International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
Index Copernicus Value (2015): 56.67 | Impact Factor (2017): 5.156

Low Complexity in Dual-Mode Double Precision Floating Point Division with Reduced Area

Shaik Asiya[1], P. Jaya Rami Reddy[2]

[1]Dept. of Electronics & Communication Engineering, Yogananda Institute of Technology and Sciences, Tirupati India
Email: shaik.asiya28[at]gmail.com
[2]Dept. of Electronics & Communication Engineering, Yogananda Institute of Technology and Sciences, Tirupati India
Email: pjrece[at]gmail.com

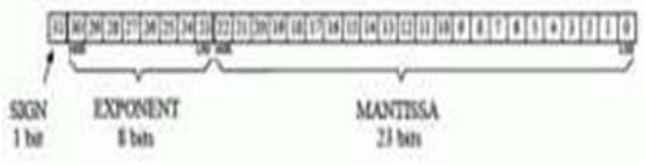
Abstract: ***Floating point is a core arithmetic widely used in scientific and engineering applications. This paper is used in proposed at architecture for dual mode double precision floating point division. It aimed to work on dual-mode functionality for single and double precision. It is used in the two pairs of single precision operands in parallel or double precision operand. A Radix-4MB multiplier is used for the mantissa computation. Other key components of floating point division flow (Such as leading one detection, left/right dynamic shifters, rounding etc.) are also re-designed for the dual mode operation. This model is synthesized under Xilinx tool and compared the results with the single precision floating point division. The proven results is proposed design in efficient over the many related works of past over the area and delay.***

Keywords: Arithmetic, configurable architecture**,** dual- mode division, floating point division, multi precision arithmetic

1. Introduction

Floating point arithmetic (FPA) architecture underwent momentous progression by specific research in the earlier period of several decades. FPA is a basic component of a large set of scientific and engineering domain application. To boost the application performance, the FPA architectures developed from scalar to vector architectures in various processing platforms. Arrays of single precision and double precision computing units are being used for the floating vector-processing. 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. This configurable block array arrangement can be lead to significant area improvement, while providing the required performance.

Our examination work is centered round the engineering outline of configurable coasting point math pieces. This paper is focused on the intent of configurable dual-mode double precision division arithmetic unit. Floating point (FP) division is a core calculation is necessary in a multitude of applications. FPD is a composite arithmetic operation which requires large area with reduced presentation than the basic arithmetic operations (such as adder, subtractor and multiplier). In this application at huge area required for division arithmetic per unit of calculation, this worked is designed for this calculation. The process of structural design is configured either for a double precision or two parallel single precision division computations, and named as DPdSP.



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

- The proposed dual-mode DPdSP division architectures with normal and sub-normal computational support, along with all the exceptional case handling. These architectures can be dynamically configured either for a double precision division or two parallel single precision division.
- A novel dual-mode Radix-4 Modified Booth multiplier architecture is proposed, which becomes the base of the proposed dual-mode mantissa division architecture.
- All the key components of the FPD flow is designed for efficient dual-mode functional with minimal overhead.
- Proposed architectures are fully pipelined and, designed in an iterative fashion for area efficient.

2. Existing System

The existing system can be proposed till now for this FPA point arithmetic especially division in our work. With this algorithm which incorporates the positive attributes of the division architectures mentioned. The floating point representation, where mantissa of dividend and divider are each 23-bit wide. The basic algorithm for the considered design is as follows: First and foremost, the sign-bit of result is obtained by performing logical XOR of sign bits of dividend and divisor. The exponential point is evaluated by subtracting 8-bit exponent of divisor from the 8 bit dividend and then adding the bias value (+127). Lastly, 23-bit mantissa of the result is computed using division logic for which as follow as: 1. Pad leading zeroes before dividend making its width, double the width of mantissa. 2. Pad leading zero before a divisor and trailing zeros after the divisor making. 3. Subtraction of dividend and divisor.4. Differences is negative put zero in the quotient else 1. 5. Right shift divisor. 6. Lift shift quotient.

A. Underlying Mantissa Division Method

The mantissa division is the most complex piece of the FP division number juggling execution.

