*International Journals of Advanced Research in Computer Science and Software Engineering*
*ISSN: 2277-128X (Volume-8, Issue-4)*

**Research Article** | **April 2018**

**Special Issue: National Conference on Emerging Trends in Engineering 2018**
**Conference Held at Sri Venkatesa Perumal College of Engineering & Technology, Puttur, A.P., India**

# VLSI Design of Dual-Mode Double Precision Floating Point Division with Reduced Area

**Asiya. Shaik, P. Jaya Rami Reddy**
*Department of Electronics and Communication Engineering, Yogananda Institute of Technology and Sciences Thirupathi, Andhra Pradesh India*
Email: shaik.asiya28@gmail.com, pjrece@gmail.com

***Abstract: Floating point division is a core arithmetic widely used in scientific and engineering applications. This paper proposed an architecture for double precision floating point division. This architecture is designed for dual-mode functionality, which can either compute on a pair of double precision operands or on two pairs of single precision operands in parallel. The architecture is based on the series expansion multiplicative methodology of mantissa computation. For this, a novel dualmodeRadix-4 Modified Booth multiplier is designed, which is used iteratively in the architecture of dual-mode 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. The proposed dual-mode architecture is synthesized using Xilinx14.7technology. Two versions of proposed architecture are presented, one with single stage multiplier and another with two stage multiplier.***

***Index Terms—Arithmetic, configurable architecture, dual-mode division, floating point division, multi-precision arithmetic.***

.

## I. INTRODUCTION

FLOATING point arithmetic (FPA) architectures underwent significant advancement by scientific research in the past several decades. FPA is a basic ingredient of a large set of scientific and engineering domain applications. To boost the application performances, 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 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. This configurable block array arrangement can lead to significant area improvement, while providing there required performance.
Our examination work is centered around the engineering outline of configurable coasting point math pieces. This paper is centered around the outline of configurable double mode twofold exactness division number juggling unit. Floating point (FP) division is a center calculation required in a huge number of uses. The proposed design can be designed either for a twofold exactness or two parallel (double)single accuracy division calculations, and along these lines named as DPdSP division design.
The main contributions of this work can be summarized as follows:

- 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 divisions.
- Anovel 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 FP division flow is designed for efficient dual-mode functionality with minimal overhead.
- Proposed architectures are fully pipelined, and designed in an iterative fashion for area-efficiency.

## II. BACKGROUND

The basic computational stream for FP division math is exhibited in Algorithm 1. This calculation is appropriate for both ordinary and sub-typical handling. It likewise incorporates the uncommon case taking care of and preparing.

**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

FPA execution includes registering independently the sign, example and mantissa part of the operands, and later consolidating them subsequent to adjusting and standardization . In the present work, all phases of the above computational stream are intended to help double mode operations.

### *A. Underlying Mantissa Division Method*

The mantissa division is the most complex piece of the FP division number juggling execution. The algorithmic procedure for this calculation is examined here. It depends on the arrangement extension technique for division, as takes after. Give m1 a chance to be the standardized profit mantissa and m2 be the standardized divisor mantissa, and after that q, the mantissa remainder, can be processed as:

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

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

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

By using Taylor Series expansion,

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

The above condition can be assessed by utilizing just multipliers, adders and subtractors, gave that the estimation of a1−1 is accessible. The preprocessed estimation of a1−1 can be gotten to from a pre-accumulated up table to play out the whole calculation; which is effectively feasible in equipment execution. The pre-registered estimation of a1−1 goes about as an underlying guess for m1−2, which additionally enhanced with residual calculation in (2). Here, the size W (bit width) of a1 (here, the shrouded "1" bit put in a1 isn't checked, as it stays as steady an incentive in the standardized configuration) decides the span of memory (for look-into table to store a1−1) and the quantity of terms from the arrangement development, to play out the calculation for a given precision. The quantity of terms (N) (with a given W) for a given precision prerequisite (2−P) can be dictated by following imbalance:

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

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 maximum numerator let a1−1 = 1),

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

## III. PROPOSED DPDSP DIVISION ARCHITECTURE (WITH 1-STAGE MULTIPLIER)

The proposed engineering is appeared in Fig. 1. It is made of three pipelined stages. The points of interest of each stage design is talked about underneath in following subsections one-by-one. Both of the input operands either contains DP operands (as whole 64-bit combine) or two parallel SP operands (as two arrangements of 32-bit match), as appeared in Fig. 2.

