International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 24 (2015) pp 44001-44006

Implementation of Low Power Multi-Precision Floating Point Multiplier

D. Gokila

Assistant Professor, Department of Electronics and Communication Engineering, United Institute of Technology, Coimbatore, Coimbatore District gokiladr@gmail. com

Dr. H. Mangalam

Professor, Department of Electronics and Communication Engineering Sri Krishna College of Engineering and Technology, Coimbatore, Coimbatore District hmangalam2@gmail.com

Abstract
Floating point multiplier is widely used in digital signal processing applications. The performance of Field Programmable Gate Arrays (FPGAs) used for floating point application is low, because of complexity in operations. This creates less interest in making FPGAs for floating point applications. So we are going for the reconfigurable floating point multiplier which provides improved deployment of the multiplier in most of the applications. This performs double precision operation or single precision operation. Further power is reduced by using the spurious power suppression technique. The implementation shows a better performance with respect to power and delay.

Keywords: Double precision, Single precision, Reconfigurable, Floating point multiplier (FPM).

Introduction

High processing performance and low power dissipation are the most important objectives in many multimedia and digital signal processing (DSP) systems, where multipliers are always the basic arithmetic unit and significantly influence the system's performance and power dissipation. Multipliers using floating point numbers are in great demand because floating point numbers have good precision, since they never deliberately discard information. So a fast and energy-efficient floating point unit is always needed in electronics industry. Field Programmable Gate Arrays (FPGA's) are broadly used for scientific computation because of the ease of customizing the hardware for the application. The limited size and architecture of FPGAs are not well-suited for floating-point applications. On the other hand, ASICs can be very efficient at floating-point operations, but lack the programmability and flexibility that is desired in many situations, and the cost of an ASIC can be prohibitively high.

By overcoming the limitations of FPGAs, it will be a very attractive platform for floating point applications. So for the better utilization of the floating point multiplier unit, reconfigurable computing is added. A new computing method using reconfigurable architectures promises an intermediate trade-off between flexibility and performance. Reconfigurable computing uses hardware that can be adapted at run-time to facilitate greater flexibility without compromising on performance. The re-configurability of the hardware permits adaptation of the hardware for specific computations in each application to achieve higher performance compared to software. Here the re-configurability is applied to perform single precision and double precision floating point multiplication. In order to further enhance the multiplier in terms of power, an efficient power reduction technique namely the spurious power suppression technique is implemented which provides an eminent improvement in power and delay.

Related Works

In FPGA's, re-configurability gives significant area utilization and delay improvements. A number of works has been proposed based on the configurability. Akkas [1] has produced the multiplier which is configured to perform either one quad precision multiplication or two double precision multiplications in parallel. It takes three cycles to perform quadruple precision multiplication and can produce a quadruple precision product every other cycle. And two double precision operations in parallel will take two cycles. Diniz and Govindu [3] have presented the design of a field programmable dual precision multiplier. By getting knowledge from these, the work is proposed for doing one double precision multiplication or two single precision multiplication. Mainly the multiplier work is based on the Akkas design [1]. But the difference is that they have used two multipliers for lower precision multiplication. These same multipliers are used in multi cycle for higher precision multiplication. Due to this structure the delay of multiplication operation is high in previous works. In proposed design, instead of two simple multipliers, Radix-4 booth concept and Wallace tree structured multiplier is used, so that, the speed of operation can be improved because of single cycle utilized for both single and double precision multiplication. This is the main advantage of this design. The reconfiguration can be obtained by just using control signal for multiplexers. Reconfiguration time is very low because it involves only changing the control signal for the multiplexers. But it has a disadvantage of having little more area than other works due to tree structured multiplier. And then the delay needed to perform the single precision operation is slightly high. The main advantage is flexibility in Floating Point Multiplier for FPGA architectures so that both double precision

multiplication and single precision multiplication can be performed.

Floating Point Representation

In general, a floating point number can be represented as
$\pm M \times B^E$
Where M is the mantissa
E is the exponent
B is the base.

