ENERGY-EFFICIENT VLSI REALIZATION OF BINARY64 DIVISION WITH REDUNDANT NUMBER SYSTEMS

[1]AVANIGADDA. NAGA SANDHYA RANI [2]BALA KRISHNA.KONDA M.Tech, Assistant Professor
[1,2]Eluru College Of Engineering And Technology, Duggirala, Pedavegi, West Godavari, Andhra Pradesh, INDIA

[1]nagasandhyarania@gmail.com [2] kbalu.crm23@gmail.com

Abstract: VLSI realizations of digit-recurrence binary division usually use redundant representation of partial remainders and quotient digits. The former allows for fast carry-free computation of the next partial remainder, and the latter leads to less number of the required divisor multiples. In studying the previous relevant works, we have noted that the binary carry save (CS) number system is prevalent in the representation of partial remainders, and redundant high radix representation of quotient digits is popular in order to reduce the cycle count. In this paper, we explore a design space containing four division architectures. These are based on binary CS or radix-16 signed digit (SD) representations of partial remainders. On the other hand, they use full or partial precomputation of divisor multiples. The latter uses smaller multiplexer at the cost two extra adders, where one of the operands is constant within all cycles. The quotient digits are represented by radix-16 SDs. Our synthesis-based evaluation of VLSI realizations of the best previous relevant work and the four proposed designs show reduced power and energy figures in the proposed designs at the cost of more silicon area and delay measures. However, our energy-delay product is 26%–35% less than that of the reference work.

Index Terms— Carry save (CS), digit recurrence binary division, energy efficiency, radix-16 signed digit (SD), redundant number system.

INTRODUCTION

DIVISION is the less frequent operation among the four basic arithmetic operations that are carried out within the execution of a typical task on digital processors. On the other hand, it is the most complex and time consuming operation. VLSI realization of dividers is generally based on two classes of algorithms, namely, subtractive (aka digit recurrence) and multiplicative (aka functional). . Quotient digit selection (QDS) is very simple in conventional binary division algorithms, such as non restoring division scheme [1], where the next quotient bit is obtained just by examining the sign of partial remainder. However, in order to reduce the number of recurrences, radix-2h (e.g., h = 4) division schemes have been proposed at the cost of more complex QDS, since one out of 2h possible digit values is to be selected. This is undertaken via comparing the partial remainder with a set of divisor multiples [2]. In order to reduce the generation cost of such multiples, the quotient digits are often selected from a signed digit (SD) set [3] (e.g., $[-2h-1,\ 2h-1]$), and converted on the fly into the desired binary output.

On the other hand, the SD representation of partial remainders has been employed to enable borrow free subtraction that shortens the cycle time. However, sign detection of SD numbers, which is required in QDS, is not a trivial operation. Despite the less frequent occurrence of division in comparison with other basic operations, the several addition operations that are embedded within a digit recurrence division contribute to extra energy consumption.

I. LITERATURE SURVEY

Power Efficient Division and Square Root Unit.

Although division and square root are not frequent operations, most processors implement them in hardware to not compromise the overall performance. Two classes of algorithms implement division or square root: digit-recurrence and multiplicative (e.g., Newton-Raphson) algorithms. Previous work shows that division and square root units based on the digit-recurrence algorithm offer the best tradeoff delay-area-power. Moreover, the two operations can be combined in a single unit. Here, we present a radix-16 combined division and square root unit obtained by overlapping two radix-4 stages. The proposed unit is compared to similar solutions based on the digit-recurrence algorithm and it is compared to a unit based on the multiplicative Newton-Raphson algorithm.

