Research Paper ENGINEERING

Implementation of Enhanced 64-bit Binary to Floating Point Converter using verilog

Venugopal Rajkumar .H ANANTHA LAKSHMI INSTITUTE OF TECH & SCIENCES Affiliated to JNTUA comment: Affiliation is replaced with Affiliated

Kumara NarayanaSwamy .C (Associate Professor).ANANTHA LAKSHMI INSTITUTE OF TECH & SCIENCES Affiliated to JNTUA

ABSTRACT

Computation with floating point arithmetic is an indispensable task in many VLSI applications and accounts for almost half of the scientific operations. Also adder is the core element of complex arithmetic circuits, in which inputs should be given in standard IEEE 754 format. The main objective of the work is to design and implement a binary to IEEE 754 floating point converter for representing 64 bit double precision floating point values. The converter at the input side of the existing floating point adder/subtractor and multiplier module helps to improve the overall design. The modules are written using Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (Verilog), and are then synthesized for Xilinx Virtex E FPGA using Xilinx Integrated Software Environment (ISE) design suite 8.2i.

KEYWORDS Floating point arithmetic; Double precision; IEEE 754 format; Verilog ; Xilinx

INTRODUCTION

The demand for floating point arithmetic operations in most of the commercial, financial and internet based applications is increasing day by day. Hence it becomes essential to find out an option to feed binary numbers directly as input for these applications. This helps in saving time and is much easier. In the current scenario, this is not possible, because, in the adder/subtractor and multiplier, inputs should be given in IEEE 754 format i.e. the binary inputs cannot be given as such, but it needs to be converted to the sign, exponent and mantissa form, about which, will be described in detail later. Hence in this project we have implemented a binary to floating point converter for single precision bits which will solve this issue to an extend.

The converter is based on IEEE double precision format and this is 64 bits wide. Various modules are written using Very High Speed Integrated Circuit (VHSIC)Hardware Description Language(VHDL)and is simulated with the help of behavioral model .They are then synthesized for Virtex E FPGA, using Xilinx Integrated Software Environment (ISE) design Suite 8.2i.The work has been carried out at Centre for Development and Advanced Computing (C DAC)where a 64-bit floating point adder/subtractor and multiplier module according to IEEE 754 format is already implemented and is currently in use for many specific applications. The proposed converter can be added into the already existing adder/subtractor and multiplier to get the full functionality of the design [10].

The fundamental difference between fixed and floating point digital signal processors (DSPs) is their respective numeric representation of data. While fixed point hardware performs strictly integer arithmetic, floating point DSPs support integer or real arithmetic, the latter normalized in the form of scientific notation. A 64-bit, binary floating point DSP, supporting industry-standard, double precision operations, provides greater accuracy and greater precision than fixed point and single precision devices due to its wider word width, exponentiation and exact internal representations of data. Fixed point devices had to implement real arithmetic indirectly through software routines which add algorithmic instructions and development time, while with floating point format, real arithmetic could be coded directly into hardware operations. So, this thesis emphasizes on utilizing the capabilities of floating point format. The binary input given will range from 0-2047 bits, which is the maximum input range that can be provided which will satisfy the exponent range in the 64 bit IEEE 754 double precision format.

The document is organized as follows: Section II summarizes the important aspects of IEEE 754 single precision format. Section III describes binary to floating point conversion. Section IV explains the design flow overview in brief. Section V presents the results. Section VI concludes the thesis and provides recommendations for further research.

II. IEEE FLOATING POINT REPRESENTATION

The Institute of Electrical and Electronics Engineering (IEEE) issued 754 standard for binary floating point arithmetic in 1985[4,7].This standardization was needed to eliminate computing industry's arithmetic vagaries. Later revisions were made to the existing standard in 2008.There are five basic formats-three binary floating point formats and two decimal floating point formats[1,5,6,8,9].The first two binary formats are called 'Single precision' and 'Double precision respectively. IEEE double precision format alone is considered in this project. Hence it is described below.

In IEEE 754-2008, the 64-bit, base 2 format is officially referred to as binary 64. Double precision format uses 1-bit for sign bit,11-bits for exponent and 52-bits to represent the fraction as shown in Fig 1[1,2].

