Datasheet of FP FMAC

Lei Li (*lile@iis.ee.ethz.ch)*

28/07/2017

1. Algorithms and functions

The design supports IEEE 754 for single precision. The three basic components in IEEE 754 are sign(S), exponent(E) and mantissa(M).

Table 1 IEEE 754 for single precision

Precision	Sign	Exponent	Mantissa	Bias
Single	1[31]	8[30:23]	23[22:0]	127

$$= (-1)^s \times (1.M)^E$$

Table 2 Special bit patterns in IEEE 754

	M=0	M≠0
E=0	0	Denormalized with real exponent=1
E=255	±∞	NaN

The IEEE 754 2008 standard supports all floating point operations. It handles all special inputs including signaling NaN, quiet NaN,+Infinity,-Infinity, positive zero and negative zero. Our design supports IEEE 754 and offers two exceptions namely overflow(OF) and underflow(UF). Besides, our design supports four different rounding modes: RNE(Round to Nearest, ties to Even, 00), RTZ(Round towards Zero, encoding as 01), RDN(Round Down, encoding as 10), RUP(Round Up, encoding as 11).
This document is organized as follows: Chapter 2 will summarize all inputs/outputs. Chapter 3 will introduce the architecture. Chapter 4 will address normalization. Chapter 5 will show the rounding modes. Chapter 6 will present exceptions. Chapter 7 will provide some waveforms for simulations. Chapter 8 will give the synthesized results.

2. Inputs and Outputs

fmac is the name of our design, which can be used to perform floating point fused multiply-add: Operand_a_DI + Operand_b_DI * Operand_c_DI. The input RM_SI is a 2-bit rounding mode.

Table 3 Inputs and outputs of fmac

Ports	width	direction	Function
Precision_ctl_SI	5	IN	Reserved for transprecision
Operand_a_DI	32bits	IN	Addend
Operand_b_DI	32bits	IN	Multiplier
Operand_c_DI	32bits	IN	Multiplicand
RM_SI	2 bits	IN	Rounding mode.
Result_DO	32bits	OUT	result
OF_SO	1 bit	OUT	Active high
UF_SO	1 bit	OUT	Active high

3. Architecture

The employed architecture is shown in Fig.1, most of which are based on the recommended architecture (Eric M. Schwarz, "Binary floating-point unit design: the fused multiply-add dataflow"). *Fmac* is consisted of *fmac_preprocess, pp_generation, Wallace, CSA, aligner, adders, LZA and fpu_norm.*

In the *fmac_preprocess* module, three operands are unpacked into three IEEE-754 encoded numbers into corresponding sign bits, biased binary exponents, and mantissas. To support denormal numbers, Operand detection is added to check each operand and give the corresponding symbols, InF_x_S, Zero_x_S, NaN_x_S and DeN_x_S. When DeN_x_S is logic 1, the corresponding exponent Exp_x_D will be set to 1, which offers a simple implementation with some extra latency. Another solution for high speed is shown in (Eric M. Schwarz, "Binary floating-point unit design: the fused multiply-add dataflow"). These signals are also used for normalization.

Radix-4 booth coding will be used to produce 13 partial products, which will be compressed by a Wallace consisted of 5 levels of CSAs. Booth encoding starts from Mant_b_DI[1:-1]. Thus the last partial product is always positive. Adding a hot one can be implemented in next partial products. Therefore, 13 partial products will be generated. *pp_generation* is used to instance *booth encoders* and *booth selector. Wallace* tree is used to compress 13 partial products into 2. The produced 2 results will be further compressed with Mant_al_D using *CSA*. Due to using the sign extension in (Eric M. Schwarz, "Binary floating-point unit design: the fused multiply-add dataflow"), all the carry outs of *wallace* and *CSA* should be taken into account.