II. EXISTING SYSTEM

The division operation can be defined as $X = Q\,D + R$, where X and D represent the dividend and divisor,

respectively, as the input operands. The results are denoted as the quotient Q and the remainder R. In hardware realization of digit recurrence division algorithms, it is often postulated that $X < D$, and the divisor is a normalized fraction, such that in radix-2^h division $1/2^h \leq D < 1$. Equation (1) describes the jth recurrence, where $W[j]$, q_{j+1}, and $Q[j]$, represent the j th partial remainder, the next quotient digit, and the partial quotient, respectively. In addition, $0 \leq j < n$, $W[0] = X$, $Q[0] = 0$, and n denotes the precision of D and Q

where the next quotient digit is obtained as $q_{j+1} = 4q^h_{j+1} + q^l_{j+1}$, with $q^h_{j+1}, q^l_{j+1} \in [-2,2]$.

$$W[j+1] = 2^h W[j] - q_{j+1} D$$
$$Q[j+1] = Q[j] + 2^{-h(j+1)} q_{j+1}. \quad (1)$$

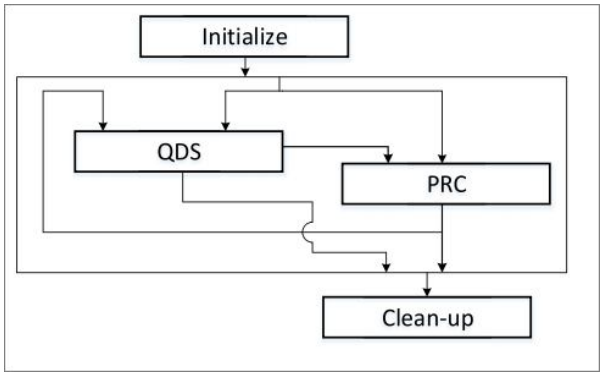
On the other hand, in order to speed up the QDS, truncated (to some t n fractional digits) partial remainders and comparison constants are used, where t is so determined to guarantee that the convergence condition (defined by (3) [1]) is not violated

$$M_k \leq 2^h W[j] < M_{k+1} \Longrightarrow q_{j+1} = k \quad (2)$$
$$-\rho D \leq W[j+1] \leq \rho D, \quad \rho = (a/(2^h - 1)). \quad (3)$$

To speed up the partial remainder computation (PRC), the partial remainders are often represented via a redundant number system. For example, use binary CS, and employ the binary SD (BSD) representations. However, we have not encountered any relevant work with high radix redundant partial remainders.

Use of Signed Digit Number Systems

Utilization of a high radix SD number system (for partial remainder representation) entails particular carry-free addition, based on (1). This regards a previous work [13] and two of the proposed designs in this paper, which use radix-16 SD representation of partial remainders. The digit set of the former is not indicated, while those of the proposed designs are MRSD. One 4-bit adder per radix-16 position, whose input bits are distinguished within dashed boxes, is used to produce the sum digits (surrounded by dashed curves). The 4-bit adders produce the negabit and three most significant posibits of the sum digit, in the same radix-16 position, and the carry-out rests at the least significant position of the next sum digit.



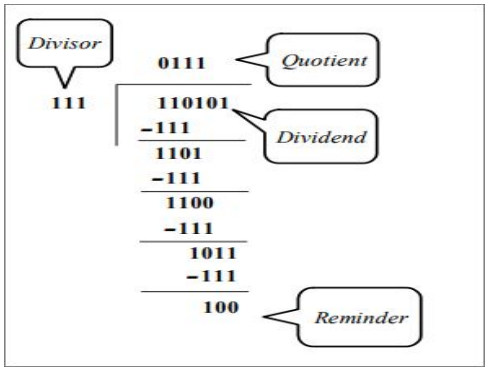
General division architecture.

III. PROPOSED SYSTEM

Division is the less frequent operation among the four basic arithmetic operations that are carried out within the execution of a typical task on digital processors. On the other hand, it is the most complex and time consuming operation.To perform binary division follow the following steps:

1. Align the divisor (Y) with the most significant end of the dividend. Let the portion of the dividend from its MSB to its bit aligned with the LSB of the divisor be denoted X.
2. Compare X and Y. a) If X >= Y, the quotient bit is 1 and perform the subtraction X-Y. b) If X < Y, the quotient bit is 0 and do not perform any subtractions.
3. Shift Y one bit to the right and go to step 2.

