2009 17th IEEE Symposium on Field Programmable Custom Computing Machines

FPGA Implementation of a Single-Precision Floating-Point Multiply-Accumulator with Single-Cycle Accumulation

Arun Paidimarri
Indian Institute of Technology, Bombay
arun_p@iitb.ac.in

Alessandro Cevrero, Philip Brisk, Paolo Ienne
École Polytechnique Fédérale de Lausanne
First_name.Last_name@epfl.ch

Abstract

This paper describes an FPGA implementation of a single-precision floating-point multiply-accumulator (FPMAC) that supports single-cycle accumulation while maintaining high clock frequencies. A non-traditional internal representation reduces the cost of mantissa alignment within the accumulator. The FPMAC is evaluated on an Altera Stratix III FPGA.

1. Introduction

This paper describes an FPGA implementation of a single-precision floating-point multiply-accumulator (FPMAC), originally designed by Intel [1]. Single-cycle accumulation with minimal frequency degradation is achieved by using a non-traditional internal FP representation that sacrifices IEEE compliance in its exception handling, rounding, and detection of overflow. The accumulator retains the result in carry-save form without normalization or rounding. The FPMAC is evaluated on an Altera Stratix III FPGA.

2. FPMAC Architecture

An IEEE-754 single-precision (32-bit) FP number has three fields—a sign bit, an 8-bit biased exponent, and a 23-bit fraction called a mantissa. We assume that the reader is familiar with the IEEE standard for floating-point numbers. The FPMAC does not support denormalized numbers, infinity, or not-a-number (NaN). If *f* is the mantissa and *exp* is the biased exponent; the value of an IEEE-754 compliant FP number is $(-1)^{sign} \times (1.f) \times 2^{exp}$. The *1* is not represented in the format, but exists internally in the FPMAC's computation logic. Thus, the bitwidth of the mantissa within the FPMAC is actually 24 bits.

The FPMAC, shown in Fig. 1, has 11 pipeline stages. The notable features of this architecture are the "Base-32" conversion and de-conversion (Stages 6 and 11) and the accumulator (Stage 8). The 48-bit mantissa multiplier outputs produced in Stage 5, and retained in carry-save form, are converted to a non-traditional FP representation shown in Fig. 2.[1] The mantissa is truncated to 24 bits, 22 to the right of the decimal point and 2 to the left. The truncated mantissas are shifted left, as specified by the 5 least significant bits of the exponent, expanding them from 24 to 55 bits: 33 bits to the left of the decimal and 22 bits to the right. Afterwards, only the three most significant bits of the exponent are needed. The value of the FP number in this format is $(-1)^{sign} \times f^* \times 2^{(32 \times exp^*)}$, where f^* is the shifted mantissa and exp^* is the truncated exponent.

The accumulator, shown in Fig. 3 and Table I, is the critical path. The Base-32 representation eliminates shifters used for mantissa alignment, which are replaced with less costly conditional constant shifters. The outputs of the sign inversion stage are the incoming exponent (E_7) and mantissa (C_7 and S_7). The accumulated result is a feedback exponent (E_8) and mantissa (C_8 and S_8). The updated exponent and mantissa are also stored to E_8, C_8 and S_8. The incoming and feedback mantissas are shifted by the difference between the incoming and feedback exponents. The exponents must be equalized before mantissa addition. The Base-32 representation ensures that all shifts are by 0 or 32 bits. Four cases are computed in parallel. Table I lists the conditions that resolve the paths. Fig. 4 shows that two 4:2 compressors can map onto three Adaptive Logic Modules (ALMs), the logic cell used in Altera's Stratix II-IV FPGAs.

[1] Intel [1] refers to this non-traditional FP format as "*Base-32*". This name is somewhat misleading, as it should indicate that the base of the exponentiation in the non-traditional format is 2^{32}, while the 3-bit exponent remains in radix-2. To avoid confusion between the two papers, we retain Intel's nomenclature.

