Decimal Floating Point Format Based on Commonly Used Precision For Embedded System Applications.

Ronald Vincent
P.G- VLSI & Embedded S/M
TKM Institute of Technology
Kollam , Kerala , India
ronaldvincent2013@gmail.com

Ms.Anju.S.L
Assistant Professor, Department of ECE
TKM Institute of Technology
Kollam , Kerala , India
anjusl.1688@gmail.com

Abstract— **The widely used IEEE standards , IEEE 754 -1985 based binary floating point format (BFP) and IEEE 754 -2008 standard based decimal floating point (DFP) format uses high precision formats, which are generally useful for astronomical or scientific calculations. For simple applications , such a precision is not required and is an additional overhead. In this paper, a new floating point format , named Immanuel decimal floating point format (I –DFP) , is presented. I –DFP can be used for embedded system applications, which uses commonly used precision . In I-DFP format , floating point numbers can be represented either in binary or binary coded decimal (BCD). A multiplier based on I-DFP , BCD based number system is developed and is functionally simulated. The multiplier developed is an LUT based BCD multiplier. The synthesis report shows that the multiplier can run with a maximum clock frequency of 198.499 MHz . Comparing the simulation results of I-DFP with the most widely used IEEE 754 -1985 BFP format based arithmetic operations, it is concluded that the floating point operations are more accurate with I-DFP. The language used for programming is VHDL , and is synthesised using Xilinx ISE 12.1 design suite. The target device selected is virtex6 family , XC6VLX75T device .The results are obtained using ModelSim SE 6.2C simulator.**

Keywords— IDFP , Immanuel decimal floating point , decimal floating point , DFP, embedded , VHDL , VLSI.

I. INTRODUCTION

During the 1960's through 1990's the main design constraints were memory and speed. The BFP format represents floating point numbers with fewer bits and also does faster arithmetic computations than decimal floating point format. Thus IEEE 754 -1985 binary floating point (BFP) standard well suited these design constrains and it is the most widely used floating point format, in worldwide systems [3]. But BFP has its own disadvantages. The main disadvantage of BFP is that fractional numbers cannot be accurately represented using BFP. Even numbers like 0.1, 0.2 or 0.3 cannot be represented accurately . Together with the error in representing fractional numbers and rounding errors, the accuracy of representing fractional numbers using BFP format is lost . As more computations, like addition, subtraction, multiplication , and division involve with fractional numbers represented in BFP , the error propagates. For example adding two numbers 0.3 + 0.6, is not exactly 0.9, but is 0.89999999999999991. This causes programming error .These types of floating point errors are not acceptable with applications like commercial and financial, which requires high accuracy. Loss of precision in these type of applications means loss of money [2].

The problem of accurately representing numbers can be solved by using decimal floating point(DFP) format . In 2008 , a new floating point standard was introduced by IEEE, known as IEEE -754-2008 . The main highlight of this floating point standard is decimal floating point standard (DFP) and a backward compatibility to already existing IEEE 754 -1985 binary floating point standard [4]. Ever since its introduction , its importance is increasing and companies like IBM , Intel has started using DFP solutions in their products. The C language committee also considers implementing the IEEE 754 -2008 DFP format and that work is known as ISO/IEC TR 24732:2009 . The large scale memory integration , multi core processor implementations which does parallel processing has loosened the once tightly held design constraints – memory and speed. DFP format is steadily gaining importance and it is expected that in this decade, processor makers will implement DFP arithmetic core in their general purpose processors.

There are common issues faced by both BFP and DFP. One such issue is rounding error. One has to agree to the fact that rounding errors cannot be avoided when representing fractional numbers with a finite number of bits, even though methods like truncating, ceiling and flooring methods are available to handle rounding of numbers . This is the same with binary floating point (BFP) and also with decimal floating point (DFP). The error and its impact in applications can be minimized by increasing the precision digits in DFP.

Both the BFP and DFP are high precision formats. Most of these notations are especially useful for astronomical and scientific calculations. The minimum exponential (emin) of a single-precision BFP is 10^{-38} and that of double precision BFP is 10^{-308}. Similarly for DFP the minimum exponentials (emin) that can be represented by using decimal64 , decimal128 , binary32 , binary64 and binary128 are 10^{-383} , 10^{-6143} , 10^{-126} , 10^{-1022} and 10^{-16382} respectively. For common day to day applications used in embedded systems the precision used are 2,4,6,7 or 8. (eg:- 1.12 , 1.1234 , 1.123456 , 1.1234567 and 1.12345678). For such applications representing fractional numbers with high precision is unnecessary, provided the

978-1-4673-5149-2/13/$31.00 ©2013 IEEE

rounding errors are relatively affordable . In this present work a new modified decimal floating point format named Immanuel decimal floating point format (I-DFP format) for commonly used precision digits is proposed. A 32bit * 32bit multiplier is designed using the proposed floating point format.