Example:



X		Y	Q	Action
110	<	111	$Q_1 = 0$	Do Not Subtract
1101	>	111	$Q_2 = 1$	Subtract
1100	>	111	$Q_3 = 1$	Subtract
1011	>	111	$Q_4 = 1$	Subtract

Digit-Recurrence Division Algorithms

In this work, we want to determine which division algorithm type should be used for a dedicated division unit in an asynchronous processor. Based on the specific unit criteria, we choose the digit-recurrence algorithms class. As the digit-recurrence algorithms produce their output digit by digit (with the most significant one first), we define the quotient after iterations $q[j]=\sum_{i=1}^{j} q_i r^{i-1}$We can notice that the value of the radix r is very important in this kind of algorithm. From the residual computed during the last iteration , we can easily deduce the remainder of the division . The digit-recurrence is based on the following residual iteration :

$$\begin{cases} w[0] = x \\ w[j+1] = rw[j] - q_{j+1}d \end{cases} \quad (1)$$

This iteration can be implemented using several solutions.

SRT algorithm: It was introduced to avoid the full-width comparisons of the restoring algorithm. The quotient is represented using a redundant number system. The idea is based on a faster choice for the values of $q_{j+1}$by the examination of a few most significant digits of the residual and the divisor. This is possible due to redundant representation of the quotient.

The computation performed in the recurrence of Equation 1 is quite simple, and it is illustrated in Figure 2. The multiplication of the residual by the the radix is straightforward if is a power of 2. In this case, the multiplication is a constant shift implemented using wires. The selection function depends on the algorithm type: restoring or SRT. It is implemented using comparators or tables. This block of the iteration can require an important part of the computation time. In our case, we only implement a radix-2 restoring algorithm and some SRT algorithms with radix between 2 and 16. The quotient digit by divisor product is quite simple for low radix values. The choice of the digits in is also important. The product should be implemented using a few simple operations (additions, constant shifts and complements). This is possible if the digits are small integers (3=2+1, 5=4+1, 7=8-1. . .).

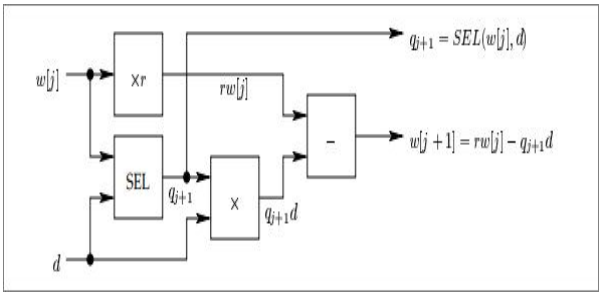


Figure 2: Architecture of the digit-recurrence stage.

Redundant Number System

Redundant number representations are used implicitly in many algorithms for which the final result appears in a conventional non-redundant form. Examples of such algorithms are carry-save schemes for multiplication and SRT division algorithms. Based on signed-digit representation, the basic operations of addition, subtraction, multiplication and division can be performed very efficiently. Implementations of these basic operations are described in the following section. Multiplication is performed in a similar fashion to carry-save multiplication so that the time required is O(w) for n digit arguments.Alternatively, the internal representation of numbers and all the basic operations can be based on a redundant number system. However, the properties of floating-point arithmetic based on a signed-digit representation.

Carry Save

In Carry Save (CS), three bits are added parallelly at a time. In this scheme, the carry is not propagated through the stages. Instead, carry is stored in present stage, and updated as addend value in the next stage. Hence, the delay due to the carry is reduced in this scheme. It is also same as full Adder.

Radix 16 Signed Digit.