978-0-7695-3716-0/09 $25.00 © 2009 IEEE
DOI 10.1109/FCCM.2009.50

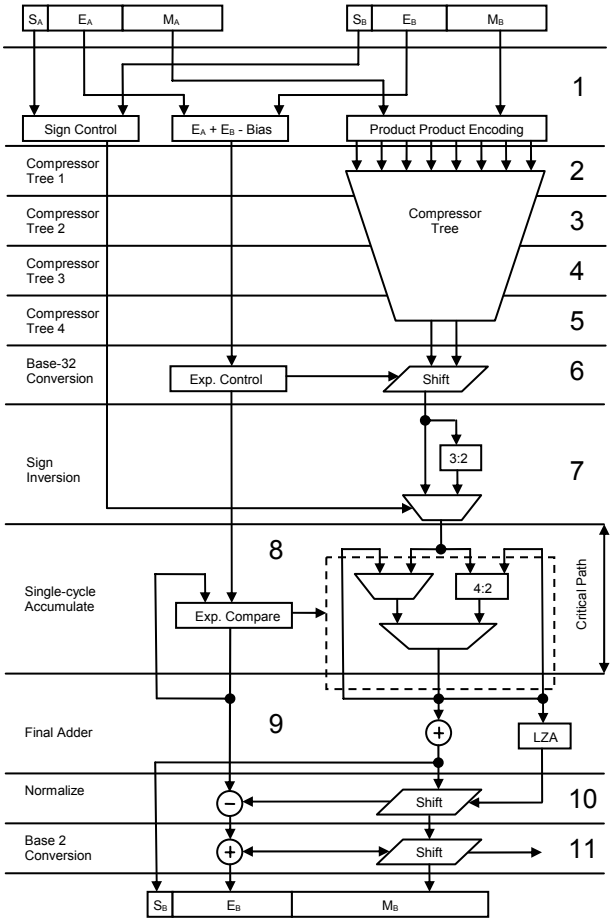

Figure 1. FPMAC Pipeline

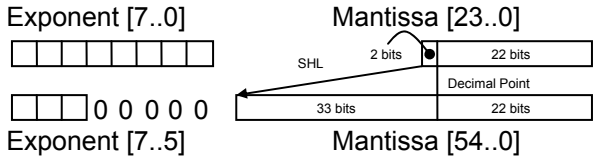

Figure 2. Base-32 Conversion

Path A: If the exponents differ by at most one, the mantissa of the smaller value is shifted right by 32 bits before the addition, and its exponent is incremented to equalize them; shifting is suppressed if the difference is zero. If overflow occurs, the 4:2 compressor output is shifted right and the exponent is incremented.

Path B: If the exponents differ by more than 1, the mantissa of the smaller number is shifted by an integer multiple of 32 bits, and the exponent is adjusted appropriately. Since the mantissas are 55 bits, this sets the smaller mantissa to zero; the larger value (exponent and carry-save mantissa) can be selected by the output.

Path C: The accumulated result is never explicitly normalized. If the number of leading zeroes or ones in $S_8 + C_8$ exceeds 31, then S_8 and C_8 are shifted left by 32 bits. If $E_8 = E_7 + 2$, S_7 and C_7 are shifted right by 32 bits as well, and E_7 is adjusted accordingly. A 4:2 compressor adds the result, which is shifted right (along with an appropriate exponent adjustment) if there is an overflow.

Path D: If $S_8 + C_8 = 0$, the result is the incoming mantissa, S_7, C_7, i.e., $E_8 = E_7$, $S_8 = S_7$, and $C_8 = C_7$. This case is handled separately to handle mismatches between the incoming and feedback exponents.

Paths A and C of the accumulator are critical, requiring 6 layers of 6-LUTs. The paths through the leading zero anticipator (LZA) and the zero detector, in contrast, require 5 layers of ALMs. One ALM can be removed from the delay of Paths A and C by dividing the datapath, as shown in Fig. 5 for Path A. In Fig. 3(a), there are three combinations of shifts on the inputs to the 4:2 compressors; all shifts are conditional 32-bit right shifts. The three cases are: (1) no shifting; (2) shift (S_7, C_7), but not (S_8, C_8); and (3) shift (S_8, C_8) but not (S_7, C_7). All three combinations are enumerated in Fig. 5, and the result is added by three 4:2 compressors in parallel, and a 3:1 multiplexor selects the result. Overflow detection and conditional shifting of the mantissa is applied to the multiplexor output.

