Architecture for Quadruple Precision Floating Point Division with Multi-Precision Support

Manish Kumar Jaiswal[1], and Hayden K.-H So[2]
Dept. of EEE, The University of Hong Kong, Hong Kong;
Email: [1]manishkj@eee.hku.hk, [2]hso@eee.hku.hk

***Abstract*—This paper proposes a FPGA based hardware architecture for quadruple precision (QP) division arithmetic which can also process a single, a double and a double-extended precision (SP, DP, DPE) computations. The mantissa division employs a series expansion methodology of division, integrated with a wide integer multiplier further optimized for FPGA implementations facilitating the built-in DSP blocks efficiently. The proposed division architecture is demonstrated using a Xilinx FPGA based implementation has shown a significant area saving and much improvement in latency with improved speed.**

***Keywords*-Quadruple Precision Arithmetic, Division, FPGA, Multi-Precision Division.**

I. Introduction

A large number of important applications demand for a higher precision computation [1] that can be supported by quadruple precision (QP) arithmetic, which provides roughly 30 decimal digits of precision. To effectively accelerate this class of high-precision, we need a efficient support in hardware accelerator. In this view, this paper proposes a multi-precision division architecture that is capable of performing up to QP operation in hardware. The main contributions of present work can be summarized as follows:

- Proposed a multi-precision quadruple precision floating point division architecture which also supports the processing of SP, DP and DPE precision computation. It is based on the series expansion methodology of division.

II. Proposed Multi-Precision Division Architecture

For the purpose of multi-precision processing, the input/output operands for the unified floating point formats are assumed as shown in Fig. 1. The proposed multi-precision division architecture works in four modes, each for SP, DP, DPE and QP processing mode. It consists of three stages: pre-processing, core computations and post-processing.

The first stage of the architecture includes data-extraction, sub-normal handling and exceptional checks and are implemented using typical methods . Since, the decimal point position (in input operands) is same for all modes (as shown in Fig. 1), unified/same signal for sign, exponent and mantissa works for all mode. This stage also includes the part of mantissa division unit, as discussed in the later part.

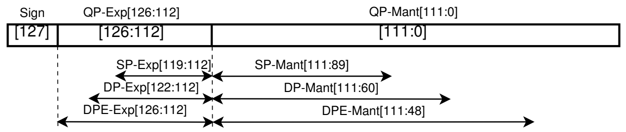

Figure 1: Input/Output Register Format

A. The Core Division Processing Architecture

The stage-2 consists of the core operation which computes sign, exponent and right-shift-amount is processed in trivial way, by using a unified "BIAS" signal for multi-precision environment ($BIAS[14:0] = \{\{4\{QP|DPE\}\},\{3\{QP|DPE|DP\}\},7'b7F\}$).

1) Mantissa Division Unit: The methodology for this is based on the [2]. Let m_1 be the normalized dividend mantissa, m_2 be the normalized divisor mantissa then q, the mantissa quotient, can be computed as (1). Here, m_2 is partitioned in to two part as a_1 (W-bit) and a_2 (all remaining bits) as in (2). Equation(1) can be solve using only multipliers, adders and subtractors, provided that the value of a_1^{-1} is available which can be access from a pre-stored look-up table. For a bit width of $W = 8$ for a_1, it requires 17 terms (up to $a_1^{-17}a_2^{16}$) for QP, 9 terms (up to $a_1^{-9}.a_2^{8}$) for DPE, 7 terms (up to $a_1^{-7}.a_2^{6}$) for DP, and 3 terms (up to $a_1^{-3}.a_2^{2}$) for SP precision requirement. For a multi-precision architectural implementation, an unified expression is structured in (3) which supports all the required precision computations. The size of look-up table (LUT) to store a_1^{-1} is taken as $2^8 \times 113$, and a full multiplier of size 114x114-bit is used iteratively using a FSM (Finite State Machine) to implement (3).

