RESEARCH ARTICLE OPEN ACCESS

Implementation of Optimized Floating Point Adder on FPGA

Deepak Mishra [1], Vipul Agrawal [2]
*(Department of Electronics and Communication Engineering
Trinity Institute of Technology & Research
Bhopal - India

ABSTRACT
This paper presents the FPGA implementation of a Decimal Floating Point (DFP) adder. The design performs addition on 64-bit operands that use the IEEE 754-2008 DPD encoding of DFP numbers. The design uses an equal bypass adder, this adder reduces the power consumption and it also reduces the delay by reducing the gate count. The design also uses barrel shifter instead of sequential shifter. The design has a maximum combinational delay of 45ns on a Virtex-5 with a latency of 1 cycle. The proposed DFP adder supports operations on the decimal64 format and it is easily extendable for the decimal128 format.

Keywords:- Floating Point Adder, FPGA, Delay, Area Overhead

I. INTRODUCTION

The binary floating point (BFP) arithmetic has certain flaws namely; it cannot provide correct decimal rounding and cannot precisely represent some decimal fractions such as 0.001, 0.0475 etc [1]. There are many applications where a precision is required such as billing, insurance, currency conversion, banking and some scientific applications. European Union requires that currency conversion to and from EURO is to be calculated to six decimal digits [2]. One study estimates that errors generating from BFP arithmetic can sum up to a yearly billing of over dollar 5 million for a large billing organization [3]. Therefore decimal floating point (DFP) arithmetic becomes very important in many current and future applications as it has ability to represent decimal fractions precisely. DFP arithmetic also has the ability to provide correct decimal rounding that will mimic the manual rounding.

Applications which cannot tolerate errors generating from BFP arithmetic, these application use software platforms to perform DFP arithmetic [1]. There are many software packages which are available for example: the java BigDecimal library [5] and IBM's decNumber library [4]. Also Intel published results for a decimal arithmetic library which uses Binary integer decimal (BID) encoding. These software packages are good enough for current applications, but trends towards globalization and e-commerce are increasing, so faster response of these systems is required. Software designs to these systems may be inadequate with the increasing performance demands of future systems. So hardware implementation of these systems is the need of the hour.

In 2008, the IEEE 754-1985 floating point standard has been revised and the new standard called the IEEE 754-2008 floating point standard was setup [6], which includes specifications for DFP formats, encoding and operations. The IEEE 754-2008 standard includes an encoding format for DFP numbers in which the significand and the exponent (and the payloads of NaNs) can be encoded in two ways namely; binary encoding and decimal encoding. [7]

Both the encoding formats break a number into a sign bit s, an exponent E, and a p-digit significand c. The value encoded is $(-1)s \times 10E \times c$. In both formats the range of possible values is identical, but the significand c is encoded differently. In the decimal encoding, it is encoded as a series of p decimal digits using the densely packed decimal encoding (DPD). In the binary encoding also known as binary integer decimal (BID) encoding, it is encoded as a binary number.

In this paper a floating point adder unit is proposed. This floating point adder unit is IEEE P754 – 2008 complaint and based on densely packed decimal (DPD) encoding for DFP arithmetic. The proposed floating point adder unit uses low power equal bypass adder to reduce the power consumption of the design.

II. DECIMAL FLOATING POINT REPRESENTATION

In IEEE 754-2008, the value of a finite DFP number with an integer significant is

$$v= (-1)s \times 10q \times c$$

Where 'S' is the sign, 'q' is the unbiased exponent, and 'C' is

Sign (s)	Combination (w + 5)	Trailing Significand (c)
1 Bit	13 bits	50 bits

the significant. The precision or the length of the significant is denoted as 'p', which is equal to 7, l6, or 34 digits, for decimal32, decimal64, or decimall28, respectively. Figure 1.1 shows the double precision decimal64 Decimal Floating Point format.

Figure 1.1: Decimal 64 – Decimal floating point format DPD encoded .The l-bit Sign Field, S indicates the sign of a number. The (w+5)-bit Combination Field, G provides the most significand digit (MSD) of the significand and a non-negative biased exponent, E such that E = q + bias. The exponent is almost always encoded in binary. The G Field also indicates special values, such as Not-a-Number (NaN) and infinity (00). The remaining digits of the significand are specified in the t-bit Trailing Significand Field, T. Table 1 shows the

Number type	Combination field	Exponents Bits	Significand MSD
finite	a b c d e	a b	0 c d e
finite	11 a b c	a b	1 0 0 e
infinite	1 1 1 1 0	. .	. . . .
NaN	1 1 1 1 1	. .	. . . .

combination field. [DPD Paper]

Table 1: Combination field

III. DECIMAL FLOATING POINT ADDITION ALGORITHM

Figure 1.2 shows the algorithm for adding two decimal floating point numbers encoded in DPD dec64 format.