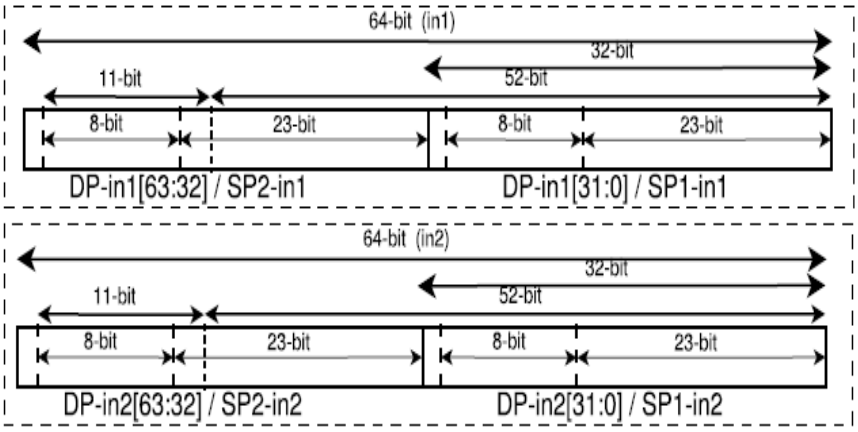



Fig. 2. DPdSP Input Output Format. The input/output encoding is based on the IEEE standard binary format.

The sub-normal (_sn) handling and exceptional checks computations are shown in Fig. 4. As the 8 MSB of DP exponent overlap with SP-2 exponent, the checks for sub-normal, infinity and NaN (Not-A-Number)have been shared among SP-2 and DP, as shown in Fig. 4.It also performs checks for divide-by-zero (_dbz) and zero (_z), and have been shared among DP and both SPs.