Sign 1-Bit	Exponent 11-Bits	mantissa 52-Bits
63	62 52	51 0

Fig 1.IEEE 754 double precision format

The double precision floating point number is calculated as (-1)s x 1.F x $2^{(E-1023)}$. Sign bit determines the sign of a number, which is either 0 for a non-negative number or 1 for a negative number. For IEEE single precision format, a bias of 1023 is added to the actual exponent. The mantissa is composed of an implicit leading bit(to the left of the binary point)with value 1,unless the exponent and 52 fraction bits to the right of the binary point is all filled with zeros. The numbers are always normalized and thus there is no need to explicitly show the implicit '1' bit, thereby precision is increased [3].

The IEEE 754 standard specifies some special values like positive infinity, negative infinity, positive zero, negative zero and Not a Number(NaN)[1-4].The standard also specifies the following rounding modes also like: Round to nearest, ties to even; Round to nearest, ties away from zero; Round toward positive infinity; round toward negative infinity and Round towards zero. These special cases are not considered in this project. The overflow bits in the result are just truncated.

III.BINARY TO FLOATING POINT CONVERSION

Converting a base 10 real number into an IEEE 754
Binary64 format using the following outline:

- Consider a real number with an integer and a fraction part such as 12.375.
- Convert and normalize the integer part into binary.
- Convert the fraction part using the following method shown below.
- Add two results and adjust them to produce a proper final conversion.

Conversion of the fractional part is done as shown below: Consider 0.375, the fractional part of 12.375. To convert it into a binary fraction, multiply the fraction by 2, take the integer part and re-multiply new fraction by 2 until a fraction of zero is found or until the precision limit is reached which is 52 fraction digits for IEEE 754 binary64 format.

0.375 x 2 = 0.750 = 0 + 0.750 => b_{-1} = 0, the integer part represents the binary fraction digit. Next step is to re-multiply

0.750 by 2 to proceed.

0.750 x 2 = 1.500 = 1 + 0.500 => b_{-2} = 1

0.500 x 2 = 1.000 = 1 + 0.000 => b_{-3} = 1,

fraction = 0.000, terminate.

We see that $(0.375)_{10}$ can be exactly represented in binary as $(0.011)_2$. Not all decimal fractions can be represented in a finite digit binary fraction. For example decimal 0.1 cannot be represented in binary exactly. So it is only approximated.

Therefore $(15.375)_{10} = (12)_{10} + (0.375)_{10} = (1100)_2 + (0.011)_2 = (1111.011)_2$

- Also in IEEE 754 binary64 format real values need to be represented in normalized form
- Hence it becomes 1.111011 x 2^3 From this
- The exponent is 3 (and in the biased form it is therefore 1023+3=1026 = $(1000\ 0000010)_2$).
- The fraction is 100011 (looking to the right of the binary point)

The resulting 64 bit IEEE 754 binary64 format representation of 15.375 as:

0-10000000010- 1110110000000000000000000000000000000000000000000000 = 40EC000000000000H.

In this project, we have done the binary to floating point conversion in IEEE 754 format. This conversion had been done using Verilog and later it was implemented in Xilinx FPGA.

IV. BLOCK DIAGRAM

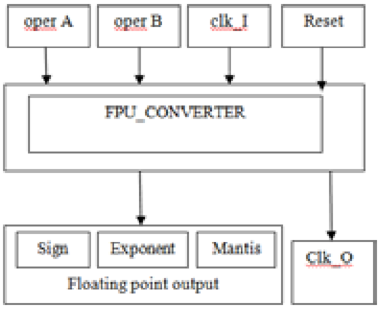

Fig 2.Block Diagram of Binary to Floating Point Converter

This is the block diagram of the binary to floating point converter which was implemented. The inputs and outputs along with the requirements are explained below.

The input fed to the floating point converter includes Input A and Input B, both of which are 2048 bits wide. Next is a clock bit, Clk_I. Another input is Reset, which will initializes all the devices to zero.

Similarly the outputs from the converter block consists of Operand A, Oper A and Operand B, Oper B , both of which should be input [63:0] of size 64. It consists of sign bit, exponent bits and mantissa bits in the output corresponding to the inputs. Next is the clock output, Clk_O, floating point operation output, is differentiated in to Sign 1-bit, Exponent 11-bit and mantissa 52bit. And inputs Clk_I is not processed by the floating point converter. They are simply bypassed by the floating point converter. The signals will obtain its full functionality when it integrated into an existing adder.