For binary case, the floating point number is represented as
$(-1)^s \times M \times 2^E$
where 2 is representing the implied base. Based on IEEE 754 standard, floating point number consist of three fields

1) A sign bit (S)
2) Biased exponent (E)
3) Mantissa (M)

The IEEE 754 floating-point standard uses 32 bits to represent a single precision floating-point number, including a single sign bit, exponent bits with bit width 8 and 23 bits of mantissa. The mantissa has effectively 24 bits including 1 implied bit to the left of the decimal point not explicitly represented in the notation.

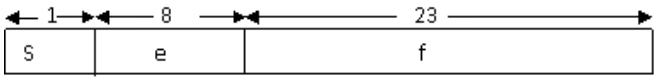


Fig.1. IEEE 754 Single Precision Floating Point Number

IEEE 754 uses 64 bits to represent double precision floating point number, including 1 sign bit, exponent bits with bit width 11 and 52 bits of mantissa. The mantissa has effectively 53 bits including 1 implied bit to the left of the decimal point not explicitly represented in the notation

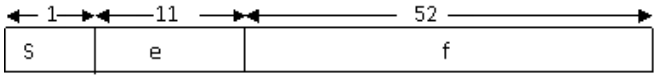


Fig.2. IEEE 754 Double Precision Floating Point Number

Bias value for the 8 bit and 11 bit exponent is 127 and 1023 respectively, then the representation is as follows
$X = (-1)^S x1.\ fx2^{(e-127)}$
$X = (-1)^S x1.\ fx2^{(e-1023)}$

Floating point numbers are having higher precision compared to fixed point numbers so that discarding of information is low.

Floating Point Multiplication

Consider the two floating point numbers X1 = (s1, e1, f1) and X2 = (s2, e2, f2) each consists of

- Sign bit
- Exponent bits
- Mantissa bits

Then Floating point multiplication X_p can be obtained using following steps.
$s_p = s_1 \oplus s_2$
$e_p = e_1 + e_2 - bias$
$1.\ f_p = 1.\ f_1 \times 1.\ f_2$

This equation can be formed as data path as shown in Fig.3.

In mantissa adjust block the normalization operation is used, based on that the exponent value has been changed. For double precision multiplication, the mantissas are getting multiplied using 53 x 53 bit multiplier. This multiplier can be configured as 24 x 24 bit multiplier; this will help to perform single precision multiplication. This single precision multiplier will take the inputs from LSB side of the inputs which is applied for double precision multiplication.

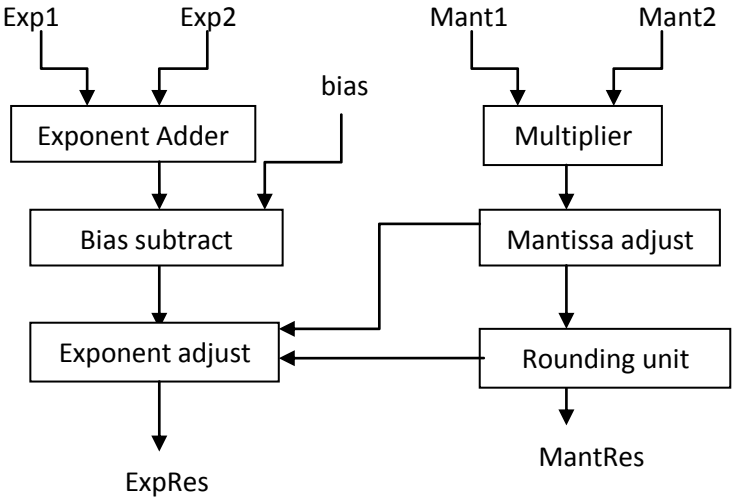


Fig.3. Data Path of Floating Point Multiplication

In this proposed work, Radix-4 modified booth algorithm with Wallace tree structure is used to perform the mantissa multiplication. Booth encoding is used to reduce the number of partial products into half the number of bits in multiplier (X). Due to this, number of levels in Wallace structure would be reduced. Then partial products have been added using number of full adders in Wallace structure to produce the final product.