Algorithm 1 F.P. Division Computational Flow [20]

1: ($IN1$ (*Dividend*), $IN2$ (*Divisor*)) Input Operands;
2: **Data Extraction & Exceptional Check-up:**
  {S1(Sign1), E1(Exponent1), M1(Mantissa1)} $\leftarrow IN1$
  {S2, E2, M2} $\leftarrow IN2$
  Check for Infinity, Sub-Normal, Zero, Divide-By-Zero
3: **Process both Mantissa for Sub-Normal:**
  Leading One Detection of both Mantissa ($\rightarrow$ L_Shift1, L_Shift2)
  Dynamic Left Shifting of both Mantissa
4: **Sign, Exponent & Right-Shift-Amount Computation:**
  $S \leftarrow S1 \oplus S2$
  $E \leftarrow (E1 - L\_Shift1) - (E2 - L\_Shift2) + BIAS$
  $R\_Shift\_Amount \leftarrow (E2 - L\_Shift2) - (E1 - L\_Shift1) - BIAS$
5: **Mantissa Computation:** $M \leftarrow M1/M2$
6: **Dynamic Right Shifting of Quotient Mantissa**
7: **Normalization & Rounding:**
  Determine Correct Rounding Position
  Compute ULP using Guard, Round & Sticky Bit
  Compute $M \leftarrow M + ULP$
  1-bit Right Shift Mantissa in Case of Mantissa Overflow
  Update Exponent
8: **Finalizing Output:**
  Determine STATUS signal & Resolve Exceptional Cases
  Determine Final Output

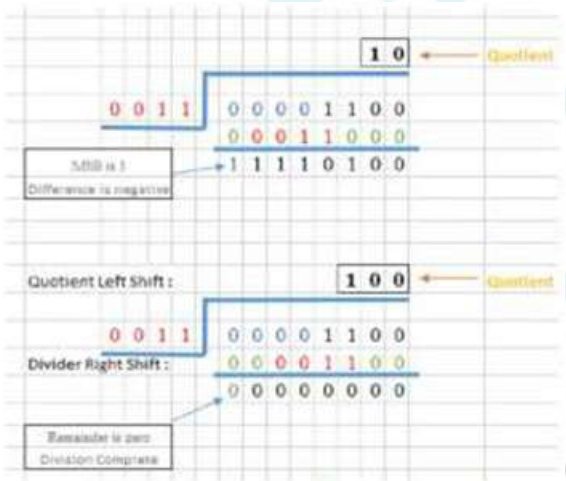


Figure 2: Floating point Vedic Divider Illustration 12 divided by 3

It depends on the arrangement extension technique for division, Where

$$q = \frac{m_1}{m_2} = \frac{m_1}{a_1 + a_2} = m_1 \times (a_1 + a_2)^{-1} \qquad (1)$$

Here, the divisor mantissa m2 is dividend in to two sections as a1 (with W+ 1-bit), and a2(every residual bit) as underneath.

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

By using Taylor series expansion,

$$(a_1 + a_2)^{-1} = a_1^{-1} - a_1^{-2}a_2 + a_1^{-3}a_2^2 - a_1^{-4}a_2^3 + \cdots \qquad (2)$$

Error occurred in the mantissa division method

$$|E_N| = |a_1^{(N+1)} a_2^N (1 - a_1^{-1}a_2 + a_1^{-2}a_2^2 - a_1^{-3}a_2^3 - \ldots)| = |\frac{a_1^{(N+1)} a_2^N}{1 + a_1^{-1}a_2}| \leq 2^{-P} \qquad (3)$$

Where, EN is error caused by all the disregarded terms. For greatest error, numerator of ought to be most extreme with the base an incentive for numerator. Thus, for most skeptical estimation (for least denominator, let (1+a1-1a2)=1, and for maxima numerator let a1-1=1).

$$|E_N| = |a_2^N| \leq 2^{-P} \qquad (4)$$

3. Proposed System

1. Proposed DPdSP Division Architecture (With 1-Stagemultiplier)

It is composed of three pipelined stages. In each stage architecture is discussed in the sub section one after one. Two 64 bit operands, one dividend (in1) and another divisor (in2) are the primary inputs along the mode control signal dp_sp.

