Dual-Mode Double Precision Division Architecture

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 presents an area efficient architecture for a dual-mode double precision floating point division, which can either process a double precision (DP) division or two parallel single precision (SP) division. The dual-mode mantissa division architecture is based on the series expansion methodology, and implemented in an iterative fashion. A dual-mode Radix-4 Modified Booth multiplier is designed for this purpose, which is used iteratively in the architecture of dual-mode mantissa computation. The proposed dual-mode division architecture is synthesized using UMC 90*nm* technology ASIC implementation. The proposed architecture shows better design metrics in terms of required area, time-period and throughput as against prior literature work.**

***Index Terms*—Floating Point Division, Dual-Mode Architecture, ASIC, Configurable Architecture.**

I. Introduction

Floating point arithmetic (FPA) is a basic ingredient of a large set of scientific and engineering domain applications. To boost the application performances, arrays of single precision and double precision computing units are being used as floating point vector processing. The current research work is aimed towards the idea of unified vector-processing units. That is, instead of having separate vector arrays of single precision and double precision, it can have an array of configurable floating point arithmetic blocks. Where each of these configurable blocks can process either a double precision or two parallel single precision computations. This configurable block array arrangement can lead to significant area improvement, while providing the required performance.

This paper is focused on the design of a dual-mode double precision division arithmetic unit. Only few literature are available on the dual-mode double precision division architecture [1], [2]. Floating point (FP) division is a core computation required in a multitude of applications. The proposed architecture can be configured either for a double precision or two parallel (dual) single precision division computations, and named as DPdSP division architecture. The proposed architecture is based on the series expansion methodology of division algorithm ([3]). Series expansion method is a multiplicative division method, which provides a hardware efficient architecture for a given precision requirement [4]. A dual-mode Radix-4 Modified Booth multiplier is designed for the purpose of mantissa division. Also, since the underlying integer multiplier in mantissa division unit has the major cost in terms of required area, an iterative architecture is proposed to achieve area efficiency. The present work is build upon the [2], with a practical approach for DPdSP division architecture. Compared to [2], which has presented an impractical single cycle implementation with large area requirement, the present work provides a different architecture for the most complex unit, the mantissa division (with some novel architectural proposal), which constitutes more than 80% of hardware resources in FP division architecture.

The main contributions of this work can be briefly summarized as follows:

- Proposed dual-mode DPdSP division architectures with sub-normal computational support, which can be configured either for a double precision division or two parallel single precision divisions.
- A novel dual-mode Radix-4 Modified Booth multiplier architecture is proposed, which is the main constituent of the proposed dual-mode mantissa division architecture.

II. Underlying Mantissa Division Method

The algorithmic methodology for mantissa division architecture is discussed here. It is based on the series expansion method of division, as follows.

Let m_1 be the normalized dividend mantissa and m_2 be the normalized divisor mantissa, then q, the mantissa quotient, can be computed as:

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

where, a_1 and a_2 are parts of division mantissa as below.

$$m_2 \rightarrow \overbrace{1.\underbrace{xxxxxxxx}_{W-bit}}^{a_1} \overbrace{xx.....xxxxxxx}^{a_2}$$

Here, the pre-computed value of a_1^{-1} acts as an initial approximation for m_2^{-1}, which further improves with remaining computation in (1). Here, the size W (bit width) of a_1 determines the size of memory (to store a_1^{-1}) and the number of terms from the series expansion, to perform the computation for a given precision. For a good balance among W and required number of terms, bit width of $W = 8$ for a_1 is selected, which requires 7 terms (up to $a_1^{-7}a_2^6$) for double precision, and 3 terms (up to $a_1^{-3}a_2^2$) for single precision requirement. For dual-mode architecture design, a unified equation for double and single precision processing is formulated as below.

$$q = \underbrace{\overbrace{m_1a_1^{-1} - m_1a_1^{-1}(a_1^{-1}a_2 - a_1^{-2}a_2^2)}^{SP}(1 + a_1^{-2}a_2^2 + a_1^{-4}a_2^4)}_{DP} \quad (2)$$

