Design and Implementation of a Floating Point ALU on a STRATIX-III FPGA

Prashanth B.U.V
Asst. Professor,
Dept. of ECE,
Vardhaman College of Engineering,
Hyderabad- 501218(A.P),
INDIA

C. Padmini
Asst. Professor (Sr. Grade),
Dept. of ECE,
Vardhaman College of Engineering,
Hyderabad- 501218(A.P),
INDIA

S. Rajendar
Associate Professor
Dept. of ECE,
Vardhaman College of Engineering,
Hyderabad-501218(A.P),
INDIA

ABSTRACT

In this paper, the implementation of DSP modules such as a floating point ALU are presented and designed. The design is based on high performance FPGA "STRATIX III".

The implementation is done after functional and timing simulation. The simulation tool used is Model Sim. The tool for synthesis and implementation is QuartusII [2]. The experimental results shows the functional and timing analysis for the fixed point to floating converter DSP module carried out using high performance synthesis software from Altera[1]. One of the most important stages of fixed-point to floating-point conversion is the evaluation of the floating-point specification accuracy. This evaluation is required to optimize the data word-length according to accuracy constraints. Classical methods for accuracy evaluation are based on floating-point simulations but they lead to very long optimization times. The use of this method in data word-length minimization processes reduces significantly the optimization time.

Keywords: Fixed point, Floating Point,

Stratix-III FPGA, Quartus-II.

1. INTRODUCTION

This paper specifies the design aspects and simulation results of fixed point to floating point converter. The IEEE 754 floating point format consists of three fields. The Sign bit:

I bit .It is 1 for a negative number and 0 for a positive number [2]. The exponent: 8bits. The exponent represents a power of two. The special case arises when the biased exponent is 255, then zero fraction represents infinity and non zero fraction represents NAN. When the biased exponent and fraction fields are zero, the number represented is a zero. De-normalized numbers are of the form O.f*2/\Emin. The significand is 23 bits[2]. Here the significand is represented as l.fffff---.The fraction part represents a number less than one. The leading one is implicit and does not appear in the representation [2]. The exponent value ranges from -126 to 127 which is biased and therefore ranges from 1 to 254.Further the exponent 0 and 255 represents special cases.

The number represented using the IEEE754 standard is (-I). as seen in * J.f* 2/\ (e-bias).

2. OBJECTIVE

The main objective of this paper illustrates the process of converting the fixed point to floating point. The process of this conversion is to read the input data first, followed by separation of sign bit and the mantissa part. Next we normalize the mantissa and adjust the exponent according to the amount of the shift. Next we add the bias value of 127 to the exponent to obtain the result.

3. TYPES OF EXCEPTIONS THAT ARISE

The four types of exceptions that arise are as follows [2]:

A.)The Overflow exception occurs when the result has an exponent of 0xFF or if any input operator is infinity.

B.) The Underflow exception occurs if the implicit bit of result is zero or if the exponent out is -126 or 0X01 the number is too small to be represented fully in single precision format.

C.)The Division by zero exception occurs when the divisor is zero the result is set to

Infinity.

D.)The Invalid operation exception occurs when the Operation cannot be performed on

operands. Ex: Subtraction of infinity and NAN inputs.

4. FLOWCHART

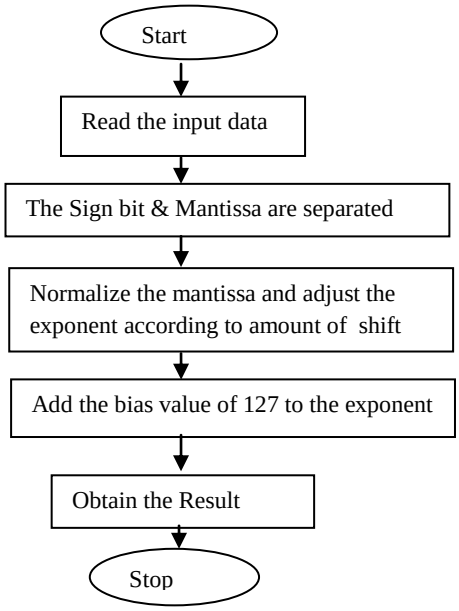

Figure 1 Flow chart for Fixed Point to Floating point conversion.