II. RELATED WORK

In 2008, IEEE proposed a new floating point standard for decimal floating point numbers. The advantage of DFP format over BFP format is, accurate representation of fractional numbers. The IEEE – 754 – 2008 DFP format has two methods for representing numbers – BID (Binary Integer Decimal) encoding and DPD (Densely Packed Decimal) encoding. The BID defines three types of representation - binary32 , binary64 and binary128. The DPD defines two types of representation - decimal64 and decimal128.

Each of these methods have their own advantages and disadvantages. With BID format , number can be represented with fewer number of bits. Multiplication and division can be done faster. But this is not the case with addition and subtraction , since shifting , rounding and normalizing with binary format is complex [6]. Regarding the second encoding technique – DPD , which is packed BCD format , it has the advantage of representing BCD numbers with fewer bits. Multiplication and division can be directly done in this format, but for addition and subtraction it has to be temporarily converted to BCD format.

Decimal floating point format as proposed by IEEE is for high precision applications. The precision (emin) that can be achieved through different DFP formats are 10^{-383} (decimal64) , 10^{-6143} (decimal128),10^{-126} (binary32) , 10^{-1022} (binary64) and 10^{-16382} (binary128) respectively. For day to day embedded applications the mentioned precision is not necessary. So a new decimal floating point format for embedded applications which uses commonly used precision, but have better accuracy than IEEE – 754-1985 BFP format , is proposed below.

III. IMMANUEL DECIMAL FLOATING POINT (I-DFP) FORMAT

A modified decimal floating point format named as Immanuel decimal floating point, (I-DFP), format is proposed here. In this modified decimal format a number can be represented either in binary format or in binary coded decimal (BCD) format. The format of a floating point number represented using the I-DFP is as follows.

TABLE I.

META NUMBER	NUMBER IN BCD/BINARY

GENERAL REPRESENTATION OF IDFP FORMAT

The meta number is a 8 bit number .The metanumber indicates the sign (+/-) bit of the number , whether the number is represented in binary / bcd , data format type in use and the number of decimal places in the number.The number can be indicated in the general form as $\pm$ (0 to $(2^N - 1)) * 10^{-E}$, where the exponential E is in the range 0 to 14.

Four data format types are defined after IDFP , they are

TABLE II.

META NUMBER	NUMBER RANGE (0 to ($2^8 - 1$))

FLOAT_CHAR8 (8 bit binary) , N = 8.

TABLE III.

META NUMBER	NUMBER RANGE (0 to ($2^{16} - 1$))

FLOAT_INT16 (16 bit binary) , N = 16.

TABLE IV.

META NUMBER	NUMBER RANGE (0 to ($2^{32} - 1$))

FLOAT_INT32 (32 bit binary) , N = 32.

TABLE V.

META NUMBER	NUMBER RANGE (0 to ($2^{64} - 1$))

FLOAT_INT64 (64 bit binary) , N = 64.

An example of number representation in I – DFP, BCD format is as follows. If the number, -42. 9496 7295 is represented in FLOAT_INT32 data type then, - 42. 9496 7295 can be rewritten as $- 42\ 9496\ 7295 \times 10^{-8}$, which when represented in I-DFP BCD format is , 1110 1000 0100 0010 1001 0100 1001 0110 0111 0010 1001 0101.

IV. COMPARISON BETWEEN IEEE FORMATS AND I-DFP FORMAT.

The main difference between IEEE -DFP and I-DFP format is that , there is only negative exponential in I-DFP. Also there is no bias in exponential , which reduces the overhead in calculations. When comparing with IEEE floating point format based arithmetic units with I-DFP BCD format based adder , subtractor and multiplier, the latter can be implemented easily in embedded applications. It is devoid of any complexity since there are no encoding techniques used in the format. On the negative side , I-DFP format uses BCD rather than Densely Packed Decimal (DPD), it requires more memory space to store numbers. But as we look into the evolution of IC fabrication technologies , the level of memory integration is rapidly growing day by day and more memory can occupy per unit area. So memory constraint might not be a tightly held rule in the coming decades as before. Combining faster architectures using parallel processing , this disadvantage could

TABLE VI.