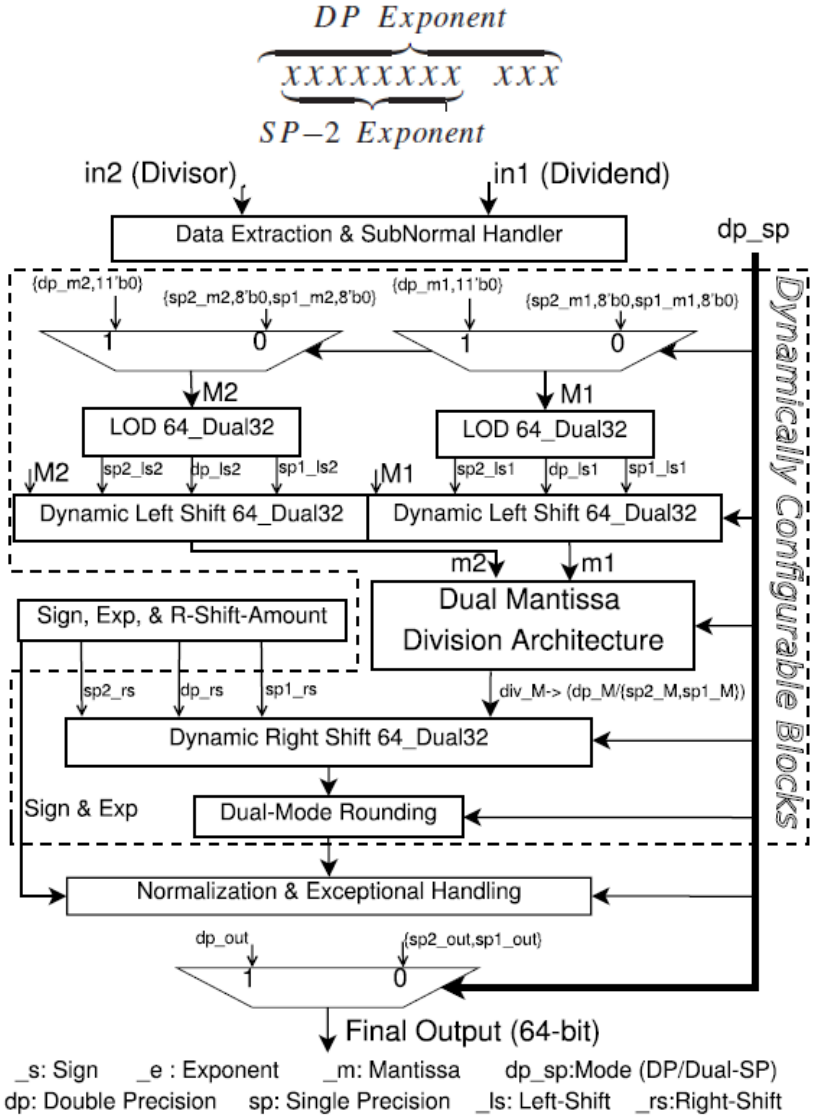



Fig. 1. DPdSP Division Architecture.

```
sp1_s1=in1[31]                    sp2_s1=in1[63]                    dp_s1=in1[63]
sp1_s2=in2[31]                    sp2_s2=in2[63]                    dp_s2=in2[63]
sp1_e1={in1[30:24],in1[23] | sp1_sn1}   sp2_e1={in1[62:56],in1[55] | sp2_sn1}   dp_e1={in1[62:53],in1[52] | dp_sn1}
sp1_e2={in2[30:24],in2[23] | sp1_sn2}   sp2_e2={in2[62:56],in2[55] | sp2_sn2}   dp_e2={in2[62:53],in2[52] | dp_sn2}
sp1_m1={~sp1_sn1,in1[22:0]}       sp2_m1={~sp2_sn1,in1[54:32]}      dp_m1={~dp_sn1,in1[51:0]}
sp1_m2={~sp1_sn2,in2[22:0]}       sp2_m2={~sp2_sn2,in2[54:32]}      dp_m2={~dp_sn2,in2[51:0]}
```

Fig. 3. DPdSP Division: Data Extraction.

After data extraction and exceptional checks, a unified set of mantissa (M1 and M2) is generated using two MUXes (as shown in Fig. 1). Based on the mode of operation, these contain the mantissa either for DP or for both SPs. This unification of mantissas helps in designing a tuned datapath processing for later stage computation, which results inefficient resource sharing.

```
SubNormal Checks:
sp1_sn1=~|in1[30:23]              sp2_sn1=~|in1[62:55]              dp_sn1=~|in1[56:52] & sp2_sn1
sp1_sn2=~|in2[30:23]              sp2_sn2=~|in2[62:55]              dp_sn2=~|in2[56:52] & sp2_sn2
sp1_sn = sp1_sn1 & sp1_sn2        sp2_sn = sp1_sn1 & sp2_sn2        dp_sn = dp_sn1 & dp_sn2

Exceptional Checks:
INFINITY:
sp1_inf1=&in1[30:23],   sp2_inf1=&in1[62:55],   dp_inf1=(&in1[56:52] & sp2_inf1)
sp2_inf2=&in2[30:23],   sp2_inf2=&in2[62:55],   dp_inf2=(&in2[56:52] & sp2_inf2)
NaN:
sp1_NaN1=&in1[30:22],   sp2_NaN1=&in1[62:54],   dp_NaN1=(&in1[55:51] & sp2_NaN1)
sp2_NaN2=&in2[30:22],   sp2_NaN2=&in2[62:54],   dp_NaN2=(&in2[55:51] & sp2_NaN2)
ZERO: sp1_z=~|in1[30:0],   sp2_z=~in1[62:32],   dp_z=(sp1_z & sp2_z & ~in1[31])
DIV-BY-ZERO: sp1_dbz=~|in2[30:0],   sp2_dbz=~in2[62:32],   dp_dbz=(sp1_dbz & sp2_dbz & ~in2[31])
```

Fig. 4. DPdSP Division: Sub-Normal and Exceptional Handling.