A two-stage algorithm for fixed point, radix-16 signed-digit division is presented. The algorithm uses two limited precision radix-4 quotient digit selection stages to produce the full radix-16 quotient digit. The algorithm requires a two-digit estimate of the (initial) partial remainder and a three-digit estimate of the

divisor to correctly select each successive quotient digit. A set of general equations for determining the ranges of normalized signed-digit numbers is derived. Another set of general equations for determining the precisions of estimates of the divisor and dividend are derived. These two sets of equations permit design tradeoff analyses to be made with respect to the complexity of the model division.

PROPOSED ARCHITECTURES

Within the framework of Fig. 2, static and semi dynamic DMGs and two different representations for partial remainders provide us with a design space based on the following options.

1) Radix-16 Quotient Digit Set: This choice, as in the previous relevant works, leads to the reduced number of cycles versus the direct generation of quotient bits.
2) SD Representation of Quotient Digits: We use [−9, 9] radix-16 SD set for the intermediate representation of quotient digits.
3) Semi dynamic DMG: The [−9, 9] multiples of divisor that are needed in the PRC are normally obtained within the initialization cycle, where ten-way multiplexer is required for selecting $q_{j+1}D$.

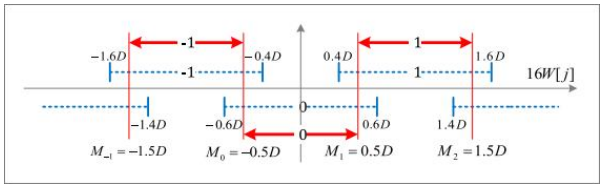


Fig. 3. Overlapped zones for specific quotient digit values

as $\pm\{6D - 2D, 6D \mp D, 6D + 2D, 6D + 3D\}$, respectively, within each recurrence.

4)Use of Redundant Number Systems for PRC: The previous relevant works have opted for CS representation of partial remainders. To be able to independently show the advantage of aforementioned semi dynamic DMG, we also use CS as one option, which due to doubling the representation storage does not seem to be a proper choice when lower power dissipation is desirable.

A. Quotient Digit Selection.

The convergence condition ($-0.6D \le W[j+1] \le 0.6D$) is partially unfolded as in Fig. 3, which suggests the comparison of the partial remainder with a set of divisor multiples (e.g., $-1.6D$ and $-0.4D$, for $q_{j+1} = -1$) in order to decide the value of the next quotient digit.

Therefore, an ease to compute choice is $M_k = (k + 0.5)D$, which leads to the case that the exact interval of $16W[j]$ (for a particular value of $q_{j+1} = -1$) falls between $M_{-2} = -1.5D$ and $M_{-1} = -.5D$ (see the corresponding bold arrows in Fig. 3)

$$D(k+0.4) \le M_k \le D(k+0.6). \quad (4)$$

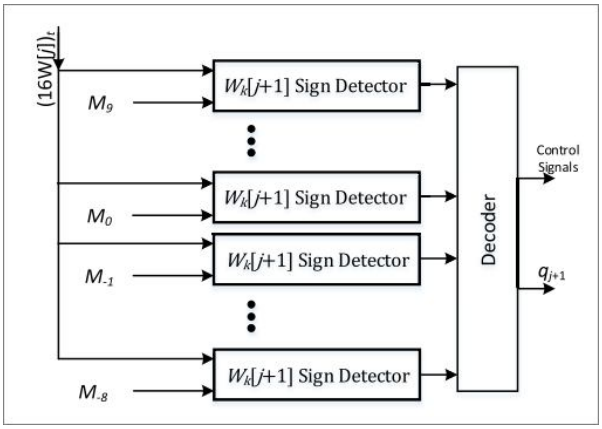


Fig. 4. QDS architecture.

