Multimode FPGAwith Flexible Embedded FPUS

Dr. G. MURUGABOOPATHI[1], S.HARIHARASITARAMAN [2] AND G.SANKAR[3]

[1]Head of Research & Development, Vel Tech Multi Tech Dr.Rangarajan Dr.Sakuntha Engineering College, Chennai, Tamil Nadu, India.

[2]Assistant Professor,Department of IT , Kalasalingam University, Srivilluputur, Tamil Nadu, India.

[3]Assistant Professor,Department of CSE, Vel Tech Hghi Tech Dr.Rangarajan Dr.Sakuntha Engineering College, Chennai, Tamil Nadu, India.

Abstract
The Performance of field-programmable gate arrays used for Floating-point applications are poor due to complexity of floating-point arithmetic. Implementing floating-point units on FPGAs consume a large amount of resources. This makes FPGAs less attractive for use in floating-point intensive applications. There is a need for embedded FPUs in FPGAs. We proposed a flexible multimode embedded FPU for FPGAs that can be configured to perform a wide range of operations. The floating-point adder and multiplier in embedded FPU can be configured to perform one double-precision operation or two single-precision operations in parallel. To increase flexibility, access to large integer multiplier, adder and shifters in the FPU is provided. Benchmark circuits were implemented on both a standard Xilinx Virtex-V FPGA and FPGA with embedded FPU blocks. We design modified to allow an unrounded product to be fed to the floating-point adder to minimize rounding error, like in a dedicated floating-point MAC unit.

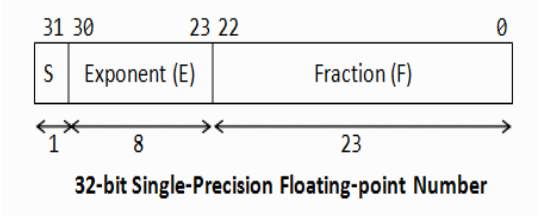
Keywords: Very Large Scale Integration Embedded Floating Point Units, Virtual Floating Point Unit, FPGA.

1. Introduction

The term floating point refers to the fact that the radix point (decimal point, or, more commonly in computers, binary point) can "float"; that is, it can be placed anywhere relative to the significant digits of the number. This position is indicated separately in the internal representation, and floating-point representation can thus be thought of as a computer realization of scientific notation. Over the years, a variety of floating-point representations have been used in computers. However, since the 1990s, the most commonly encountered

representation is that defined by the IEEE 754 Standard[4].

The IEEE754 standard floating-point format consists of three fields—a sign bit(s) , a biased exponent(e) , and a mantissa(f) . Single-precision numbers have a 1-bit sign, 8-bit exponent, and 23-bit mantissa as shown in Fig. 1. Double-precision numbers have a 1-bit sign, 11-bit exponent, and 52-bit mantissa as shown in Fig. 1.



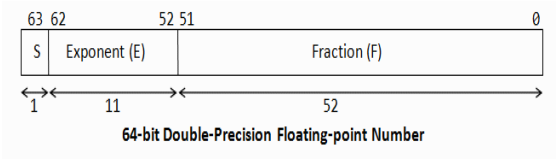


Fig.1 IEEE floating point formats.

The three fields make up a floating-point number according to Eq.(1) single-precision and Eq.(2) double-precision. There is an implied "1" to the left of the binary point (except in the special case of denormal numbers)

$$X=(-1)^s \times 1.f \times 2^{(e-127)} \quad (1)$$

$$X=(-1)^s \times 1.f \times 2^{(e-1023)} \quad (2)$$

Floating-point numbers have an advantage of being able to cover a much larger dynamic range compared

to fixed-point numbers. The disadvantage is that floating-point computations are much more complex to implement in hardware.

2. Modelling

The proposed architecture for the FPGA with embedded multimode FPUs is an island-style FPGA structure based on the Xilinx Virtex-V. The embedded FPUs would be distributed in a regular arrangement around the FPGA [2], surrounded by fine-grained configurable logic blocks (CLBs) as illustrated in Fig.2.

The number and arrangement of the the embedded FPUs would be decided by the FPGA vendor and would depend on the size of the FPGA. In this work, embedded FPUs are placed evenly spaced in two columns near the center of the FPGA.