In this the model based design approach is provided for an integrated workflow. This model based design as shown in figure 2 speeds up the algorithm development with a unified design environment.

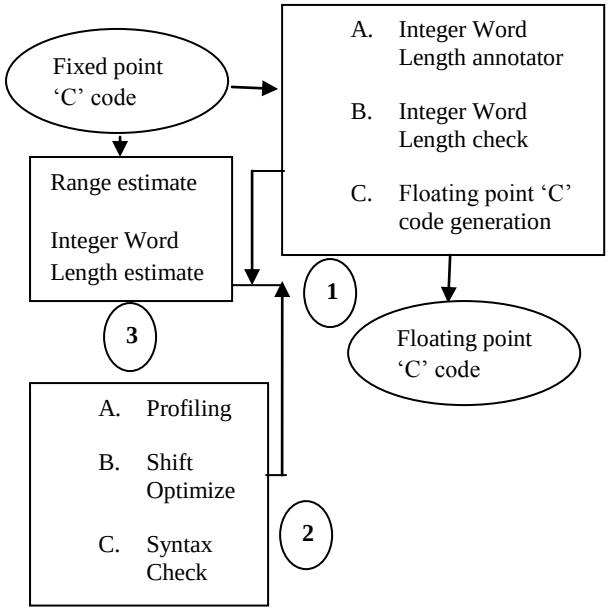

Figure 2 Block diagram of model based design approach

In the above block diagram the block 1 represents the code conversion block which consists of modules such as integer word length annotator, integer word length check, and floating point 'C' code generation. The block 2 illustrates the shift reduction block in which we do the syntax analysis with shift optimize and profiling. Block 3 mainly concentrates on estimation of range and integer word length.

5. ALGORITHM

The algorithm steps provided describes this process are implemented in the modular format.

1. Start
2. Read the input Value
3. Separate Signed bit and the mantissa part.
4. Normalize the mantissa part.
5. Adjust the exponent according to the am-ount if the shift.
6. Add the bias value of 127 to exponent.

The floating-point conversion process is made up of two main stages. The first stage corresponds to the definition of the data binary-point position through the determination of the integer part word-length. For this, the data dynamic range is evaluated to obtain the extreme values which have to be represented. Then, the data word-lengths are determined from the definition for each data of the fractional part word length. The efficient application implementation in hardware architectures such as Field Programmable Gate Array (FPGA) requires the minimization of the chip size and power consumption. Thus, for the implementation, the goal is to minimize the data word length as long as the computation accuracy is maintained.

The conversion procedure is explained in detail as follows. First, the ranges of fixed-point variables are estimated by the simulation of the range-estimation program that is automatically generated from the original fixed point version. The integer word-lengths which are the number of bits used for the integer part, of the floating-point variables are initially determined using the range-estimation results. Second, the integer word-length of each variable is optimized to minimize the number of scaling shift operations using a data-path specific cost function. Finally, the fixed-point variables and constants are replaced by the corresponding integer types, and appropriate scaling codes are inserted.

The models which are proposed earlier such as filters [6], FFT [7], and LMS based adaptive filters [8] do not allow the automatic computation of fixed point accuracy of any signal processing application on FPGA. This approach is interesting in case of implementing fixed point to floating point conversion on a specific selected target such as STRATIX-III FPGA by generating the functional and timing simulations as shown in figure 3 and RTL schematic as shown in figure 4.

6. RESULTS

The obtained results are as follows:

Logic Utilization : <1%

Combinational ALUT's: 122/38,000(<1%)

Total pins: 56/296(19%)

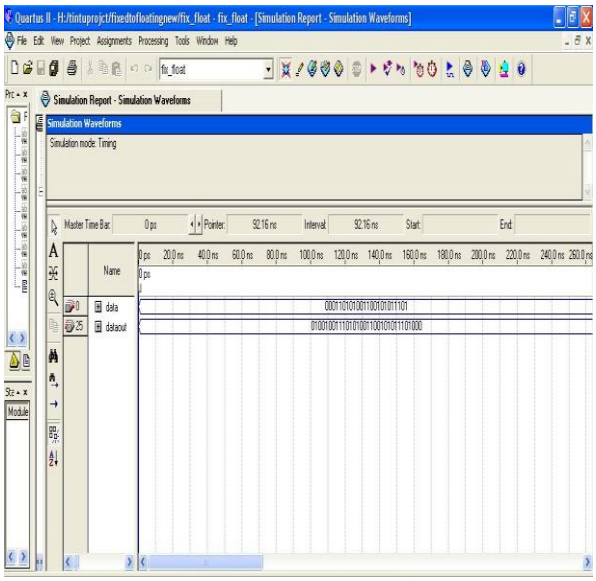

Figure 3 Functional and Timing simulations.