A. First-Stage Architecture

First-stage includes the basic processing for data extraction, exceptional case handling and subnormal processing. It also includes the part of the mantissa division unit, the pre-fetching of initial approximation of divisor mantissa inverse from look-up table. A novel dual-mode Radix-4 Modified Booth multiplier is intended for the purpose of mantissa division, which has negligible area and recital overhead over single-mode multiplier, in Radix-4 which is redesigned here, to put up dual-mode processing.

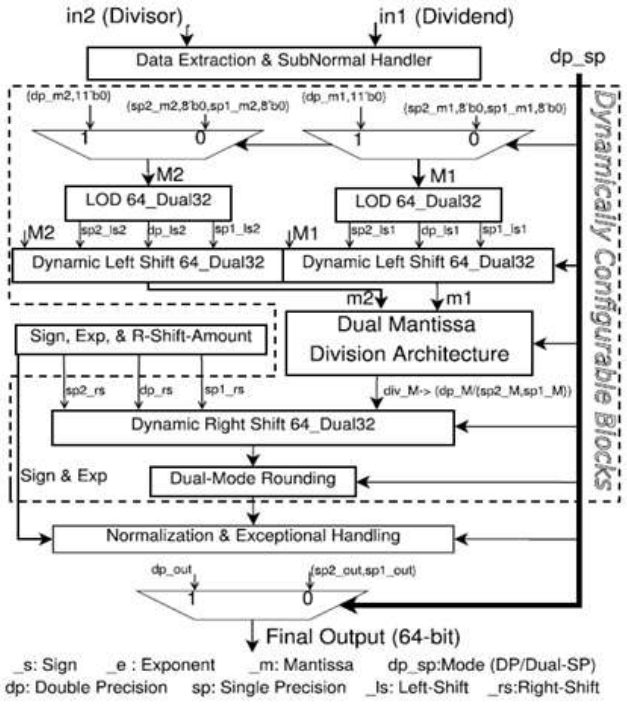


Figure 3: DPdSP division architecture

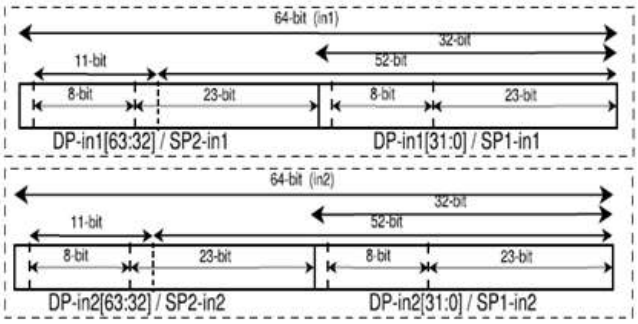


Figure 4: DPdSP input output format

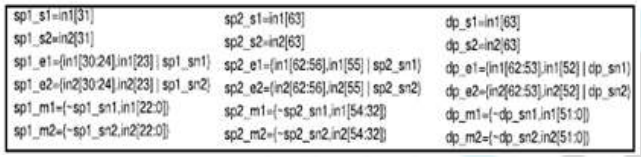
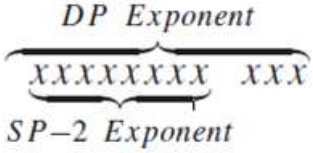


Figure 5: DPdSP division: Data Extraction

The subnormal (_sn) handling and exceptional checks are shown in Figure 6. As the 8 MSB of DP exponent overlap with SP-2 exponent, the checks for subnormal infinity and NaN have been shared among SP-2 and DP.



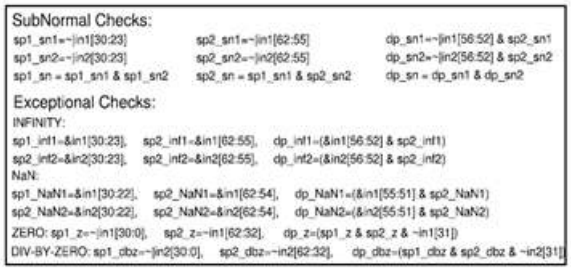


Figure 6: DPdSP division subnormal and Exceptional Handling

After left shifting, mantissa appears in to normalized form m1 and m2, as shown in Figure 3. In the next unit in this stage of division architecture, the 8-bit MSB part (a1) of normalized divisor mantissa (m2) are used to fetch the pre-computed initial approximation of their inverse, as discussed in the section II.