1) On the other hand, the comparison of an SD or CS number (i.e., partial remainder) with a non redundant one (i.e., comparison constants) cannot be trusted to a digit by digit comparator (most significant digits first). Replacing the operands of (2) with the corresponding truncated operands, w we get at (5) and (6), respectively

$$(M_k)_t \le (16W[j])_t, \quad \text{for } q_{j+1} = k \quad (5)$$

$$(16W[j])_t < (M_k)_t, \quad \text{for } q_{j+1} = k - 1. \quad (6)$$

After some elaborations on (5) and (6), as follows, we get at $M_k - 2\times16^{-t} - k\,D < W[j+1]$ and $M_k + 16^{-t} - (k-1)\,D > w[j+1]$, respectively. Applying these results on the convergence condition in (3) (i.e., $-\rho\,D < W[j+1]$

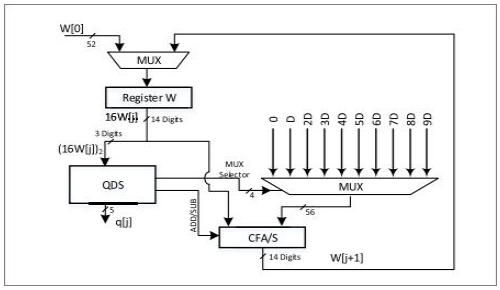


Fig. 5.PRC for designs CS-10 and MRSD-10.

and $W[j+1] < \rho D$) leads to (7) and (8), respectively

$$(5) \Rightarrow M_k - 16^{-t} < (M_k)_t \le (16W[j])_t$$

$$< 16W[j] + 16^{-t} \Rightarrow M_k - 2$$

$$\times 16^{-t} - kD < 16W[j] - kD$$

$$= W[j+1]$$

$$(6) \Rightarrow 16W[j] - 16^{-t} < (16W[j])_t < (M_k)_t \le M_k \Rightarrow$$

$$M_k + 16^{-t} - (k-1)D > 16W[j] - (k-1)D = W[j+1] - \rho D < M_k - 2$$

$$\times 16^{-t} - kD \Rightarrow 2 \times 16^{-t}$$

$$+ kD\rho D < M_k \qquad (7)$$

$$M_k + 16^{-t} - (k-1)D < \rho D \Rightarrow M_k$$

$$< \rho D - 16^{-t} + (k-1)D. \qquad (8)$$

Recalling that $\rho = 0.6$ and $1/16 \le D < 1$, and combining the inequalities (7) and (8), we get at the following:

$$\rho D + 2 \times 16^{-t} + kD < \rho D - 16^{-t} + (k-1)D$$
$$\Rightarrow 3 \times 16^{-t} < (2\rho - 1)D = 0.2 \times D \Rightarrow 16^t > 15/D$$

Since the latter should hold for all the values of D (in particular $D = 16^{-1}$), we need to assert $16^t > 240$, which leads to $t > 1$.

B. Partial Remainder Computation

The aforementioned four proposed architectures are mainly described within the PRC discussion as follows.

CS-10 and MRSD-10 Designs: The straightforward PRC would use a unified carry-free adder/subtractor (CFA/S) and a 10:1 multiplexer, for the quotient digit set [−9, 9].

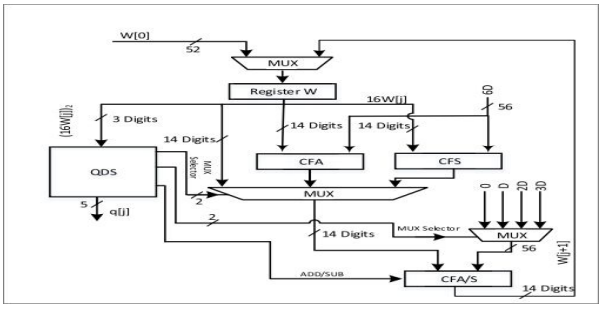


Fig. 6. PRC for designs CS-4 and MRSD-4

2)CS-4 and MRSD-4 Designs: In order to utilize a smaller selector of divisor multiples, we propose the architecture of Fig. 6, where again QDS box represents Fig. 4, whose output control signals are shown as "MUX Selector" and "ADD/SUB." Let $q_{j+1} = 6\alpha + \beta$, where $\alpha \in [-1, 1]$ and $\beta \in [0, 3]$.