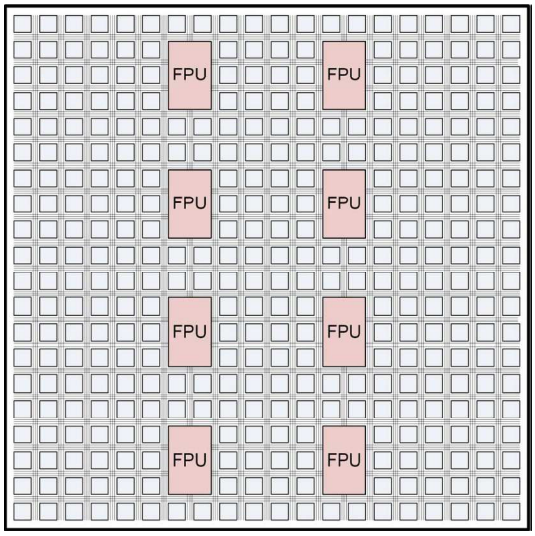


Fig 2. Architecture of FPGA with embedded FPU blocks.

3. Embedded FPU blocks

The FPU block contains one floating-point multiplier and one floating-point adder. These can be used independently, or configured in a multiply-add configuration by enabling a bus connecting the output of the multiplier to an input of the adder.

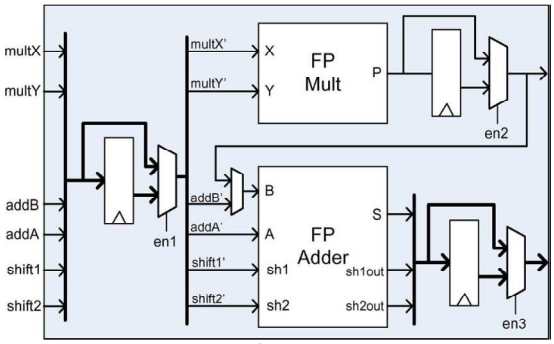


Fig 3. Structure of embedded FPU block

Fig.3 shows the structure of the embedded FPU block. Optional registers are available at the inputs and outputs of the FPU block to allow for easy implementation of pipelined or multicycle circuits. To increase the usefulness of the floating-point units, several key integer components within the floating-point units were made accessible. These components are as follows.

1. Dual-mode 53 53-bit integer multiplier in the floatingpoint multiplier. Can be configured as two independent 24×24-bit multipliers.
2. Dual-mode 53-bit integer adder in the floating-point adder. Can be configured as independent 27-bit and 26-bit adders.
3. The 106-bit right shifter in the pre-alignment stage of the floating-point adder. Since maximum shift is 64 places, access is given to 64-bits of the shifter so that it appears to be a 64-bit shifter with maximum shift of 64 places.
4. The 54-bit left shifter in the normalization stage of the floating-point adder. Maximum shift is 54 places.

3.1 Floating Point Adder Datapath

The conventional floating-point addition algorithm consists of five stages—exponent difference, pre-alignment, addition, normalization and rounding .

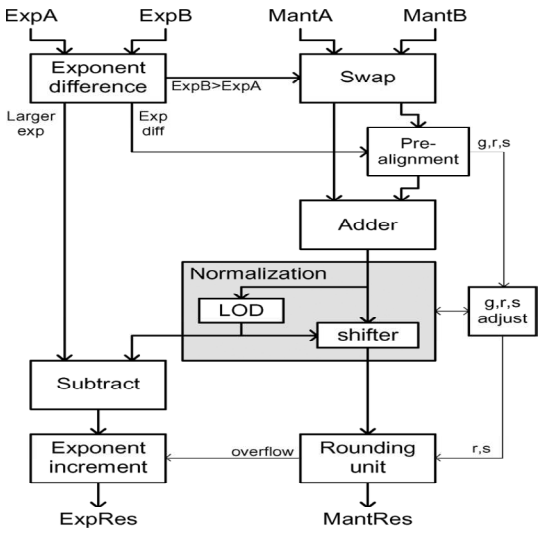


Fig 4..Floating point adder datapath.

Given floating-point numbers $X_1=(s_1,e_1,f_1)$ and $X_2=(s_2,e_2,f_2)$, the stages for computing X_1+X_2 are described as follows.

1) Find exponent difference $d= e_1-e_2$. If $e_1<e_2$, swap position of mantissas. Set larger exponent as tentative exponent of result.

2) Prealign mantissas by shifting smaller mantissa right by 'd' bits.