This design removes the 3-bit comparator from the critical path. Replacing the conditional shifter (a 2:1 multiplexor) on each path is offset by the introduction of the 3:1 multiplexor; the benefit comes from resolving the conditional shifting *after* the 4:2 compressors. The 3-bit comparison can be performed in parallel with the 4:2 compressors, eliminating their dependence in Fig. 3(a). The 3-bit comparator now computes the two multiplexor control signals, rather than the 3 output bits shown in Fig. 3(b).

3. Experimental Results

Six versions of the FPMAC were synthesized on a Stratix III FPGA using Altera's Quartus II software; they are compared with an FPMAC built from Altera's addition and multiplication Megafunctions. The results are shown in Table II. Three mantissa multipliers are considered: soft logic using traditional partial product generation and Booth encoding, and DSP blocks; each is evaluated with and without the "Divided Datapath" optimization in Fig. 5. Dividing the datapath increases the clock frequency in all cases, but also increases the number of adaptive LUTs (ALUTs) required.

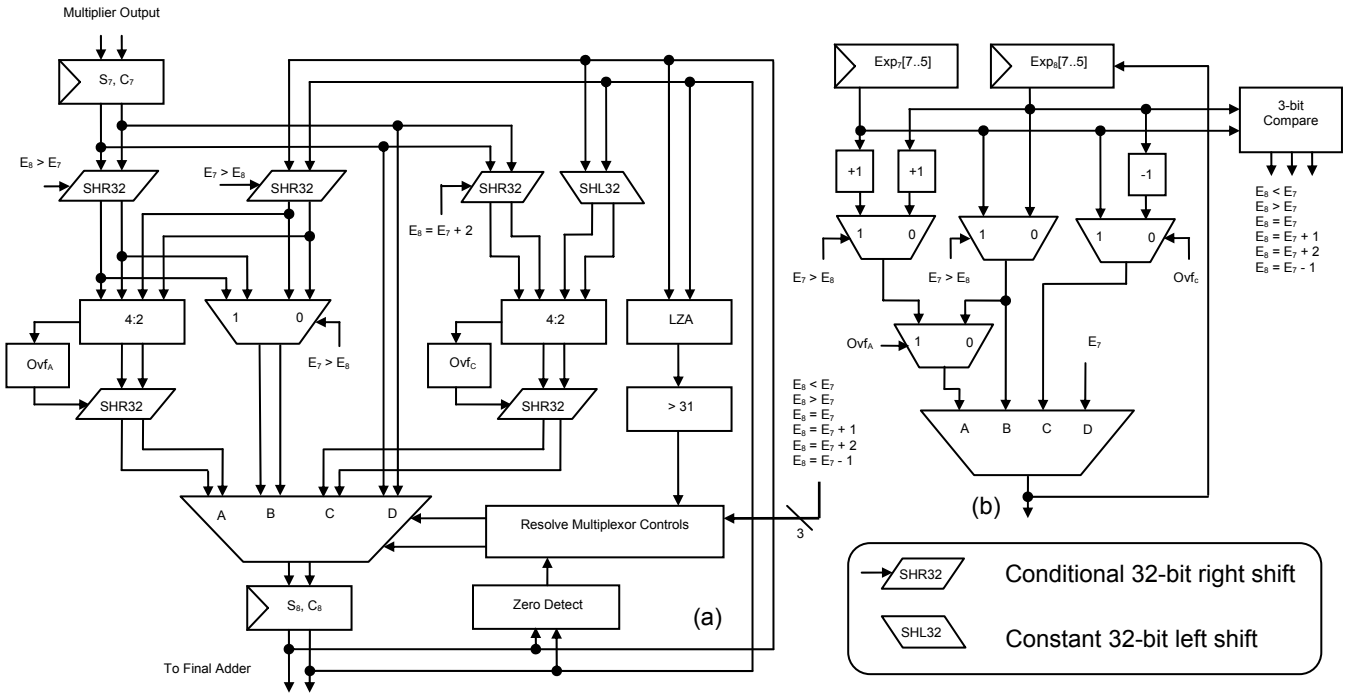

Figure 3. Single-cycle accumulator datapath for the mantissa (a) and exponent (b).