C. Initialization and Clean-Up

The initialization cycle is responsible for the following tasks.

1) Converting X to W [0]: This is a cost- and delay-free operation. In case of CS designs, zero-valued bits are inserted as the second bits of CS representation.
2) Parallel Computation of the Divisor Multiples:
   a) CS-10 and MRSD-10 Designs: There are four shifters (for $2D$, $4D$, $6D$, and $8D$) and four parallel adders (for $3D = 2D + D$, $5D = 4D + D$, $7D = 8D - D$, and $9D = 8D + D$).
   b) CS-4 and MRSD-4 Designs: There are only two shift operations (for $2D$ and $6D$), and one addition (for $3D = 2D + D$).

3)Parallel Precomputation of the Truncated Comparison Constants: This is done, as in the following expressions for M_0 to M_8.

a) *CS-10 and MRSD-10 Designs*:

$$M_1 = D/2, \quad M_2 = 3D/2, \quad M_3 = 5D/2$$
$$M_4 = 7D/2, \quad M_5 = 9D/2, \quad M_6 = 5D + M_1$$
$$M_7 = 6D + M_1, \quad M_8 = 7D + M_1$$
$$M_9 = 8D + M_1.$$

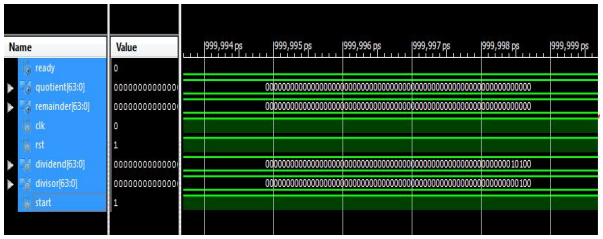
b) *CS-4 and MRSD-4 Designs*:

$$M_1 = D/2, \quad M_2 = 3D/2, \quad M_3 = 2D + M_1$$
$$M_4 = 3D + M_1, \quad M_5 = 4D + M_1$$
$$M_6 = 4D + M_2, \quad M_7 = 6D + M_1$$
$$M_8 = 6D + M_2, \quad M_9 = 8D + M_1.$$

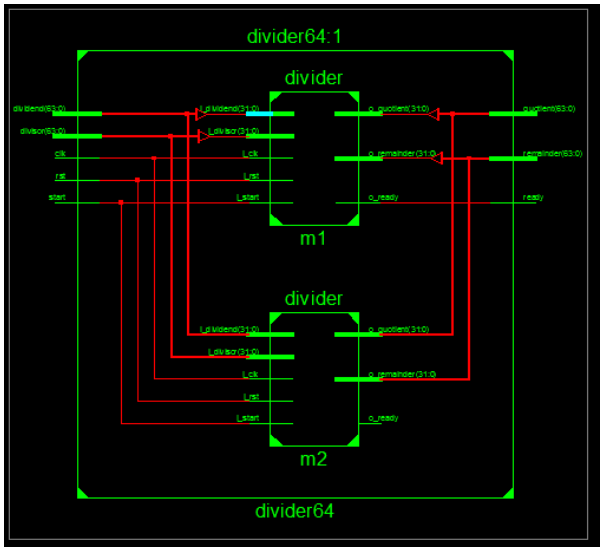
Other comparison constants are obtained as $M\text{-}k = -M_{k+1}$, for $0 \le k \le 8$.