3) Add or subtract mantissas to get tentative result for mantissa.

4) Normalization. If there are leading-zeros in the tentative result, shift result left and decrement exponent by the number of leading zeros. If tentative result overflows, shift right and
increment exponent by 1 bit.

5) Round mantissa result. If it overflows due to rounding, shift
right and increment exponent by 1 bit.

Fig.4. shows the datapath for a floating-point addition. Only the main parts of the datapath are shown for clarity. The prealignment and normalization stages require large shifters. The prealignment stage requires a right shifter that is twice the number of mantissa bits (i.e., 48 bits for single-precision, 106 bits for double-precision) because the bits shifted out have to be maintained to generate the guard, round and sticky bits needed for rounding. The shifter only needs to shift right by up to 24 places for single-precision or 53 places for double-precision.

The normalization stage requires a left shifter equal to the number of mantissa bits plus 1 (to shift in the guard bit), i.e., 25-bits for single-precision and 54-bits for double-precision.

The shift amount is determined by the leading one detector (LOD) circuit [7], which outputs the number of leading zeros before the first one in the bit string.

The final stage of the floating-point adder is the rounding unit. It makes a rounding decision based on the rounding mode, the LSB of the mantissa, the round bit and the sticky bit. If rounding is necessary, "1" is added at the LSB of the mantissa. There are other variations of the conventional floating-point adder architecture that improve performance, such as the leading one predictor (LOP) architecture [8] and the dual-path architecture [9]. The tradeoff involved with these two architectures is that they require additional hardware and area for them added performance. The conventional architecture was chosen over the faster architectures for area savings and reduced complexity, which simplifies the conversion to a dual-precision structure.

3.2 Floating Point Multiplier Datapath*:*

Algorithmically, floating-point multiplication is much simpler than floating-point addition. However, a very wide integer multiplier is required. Given floating-point numbers $X_1=(s_1,e_1,f_1)$ and $X_2=(s_2,e_2,f_2)$, $X_P=X_1+X_2$
and , can be computed using,

$$S_P=S_1 \oplus S_2 \quad (3)$$

$$e_p=e_1+e_2-bias \quad (4)$$

$$1.f_p=1.f_1\times 1.f_2 \quad (5)$$

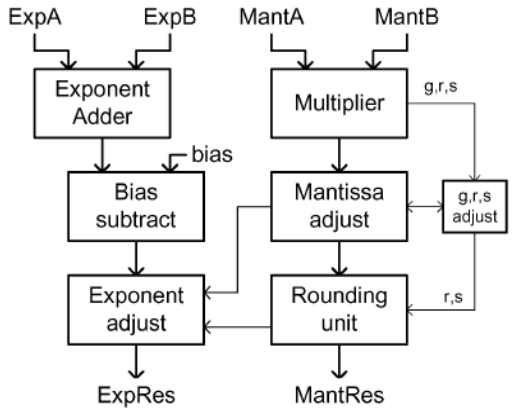


Fig 5. Floating point multiplier datapath.

Fig.5. shows the datapath for a floating-point multiplier. Only the main parts of the datapath are shown for clarity. If the result from the multiplier has two bits left of the binary point, the mantissa has to be shifted right to compensate and the exponent is incremented. If the rounding of the mantissa results in an overflow, the mantissa is shifted right by one and the exponent is incremented. Eq.(5) calls for a very wide multiplier—53×53-bit unsigned multiplier for double-precision and 24×24-bit for single-precision. Therefore, an efficient multiplier must be employed.

4. Multimode FPU:

The multimode embedded FPU was designed to include a dual-precision floating-point multiplier and a dual-precision floating-point adder. They were designed to be IEEE754 compliant , except that hardware support for denormals was not included and only the IEEE754 default rounding mode (round-to-nearest even) was implemented.

The design can be easily modified to support the other rounding modes specified in IEEE754. The dual-precision FPU accepts 64-bit inputs, where double-precision operands occupy the full 64-bits and single-precision operands each occupy half of the 64-bits as shown in Fig.6.

The following two sub-sections explain the modifications made to a standard floating-point adder and multiplier to convert them into dual-precision versions capable of performing one double-precision operation or two single-precision operations in parallel.