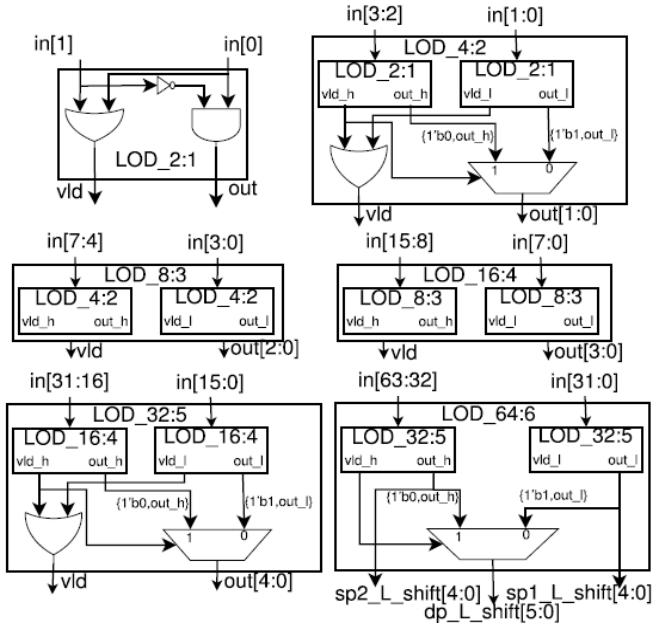



Fig. 5. DPdSP Division: Dual-Mode LOD.

The architecture for dual mode dynamic left shifter is shown in Fig. 6. It is a 6-stage barrel shifter unit, in which first stage is a single mode unit, and the 5 remaining stages are dual mode units. The first stage unit is a simple left-shift barrel shifter which performs shifting only in double precision mode, based on the MSB of DP shift bits.. It works either for DP or for dual SP dynamic left shifting. A dual mode stage contains two multiplexers for each 32-bit blocks, which shift their inputs based on the corresponding shifting bits (either of DP or both SPs). The shifting amount for a given dual-mode stage is given by $y = 2x$, where "x" is the shifting-bit position for that stage. The dual-mode stage also contains a multiplexer which selects between 32-bit MSB shifting output or their combination with primary 64-bit input to the stage, based on the true *dp, sp* and corresponding shifting bit of DP left shift. Except this multiplexer, the dual-mode stages behaves like two separate32-bit barrel shifter, which are constructed to support dual mode left shifting operation.

***B. Second-Stage Architecture***

The second stage architecture performs core operation of division architecture. It computes on the core sign, exponent and mantissa processing of FP division arithmetic and the computation related to right shift amount, which all correspond to the steps 4 and 5 of Algorithm 1. The mantissa computation is the most complex part of the floating point division arithmetic. Here, its related architecture includes the unified and dual-mode implementation of (7).As shown in Fig, the initial inverse approximation of divisor mantissa is fetched in first stage of architecture. The remaining computation is built around a dual-mode booth multiplier, in an iterative fashion. A dual-mode finite state machine (FSM)is designed which decides the effective inputs for multiplier in each state and which will be discussed shortly after the description of dual mode multiplier architecture.

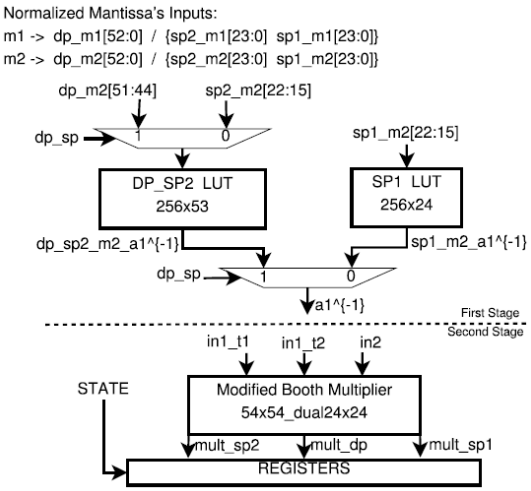



Fig.6. DPdSPDualModeMantissaDivision Architecture.

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

The architecture is based on the Radix-4Modified Booth Encoding, which reduces the number of partial products at most to ( $n2+ 1$) [8]. Here, it is a 54-bit integer multiplier (for DP processing), which is also designed to process two parallel sets of 24-bit unsigned operands (for two SPs processing) multiplication. In fact it can process two parallel sets of 26-bit unsigned operand multiplication, using entire 14 partial products of PP1 for first set and entire PP2 for second set operand, without corrupting each other. However, here it is presented for current requirement only. The presented dual-mode Booth architecture has three input 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". The computations related to the sign, exponent and right shift amount processing are shown in Fig..