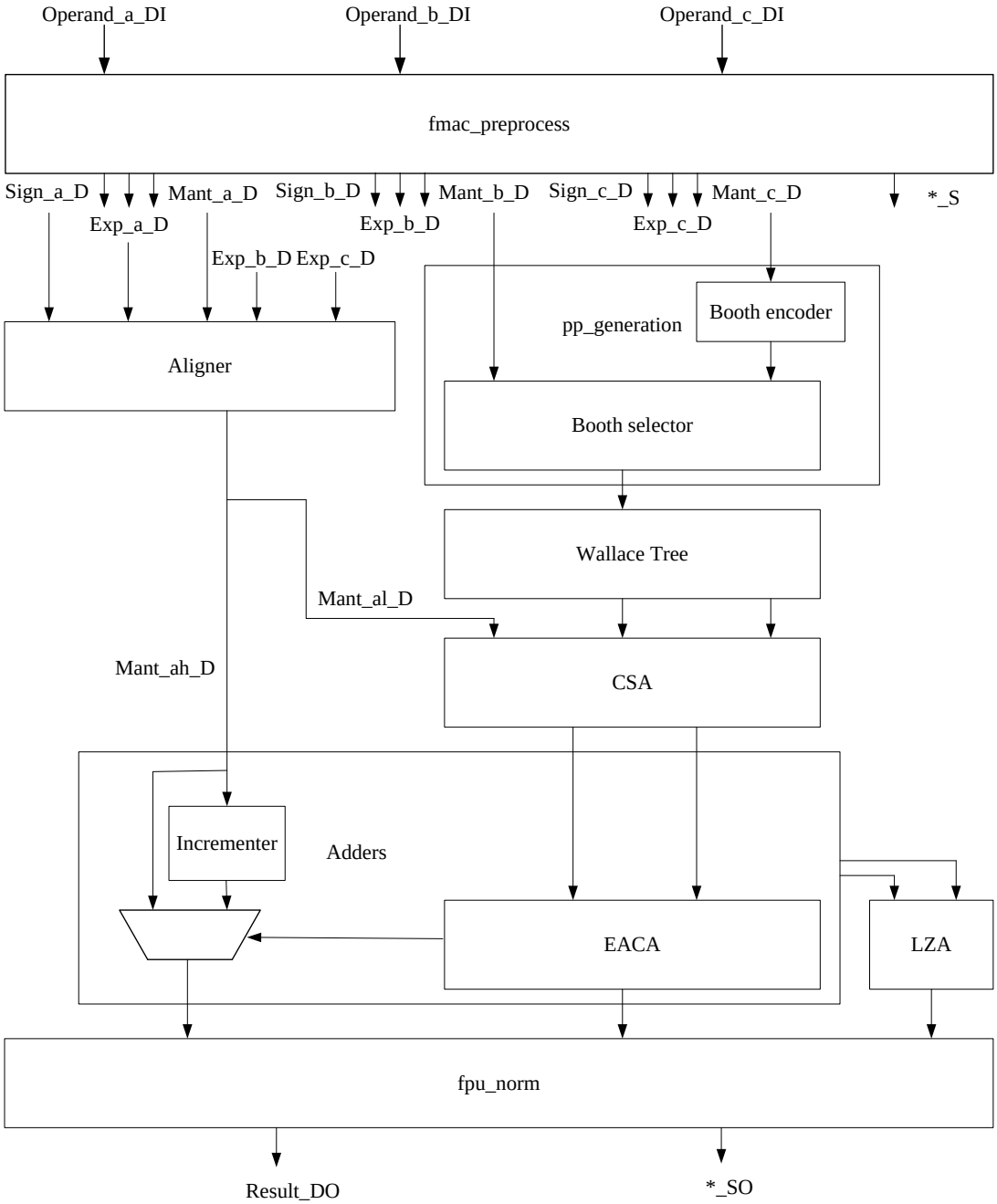

Fig.1 The architecture for FMAC

Aligner is used to implement the alignment. The alignment scheme in (Gongqiong Li, Zhaolin Li, “Design of a fully pipelined single-precision multiply-add-fused unit,”IEEE VLSID’07,2007) is employed. Most of detailed implementation can be based on (Gongqiong Li, Zhaolin Li, “Design of a fully pipelined single-precision multiply-add-fused unit,”IEEE VLSID’07,2007) , except normalization, since (Gongqiong Li, Zhaolin Li, “Design of a fully pipelined single-precision multiply-add-fused unit,”IEEE VLSID’07,2007) does not support denormal numbers.

The difference

$$d = Exp\_a\_D - Exp\_b\_D - Exp\_c\_D + Cbias \quad (1)$$

The right shift amount for alignment is

$$mv = 27 - d \tag{2}$$

.

The exponent after alignment can be given for two cases:

(1) If mv is negative, the result exponent is Exp_a_DI with the result mantissa of Mant_a_DI;

(2) Others: the result exponent is $\text{Exp\_b\_D} + \text{Exp\_c\_D} - \text{Cbias} + 27$.

If $\text{Sign\_a\_D} \oplus (\text{Sign\_b\_D} \oplus \text{Sign\_c\_D})$, there will be a substraction. The scheme in (Gongqiong Li, Zhaolin Li, "Design of a fully pipelined single-precision multiply-add-fused unit,"IEEE VLSID'07,2007) is employed to address this issue.