Multiplier Unit

The structure consists of the components for double precision multiplication. One of the inputs is given as input for booth encoder and then output from this will drive the partial product generator. Another input for partial product generator is given which is same as the second mantissa value. This will produce the partial products in 27 rows. And then these all the

partial products are compressed using number of full adders and half adders to get the final sum.

In this design multiplier [10] block consists of following blocks

1) Booth Encoder
2) 53 x 53 bit Partial Product Generator
3) Wallace structure
4) CSA adder

Booth Encoder

Parallel Multiplication using basic Booth's Recoding algorithm technique based on the fact that partial product can be generated for group of consecutive 0's and 1's which is called as Booth's recoding. These Booth's Recoding algorithm [6] is used to generate efficient partial product. These Partial Products always have large number of bits than the input number of bits. This partial product width usually depends upon the radix scheme used for recoding.

Modified Booth algorithm:

One of the solutions of realizing high speed multipliers is to enhance parallelism which helps to decrease the number of subsequent calculation stages. The first version of the booth algorithm [8] (radix-2) had two disadvantages. They are:

1) The number of add subtract operations and the number of shift operations become variable and becomes inconvenient in designing parallel multipliers.
2) The algorithm becomes inefficient when there are isolated 1's.

These drawbacks are overcome by using modified radix-4 booth algorithm which scans strings of three bits with the algorithm.

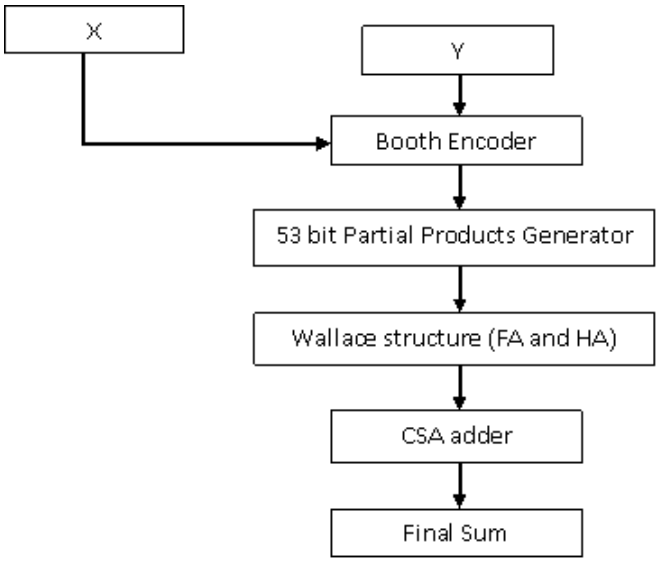


Fig.4. Proposed multiplier structure

Algorithm

- Extend the sign bit by one position if necessary to ensure that n is even.
- Add a zero to the right of the LSB of the multiplier.
- Based on the value of each vector, each partial product will be 0, +y,-y, +2y or-2y.

The negative values of y are made by taking the two's complement. The multiplication of y is done by shifting y by one bit to the left. Thus, in any case, only n/2 partial products are generated in designing of n-bit parallel multipliers. The least significant block (LSB) uses only two bits of the multiplier, and assumes a zero for the third bit. The overlap is necessary so as to know what happened in the last block, since the MSB of the block acts like a sign bit. The modified booth algorithm using radix 4 method is the efficient technique. Based on this the booth encoder [10] is designed with three basic operator signals.

1. Direction-Direction operator is used to choose either the normal multiplicand(X) or inverse of multiplicand (~X).
2. Shift-Shift operator is shifting the bits by one position to left side.
3. Addition-Addition operator perform addition of one to partial product.

