IJSTE - International Journal of Science Technology & Engineering | Volume 2 | Issue 10 | April 2016
ISSN (online): 2349-784X

Design of Floating Point Arithmetic Unit using VHDL

Amal Kurian Mathew
B. Tech Student
Department of Electronics and Communication Engineering
Saintgits College of Engineering, Kottayam, Kerala

Amitha Saleem
B. Tech Student
Department of Electronics and Communication Engineering
Saintgits College of Engineering, Kottayam, Kerala

Priya Susan Mathew
B. Tech Student
Department of Electronics and Communication Engineering
Saintgits College of Engineering, Kottayam, Kerala

Stephanie Ann Kuruvilla
B. Tech Student
Department of Electronics and Communication Engineering
Saintgits College of Engineering, Kottayam, Kerala

Dhanusha P. B
Assistant Professor
Department of Electronics and Communication Engineering
Saintgits College of Engineering, Kottayam, Kerala

Abstract

The main objective of project is to design and verify different operations of floating point arithmetic unit (FPAU). We have designed a 64-bit AU which accepts two floating point 64 bits numbers and the code corresponding to the operation which it has to perform from the user. The AU performs the desired operation and generates the result accordingly. A pipeline floating point arithmetic unit (AU) design using very high speed hardware description language (VHDL) is introduced. The novelty of the AU is it gives high performance through the pipelining concept. Pipelining is a technique where multiple instruction executions are overlapped. In the top-down design approach, four arithmetic modules: addition, subtraction, multiplication, and division: are combined to form the floating-point AU. Each module is divided into smaller modules. Two bits selection determines which operation takes place at a particular time. The pipeline modules are independent of each other. All the modules in the AU design are realized using VHDL. Design functionalities are validated through simulation and compilation. After the coding was done, the synthesis of the code was performed using Xilinx-ISE. Test vectors are created to verify the outputs as opposed to the calculated results. Successful implementation of pipelining in floating point AU using VHDL fulfills the needs for different high-performance applications.

Keywords: Addition algorithm, ALU, Division algorithm, FPAU, Multiplication algorithm, Subtraction algorithm

I. INTRODUCTION

Floating-point numbers are widely adopted in many applications due to its dynamic representation capabilities. Floating-point representation is able to retain its resolution and accuracy compared to fixed-point representations. Based on this standard, floating-point representation for digital systems should be platform-independent and data are interchanged freely among different digital systems. ALU is a block in a microprocessor that handles arithmetic operations. It always performs computation of floating-point operations. Some CPUs such as AMD Athlon have more than one floating point unit that handles floating point operations. The use of VHDL for modeling is especially appealing since it provides a formal description of the system and allows the use of specific description styles to cover the different abstraction levels (architectural, register transfer and logic level) employed in the design. In the computation of method, the problem is first divided into small pieces; each can be seen as a submodule in VHDL. Pipelining is one of the popular methods to realize high performance computing platform. Implementing pipelining requires various phases of floating-point operations be separated and be pipelined into sequential stages. This increases the throughput and efficiency of the overall operation. Hence to realize an ALU design, this research proposes a pipeline floating point ALU design using VHDL to ease the description, verification, simulation and hardware realization. VHDL is a widely adopted standard and has numerous capabilities that are suited for designs of this sort.

II. ARITHMETIC LOGIC UNIT

An arithmetic logic unit (ALU) is a digital electronic circuit that performs arithmetic and bitwise logical operations on integer binary numbers. This is in contrast to a floating-point unit (FPU), which operates on floating point numbers. An ALU is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs). A single CPU, FPU or GPU may contain multiple ALUs.

The inputs to an ALU are the data to be operated on, called operands, and a code indicating the operation to be performed; the ALU's output is the result of the performed operation. In many designs, the ALU also exchanges additional information with a status register, which relates to the result of the current or previous operations.

A. Signals

An ALU has a variety of input and output nets, which are the shared electrical connections used to convey digital signals between the ALU and external circuitry. When an ALU is operating, external circuits apply signals to the ALU inputs and, in response, the ALU produces and conveys signals to external circuitry via its outputs.