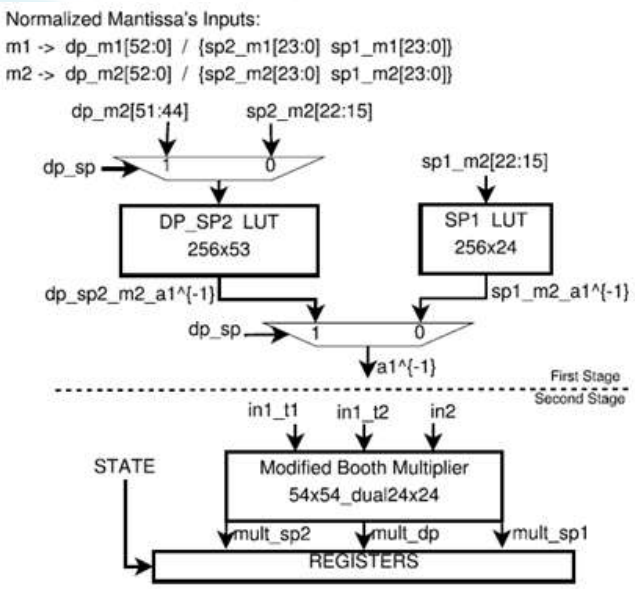


Figure 7: Dual mode mantissa Division Architecture

B. Second-Stage Architecture

In this second stage main building blocks(like like mantissa division, leading-one-detection, dynamic right/left shifting, rounding) are intended and optimized for the competent dual-mode processing. After data extraction and exceptional checks, a unified set of mantissa (M1 and M2) I generated using two MUXes.

The computations related to the sign, exponent and right shift amount processing are shown in Fig. 8. The sign computation is a simple XOR operation among both input operands sign bits. The related exponent computation is the difference of dividend (in1) exponent and divisor (in2) exponent, with proper BIAS ing and the adjustment of mantissa left shift amount (LSA):

$$BIAS + (Exp\_in_1 - LSA\_in_1) - (Exp\_in_2 - LSA\_in_2)$$

The right shift amount in subnormal output is

RSA=(Exp_in2-LSA_in2)-(Exp_in1-LSA_in1)-BIAS

All this computation is done separately for DP and both SPs.

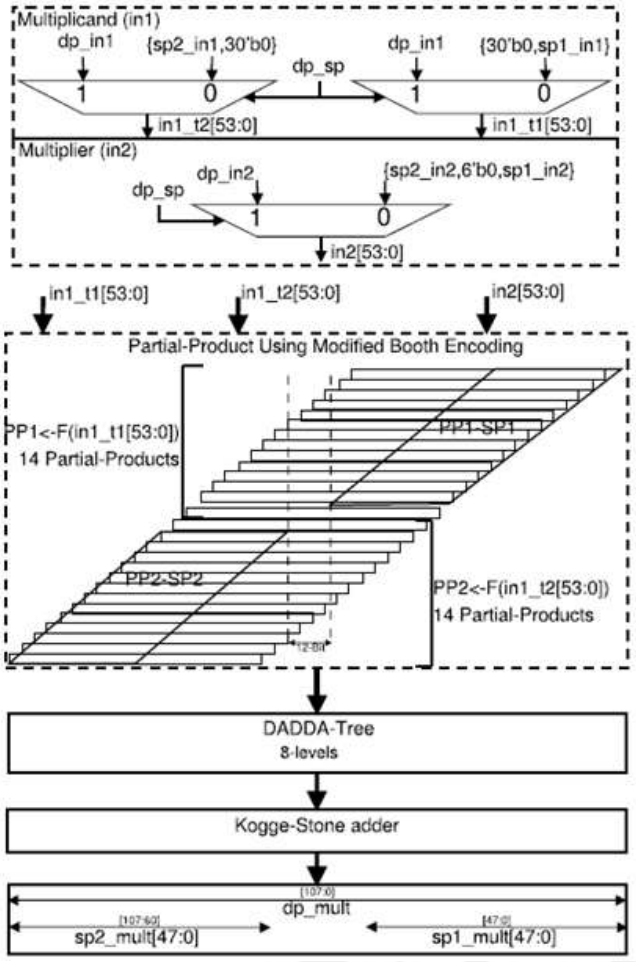


Figure 8: modified Booth Multiplier Architecture

1) Dual-Mode Radix-4 Modified Booth Multiplier Architecture

The architecture is based on the Radix-4 modified 54 bit integer multiplier which is also designed for the two parallel single mode at 24 bit unsigned operands.