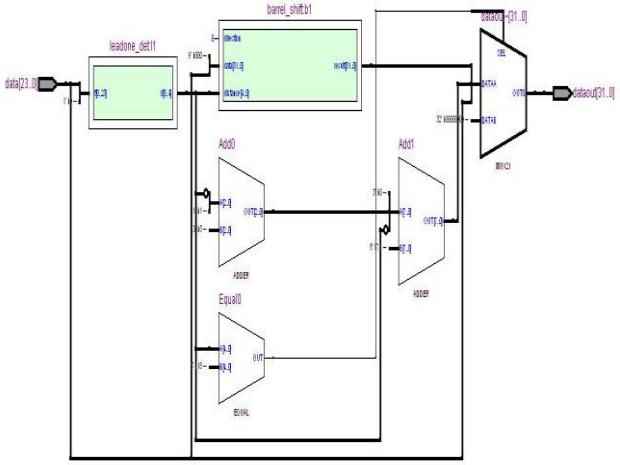

Figure 4 RTL Schematic of Fixed point to floating point converter.

7. CONCLUSIONS

In this paper the implemented DSP modules are Floating point based computational systems. These DSP modules are designed from the block diagram approach to the synthesis and simulation aspects [4]. The functional timing analysis results and the synthesis results are measured in precise and accurate manner. Finally the simulation waveforms are obtained in the FPGA simulation tools and the simulation waveforms are verified with the hardware design aspects, and matching results are obtained. All the simulations are carried out in the Quartus-II software [5]. The target device selected is Stratix-III. The functional and timing analysis for the modules is carried out and accurate measurements are obtained. With our method, the time required to minimize the data word length is definitively lower as seen from the simulation and timing diagrams obtained. It allows a complete exploration of the design space and the determination of an optimized solution.

Further research is aiming towards the further reduction of the acquisition time by a more accurate prediction of the measurement domain.

8. REFERENCES

[1] PrashanthB.U.V. Article: Implementation of FIR Filter and FFT Systems on a STRATIX-IIIFPGAProcessor. *International Journal of Computer Applications* 39(3):16-19, February 2012. Published by Foundation of Computer Science, New York, USA.

[2] Prashanth, B.U.V; Kumar, P.A.; Sreenivaslu G. Article: Design & Implementation of floating point ALU on a FPGA Processor. *The 2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET),* 21-22 March 2012.Published in IEEE Xplore, **DOI:** 10.1109/ICCEET.2012.6203790,Kum-arkoil, Kanyakumari, India.

[3] "A New Common Sub expression Elimination Algorithm for Realizing Low-Complexity Higher Order Digital Filters"IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 29, No. 5, pp 844 - 848, May 2010.

[4] Vinay K. Ingle**,** John G. Proakis,(2009), DigitalSignalProcessingUsingMATLAB,3e, Cengage Learning.

[5] Lawrence R. Rabiner**,** Ronald W. Schafer,(2011), Theory and Applications of Digital Speech Processing, 1e, Prentice Hall.

[6] B. Liu, "Effect of finite word length on the accuracy of digital filters –A review," *IEEE Trans, Circuit Theory*, vol. CT-18, no. 6, pp. 670–677, Nov. 1971.

[7] Tran-Thong and B. Liu, "Fixed-point fast Fourier transform error analysis," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-24, no. 6, pp. 563–576, Dec. 1976.

[8] C. Caraiscos and B. Liu, "A round off error analysis of the LMS adaptive algorithm," *IEEE Trans. Acoust. Speech, Signal Process.* vol. ASSP-32, no. 1, pp. 34–41, Feb. 1984.