According to the shift amount, Mant_a_D is divided into two parts: Mant_ah_D and Mant_al_D. Mant_ah_D is inputted into an incrementer. Mant_al_D is taken as another inputs of the final CSA. The carry-out of EACA is used to choose the right result for Mant_ah_D. The corresponding theory is shown in (Eric M. Schwarz, "Binary floating-point unit design: the fused multiply-add dataflow,").

Adders is used to implement the functions of EACA and deal with *Mant_ah_D* to produce a positive result, as well as offering the inputs of *LZA*. Two channels are needed to implement EACA to produce a positive result. The inputs of either channel can be taken as the inputs of *LZA*.

LZA is used to finish the leading one detection. The method in (M. Schmookler, K. J. Nowka, "Leading Zero anticipation and detection—A comparison of method," IEEE,2001) is used for high speed. One *LZD* is introduced to produce the index. Since *LZA* cannot predict an exact position and two channels are used for normalization for this part.

4. Normalization

(1) $d \geq 27$, there is no overlap for Operand_a_DI and $\text{Operand\_b\_DI} * \text{Operand\_c\_DI}$.

Return Operand_a_DI and set the value for rounding based on $\text{Operand\_b\_DI} * \text{Operand\_c\_DI}$.

(2) $d \leq -48$, there is no overlap for Operand_a_DI and $\text{Operand\_b\_DI} + \text{Operand\_c\_DI}$.

Return $\text{Operand\_b\_DI} * \text{Operand\_c\_DI}$ and set the value for rounding based on Operand_a_DI.

(3) $-48 < d < 27$,There is overlap for Operand_a_DI and $\text{Operand\_b\_DI} * \text{Operand\_c\_DI}$.

The exponent after normalization can be given as:

$$\text{Exp\_norm\_D} = \text{Exp\_prenorm\_D} + 27 - \text{LZ}$$

Where LZ is the leading zero detection for the 74-bit mantissa.

$$\text{Mant\_norm\_D} = \text{Right shift}(27 - \text{LZ})\ \text{Mant\_prenorm\_D}$$

If $\text{Exp\_norm\_D} > 255$, OF is signaled and return E=255,M=0;
If $\text{Exp\_norm\_D} = 255$ and M≠0, NaN is signaled and return qNaN;
If $\text{Exp\_norm\_D} == 1$ and $\text{Mant\_norm\_D}[C_{MANT} - 1] == 0$, the result is a denormal number, return $\text{Exp\_norm\_D} == 0$ and Mant_norm_D;
If $1 =< \text{Exp\_norm\_D} <= 254$ and $\text{Mant\_norm\_D}[C_{MANT} - 1] == 1$, it is a normal result, return Exp_norm_D and Mant_norm_D;
If $\text{Exp\_norm\_D} == 0$ and M≠0, it is a denormal number, return >> (M) and the adjusted E;
(4)Special cases

Table 4 Special cases for FMAC

FMAC	operation	return
1	NaN +b*c	qNaN
2	a+NaN*c	qNaN
3	a+b*NaN	qNaN
4	Inf+ b*c	Inf
5	a+Inf*c	Inf
6	a+ b*Inf	Inf
7	DeN+DeN*c	DeN
8	DeN+DeN*DeN	DeN

*qNaN=0x7fc00000
(5) The above analysis is based on the conventional method. For high speed purpose, LZA based on the inputs of EACA can be used. The method in (M. Schmookler, K. J. Nowka, “Leading Zero anticipation and detection—A comparison of method,” IEEE,2001) can be used.

5. Rounding

The design supports four different rounding modes: RNE(Round to Nearest, ties to Even, 00), RTZ(Round towards Zero, encoding as 01),RDN(Round Down, encoding as 10),RUP(Round Up, encoding as 11).

Table 5 Rounding modes

Mode	Code
RNE	00
RTZ	01
RDN	10
RUP	11

6. Exceptions

The design supports two exceptions namely overflow(OF), underflow(UF).
Exp_OF_SO is signaled if the exact result has an exponent that cannot be represented

in the format. Returns infinity (positive or negative) as result.

Exp_UF_SO is signaled when the result is denormal and rounded.

7. Waveforms

The waveforms are based on fmac. Fmac.sv has been tested on the pulpino using benchmark programs.