2) Dual-Mode Iterative Mantissa Division Architecture

The mantissa architecture is based on the unified implementation of quotient which can process either a DP mantissa division or two parallel SPs mantissa division, with the help of above discussed dual mode modified booth multiplier.

$$\overbrace{m_1 a_1^{-1} - m_1 a_1^{-1}[(a_1^{-1}a_2 - a_1^{-2}a_2^2) \times (1 + a_1^{-2}a_2^2 + a_1^{-4}a_2^4)]}^{Double\ Precision}$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{Single\ Precision}$$

Here, m1 is the normalized dividend mantissa; and m2 is the normalized mantissa, where m2 is partitioned into a1 (first 8-bit right to the decimal point) and a2 (all remaining bits right to the a1).

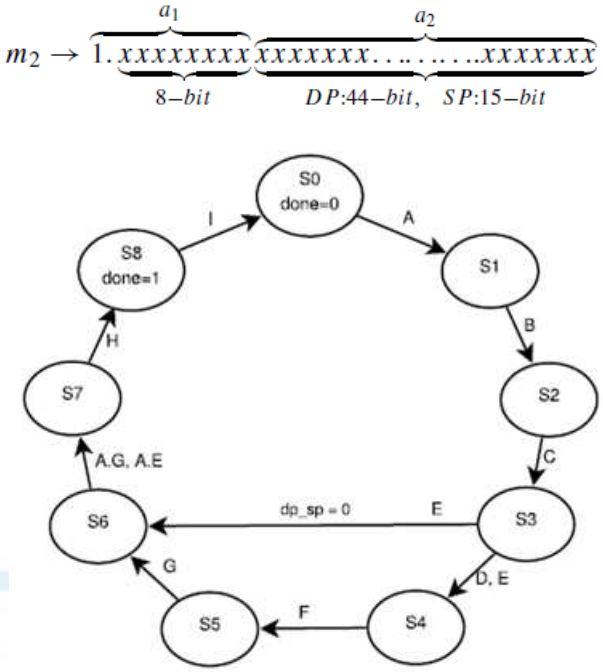


Figure 9: DPdSP dualmode Iterative Mantissa Division FSM

The FSM bits is based on the position of decimal point and processing mode for DP mode multiplication are done in 54-bit and add/sub are performed in 64-bit.

c. Third stage Architecture

In this stage, for the case of exponent underflow, mantissa division quotient is first process for the dynamic right shifting.

1) Dual mode Dynamic Right Shifting:

The right shift amount and mantissa quotient acts as primary inputs to the dual mode dynamic right shifters. The top two MUXs work for each SP mantissa quotient dynamic right shifting, which are combined by 3rd MUX to produce the DP mantissa dynamic right shifting.

2) Dual-mode Rounding:

We are using round-to nearest method. It is based up two steps, first unit-at-last-place (ULP) addition with quotient mantissa. It has two logic gates and two one-bit 2:1 MUXs as an overhead over single-mode DP rounding.

3) Final processing:

In rounded mantissa quotient is normalized for both DPdSP, in case of any mantissa overflow rounding ULP addition, it requires 1-bit right shift overflow. Mantissa is updated to exceptional cases infinity, divided by zero, NaN, along with overflow handling, which needs separate from DPdSP.