Step 1: Decode the inputs A and B to obtain

(A_s, A_E, A_c) and (B_s, B_E, B_c)

Step 2: Define effective operation (EOP)

EOP <= As XOR Bs; EOP = 0 → Addition, EOP = 1 → Subtract

Step 3: if Ae < Be, then Swap A and B

Step 4: Calculate d <= Ae – Be

Step 5: Shift right 'Bc' by d

Step 6: Add 'd' to 'Be'

Step 7: Compute Zc <= Ac ± Bc (Depends on EOP)

Step 8: Ze <= Ae

Step 9: Zs <= As XOR Bs

Step 10: Encode in decimal 64 – IEEE P754 format

Figure 1.2: Floating Point Addition – Algorithm

IV. DECIMAL FLOATING POINT ADDER IMPLEMENTATION

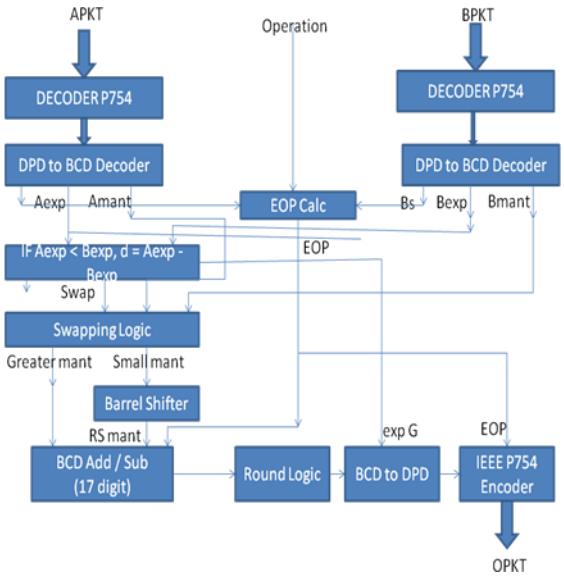

Figure 1.3: Floating point adder: Architecture

This architecture of floating point adder is proposed for IEEE 754-2008 decimal64 format and it can easily be extended to 128 bit format.

Arrows shows the direction of data flow. The adder is designed as follows: the decoder unit decodes the decimal 64 bit number and extracts sign (As, Bs), exponent (Aexp, Bexp) and mantissa (Amant, Bmant) information. The effective operation (EOP) logic defines the actual operation by xoring the two signs As and Bs. If both number are either positive or negative then the EOP is addition and if one of them is negative then the EOP is subtraction. We have two channels in our design Channel A and channel B, channel A is for large number and channel B is for smaller number. So next the two exponents are compared, if Ae is smaller than Be then numbers are swapped, the larger of two number is assigned to channel A and the smaller one is assigned to channel B using a swapping logic. Here the difference between two exponents is also computed and assigned to variable d. Next the mantissa Bm (smaller of two mantissa's) is shifted right by amount "d" by barrel shifter. Now the two numbers are aligned for addition/subtraction.

The effective operation is performed using a low power BCD adder. The effective operation is determined earlier using EOP. Next the rounding logic rounds the resultant mantissa. The sign, exponent and mantissa of resultant are encoded in IEEE P754-2008 decimal64 format.

V. LOW POWER ADDER DESIGN

The floating point adder shown in figure 1.3 uses a 16 digit BCD adder for the addition of mantissa Am and Bm. Figure 1.4 shows the 4 bit BCD adder, this BCD adder uses two 4 bit ripple carry adder, these 4 bit ripple carry adder uses conventional full adder

.

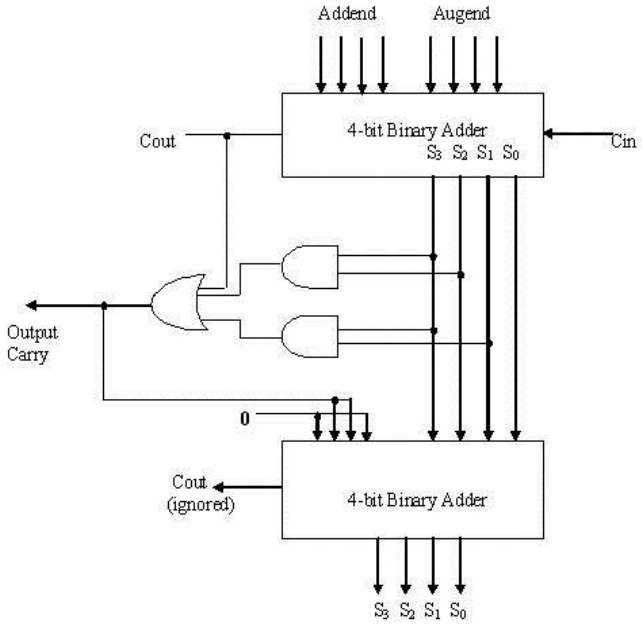

Figure 1.4: BCD adder

Figure 1.5 shows the conventional full adder. All the logic gates in this design are applied with inputs all the time and this consumes power at all times, also the gate count for sum is two and the gate count for carry is three. We are replacing the conventional full adder by equal bypassing full adder. Figure 1.5 shows the low power low delay equal bypassed full adder.