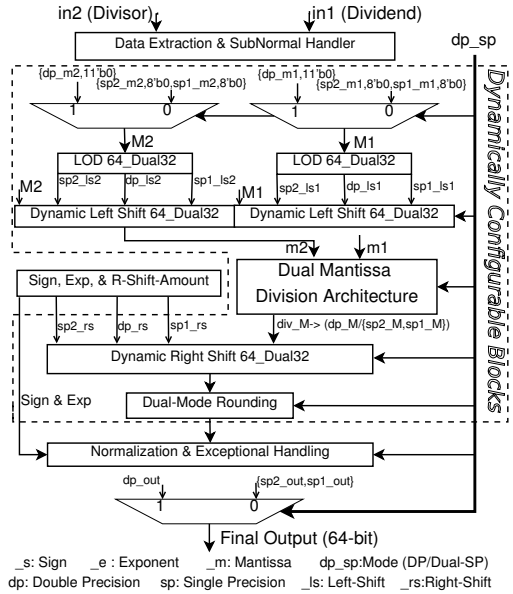

Fig. 1: DPdSP Division Architecture

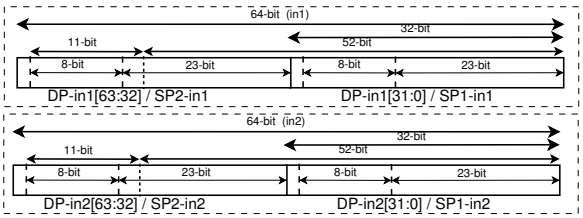

Fig. 2: DPdSP Input Output Format

Here, SP part computes for single precision, while entire equation process the double precision. The interesting feature of (2) forms the basis of sharing hardware resources to efficiently model the dual-mode architecture for mantissa division computation, which is capable of processing either a DP mantissa or two SP mantissa divisions. The size of look-up table to store a_1^{-1} is taken as $2^8 \times 53$ for DP and $2^8 \times 24$ for SP, which is sufficient for both precision.

III. Proposed DPdSP Division Architecture

The proposed architecture is shown in Fig. 1. It is composed of three pipelined stages. Two 64 bit operands, one dividend (in_1) and another divisor (in_2) are the primary inputs along with the mode-control signal dp_sp (double precision or dual single precision). Both of the input operands either contains DP operands (as entire 64-bit pair) or two parallel SP operands (as two sets of 32-bit pair), as shown in Fig. 2.

A. First-Stage Architecture

First stage process for data-extraction, exceptional case handling, and sub-normal processing. It also includes the part of mantissa division unit, the pre-fetching of initial approximation of divisor mantissa inverse from look-up table. The data extraction computation takes the primary operands and extract the signs, exponents and mantissas components

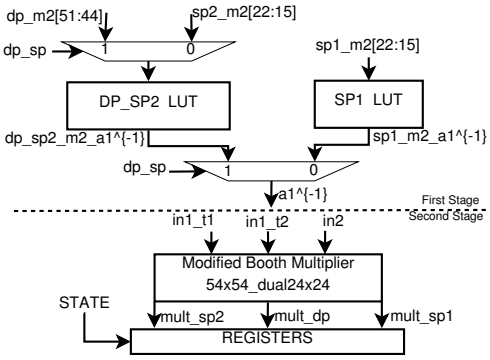

Fig. 3: DPdSP Dual Mode Mantissa Division Architecture

for double precision and both single precision, based on their standard formats. The sub-normal (_sn) handling and exceptional checks computations are done using traditional methods.However, as the 8 MSB bits of DP exponent overlap with SP-2 exponent (as shown in Fig 2), the checks for sub-normal, infinity and NaN (Not-A-Number) have been shared among SP-2 and DP. Similarly, it also performs checks for `divide-by-zero` (_dbz) and `zero` (_z), and have been shared among DP and both SPs.