The booth encoding can be simplified using the expressions that follows

Direction $D_m = Y_{m+1}$;

Addition $A_m = Y_{m-1} \oplus Y_m$;

Shift $S_m = Y_{m+1} \oplus Y_m$

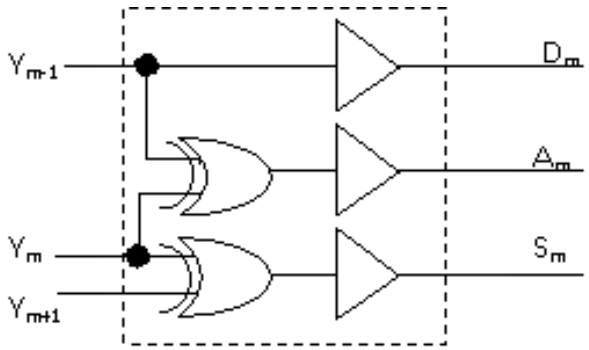


Fig.5. Booth encoder circuit

These signals are given to the partial product generator to produce the partial products based on the operator signal. Totally we are having 53 bits of mantissa, by grouping it as 3 bits; we will get the 27 groups of 3 bits inputs. So for 27 groups of bits all the 3 signals are generated and then it will trigger the partial product generator. 27 rows of partial products are generated according to the signals from the booth encoder.

Partial Product Generator[PPG]

Partial products are the intermediate results in the multiplications which are added to produce the final product.

In this design the partial products are generated based on the signals from booth encoder.

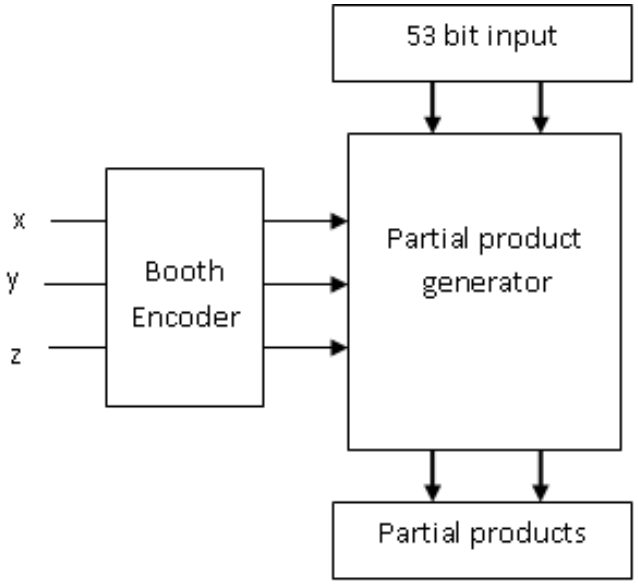


Fig.6. Block diagram of PPG

Here the output from the booth encoder is acting as one of the inputs to PPG and another input is given by the second mantissa value. Based on the encoder input value, the partial product selector has to be considered. Based on these 3 bits of groups, the partial products are produced with help of partial product selectors [6] such as 0, +1,-1, +2,-2. It illustrates how to calculate partial products from the bits of multiplicand B according to the values of the recoded digit.

Table 1. Relationship of partial-product and recoded signed bits

Multiplier bits	Selection
000	+0(0)
001	+1(p1)
010	+1(p1)
011	+2(p2)
100	-2(m2)
101	-1(m1)
110	-1(m1)
111	-0(0)

The computation of partial products given in Table 1 is simple:
For p1, the partial products equal the bits of B.
For p2, we obtain the partial products by a left shift of B.
For m1, we need to invert the bits of B and add the value 1 at the least significant bit.
For m2, we need to invert the bits of B, shift them left, and add the value 1 at the least significant bit.
For the encoded digit equal to 0, all partial products bits are equal to 0.

By doing these operations regarding to the table all the partial products in 27 rows is obtained, because of the 27 groups of bits of inputs to the partial product generator. These partial products are needed to be arranged in the proper format to get the correct result at the output side.

Wallace tree structure
Fast process for multiplication of two numbers was developed by Wallace [9].