Table I. Multiplexer control resolution logic

Path	Zero-Detect	Conditions
D	1	-
C	0	($E_8 = E_7 + 1$ or $E_8 = E_7 + 2$) and LZA > 31
B	0	$E_8 > E_7 + 2$ or $E_7 > E_8 + 2$ or ($E_8 = E_7 + 2$ and LZA $\leq$ 31)
A	0	Otherwise: i.e., $E_8 = E_7$ or $E_7 = E_8 + 1$ or ($E_8 = E_7 + 1$ and LZA $\leq$ 31)

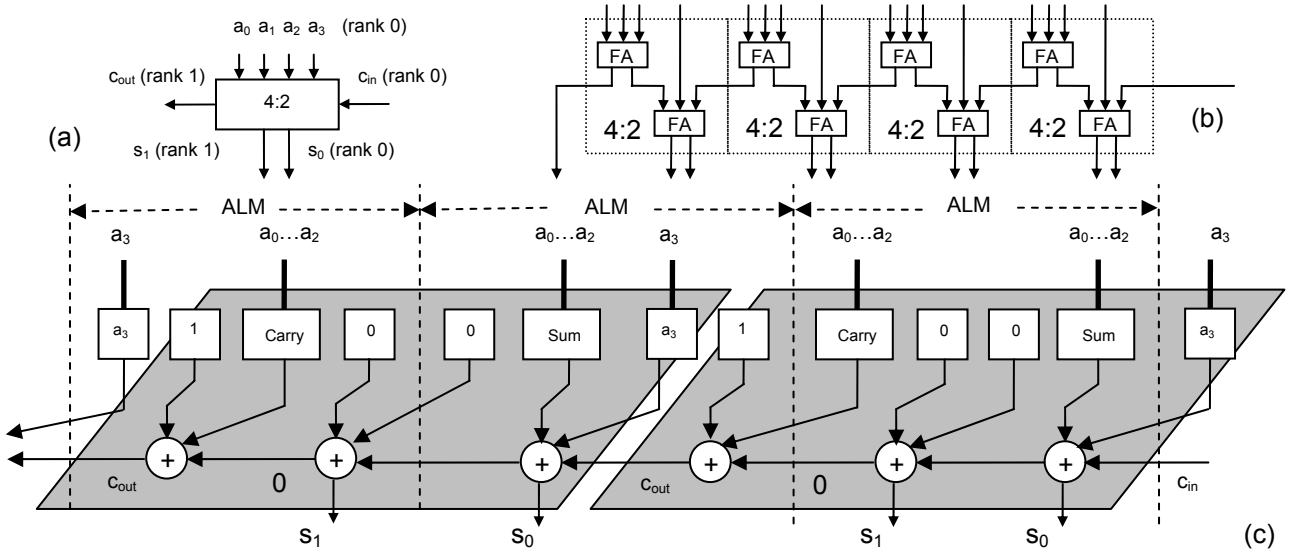

Figure 4. (a) 4:2 compressor I/O diagram; (b) the interconnection scheme of an array of 4:2 compressors (the critical path includes just two full adders); (c) two 4:2 compressors can map onto three ALMs in shared arithmetic mode. The rank of a bit is its position, e.g., the least significant bit is rank 0, the second least significant bit is rank 1, etc.