After above processing, a unified set of mantissa (M1 and M2) is generated using two 2:1 MUX (as shown in Fig. 1), which contain the mantissa either for DP or for both SPs. This unification of mantissas helps in designing a tuned data-path processing for later stage computation, which results in efficient resource sharing. The next two units, the leading-one-detector (LOD) and dynamic left shifter, in this stage perform sub-normal processing. They bring the sub-normal mantissa (if any) into the normalized format. The details on dual-mode LOD and dual-mode dynamic left shifter architecture can be sought from [2].

Above processing produces mantissas into normalized form m_1 and m_2, as shown in Fig. 1. Further, in this stage of architecture, the 8-bit MSB part (a_1) of normalized divisor mantissas (m_2) are used to fetch the pre-computed initial approximation of their inverse. It is shown in the first-stage part of Fig. 3, DP_SP2 LUT (256x53) is shared for DP and SP-2 initial approximation, and SP1 LUT (256x24) works for SP-1 only.

B. Second-Stage Architecture

This stage of architecture computes the sign, exponent and mantissa processing of FP division arithmetic and the computation related to right shift amount. The computations related to exponent and right shift amount processing are done using traditional methods. These computations are processed separately for DP and both SPs.

The dual mode mantissa division processing is the most crucial component of the FP division architecture. The mantissa computation architecture includes the unified and dual-mode implementation of (2). This computation is built around a dual-mode booth multiplier, in an iterative fashion. A dual-

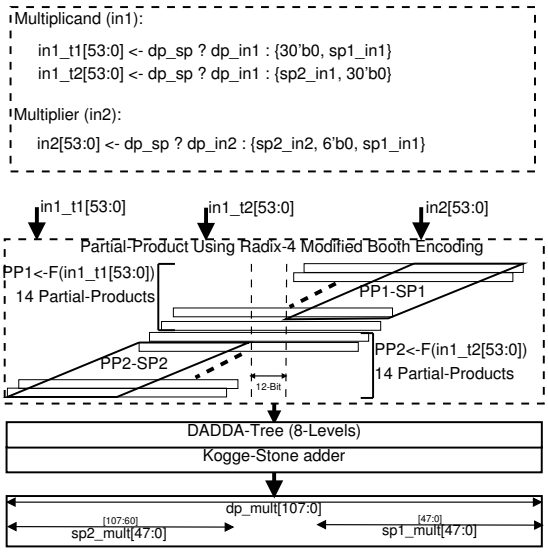

Fig. 4: Dual-Mode Modified Booth Multiplier Architecture

mode finite state machine (FSM) is designed which decides the effective inputs for multiplier in each state.

1) Dual-Mode Radix-4 Modified Booth Multiplier Architecture: The architecture is based on the Radix-4 Modified Booth Encoding and shown in Fig 4. It is a 54-bit integer multiplier (for DP processing), which can also process two parallel sets of 24-bit unsigned operands (for two SPs processing) multiplication. The presented dual-mode multiplier has three input operands (two multiplicands and a multiplier). A set of two inputs ($in_1\_t_1$ and $in_1\_t_2$) forms the multiplicand operands. Here, $in_1\_t_1$ consists of either 'DP multiplicand operand' or 'SP-1 multiplicand operand at the LSB side', and $in_1\_t_2$ consists of either 'DP multiplicand' or 'SP-2 multiplicand operand at the MSB side'. While, the multiplier input (in_2) contains multiplier operands either for DP, or for both SPs with 6-bit zero in between (see top portion of Fig. 4). Correspondingly, two-sets of partial products (PP1 and PP2) are generated. Partial products PP1 are the result of $in_1\_t_1$ and in_2, and PP2 is derived from $in_1\_t_2$ and in_2. Here, the inputs $in_1\_t_1$, $in_1\_t_2$ and in_2 are built so that, in dual-SP mode processing the single precision partial products (PP1-SP1 and PP2-SP2) and their reduction do not overlap (Fig. 4), and produce two distinct results for SP-1 and SP-2 multiplication, respectively.