IV. RESULTS

Simulation results:



RTL Schematic:



Design and Summary:

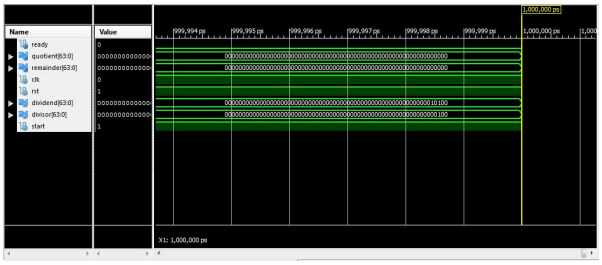
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1793	4656	38%
Number of Slice Flip Flops	666	9312	7%
Number of 4 input LUTs	3427	9312	36%
Number of bonded IOBs	260	232	112%
Number of GCLKs	1	24	4%

Timing report:

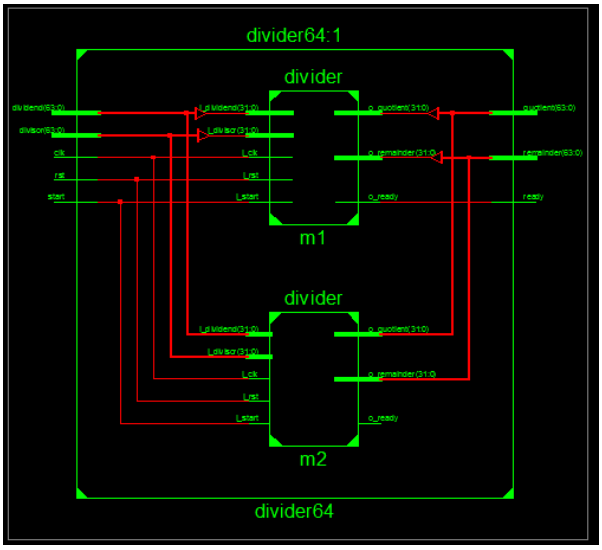
```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 9.991ns (Maximum Frequency: 100.092MHz)
   Minimum input arrival time before clock: 7.606ns
   Maximum output required time after clock: 5.373ns
   Maximum combinational path delay: 6.106ns
```

Extension Results

Simulation:



RTL Schematic:



Design and summary:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	0	4656	0%
Number of bonded IOBs	129	232	55%

Timing report:

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: No path found
```

V. CONCLUSION

We studied (via analytical and synthesis-based evaluation of the corresponding VLSI realizations) the impact of the following two design options on the figures of merit of binary digit-recurrence division hardware.

1) Representation of partial remainders via high radix redundant number systems. Our representation choice is maximally redundant radix-16 SD number system with digit set [−15, 15].

2) Dynamic generation of some divisor multiples in $[-9, 9] \times D$ around the precomputed multiple $6D$.

We also studied the relevant previous designs, which have opted for binary CS representation of partial remainders and representation of radix-16 quotient digits via minimally redundant radix-4 $[-2, 2]$ digits, which leads to partial dynamic generation of divisor multiples.

REFERENCES

[1] M. Ercegovac and T. Lang, Digital Arithmetic. San Mateo, CA, USA: Morgan Kaufmann, 2003.

[2] D. L. Harris, S. F. Oberman, and M. A. Horowitz, "SRT division architectures and implementations," in Proc. 13th IEEE Symp. Comput. Arithmetic, Jul. 1997, pp. 18–25.

[3] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," IRE Trans. Electron. Comput., vol. 10, no. 3, pp. 389–400, Sep. 1961.

[4] H. A. H. Fahmy and M. J. Flynn, "The case for a redundant format in floating point arithmetic," in Proc. 16th IEEE Symp. Comput. Arithmetic, Santiago de Compostela, Spain, Jun. 2003, pp. 95–102.

[5] S. Gorgin and G. Jaberipur, "A family of high radix signed digit adders," in Proc. 20th IEEE Symp. Comput. Arithmetic, Tübingen, Germany, Jul. 2011, pp. 112–120.

[6] W. Liu and A. Nannarelli, "Power efficient division and square root unit," IEEE Trans. Comput., vol. 61, no. 8, pp. 1059–1070, Aug. 2012.