4.1 Dual Precision Floating Point Adder

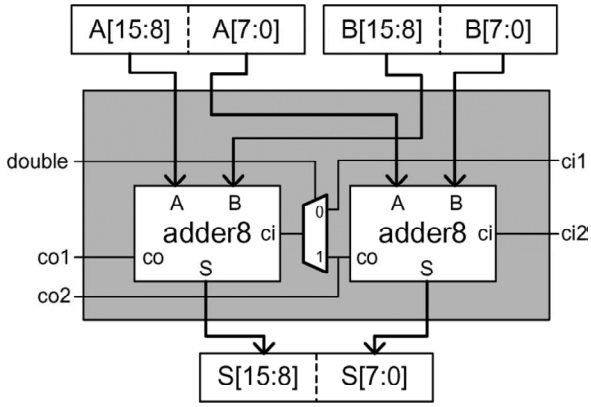


Fig.6. 2X 8-bit/1 X 16-bit adder—capable of performing two independent 8-bit additions in parallel or one 16-bit addition.

The method for converting a standard floating-point adder into a dual-precision adder involves duplicating the data path for a single-precision adder and then linking duplicated functional blocks together (and widen them where necessary) to accommodate double-precision. Multiplexers controlled by a mode signal (double) selects between single-precision mode and double-precision mode. A double-precision exponent is 11-bits, while a single-precision exponent is 8-bits. For all the operations on the exponents that involve adding or subtracting, we use two 8-bit adders that can combine into one 16-bit adder fig7.

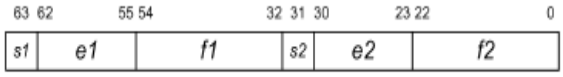


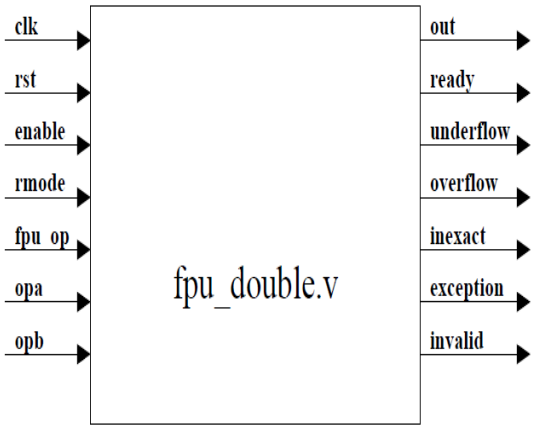
Fig .7. Two single-precision numbers in one 64-bit word.

5. Results

5.1.FPU Double Top Module

To evaluate the proposed FPGA architecture, Xilinx Virtex-V FPGA device is used. Xilinx ISE 9.2i(or

more) is used for synthesis and codings are done in verilog. Both single-precision and double-precision versions of each circuit were built in order to evaluate the multimode embedded FPUs in both precision modes. FPU double top level module is shown in the figure below,



5.2. Input Signals:

1. clk (global)
2. rst (global)
3. enable (set high to start operation)
4. rmode (rounding mode, 2 bits, 00 = nearest, 01 = zero, 10 = pos inf, 11 = neg inf)
5. fpu_op (operation code, 3 bits, 000 = add, 001 = subtract, 010 = multiply, 011 = divide, others are not used)
6. opa, opb (input operands, 64 bits)

5.2. Operation and Rounding:

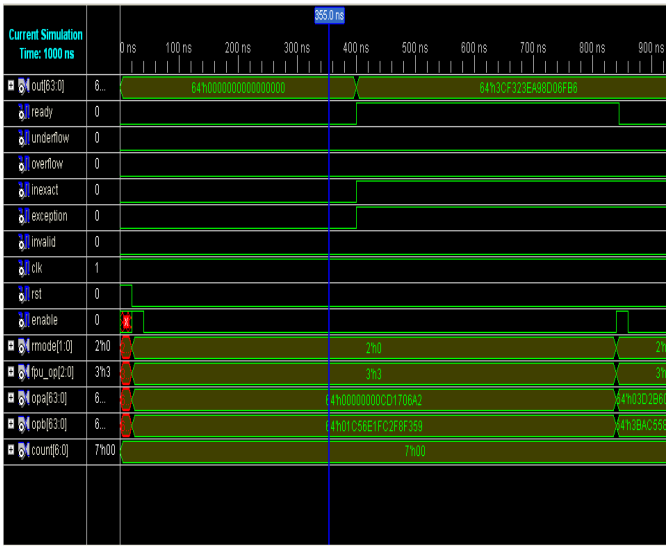
FPU Operations (fpu_op):

1. 0 = add
2. 1 = sub
3. 2 = mul
4. 3 = div

Rounding Modes (rmode):

1. 0 = round_nearest_even
2. 1 = round_to_zero
3. 2 = round_up
4. 3 = round_down

5.3. Simulation Results:



6. Conclusions

Flexible multimode embedded floating-point unit for FPGAs is designed. Each embedded FPU contains a dual-precision floating-point adder and multiplier, which can each perform one double-precision operation or two single-precision operations in parallel. To further increase flexibility of the embedded FPU, access to integer components of the FPU are provided.

