proposed design.

VII. CONCLUSION

Generic high performance multimode floating point unit is designed to perform two single precision and one double precision floating point arithmetic operations in parallel. IEEE 754 standard based floating point representation has been used. This unit has been coded in VHDL. Code has been implemented in Xilinx Virtex 4 which minimizes the delay and improve the accuracy with high performance. The designed multimode floating point unit operates on 32-bit operands. It has to be implemented in the Virtex 4 FPGA board using the Xilinx ISE. It can be designed for 64-bit operand to enhance precision.

REFERENCES

[1] Chun Hok Ho, Chi Wai Yu, Philip Leong, Wayne Luk, Steven J.E. Wilton “Floating Point FPGA: Architecture and Modelling” in the UK EPSRC (grant EP/C549481/1 and grant EP/D060567/1)


[5] Michael J. Beauchamp Scott Hauck, University of Washington and Keith D.Underwood K. Scoot Hemmert, Sandia National University —Embedded Floating Point Units in FPGA in International Conference on FPGAs at California, USA.

[6] Per Karlstrom, Andreas Ehliar, Dake Liu of Linkping University, —High Performance, low latency FPGA based Floating point Units in Virtex 4” in International conference on FPGAs.