Therefore, the sum of all partial products will generate product for DP operands in DP-mode or for both SPs in dual-SP mode. A DADDA-tree of 8 levels is designed to compress all the partial products into two operands, which are further added using a parallel-prefix Kogge-Stone final adder. The final product contains either DP or dual-SP results as shown in Fig. 4. Compared to the contemporary Modified Booth multiplier, the proposed dual-mode Modified Booth multiplier requires only three 2:1 MUXs as an area overhead, which are needed for the input operands multiplexing.

2) Dual-Mode Iterative Mantissa Division Architecture: The mantissa division is designed in an iterative fashion to have an area efficient architecture. The architecture is based on the unified implementation of (2), which can process either a DP mantissa division or two parallel SPs mantissa divisions by inclusion of above discussed dual-mode modified Booth multiplier. Here, m_1 (dividend) and m_2 (divisor) are normalized mantissas which contain either DP mantissas ($dp\_m_1[52:0]$ and $dp\_m_2[52:0]$) or both SPs mantissas ($sp_1\_m_1[23:0]$, $sp_1\_m_2[23:0]$, $sp_2\_m_1[23:0]$ and $sp_2\_m_2[23:0]$), as shown in Fig. 3. Divisor mantissa (m_2) is partitioned into a_1 (first 8-bit right to the decimal point) and a_2 (all remaining bits right to the a_1), for DP and both SPs, as below.

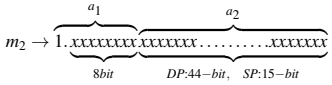

Fig. 5: DPdSP Dual-Mode Iterative Mantissa Division FSM

$$m_2 \rightarrow 1.\overbrace{xxxxxxxx}^{a_1}\overbrace{xxxxxxx.....xxxxxxx}^{a_2}$$
$$\underbrace{\qquad}_{8bit}\quad\underbrace{\qquad\qquad}_{DP:44-bit,\ SP:15-bit}$$

For the ease of understanding, various terms of (2) are listed in (3). From these abbreviations in (3), for SPs computation, it only requires to skip the computation of D, F, G and H_{DP} from DP flow. A 9 state (S0 to S8) FSM is designed for this purpose. Each state of FSM determines the inputs ($in_1\_t_1, in_1\_t_2$ and in_2) for dual-mode modified booth multiplier, and assigned its output to the designated terms, which proceeds as follows:

$$A = m_1.a_1^{-1}, \quad B = a_1^{-1}.a_2, \quad C = B^2 = a_1^{-2}.a_2^2, \quad D = B^4 = C^2 = a_1^{-4}.a_2^4$$
$$E = B - C = a_1^{-1}a_2 - a_1^{-2}a_2^2, \quad F = 1 + C + D = 1 + a_1^{-2}a_2^2 + a_1^{-4}a_2^4$$
$$G = EF, \quad H_{DP} = AG \quad H_{SP} = AE \quad I = A - H \tag{3}$$