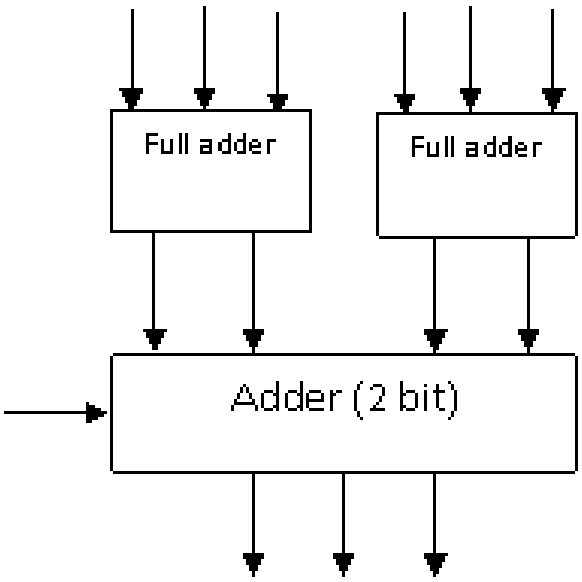


Fig.7. Wallace element

Two step process is used to multiply two numbers:
(1) The bit products are formed.
(2) The bit product matrix is "reduced" to a two row matrix by using carry-save adders.
(3) The last two rows are summed using a fast carry-propagate adder to produce the product.

The Wallace-Tree binary adder is a usual building block in the implementation of the binary multiplier, and is an integral element in the efficient implementation of high-speed binary multipliers.

Proposed Spurious Power Suppression Technique (SPST)
The problem of power consumption can be overcome with the use of different power consumption reduction techniques like Boolean Techniques, Guarded Evaluation, Partially Guarded Computation, Dynamic Range Determination and Glitching Power Minimization in the multiplier circuit. These above techniques has its won trade off like need of additional circuit, low percentage of power consumption reduction. The SPST [25] uses a logic circuit to detect the effective data range of arithmetic units like adders or multipliers. When a portion of data does not affect the final computing results, the data controlling circuits of the SPST latch this portion to avoid useless data transitions occurring inside the arithmetic units.

Besides, there is a data asserting control realized by using registers to further filter out the useless spurious signals of arithmetic unit every time when the latched portion is being turned on. This asserting control brings evident power reduction. Fig.8 shows the design of low power adder/subtractor with SPST.

The adder /subtractor is divided into two parts, the most significant part (MSP) and the least significant part (LSP). The MSP of the original adder/subtractor is modified to include detection logic circuits, data controlling circuits, sign extension circuits, logics for calculating carry in and carry out signals. The most important part of this study is the design of the control signal asserting circuits, denoted as asserting circuits in Fig.8. Although this asserting circuit brings evident power reduction, it may induce additional delay.

There are two implementing approaches for the control signal assertion circuits. The first implementing approach of control signal assertion circuit is using registers and the second using gates. Here the second method is used. The three output signals of the detection logic are close, Carr_ctrl, sign. The three output signals in the detection logic unit are given a certain amount of delay before they assert. This will filter out the glitch signals as well as keep the computation results correct.

Implementation and Discussion

For implementation Xilinx ISE Design Suite 13.1with VHDL programming was used. Simulation process was done using ISIM tool.