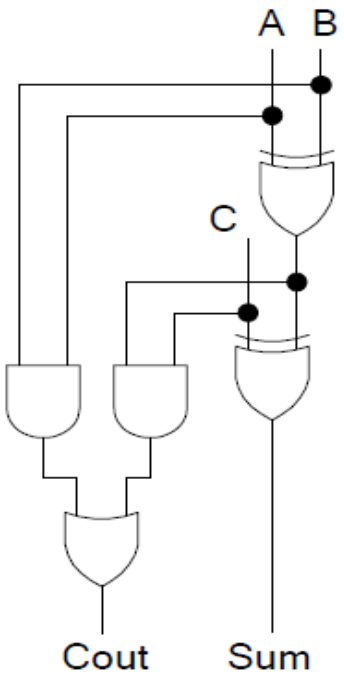

Figure 1.5: Conventional full adder

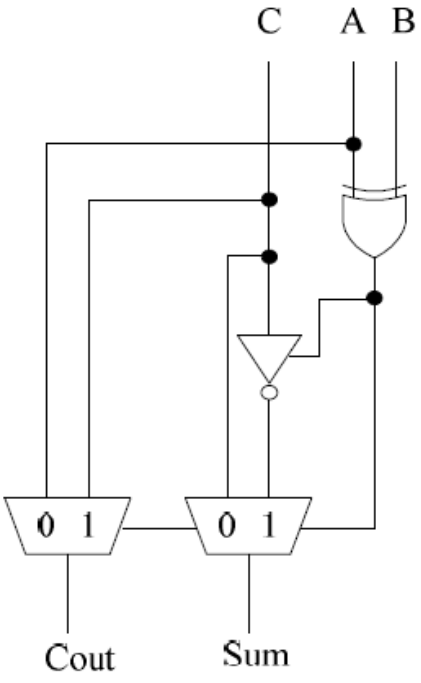

Figure 1.6: Low power – low delay full adder

In this full adder when input 'A' and input 'B' are equal then the output of XOR gate is '0', this makes the control input of tri-state buffer '0', now the output of tri-state inverter is high impedance 'Z', this blocks one channel of the multiplexer and reduces the power consumption. And if the two inputs 'A' and 'B' are different then the output of the XOR gate is '1' and

control input of tri-state inverter is also '1', the input 'C' is complemented.

Also at all times the gate count of Cout is reduces to 1, this is 2 less than the conventional full adder. So the power consumption and delay of the BCD adder is reduced. [Low power adder]

VI. IMPLEMENTATION & RESULTS

All logics were described in VHDL. The design has been implemented on Xilinx Virtex-5 device XC-5VLX30FF324-3. Resource utilization is shown in table 2. Design statics are shown in table 3.

Table 2: Device Utilization Summary

S.No.	Resource	Utilization
1	Number of slice registers	5/19200
2	Number of slice LUTs	803/19200

Table 3: Design Statistics

S.No.	Cell	Usage
1	BELS	840
1a	GND	1
1b	LUT2	9
1c	LUT3	101
1d	LUT4	82
1e	LUT5	180
1f	LUT6	431
1g	MUXF7	36
2	FLIP FLOPS/LACTHES	7
2a	LD	5
2b	LDCP	2
3	IO Buffers	193
3a	IBUF	129
3b	OBUF	64

Table 4: Power Consumption

S.no	Frequency	Static	Dynamic	Total
1	10 Mhz	379 mw	24 mw	403mw
2	50 Mhz	380 mw	118 mw	498 mw
3	100 Mhz	382 mw	235 mw	617 mw
4	200 Mhz	385 mw	471 mw	856 mw

VII. CONCLUSION

IEEE P754 compliant decimal floating point adder is successfully implemented on Virtex-5 device. The design was tested with several test vectors and no errors are found, so the design is behaving correctly. We have replaced the full adder of the BCD adder by a low power – low delay full adder to reduce the power consumption and the delay of the design. The design has a maximum combinational delay of 45 ns with the latency of 1 clock cycle.

The low power – low delay adder can be further used in the implementation of multiplier and divider circuit and the complete floating point arithmetic and logic unit can be implemented on FPGA. This will reduce the power consumption of the complete design. Further clock gating techniques can be used to reduce the clock power.