B. Data

A basic ALU has three parallel data buses consisting of two input operands (*A* and *B*) and a result output (*Y*). Each data bus is a group of signals that conveys one binary integer number. Typically, the A, B and Y bus widths (the number of signals comprising each bus) are identical and match the native word size of the encapsulating CPU (or other processor).

C. Opcode

The opcode input is a parallel bus that conveys to the ALU an operation selection code, which is an enumerated value that specifies the desired arithmetic or logic operation to be performed by the ALU.

D. Status

The status outputs are various individual signals that convey supplemental information about the result of an ALU operation. These outputs are usually stored in registers so they can be used in future ALU operations or for controlling conditional branching.

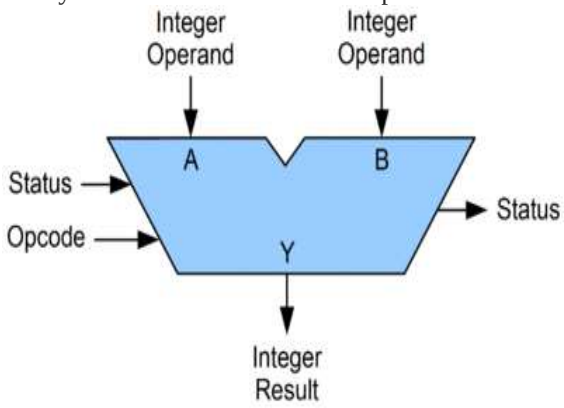

Fig. 1: a symbolic representation of an alu and its input and output signals, indicated by arrows pointing into or out of the ALU, respectively. Each arrow represents one or more signals.

E. Limitation Of Alu

It operates only on integer values, hence less precise.
When forced to be operated on floating point numbers, then it results in slow operations and lags in accuracy.

III. FLOATING POINT AU

A floating-point unit (FPU, colloquially a math coprocessor) is a part of a computer system specially designed to carry out operations on floating point numbers. Typical operations are addition, subtraction, multiplication, division, square root, and bit shifting

Single-precision floating-point format is a computer number format that occupies 4 bytes (32 bits) in computer memory and represents a wide dynamic range of values by using a floating point. In IEEE 754-2008 the 32-bit base 2 format is officially referred to as binary32. It was called single in IEEE 754-1985.

Double-precision floating-point format is a computer number format which occupies 8 bytes (64 bits) in computer memory and represents a wide, dynamic range of values by using a floating point. Computers with 32-bit storage locations use two memory locations to store a 64-bit double-precision number; each storage location holds a single-precision number.

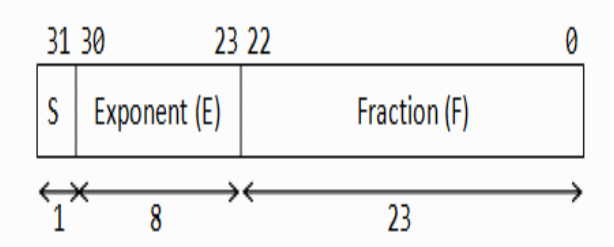

Fig. 2: 32-bit single precision floating point number.