Algorithm 2 Exceptional Case Processing at Output

1: (*IN1* (*Dividend*), *IN2* (*Divisor*));
2: **if** IN1 = NaN or IN2 = NaN or IN1=IN2=INFINITY or IN1=IN2=ZERO **then**
3: IN1/IN2 ← NaN
4: **else if** (IN2 = INFINITY & IN1 != INFINITY/NaN) or IN1 = ZERO **then**
5: IN1/IN2 ← ZERO
6: **else if** (IN1 = INFINITY & IN2 != INFINITY/NaN) or (IN2 = ZERO → DIV-BY-ZERO) or (Output OVER-FLOW) **then**
7: IN1/IN2 ← INFINITY
8: **else if** For Normal or Underflow Case **then**
9: IN1/IN2 ← Computed Results

4. Proposed DPdSP Division Architecture (With 2-Stage Multiplier)

All stages of previous architecture are participated into two pipelined stages as given below:

First stage is split in to two stages by inserting a pipelined register in the dual mode dynamic left shifter. Second stage is used in the sign; exponent and right shift amount related processing are still compressed in a single stage due the similar delay of the components.

5. DP Division Implementation

Both the architectures, with 1-stage multiplier and two stage multiplier are constructed respectively, for the DP related processing and easily sought from the architectures in DPdSP.

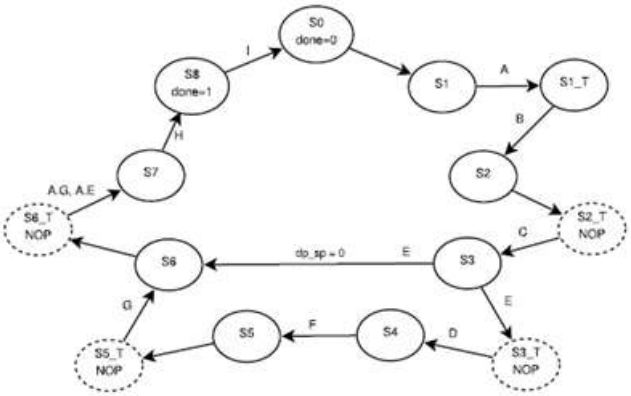


Figure 10: DPdSP Iterative Mantissa FSM with 2-stage multiplier

6. Simulation Results

The simulation and synthesis result are proposed architecture of floating point division in a double precession with compression drawn out for existing system.

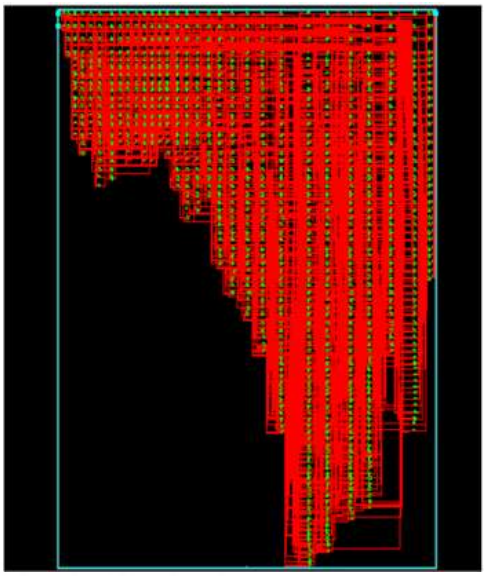


Figure 11: Schematic of the proposed model

S. No.	DESIGN	AREA(Slices & LUTS)	DELAY Ns
1	Existing FP Division	1278	33.25
2	Bandwidth	797	18.77

Figure 12: simulation result of proposed floating point division

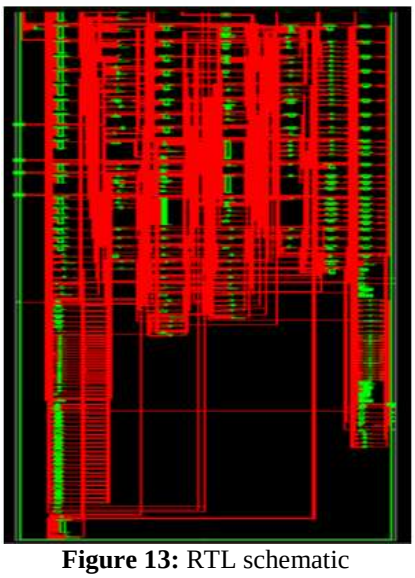


Figure 13: RTL schematic

7. Conclusion

Finally, the conclusion is presented two dual-mode iterative architecture for double precision floating point division arithmetic. We discussed in this two architecture with different pipeline levels, 3 and 6 stages architectures, comprised of 1-stage and 2 stage multiplier respectively, with area, period and throughput trades offs.