REFERENCES

[1] M.F. Cowlishaw, "Decimal Floating-Point: Algorism for Computers," Proc. IEEE 16th Symp. Computer Arithmetic, pp. 104-111, 2003.

[2] IBM Corporation, The 'Telco' benchmark, http://speleotrove.com/ Decimal/telcoSpec.html, 2005.

[3] D.-G. for Economic and F. A. C. from the Commission to the European Council, "Review of the Introduction of Euro Notes and Coins," EURO PAPERS, Apr. 2002.

[4] M.F. Cowlishaw The decNumber library, v3.68. IBM, http://speleotrove.- com/decimal/decnumber.pdf, 2013.

[5] S. Microsystems BigDecimal Class, Java 2 Platform Standard ed. 5.0, APISpecification,http://docs.oracle.com/javase/1.5.0/docs /api/java/math/BigDecimal.html, 2013.

[6] M. Cornea, C. Anderson, J. Harrison, P.T.P. Tang, E. Schneider, and C. Tsen, "A Software Implementation of the IEEE 754R Decimal Floating-Point Arithmetic Using the Binary Encoding Format," Proc. IEEE 18th Symp.

[7] ANSI/IEEE 754-1985, "Standard for Binary Floating-Point Arithmetic".

[8] R.K. Yu, G.B. Zyner, 167 MHz radix-4 floating point multiplier, Proceedings 12th Symposium on Computer Arithmetic, 1995, pp. 149-154.

[9] C. Gamez, R. Pang, Apparatus and method for rounding operands, U.S. patent 5258943, 1993.

[10] M. Saishi, T. Minemaru, Multiplication circuit having rounding function, U.S. patent 5500812, 1996.

[11] Guy Even, Silvia M. Mueller, Peter-Michael Seidel "A dual precision IEEE Floating-point multiplier"

Elsevier INTEGRATION, the VLSI journal 29 (2000) 167-180.

[12] C. Tsen, M.J. Schulte, and S.G. Navarro, "Hardware Design of a Binary Integer Decimal Based IEEE P754 Rounding Unit," Proc. IEEE 18th Int'l Conf. Application-Specific Systems, Architectures and Processors, pp. 115-121, 2007.

[13] B.J. Hickmann, A. Krioukov, M.J. Schulte, and M.A. Erle, "A Parallel IEEE P754 Decimal Floating-Point Multiplier," Proc. IEEE 25th Int'l Conf. Computer Design, 2007.

[14] C. Tsen, S.G. Navarro, M.J. Schulte, B. Hickmann, and K. Compton, "A Combined Decimal and Binary Floating-Point Multiplier," Proc. IEEE 20th Int'l Conf. Application-Specific Systems, Architectures, and Processors, pp. 8-15, 2009.

[15] J. Di and J. S. Yuan, "Power-aware pipelined multiplier design based on 2-dimensional pipeline gating," in *13th Great Lakes Symposium on VLSI*. ACM, 2003, pp. 64–67.

[16] Sunjoo Hong, Taehwan Roh and Hoi-Jun Yoo, "a 145w 8×8 parallel multiplier based on optimized bypassing architecture", department of electrical engineering, Korea advanced institute of science and technology (KAIST), Daejeon, Republic of Korea, IEEE, pp.1175-1178, 2011.

[17] Yin-Tsung Hwang, Jin-Fa Lin, Ming-Hwa Sheu and Chia-Jen Sheu, "low power multipliers using enhenced row bypassing schemes", department of electronic engineering, National Yunlin University of science & technology, Touliu, Yunlin, Taiwan, IEEE, pp.136-140, 2007.

[18] George Economakos, Dimitris Bekiaris and Kiamal Pekmestzi, "a mixed style architecture for low power multipliers based on a bypass technique", national technical University of Athens, school of electrical and computer engineering, heroon polytechniou 9, GR-15780 Athens, Greece, IEEE, pp.4-6, 2010.

[19] Meng-Lin Hsia and Oscal T.-C. Chen, "low power multiplier optimized by partial-product summation and adder cells", dept. of electrical engineering, national chung cheng University, chia-yi, 621, Taiwan, IEEE, pp.3042-3045, 2009. [12] P. C. H. Meier, "analysis and design of low power digital multipliers", Ph.D. thesis, Carnegie Mellon University, dept. of electrical and computer engineering, Pittsburgh, Pennsylvania, 1999.

[20] Carlos Minchola, Martin Vazquez and Gustavo Sutter "A FPGA IEEE 754 2008 decimal floating point adder subtractor" 2011 IEEE.

[21] Yanyu Ding, Deming Wang, Jianguo Hu and Hongzhou Tan, "A Low power Parallel Multiplier Based on Optimized-Equal-Bypassing-Technique", Third International Conference on Information Science and Technology March, 2013 IEEE, China