$$q = \frac{m_1}{m_2} = \frac{m_1}{a_1 + a_2} = m_1(a_1 + a_2)^{-1} = m_1(a_1^{-1} - a_1^{-2}a_2 + a_1^{-3}a_2^2 - a_1^{-4}a_2^3 \ldots) \tag{1}$$

$$m_2 = \overbrace{1.xxxxxxxx}^{a_1} \quad \overbrace{xxxxxxx\ldots\ldots\ldots\ldots\ldots\ldots\ldots xxxxxxx}^{a_2:\ QP:104-bit,\ DPE:56-bit,\ DP:44-bit,\ SP:15-bit} \tag{2}$$

$$q = \underbrace{\underbrace{\overbrace{\overbrace{m_1.a_1^{-1} - m_1.a_1^{-1}(a_1^{-1}.a_2 - a_1^{-2}.a_2^2)}^{SP}(1 + a_1^{-2}.a_2^2 + a_1^{-4}.a_2^4}^{DP} + a_1^{-6}.a_2^6)}_{DPE}(1 + a_1^{-8}.a_2^8)}_{QP} \tag{3}$$

$$\begin{array}{lllll} A = m_1a_1^{-1}, & B = a_1^{-1}a_2, & C = a_1^{-2}a_2^2, & D = a_1^{-4}a_2^4, & E = a_1^{-6}a_2^6 \\ F = a_1^{-8}a_2^8, & G = B - C, & H_T = 1 + C + D, & H = H_T + F, & I = 1 + F \\ J = GH, & K = JI, & L = AK, & M = A - L & \end{array} \tag{4}$$

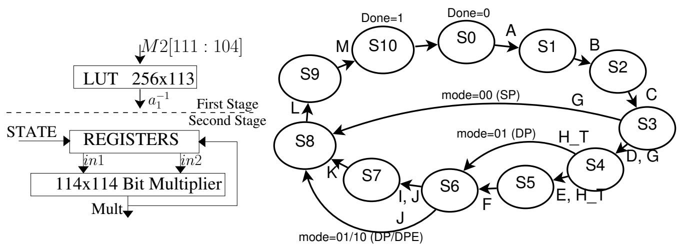

Figure 2: Mantissa Division Architecture and FSM

The implementation of eq.(3) (as shown in Fig. 2), incorporates a LUT, a 114x114 multiplier and a FSM. Based on the mode of operation, the FSM decides the effective inputs for the multiplier in each state and assigned its output to the designated terms in (4). A single stage, 114x114 multiplier is designed around DSP48E IPs, using combination of 3-partition and 2-partition Karatsuba method [3]. Initially, multiplier operands are partitioned into 3 sets of 38-bit, which requires 3 38x38 and 3 39x39 multipliers. The 39x39 (also used as 38x38) multiplier is designed using two partition method, which needs one 19x19, one 20x20 and one 21x21 multipliers. The 19x19, 20x20 and 21x21 multipliers are implemented by using a DSP48E and some logic resources. It requires only 18 DSP48E blocks for 114x114 multiplier.

$$\begin{aligned}
S0:&\ in1=\{1'b0,m_1\}, \qquad in2=\{1'b0,m_2\_a_1\_i\}\\
S1:&\ in1=\{10'b0,m_2\_a_2\},\quad in2=\{1'b0,m_2\_a_1\_i\},\quad A[127:0]=mult[225:98]\\
S2:&\ in1=in2=B=mult[216:103]\\
S3:&\ in1=in2=\{18'b0,mult[227:132]\},\quad C=mult,\quad G=B-\{8'b0,C[227:122]\}\\
S4:&\ in1=\{50'b0,mult[227:132]\},\quad in2=\{50'b0,C[227:164]\},\quad D=mult[191:0]\\
&\ H_T=\{1'b1,16'b0,C[227:130\}+\{33'b0,D[191:110\}\\
S5:&\ in1=in2=\{66'b0,D[191:144]\}, \qquad E=mult[127:0]\\
S6:&\ in1=G, \qquad in2=DP\ ?\ H_T\ :\ (H\leftarrow H_T+\{49'b0,E[127:62]\})\}\\
&\ F=mult[95:0], \qquad I=\{1'b1,64'b0,F[95:47]\}\\
S7:&\ in1=J=mult[227:114], \qquad in2=I\\
S8:&\ in1=SP\ ?\ G\ :\ (J\ or\ K\leftarrow mult[227:114]), \qquad in2=A[127:14]\\
S9:&\ L=(QP|DPE)\ ?\ \{6'b0,mult227,106]\}\ :\ (DP\ ?\ \{7'b0,mult[227:107]\}\\
&\quad :\ \{8'b0,mult[227:108]\}), \qquad in1=in2=0\\
S10:&\ in1=in2=0,
\end{aligned} \tag{5}$$