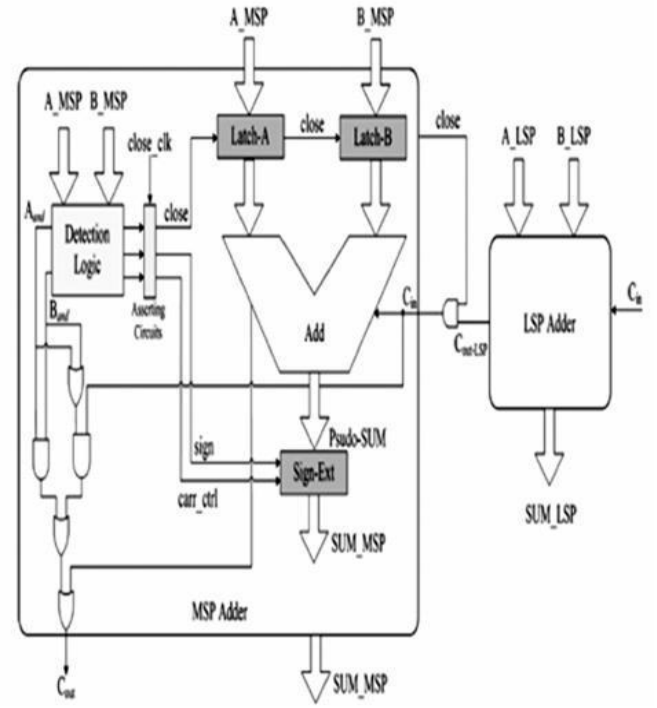
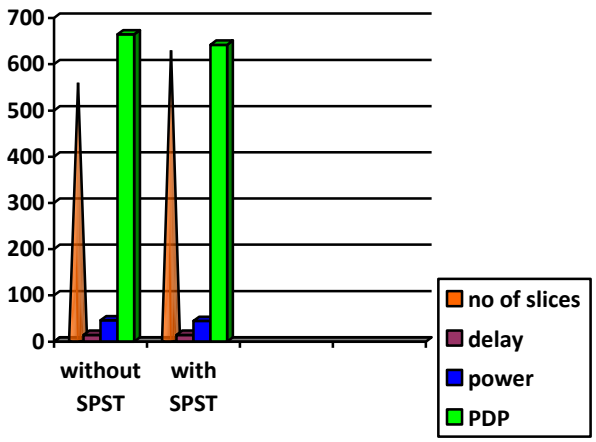


Fig.8. Low Power Adder/Subtractor Adopting SPST

Table 2. Comparison of floating point multiplier using and not using SPST technique

	Number of Slice FFs	Delay (ns)	Power (mW)	PDP (pJ)
Without SPST	555	14. 50	45. 83	664. 5
With SPST	625	14. 40	44. 59	642. 1



Comparison of configurable floating point multiplier using and not using SPST technique is shown in Table 2. The results show that the floating point multiplier with power suppression technique shows improved results than compared to floating point multiplier without using suppression technique with respect to power and delay with acceptable increase in area.

Conclusion

This paper presents a flexible multimode floating-point multiplier for FPGAs. Each floating point multiplier can perform double-precision operation or single-precision operation. Results show that the FPGA with embedded multimode FPUs incorporated with power reduction technique provide considerable performance with respect to power and delay with the benefits seen in single-precision, double-precision, fixed-point, and integer applications.

References

1. A.Akkas¸ and M. J. Schulte, "A quadruple precision and dual double precision floating-point multiplier, " in *Proc. Euromicro Symp. Digit. Syst. Des. (DSD)*, p. 76, 2003.
2. G. Even, S. M. Mueller, and P.M. Seidel, "A dual precision IEEE floating-point multiplier, " *Integr. VLSI J.*, vol. 29, no. 2, pp. 167-180, 2000.
3. P. C. Diniz and G. Govindu, "Design of a field-programmable dual-precision floating-point arithmetic unit, " in *Proc. Int. Conf. Field Program. Logic Appl. (FPL)*, pp. 1-4, 2006.
4. *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Std 754, 1985.