Different round of modes are added as per IEEE754, hence the round off errors are eliminated finally .This elimination will makes our floating point unit faster and efficient when compared with existing architecture. Our design is synthesised and simulated using Xilinx ISE.

References

[1] Y. Dou, S. Vassiliadis, G. Kuzmanov, and G. Gaydadjiev, “64-bit floating-point FPGA matrix

multiplication," in Proc. ACM/SIGDA 13th Int. Symp. Field-Program. Gate Arrays, 2005, pp. 86–95.
[2]M. J. Beauchamp, S. Hauck, and K. S. Hemmert, "Embedded floatingpoint units in FPGAs," in Proc. IEEE Symp. Field Program. Gate Arrays (FPGA), 2006, pp. 12–20.
[3] A. Akkas¸, "Dual-mode quadruple precision floating-point adder," in Proc. 9th Euromicro Conf. Digit. Syst. Des. (DSD), 2006, pp. 211–220.
[4] IEEE Standard for Binary Floating-Point Arithmetic,ANSI/IEEE Std754, 1985.

[5] P. C. Diniz and G. Govindu, "Design of a field-programmable dual-precisionfloating-point arithmetic unit," in Proc. Int. Conf. Field Program. Logic Appl. (FPL), 2006.
[6] D. A. Patterson and J. L. Hennessy, "Computer Organization and Design", 3rd ed. San Francisco, CA: Morgan Kaufmann, 2005, ch. H.5.
[7] V. G. Oklobdzija, "An algorithmic and novel design of a leading zero detector circuit: Comparison with logic synthesis," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 2, no. 1, pp. 124–128, Mar. 1994.
[8] H. Suzuki, H. Morinaka, H. Makino, Y. Nakase, K. Mashiko, and T. Sumi, "Leading-zero anticipatory logic for high speed floating-point addition," IEEE J. Solid-State Circuits, vol. 31, no. 8, pp. 157–1164, Aug. 1996.
[9] M. Farmwald, "On the design of high performance digital arithmetic units," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, Aug. 1981.
[10] Xilinx, San Jose, CA, "Virtex-II platform FPGAs: Complete data sheet," 2007. [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds031.pdf

G.Murugaboopathi received the Undergraduate Degree in Computer Science and Engineering from Madurai Kamaraj University, in 2000, the Post Graduate degree in Digital Communication and Network from Madurai Kamaraj University, in 2002 and Ph.D in Computer Science and Engineering at Bharath University, Chennai. He has more than 17 publications in National Conferences International Conference and International Journal proceedings. He has more than 10 years of teaching experience. His areas of interest include Wireless Sensor Networks, Mobile Communication, Mobile Adhoc Networks Computer Networks, Network Security, High Speed Networks, Network and Data Security Cryptography and network security DBMS and etc., He is currently working as an Head R & D and Associate Professor in the Department of Information Technology at Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering College Chennai, India.

S.Hariharasitaraman received his B.E. degree in Computer Science and Engineering from Madurai Kamaraj University, Tamilnadu, India, in 2003, M.E. degree in Computer Science and Engineering from Anna University, Tamilnadu, India, in 2005. He is working as Assistant professor, in the Department of Information Technology, Kalasalingam University y. His research interests include Distributed Computing, Cloud Computing. At present, He is engaged in the area of Security Mechanisms in Cloud Computing.

Mr.G.Sankar is working as Assistant Professor in the CSE department of veltech Hightech EngineeringCollege,Chennai.Mr.Sankar has Masters of engineering degree in computer science and engineering from annamalai university, Master of business administration in project management from alagappa university and Bachelor of engineering degree in computer science and engineering from bharathidasanuniversity.Mr.sankar has over seven and half years of teaching and research experience. He has authored several research papers in national and international Conference