$$\begin{aligned}
S_0:\ & in_1_t_1 = dp_sp\ ?\ \{1'b0, dp_m_1\}\ :\ \{30'b0, sp_1_m_1\}\\
& in_1_t_2 = dp_sp\ ?\ \{1'b0, dp_m_1\}\ :\ \{sp_2_m_1, 30'b0\}\\
& in_2 = dp_sp\ ?\ \{1'b0, dp_m_2_a_1^{-1}\}\ :\ \{sp_2_m_2_a_1^{-1}[52:29], 6'b0, sp_1_m_2_a_1^{-1}\}\\
S_1:\ & in_1_t_1 = dp_sp\ ?\ \{10'b0, dp_m_2_a_2\}\ :\ \{30'b0, 9'b0, sp_1_m_2_a_2\}\\
& in_1_t_2 = dp_sp\ ?\ \{10'b0, dp_m_2_a_2\}\ :\ \{9'b0, sp_2_m_2_a_2, 30'b0\}\\
& in_2 = dp_sp\ ?\ \{1'b0, dp_m_2_a_1^{-1}\}\ :\ \{sp_2_m_2_a_1^{-1}[52:29], 6'b0, sp_1_m_2_a_1 - 1\}\\
& A[63:0] = dp_sp\ ?\ dp_mult[105:42]\ :\ \{sp_2_mult[47:16], sp_1_mult[47:16]\}\\
S_2:\ & in_1_t_1 = dp_sp\ ?\ dp_mult[96:43]\ :\ \{30'b0, sp_1_mult[38:15]\}\\
& in_1_t_2 = dp_sp\ ?\ dp_mult[96:43]\ :\ \{sp_2_mult[38:15], 30'b0\}\\
& in_2 = dp_sp\ ?\ dp_mult[96:43]\ :\ \{sp_2_mult[38:15], 6'b0, sp_1_mult[38:15]\}\\
& B[63:0] = dp_sp\ ?\ dp_mult[96:43]\ :\ \{sp_2_mult[38:12], sp_1_mult[38:12]\}\\
S_3:\ & in_1_t_1 = in_1_t_2 = in_2 = dp_mult[107:54] \quad C_{DP} = dp_mult \quad E = B - C\\
& C = dp_sp\ ?\ \{8'b0, dp_mult[107:62]\}\ :\ \{8'b0, sp_2_mult[47:29], 8'b0, sp_1_mult[47:29]\}\\
S_4:\ & in_1_t_1 = in_1_t_2 = in_2 = 0 \quad D_{DP} = dp_mult[107:87]\\
& F_{DP}[53:0] = \{1'b1, 16'b0, C_{DP}[107:71]\} + \{33'b0, D_{DP}\}\\
S_5:\ & in_1_t_1 = in_1_t_2 = E \quad in_2 = F_{DP}\\
S_6:\ & in_1_t_1 = dp_sp\ ?\ G\ :\ \{30'b0, E[26:3]\} \quad G = dp_mult[107:54]\\
& in_1_t_2 = dp_sp\ ?\ G\ :\ \{E[26:3], 30'b0\} \quad in_2 = A\\
S_7:\ & in_1_t_1 = in_1_t_2 = in_2 = 0, \quad AE = \{8'b0, sp_2_mult[47:24], 8'b0, sp_1_mult[47:24]\}\\
& AG = \{7'b0, dp_mult[107:51]\} \quad H = dp_sp\ ?\ AG\ :\ AE\\
S_8:\ & I = A - H \quad in_1_t_1 = in_1_t_2 = in_2 = 0
\end{aligned} \tag{4}$$

The finite state machine (FSM) is shown in Fig. 5. For DP processing it goes through all the states, whereas for dual-SP it skips states $S4$ and $S5$ which performs only DP related computations. The selection of bits for a term is based on the position of decimal point and mode of the processing. Generally, for DP mode, the multiplications are done in 54-bit (sufficient for it's precision requirement) and add/sub are performed in 64-bit (to preserve precision), whereas, for dual-SPs, the multiplications are done in 24-bit and add/sub are performed in 32-bit. The mantissa division requires 9 cycles for DP-mode processing, while only 7-cycles for dual-SPs processing. Compared to the only DP mantissa division FSM, the DPdSP mantissa division FSM requires 14 54-bit 2:1 MUXs as an overhead.