The proposed architecture outperforms the prior art on this in terms of required area, period and throughput in cycles, and unified metric AREA*PERIOD*TROUGHPUT (in cycles).

Based on the current proposed DPdSP division architecture, similar architectures for dual-mode division can be formed using other multiplicative based methods (like Newton-Raphson, Goldschmidt) of division. Future we will targets on the architectures.

References

[1] J.-C. Jeong, W.-C. Park, W. Jeong, T.-D. Han and M.-K. Lee, “A cost effective pipelined divider with a small lookup table, ” IEEE Trans. Comput., vol. 53, no. 4, pp. 489–495, Apr. 2004.
[2] S. F. Oberman and M. Flynn, “Division algorithms and implementations, ”IEEE Trans. Comput., vol. 46, no. 8, pp. 833–854, Aug. 1997.
[3] M. K. Jaiswal and R. C. C. Cheung, “High performance reconfigurable architecture for double precision floating point division, ” in Proc. 8thInt. Symp. Appl. Reconfigurable Comput. (ARC), Hong Kong, China, Mar. 2012, pp. 302–313.
[4] X. Wang and M. Leeser, “VFloat: A variable precision fixed- and floating-point library for reconfigurable hardware, ” ACM Trans. Reconfigurable Technol. Syst., vol. 3, no. 3, pp. 16:1–16:34, Sep. 2010.
[5] M. K. Jaiswal, R. Cheung, M. Balakrishnan, and K. Paul, “Series expansion based efficient architectures for double precision floating point division, ” Circuits, Syst., Signal Process., vol. 33, no. 11, pp. 3499–3526, 2014. [Online]. Available: http://dx.doi.org/10.1007/s00034-014-9811-8
[6] J.-M. Muller et al., Handbook of Floating-Point Arithmetic, 1st ed. Basel, Switzerland: Birkhäuser, 2009.
[7] P. Soderquist and M. Leeser, “Area and performance tradeoffs in floating-point divide and square-root implementations, ” ACM Comput. Surv., vol. 28, no. 3, pp. 518–564, Sep. 1996.
[8] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.
[9] B. Pasca, “Correctly rounded floating-point division for DSP-enabled FPGAs, ” in Proc. 22nd Int. Conf. Field Program. Logic Appl. (FPL), Aug. 2012, pp. 249–254.
[10] A. Akka¸s, “Dual-mode quadruple precision floating-point adder,” in Proc. Euromicro Symp. Digit. Syst. Design, 2006, pp. 211–220.
[11] M. Ozbilen and M. Gok, “A multi-precision floating-point adder,” in Proc. Ph.D. Res. Microelectron. Electron. (PRIME), 2008, pp. 117–120.
[12] M. K. Jaiswal, R. C. C. Cheung, M. Balakrishnan, and K. Paul, “Unified architecture for double/two-parallel single precision floating point adder, ” IEEE Trans. Circuits Syst. II, Express Briefs, vol. 61, no. 7, pp. 521–525, Jul. 2014.
[13] A. Akka¸s, “Dual-mode floating-point adder architectures, ” J. Syst. Archit., vol. 54, no. 12, pp. 1129–1142, Dec. 2008.
[14] M. Jaiswal, B. Varma, H.-H. So, M. Balakrishnan, K. Paul, and R. Cheung, “Configurable architectures for multi-mode floating point adders,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 62, no. 8, pp. 2079–2090, Aug. 2015.
[15] A. Baluni, F. Merchant, S. K. Nandy, and S. Balakrishnan, “A fully pipelined modular multiple precision floating point multiplier with vector support, ” in Proc. Int. Symp. Electron. Syst. Design (ISED), 2011, pp. 45–50.

Authors Profile

Mrs. Shaik. Asiya, pursuing MTech in the department of ECE at Yogananda Institute of Technology & Sciences, Tirupati. My research area of interests includes of VLSI and Digital and Analog circuits.

Mr. P. Jaya Rami Reddy, working as Associate Professor and Head of the department of ECE at Yoganandha Institute of Technology and Sciences in Tirupati. He is member in the institute of Engineers (IE) and has worked experience of 17 years. His area on interest includes VLSI, Embedded systems and Image Processing.