IV. BLOCK DIAGRAM

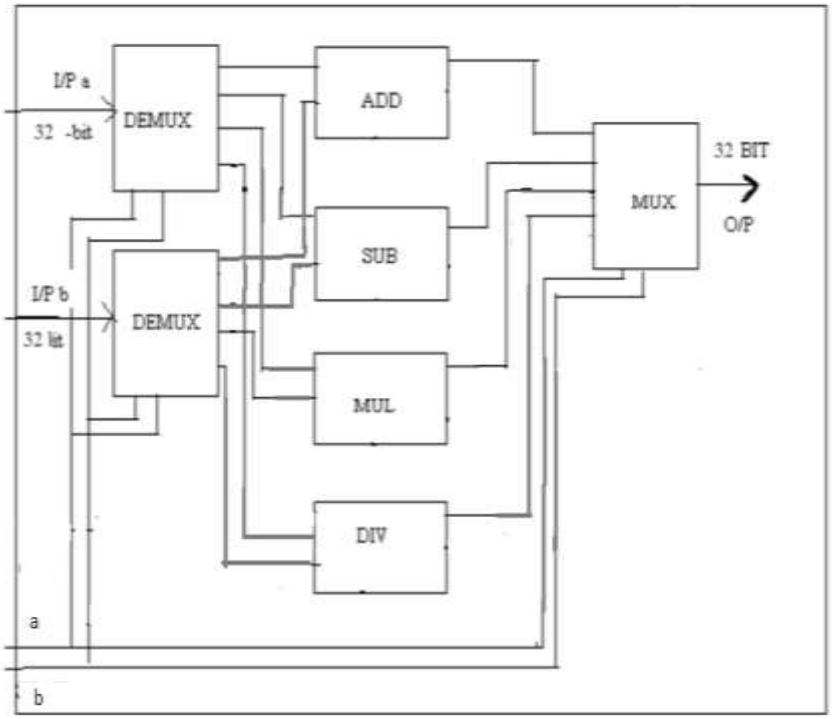

Fig. 3: Block diagram of FPAU.

There are four modules in FPAU, addition, subtraction, multiplication and division. The algorithms for each are explained below.

A. Addition Algorithm

1) Take 2 floating point numbers.
2) Convert them to binary.
3) Normalize them.
4) Floating point is represented as S (sign bit-1), Exponent (*8 bit) and significant (23 bits).
5) 5.127 is added to each exponent number.
6) The exponent is converted to binary and the smallest binary exponent is subtracted from the largest and is taken as ‘d’.
7) The number which is the smaller is shifted to right d times.
8) It is then added to the larger number and its exponent is equal to the larger exponent.

B. Subtraction Algorithm

1) Add 127 to each exponents and find difference between them.
2) Shift the smallest significant according to difference of exponent to the right.
3) Find the twos compliment of largest significant.
4) Add the new significant.
5) Normalize it.
6) Results sign bit will sign bit of greatest significant.

C. *Multiplication Algorithm*

1) XOR the sign bits
2) Add exponents and subtract 127 from them
3) Multiply the significant
4) Normalize the result

D. *Division Algorithm*

1) XOR the sign bits
2) subtract exponents and add 127 from them
3) Divide the significant
4) Normalize the result

V. CONCLUSION

Design and implementation of floating point ALU is used instead of basic ALU because of its capability to deal with floating numbers. Also we have chosen VHDL for simulation and analysis due to its various advantages. We have studied about FPU, VHDL and about Xilinx software. Then we studied the algorithm of each operation such as addition, subtraction, multiplication and division and wrote its corresponding VHDL code. And later we used Xilinx to simulate it.

REFERENCES

[1] Akter, M., M.B.I. Reaz, F.M. Yasin, F. Choong, 2008. Hardware Implementations of Image Compressor for Mobile Communications. Journal of Communications Technology and Electronics, 53(8): 899-910.
[2] Alvarez, J., O. Lopez, F.D. Freijedo and J. Doval-Gandoy, 2011. Digital Parameterizable VHDL Module for Multilevel Multiphase Space Vector PWM. IEEE Transactions on Industrial Electronics, 58(9): 3946-3957.
[3] AMD Athlon Processor technical brief, 1999. Advanced Micro Devices Inc., Publication no. 22054, Rev. D. ANSI/IEEE Std 754-1985, 1985. IEEE Standard for Binary Floating-Point Arithmetic, IEEE, New York.
[4] Arnold, M.G. and S. Collange, 2011. A Real/Complex Logarithmic Number System ALU. IEEE Transactions on Computer, 60(2): 202-213.
[5] Chen, S., B. Mulgrew and P.M. Grant, 1993. A Clustering Technique for Digital Communications Channel Equalization Using Radial Basis Function Networks. IEEE Transactions on Neural Networks, 4: 570-578.
[6] Daumas, M., C. Finot, 1999. Division of Floating Point Expansions with an Application to the Computation of a Determinant. Journal of Universal Computer Science, 5(6): 323-338.