Operand X	Operand Y	Operation	IEEE -754 -1985 BFP Erroneous result.	I-DFP Correct Result
1.345	1.123	-	0.22199999999999998	0.2220
0.1235	0.1234	-	0.00010000000000000286	0.0001
3.537	3.523	-	0.01399999999999979	0.0140
0.3	0.6	+	0.89999999999999991	0.9
2.24	22.4	(X*10)-Y	0.00000000000000035527136788	0

COMPARISON BETWEEN BFP AND IDFP FORMATS

be overlooked in the near future when we consider the ease of representing number in human readable form. But the overall comparison between IEEE DFP and I-DFP formats would only be complete if a study on the rounding errors in both of these formats is done, considering all the options like truncating, ceiling, flooring etc. Also its impact in applications, which widely vary in the requirement of precision should also be considered. But in that case, for common precision applications , the result would be comparable.

V. I-DFP BASED BCD MULTIPLIER

In order to verify the accuracy of I-DFP format - an adder, subtractor and a multiplier is build. The multiplier design is explained below. The multiplier build is an LUT based BCD multiplier. Here partial products are stored in the look up table as shown in figure 1, below.

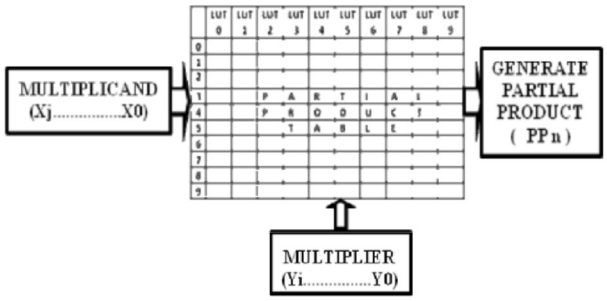

Figure 1. Partial Products stored in Look Up Table for Multiplier.

The partial products are repeatedly added together using conventional BCD adder circuit. The BCD multiplier built is a 32bit * 32bit multiplier , which gives a 64 bit result.

The basic BCD adder shown in figure 2, adds two BCD nibbles. If the result exceeds the limit of BCD number representation ie, (0 -9) then a BCD correction is required. For BCD correction, six is added to (ie, nine is subtracted from) the sum.

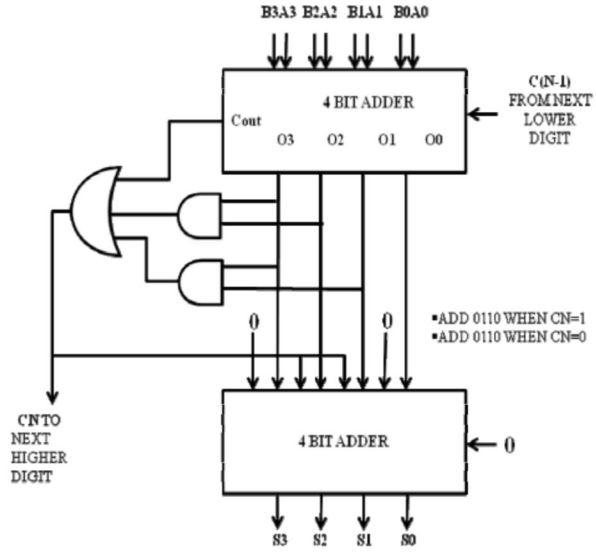

Figure 2. Four bit Nibble BCD Adder Circuit.

If more than two BCD nibbles are to be added, then BCD adders are cascaded as shown in figure 3. For floating point addition the operands has to be normalized to required decimal places by shifting and rounding.

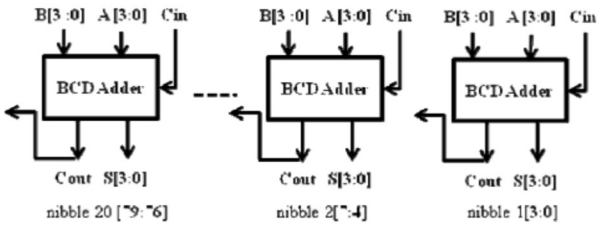

Figure 3. Cascading four bit BCD Adder Circuit.