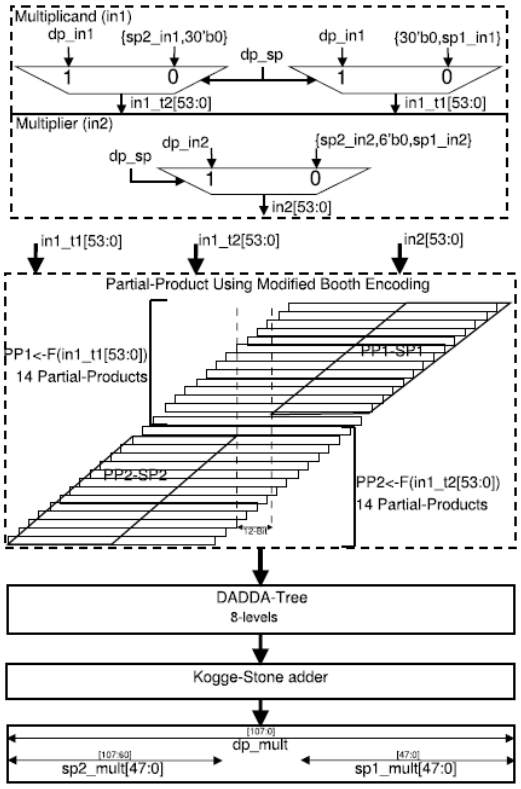


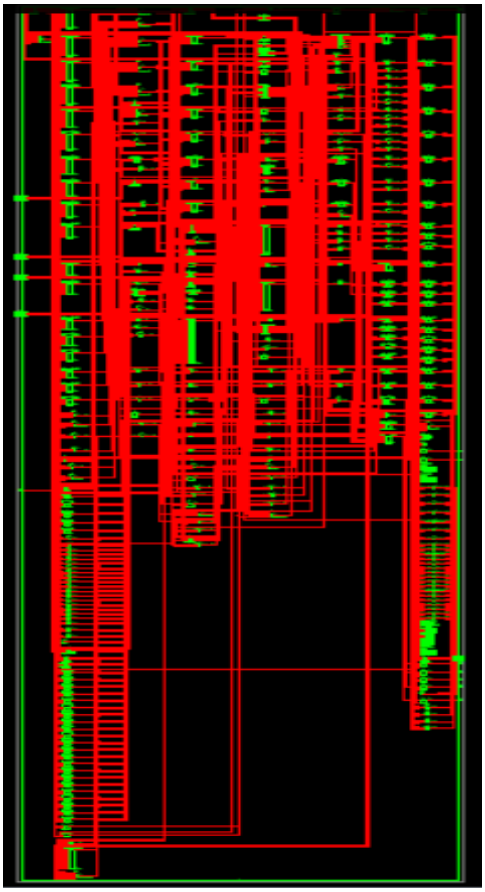

Fig.7. Dual-Mode Modified Booth Multiplier Architecture.

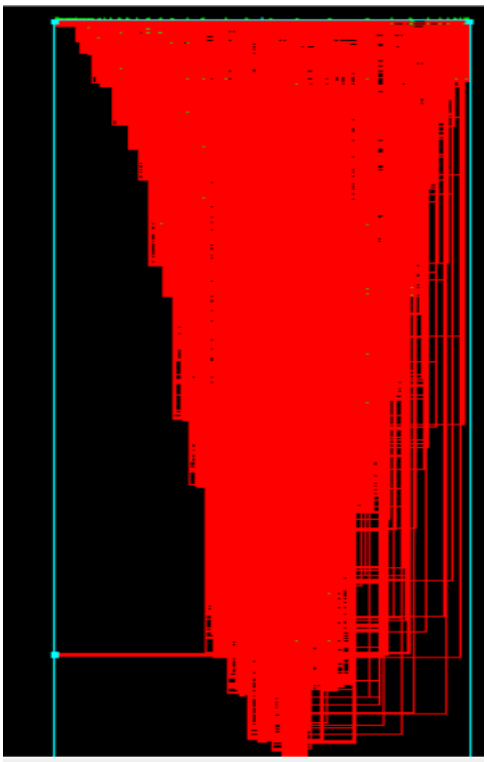
The sign computation is a simple XOR operation among both input operand assign bits. The related exponent computation is the difference of dividend (*in*1) exponent and divisor (*in*2) exponent, with proper BIASing and the adjustment of mantissa left shift amount (LSA):