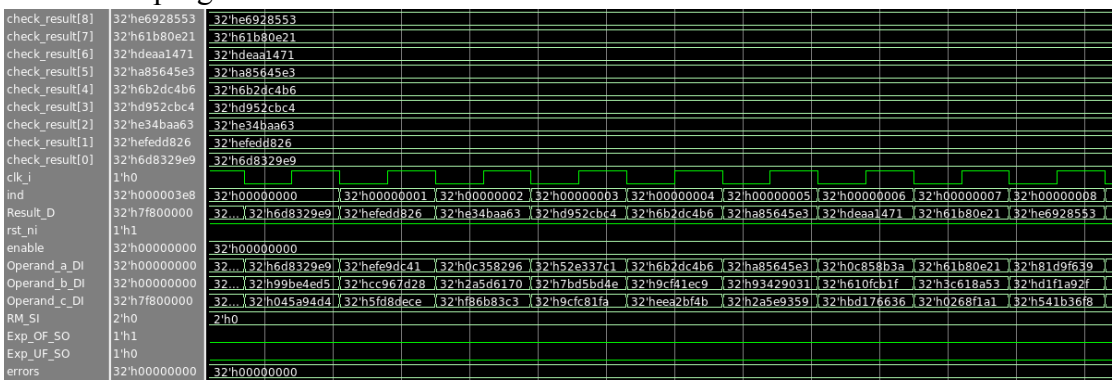

Fig.2 The waveform for the first 9 test cases (normal cases)

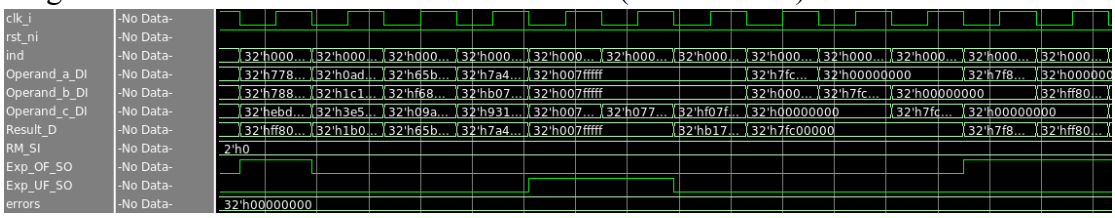

Fig.3 The waveform for the last test cases (including normal cases and special cases)

8. Synthesized results

UMC65nm process technology was used for synthesis.
Operating Conditions: uk65lscllmvbbl_108c125_wc
Library: uk65lscllmvbbl_108c125_wc
Input_delay:0.5ns
Output_delay:0.5ns

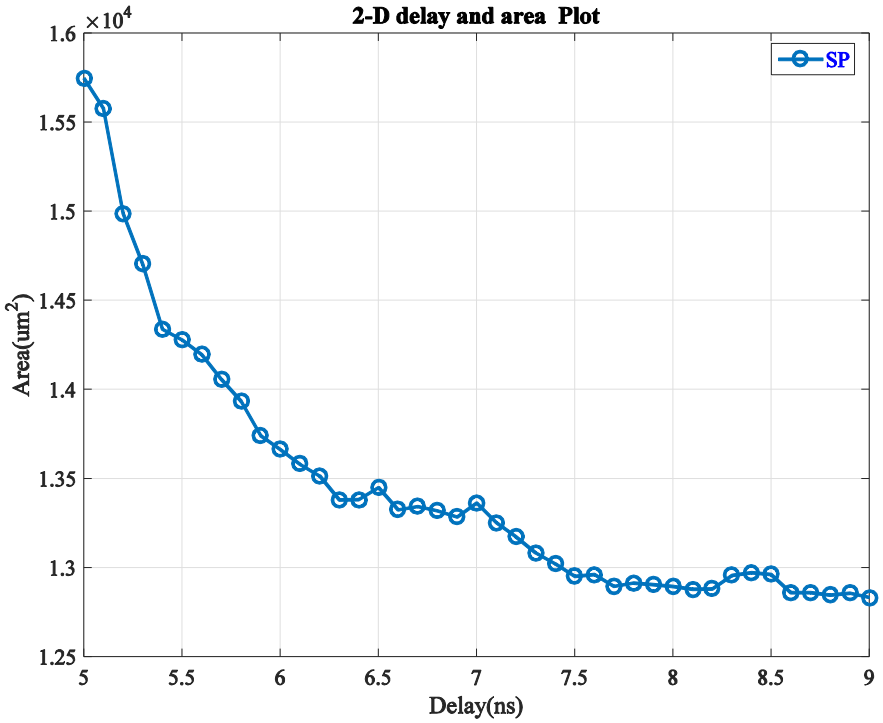

Fig.4 The synthesized results