V. IMPLEMENTATION RESULTS

The design was simulated using Behavioral model XILINX ISE 8.2i and was targeted towards Virtex E FPGA. The device and package used are XCV600E and BG560 respectively. The speed grade is -6. Synthesis and implementation was performed using Xilinx ISE 8.2i version. The place and route simulation result obtained in Behavioral model, testbench waveform after incorporating the delay file is shown in Figure 3. The implementation results are shown in Table1.

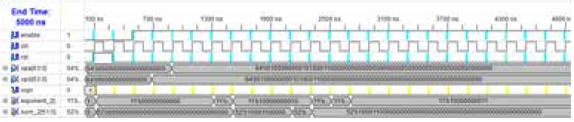

Fig 3: Testbench waveform of Floating-point Addition.

Here the output gives the 64 bit output which includes the sign 1bit, exponent 11bit as well as the mantissa 52bit bits.

TABLE1. DEVICE UTILIZATION SUMMARY

Logic Utilization	Used	Available	Utilization
Number of slice flipflops	437	13824	3%
Number of 4 inputLUTs	660	13824	4%
Logic Distribution			
Number ofoccupied slices	436	6912	6%
Number of slicescontaining onlyrelated logic	436	436	100%
Number of slicescontainingunrelated logic	0	436	
Total number of 4input LUTs	662	13824	4%
Number used aslogic	660		
Number of bondedIOBs	193	404	47%
Number of GCLKs	1	4	25%
Number ofGCLKIOBs	1	4	25%

Total power utilized was 0.0014 W and it had maximum combinational path delay of 6.514 ns.

VI. CONCLUSION

In this project, we have implemented a binary to floating point converter which is based on IEEE 754 double precision format. The unit had been designed to perform the conversion of binary inputs to IEEE 754 64-bit format, which will be given as inputs to the floating point adder/subtractor and multiplication block. The unit was coded using Verilog. Later the codes were synthesized for Virtex E FPGA using Xilinx ISE 8.2i and results were verified.

VII. FUTURE ENHANCEMENTS

In this work we have done the conversion for double precision format (64 bit operands) only. It can be designed for quadruple precision (128 bit operands) in order to enhance precision. Also in target FPGA platform, resources are not fully utilized. Moreover this converter needs to be integrated into the existing floating point adder/subtractor and multiply.

In future it can be enhanced to design division, square root and trigonometry blocks.

ACKNOWLEDGEMENT

We would like to thank the Principal , HOD and all the teaching and non-teaching staffs of Anantha lakshmi Institute of Technology & sciences for helping to complete the work as mentioned in the paper.

REFERENCES

| [1] Charles Farnum, “Compiler Support for Floating-Point Computation” Software Practices and Experience,” pp. 701-9 vol. 18, July 1988. | [2] D. Goldberg, “What every computer scientist should know about floating-point Arithmetic,” pp. 5-48 in ACM Computing Surveys vol.23-1 (1991). | [3] Guillermo Marcus, Patricia Hinojosa, Alfonso Avila and Juan Nolazco- Flores “ A Fully Synthesizable Single-Precision, Floating Point Adder/Substractor and Multiplier in VHDL for General and Educational Use,” Proceedings of the Fifth IEEE International Caracas Conference on Devices, Circuits and Systems, Dominican Republic, Nov.3-5, 2004. | [4] IEEE Computer Society (1985), IEEE Standard for Binary Floating- Point Arithmetic, IEEE Std 754-1985. | [5] Jim Hoff; “A Full Custom High Speed Floating Point Adder” Fermi National Accelerator Lab, 1992. | [6] John Thompson, Nandini Karra, and Michael.J.Schulte “A 64-bit decimal floating-point adder,” Proceedings of the IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design (ISVLSI’04) . | [7] W. Kahan “IEEE Standard 754 for Binary Floating-Point Arithmetic,” 1996 | [8] Subhash Kumar Sharma,Himanshu Pandey,Shailendra Sahni ,Vishal Kumar Srivastava, “Implementation of IEEE_754 Addition and Subtraction for Floating Point Arithmetic Logic Unit”, Proceedings of International Transactions in Material Sciences and Computer,pp.131- 140,vol.3,No.1,2010. |