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

## IV. RESULTS

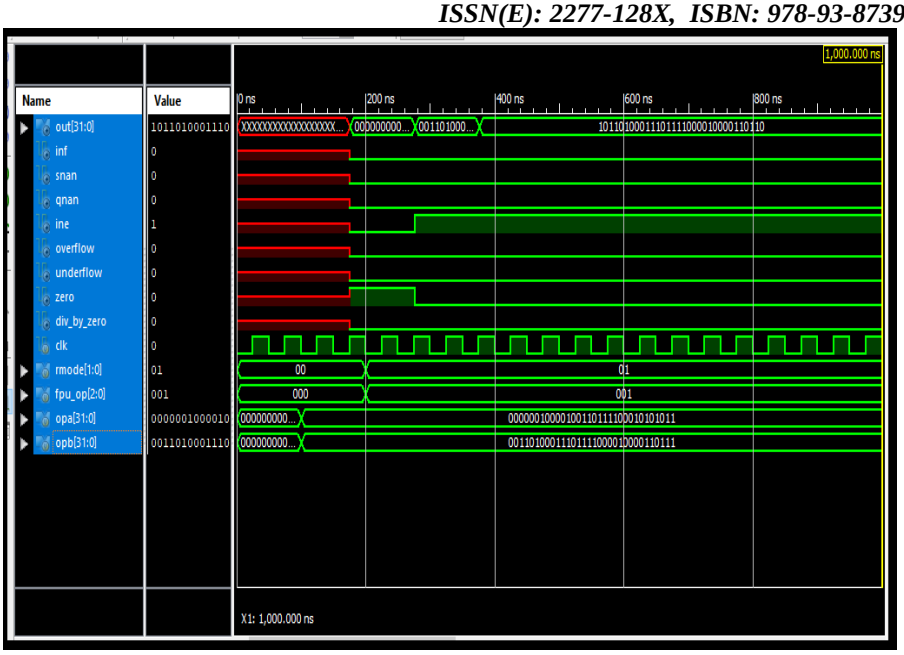
Simulation output





RTL Schematic




Technological schematic

## V. CONCLUSION

This paper has presented two dual-mode iterative architectures for double precision floating point division arithmetic. It can be dynamically configured for double precision with dual single precision (DPdSP) floating point division arithmetic. All the components are designed for efficient dual mode processing. A novel dual-mode Radix-4 Modified Booth multiplier architecture is proposed with minimal overhead, forthe purpose of dual-mode mantissa processing. The entire data path has been tuned to perform the dual mode computation with minimal hardware overhead. The proposed architecture outperforms the prior art on this in terms of required area, period, throughput in cycles, and unified metric *Area ×Period (FO4) ×Throughput (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. Our future work on this will be targeting these architectures.

## REFERENCES

[1] J.-C. Jeong, W.-C. Park, W. Jeong, T.-D. Han, and M.-K. Lee, "A costeffectivepipelined 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 reconfigurablearchitecture for double precision floating point division," in *Proc. 8th Int. Symp. Appl. Reconfigurable Comput. (ARC)*, Hong Kong, China,Mar. 2012, pp. 302–313.

[4] X. Wang and M. Leeser, "VFloat: A variable precision fixed- andfloating-point library for reconfigurable hardware," *ACM Trans. ReconfigurableTechnol. Syst.*, vol. 3, no. 3, pp. 16:1–16:34, Sep. 2010.

[5] M. K. Jaiswal, R. Cheung, M. Balakrishnan, and K. Paul, "Seriesexpansion based efficient architectures for double precision floating pointdivision," *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 infloating-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-enabledFPGAs," 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 floatingpoint 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, andR. Cheung, "Configurable architectures for multi-mode floating pointadders," *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 fullypipelined modular multiple precision floating point multiplier withvectorsupport," in *Proc. Int. Symp. Electron. Syst. Design (ISED)*, 2011,pp. 45–50.

## AUTHORS BIOGRAPHIES

**Mrs Shailk.Asiya,** puruing MTech in the dept of ECE at Yogananda Institute of Technology & Sciences, Tirupati. My research area of interests includes Analog and Digital Circuits.

**Mr P.Jaya Rama Reddy**, worked as associate professert and head of the departmant of ECE at yogananda Institute of Technology & Sciences, Tirupathi. He has member of the institute of engineers (IE) and has working exprence of 17 years. His aera instrest includes VLSI, Embedded systems and image processing.