C. Third-Stage Architecture

In this stage, for the case of exponent underflow, mantissa division quotient is first process for the dynamic right shifting. This is followed by the dual-mode rounding (rounding to nearest is implemented) of the quotient mantissa, and then it undergoes normalization and exceptional case processing. The architectural details of dual-mode dynamic right shifter can be sought from [2], which can shift either a DP mantissa or two-parallel SP mantissa. It takes right-shift-amount and mantissa quotient as primary inputs. Rounding first computes the unit-at-last-place (ULP) separately for DP and both SPs, and performs ULP addition. The ULP-addition with quotient mantissa is shared among DP and both SPs by using two 32-bit incrementer, which individually acts like a SP ULP-adder, however, their combination (by propagating carry) also performs for DP ULP-addition. The rounded mantissa quotient is further normalized separately for DP and both SPs, which requires 1-bit right shifting. And corresponding exponents are incremented by one, separately for DP and both SPs. Further to this, each exponent and mantissa is updated for exceptional cases (either of infinity, subnormal or underflow cases), which needs separate units for DP and both SPs. Finally, the computed signs, exponents and mantissas for double precision and both single precision are multiplexed using a 64-bit 2:1 MUX to produce the final 64-bit output floating point quotient result, which either contains the DP quotient or two SPs quotients .

IV. Implementation Results

The proposed architecture is synthesized with UMC 90*nm* standard cell ASIC library, using Synopsys Design Compiler, with best achievable timing constraints. It has a latency of 11 cycles and throughput of 10 cycles for DP computation, a latency of 9 cycles and throughput of 8 cycles for dual-SP computations. The functional verification is carried out using 5-millions random test cases for each of the normal-normal, normal-subnormal, subnormal-normal and subnormal-subnormal operands combination, along with the other exceptional case verification, for both DP and dual-SP mode. It produces a maximum of 1-ULP (unit at last place) precision loss which is sufficient for a large amount of applications.

TABLE I: Comparison of DPdSP Division Architecture

	[1] (Only Normal)	[2] (SubNormal)	Proposed (SubNormal)
Gate Count[1]	212854	163194	66416
Period (FO4)[2]	31.4	437.5	38.22
Throughput[3]	29/15 (DP/dSP)	1/1 (DP/dSP)	10/8 (DP/dSP)
Area × Period × Throughput [4]	193.82×10^6	71.39×10^6	25.38×10^6

[1] Based on minimum size inverter [2] 1 FO4 (ns) ≈ (Tech. in μm) / 2
[3] in clock-cycle [4] Gate Count × Period (FO4) × Throughput

A technological independent comparison is presented in Table-I, in terms of Gate-Count for area, FO4-delay for timings, cycle counts for latency & throughput and in terms of an unified metric $Area \times Period\ (FO4) \times Throughput\ (in\ clock-cycle)$ (which should be smaller for a better design). Isseven *et. al.* [1] has presented an iterative DPdSP division architecture using Radix-4 SRT division algorithm, without subnormal support. Compared to proposed architecture, Isseven *et. al.*'s architecture requires much larger area and has poor $Area \times Period \times Throughput$ metric. The prior work presented in [2] also requires a significantly large area with a poor $Area \times Period \times Throughput$. Also, due to its single cycle implementation of [2], this design is not practical. Thus, the currently proposed architecture is better in terms of design metrics. To the best of author's knowledge, literature does not contains any other dual-mode division architecture, which can support DP with two parallel SP divisions.

V. Conclusions

This paper has presented a dual-mode iterative architecture for DP FP division arithmetic. It can process either a DP or two-parallel SPs floating point division. All the components are designed for efficient dual-mode processing and a novel dual-mode Radix-4 Modified Booth multiplier architecture is proposed with minimal overhead. The proposed dual-mode architecture outperforms the prior arts in terms of various design metrics.

VI. 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] A. Isseven and A. Akkaş, "A dual-mode quadruple precision floating-point divider," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, 2006, pp. 1697–1701.

[2] M. Jaiswal, R. Cheung, M. Balakrishnan, and K. Paul, "Configurable architecture for double/two-parallel single precision floating point division," in *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, July 2014, pp. 332–337.

[3] S. F. Obermann and M. J. Flynn, "Division algorithms and implementations," *Computers, IEEE Transactions on*, vol. 46, no. 8, pp. 833–854, Aug. 1997.

[4] M. K. 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