5. D. A. Patterson and J. L. Hennessy, *Computer Organization and Design*, 3rd ed. San Francisco, CA: Morgan Kaufmann, ch. H. 5, 2005.
6. M. Nicolaidis and R. O. Duarte, "Fault-secure parity prediction booth multipliers, " *IEEE Des. Test*, vol. 16, no. 3, Jul., pp. 90-101, 1999.
7. W.C. Yeh and C.W. Jen, "High-speed booth encoded parallel multiplier design, " *IEEE Trans. Comput.*, vol. 49, no. 7, Jul, pp. 692-701, 2000.
8. A.D. Booth, "A signed binary multiplication technique, " *Quarterly J. Mechan. Appl. Math.*, vol. 4, pp. 236-240, 1951.
9. C.S. Wallace, "A suggestion for a fast multiplier, " *IEEE Trans. Electron. Comput*, vol. EC-13, no. 1, Feb., pp. 14-17, 1964.
10. Ki-seon Cho, Jong-on Park, Jin-seok Hong and Goang-seog Choi, "54x54-Bit radix-4 multiplier based on modified booth algorithm", *ACM, GLSVLSI'03*, Washington, DC, USA, pp. 233-236, 2003.
11. Padma Devi, AshimaGirdher and Balwinder Singh. "Improved carry select adder with reduced area and low power consumption", *International Journal of Computer Applications*, vol. 3, no. 4, 2010.
12. Sudharsanarani B. and Vijayakumarraju V. "Reducing the size of partial product array in two's complement multipliers", *International Journal of Logic and Computation (IJLP)*, vol. 5, issue 2, pp. 714-727, 2012.
13. Vojin G. Oklobdzija "High-Speed VLSI arithmetic units: Adders and Multipliers", Sep 1999.
14. Wen M. C., Wang S. J. and Lin Y. N. "Low-power parallel multiplier with column bypassing", *Electronics letter*, vol. 41, no. 10, 2005.
15. Yee Jern Chong and Sri Parameswaran "Configurable multimode floating point units for FPGAs*", IEEE Trans. VLSI*, vol. 19, no. 11, Nov, pp. 2033-2044, 2011.
16. Y. Dou, S. Vassiliadis, G. Kuzmanov, and G. Gaydadjiev, "64-bit floating point FPGA matrix multiplication, " in Proc. ACM/SIGDA13thInt. Symp. Field-Program. GateArrays, pp. 86-95, 2005.
17. K. S. Hemmert and K. D. Underwood, "Analysis of the double-precision floating point FFT on FPGAs, " presented at the ACM Int. Symp. Field Program. Gate Arrays, Monterey, CA, Feb 2004.
18. G. Govindu, S. Choi, V. K. Prasanna, V. Daga, S. Gangadharpalli, andV. Sridhar, "Ahigh-performance and energy efficient architecture for floating-point based LU decomposition on FPGAs, " in Proc. 11th Reconfigurable Arch. Workshop(RAW), SantaFe, NM, Apr, p. 149a, 2004.
19. M. deLorimer and A. DeHon, "Floating point sparse matrix-vector multiply or FPGAs, " in Proc. ACM Int. Symp. Field Program Gate Arrays, Monterey, CA, Feb, pp. 75-85, 2005.
20. Convey Computer Corporation, "Convey computer, " Richardson, TX, [Online]. Available: http:// www. conveycomputer. com/2008-2010.
21. K. Underwood, "FPGAs vs. CPUs: Trends in peak floating-point performance, " in Proc. ACM/SIGDA 12th Int. Symp. Field Program. Gate Arrays, Monterey, CA, Feb. pp. 171-180, 2004.
22. M. J. Beauchamp, S. Hauck, and K. S. Hemmert, "Embedded floating-point units in FPGAs, " in Proc. IEEE Symp. Field Program. Gate Arrays (FPGA), pp. 12-20, 2006.
23. C. H. Ho, P. H. W. Leong, W. Luk, S. J. E. Wilton, and S. Lopez-Buedo, "Virtual embedded blocks: A methodology for evaluating embedded elements in FPGAs, " in Proc. IEEE Symp. Field-Program. Custom Comput. Mach. (FCCM), pp. 35-44, 2006.
24. C. H. Ho, C. W. Yu, P. H. W. Leong, W. Luk, and S. J. E. Wilton, "Domain-specific hybrid FPGA: Architecture and floating point applications, " in Proc Int. Conf. Field Program. Logic Appl. (FPL), pp. 196-201, 2007.
25. Kuan-Hung Chen and Yuvan-Sun Chu, " A Low Power Multiplier with the Spurious Power Suppression Technique" *IEEE Trans.* VLSI systems, vol. 15, no. 7, July 2007.