FSM consists of 11 states (S0 to S10). For QP-mode it passes through all the states (S0 to S10). Whereas, for DPE mode it skips the state S7. DP-mode does not requires the processing of state S5 and S7; and the states S4-to-S7 are not required in SP-mode. Some mode specific assignments can also be seen in the states S6, S8, and S9 using mode control signals (QP, DPE, DP, and SP). This FSM requires 11 cycles, 10 cycles, 9 cycles and 7 cycles respectively for QP-mode, DPE-mode, DP-mode and SP-mode processing. The post-processing stage performs normalization, rounding (`round-to-nearest`) and final-processing, which all are done using trivial methods, over unified mantissa for multi-precision processing.

Table I: Comparison of Division Architecture

	[4]	Proposed Multi-Precision Arch.
Latency	118 (QP)	9/11/12/13 (SP/DP/DPE/QP)
Throughput	NA (QP)	8/10/11/12 (SP/DP/DPE/QP)
LUTs	26811	7440
FFs	13809	2584
DSP48	-	18
Freq (MHz)	50	89

III. Implementation Results

The proposed multi-precision QP division architecture is implemented using Xilinx Virtex-7 FPGA device. The functional verification of the proposed architecture is carried out using 5-millions random test cases with various combinations of operands, which produces a faithful rounded result (max 1-ULP, unit at last place, precision loss). To the best of author's knowledge, literature does not contains any multi-precision quadruple precision division architecture. Diniz *et al.* [4] is only work available which has shown the results for a single-mode quadruple precision division architecture implementation on a FPGA device. A comparison is shown against it in Table I, which shows that proposed work provides better latency, throughput, area and speed metric along with providing multi-precision support.

IV. Conclusions

This paper presented an iterative multi-precision quadruple precision division architecture for the hardware accelerators, which is based on the series expansion division methodology of mantissa division. Compared to the available literature, the proposed architecture out-performs them in terms of area, speed, latency and throughput.

Acknowledgments: This work is party supported by the "The University of Hong Kong" grant (Project Code. 201409176200), the "Research Grants Council" of Hong Kong (Project ECS 720012E), and the "Croucher Innovation Award" 2013.

References

[1] D. H. Bailey, R. Barrio, and J. M. Borwein, "High-precision computation: Mathematical physics and dynamics," *Applied Mathematics and Computation*, vol. 218, no. 20, pp. 10 106–10 121, 2012.

[2] M. Jaiswal, R. Cheung, M. Balakrishnan, and K. Paul, "Series expansion based efficient architectures for double precision floating point division," *Circuits, Systems, and Signal Processing*, vol. 33, no. 11, pp. 3499–3526, 2014. [Online]. Available: http://dx.doi.org/10.1007/s00034-014-9811-8

[3] A. Karatsuba and Y. Ofman, "Multiplication of Many-Digital Numbers by Automatic Computers," in *Proceedings of the USSR Academy of Sciences*, vol. 145, 1962, pp. 293–294.

[4] P. Diniz and G. Govindu, "Design of a field-programmable dual-precision floating-point arithmetic unit," in *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on*, Aug 2006, pp. 1–4.