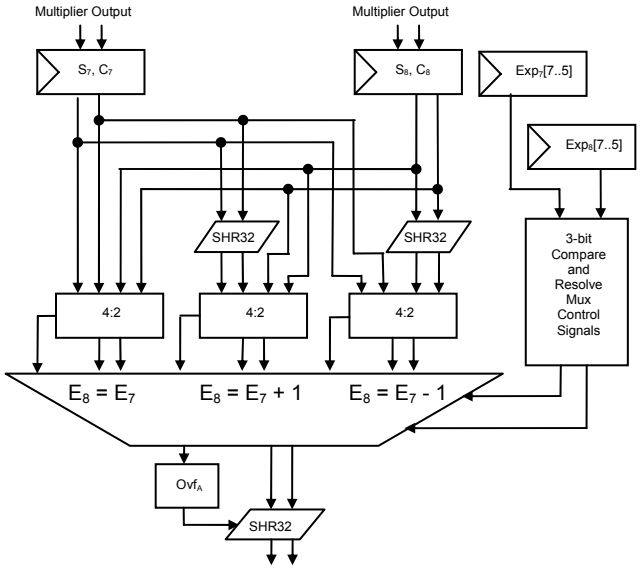

Figure 5. Path A from Fig. 3(a) optimized for delay reduction on an FPGA; Path C is reorganized similarly.

Switching from partial product generation (PPG) to Booth Encoding and DSP blocks reduces the number of pipeline stages in the design, but degrades the clock frequency. The clock frequencies range from 221-263MHz; these are typical frequencies for complete designs running on Stratix III FPGAs.

In contrast, the FPMAC built from Altera's Megafunctions can achieve a frequency of 375MHz, but require 14 cycles for accumulation; however, they require considerably fewer ALUTs than ours. A complete system employing this FPMAC is unlikely to achieve a frequency of 375 MHz.

4. Related Work

Luo and Martonosi designed an FPMAC using carry-save arithmetic in the accumulator and a delayed final adder [2]; Intel extended their design with the non-traditional internal representation to reduce shifting in the accumulator [1]. The FPMAC presented here is based on Intel's design.

de Dinechin et al. [3] developed a variable-precision FP accumulator without internal shifting. The accumulator includes a carry-propagate adder with a partial carry-save representation when the bitwidth constrains the critical path.

Altera's floating-point datapath compiler [4] uses an extended mantissa representation, to reduce the amount of shifting; the goal is to extend the mantissa bitwidth to match that of the DSP blocks. It also eliminates normalization between dependent FP operations when possible.

5. Conclusion

This paper has described an FPGA-synthesizable FPMAC that achieves single-cycle accumulation with minimal clock frequency degradation by sacrificing IEEE-754 compliance in its rounding and error handling. It is appropriate for use in designs where IEEE compliance is not mandatory and the peripheral circuitry can stream two numbers to multiply every cycle into the FPMAC.

6. References

[1] S. R. Vangal, Y. V. Hoskote, N. Y. Borkar, and A. Alvandpour, "A 6.2-GFlops floating-point multiply-accumulator with conditional normalization." *IEEE Journal of Solid State Circuits*, vol. 41, no. 10, Oct. 2006, 2314-2323.

[2] Z. Luo, and M. Martonosi, "Accelerating pipelined integer and floating-point accumulations in configurable hardware with delayed addition techniques." *IEEE Trans. Computers*, vol. 49, no. 3, March, 2000, 208-218.

[3] F. de Dinechin, B. Pasca, O. Cret, and R. Tudoran, "An FPGA-specific approach to floating-point accumulation and sum-of-products." *Int. Conf. Field-Programmable Technology*, Dec. 2008, pp. 33-40.

[4] M. Langhammer, "Floating point datapath synthesis for FPGAs," *Int. Conf. Field Programmable Logic and Applications*, Sept. 2008, pp. 355-360.

Table II. Comparison of different implementations of the FPMAC and implementations based on

Mantissa Multiplier	Divided Datapath	Freq. (MHz)	Pipeline Stages	Area (ALUTs)	DSP Blocks
PPG	Yes	263	11	3437	0
	No	234	11	3141	0
Booth Encoding	Yes	259	10	3037	0
	No	230	10	2883	0
DSP Blocks	Yes	234	9	1924	6
	No	221	9	1890	6

Single-cycle accumulation latency

Soft Logic Megafunction		
Freq. (MHz)	Area (ALUTs)	DSP Blocks
375	1207	0

DSP Block Megafunction		
458	691	4

14-cycle accumulation latency
25 pipeline stages