The exponent of the result is the summation of the exponent of the two operands involved in multiplication. The number of decimal places in the number representated in the meta number

is encoded in binary format. So when two numbers are multiplied the exponents are simply added together.

VI. IMPLEMENTATION RESULTS

The functional simulation of LUT based bcd multiplier is done using Xilinx ISE design suit 12.1. The hardware description language used is VHDL. The device selected for functional simulation is from virtex6 family , XC6VLX75T device. The multiplier is a 32 bit * 32 bit multiplier. The simulation result shows the multiplier runs with a maximum clock frequency of 198.499MHz. The following list shows the device utilization summary.

TABLE VII.

Device utilization summary		
Selected Device: 6vlx75tff484-3		
Slice Logic Utilization		
Number of Slice Registers	169 out of 93120	0%
Number of Slice LUTs	8407out of 46560	18%
Number used as Logic	8407 out of 46560	18%
Slice Logic Distribution:		
Number of LUT Flip Flop pairs used	8410	
Number with an unused Flip	8241 out of 8410	97%
Number with an unused LUT	3 out of 8410	0%
Number of fully used LUT-FF pairs	166 out of 8410	1%
Number of unique control sets	4	
IO Utilization		
Number of IOs	192	
Number of bonded IOBs	192 out of 240	80%
Specific Feature Utilization		
Number of BUFG/BUFGCTRLs	1 out of 32	3%

Device Utilization Summary

VII. CONCLUSION

The BFP format proposed by IEEE has the advantage of high processing power but it cannot represent fractional numbers accurately. Also it needs conversion algorithms to represent numbers in human readable form, especially in the case of embedded system applications. The IEEE formats , both BFP and DFP are high precision formats suitable especially for astronomical and scientific calculations. So a new floating point format named Immanuel Decimal floating point (I-DFP) format, for embedded system applications, which uses commonly used precision is presented in this work.

In high level languages, there exits different formats for representing integers (positive and negative numbers) and floating point numbers (IEEE BFP and DFP formats) and when implementing these formats using hardware description languages, it requires separate hardware . Also complex hardware is required for converting integer numbers (represented in binary) to floating point numbers (represented in terms of mantissa , exponential ,bias, normalization) and viceversa. Using I-DFP format for representing both integer and floating point numbers these separate hardware parts are not required, and hence overall hardware area reduction. Also there is no need for a separate hardware for signed and unsigned number arithmetic operations in the case of I-DFP format. The cost of this advantage is, meta-number should always be part of the number representation. Even if we use separate formats, the conversion from integer to I-DFP format is easier compared to IEEE formats. All data types used in high level languages like char , integer , signed char , unsigned integer and floating point numbers can easily be represented using the four data type formats defined using I-DFP.

To verify I-DFP format , a 32bit * 32bit LUT based BCD multiplier is developed using VHDL. The code is synthesized using Xilinx ISE design suit 12.1 and simulated using ModelSim simulator. The functional simulation result shows that the multiplier can run with a clock frequency of 198.499 MHz . Comparison results shows that the accuracy of I-DFP format is better than IEEE -754-1985 BFP format and is more ideal for embedded applications which uses commonly used decimal places .

REFERENCES

[1] Ahmad M. Zaki, Ayman M. Bahaa-Eldin,"Accurate Floating-point Operation using Controlled Floating-point Precision",IEEE, pp 696 - 701, 2011.

[2] Charles Tsen, Sonia González-Navarro, Michael Schulte1, Brian Hickmann, Katherine Compton ,"A Combined Decimal and Binary Floating-point Multiplier ", IEEE , pp 8 -15 , 2009.

[3] Michael J. Anderson, Charles Tsen, Liang-Kai Wang, Katherine Compton, and Michael J. Schulte "Performance Analysis of Decimal Floating-Point Libraries and Its Impact on Decimal Hardware and Software Solutions" , IEEE , pp 465 – 471 , 2009.

[4] IEEE Standard for Floating-Point Arithmetic, IEEE Std 754™-2008.

[5] Muneeb Ahmed Awan, Muhanmmad Raheel Siddiqui ,"Resolving IEEE Floating-Point Error using Precision-Based Rounding Algorithm", IEEE, pp 329 – 333 , 2005.

[6] Michael F. Cowlishaw, Eric M. Schwarz, Ronald M. Smith, Charles F. Webb ,"A Decimal Floating-point Specification" , IEEE, pp 147 -154, 2001.

[7] ANSI/IEEE standard for binary floating point arithmetic 754-1985.