International Journal of Engineering Innovation & Research
Volume 6, Issue 5, (September) ICEMS-2017, ISSN (Online) : 2277 – 5668

International Conference on Advances in Engineering Management & Sciences - ICEMS -2017

A VLSI Implementation of Binary Division with Radix-16 Signed Digit Number Systems

[1]B. Prathap Reddy
[1]Assistant Professor, Department of ECE, SREC, NANDYAL, A.P., India
[1]prathap.bogala@gmail.com

***Abstract* – VLSI Implementation of binary division generally uses redundant representation of partial remainders and quotient digits. This method allows for fast carry-less computation of the next partial remainder, and leads to less number of the required divisor multiples. In studying the existed relevant works, I have noted that the redundant high radix representation of quotient digits is popular in order to reduce the cycle count. In this paper, we propose a design containing two division architectures. These are based on radix-16 signed digit (SD) representations of partial remainders. They use full or partial pre-computation of divisor multiples. The other uses smaller multiplexer and two extra adders, where one of the operand is constant within all cycles. The quotient digits are represented by radix-16 SDs. Our synthesis results of VLSI Implementations of the existed work and the two proposed designs show reduced power and energy figures at the cost of more silicon area and delay measures. Finally, our energy-delay product is less than that of the existed work.**

***Keywords* – Binary, Division, Signed Digit, Power, Energy.**

I. 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 or digit recurrence and multiplicative or functional. The former is less costly, but slower, such that obtaining each digit of the quotient requires a separate recurrence cycle, while the number of iterations in the latter is logarithmically proportional to the number of quotient digits. 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. To reduce the number of recurrences, radix-16 division schemes have been proposed. The current trend in VLSI design favors low-power products that are often more reliable due to the reduction of die temperature. The reduction of power dissipation is considerably influential within the arithmetic/logic units, which is often recognized as the hot spot of digital processors.

II. BACKGROUND

The division operation can be defined as X = QD+ 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-2h division $1/2h \leq D < 1$. Equation (1) describes the jth recurrence, where W[j], qj+1, and Q[j], represent the jth 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

$$W[j+1]=2hW[j]-qj+1D$$

$$Q[j+1]=Q[j]+2-h(j+1)qj+1 \quad \ldots\ldots\ldots\ldots (1)$$

QDS can be simplified via selecting qj+1 from an SD set [−α, α]. For example, radix-16 digit set [−10,10], where the next quotient digit is obtained as qj+1 = 4qh j+1 +q l j+1, with qh j+1,ql j+1 ∈[− 2,2]. The common QDS approach is via comparing the shifted partial remainder with a set of comparison constants (i.e., a set of selected divisor multiples), such that qj+1 is selected based on (2), where Mk and Mk+1 represent two consecutive comparison constants, and k ∈ [−α,α]. In practice, all the required comparisons take place in parallel. On the other hand, in order to speed up the QDS, truncated to some fractional digits, partial remainders and comparison constants are used, where t is so determined to guarantee that the convergence condition is not violated.

$$Mk \leq 2hW[j] < Mk+1 \Rightarrow qj+1 = k \quad (2)$$

$$\rho D \leq W[j+1] \leq \rho D, \rho = (\alpha/(2h-1)) \quad (3)$$

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

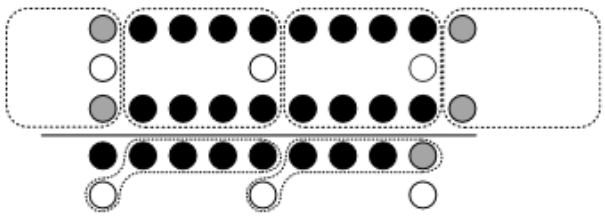

Fig. 1. Two digit slices of CFA required for PRC

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 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. Therefore, an addition of a radix-16 MRSD number (i.e., the shifted partial remainder) with a binary number (i.e., the divisor multiple) is in order. In the previous works that use radix-16 MRSD number system, each radix-16 MRSD digit is represented by a 5-bit two's complement encoding. The most significant bit is a negative bit with arithmetic value −1(0) corresponding to logical status 0(1) . Fig. 1 shows two digit slices of the required carry-free addition, for the latter so-called negative bit two's complement representation, where white (black) dots represent negative bits (positive bits). 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 negative bit and three most significant positive bits 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. The use of the aforementioned adder is expected to lead to less power dissipation with respect to designs that use CS adder, which is mainly due to the use of 5 bits per digit in the former compared with 8 bits in the latter.

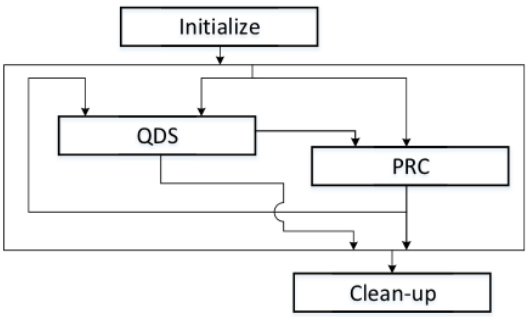

Fig. 2. General Architecture for Binary Division

III. PROPOSED ARCHITECTURE

The general digit recurrence division architecture is shown in Fig. 2, where the pertinent discussion regarding the proposed designs will be provided as appropriate. Double precision operands (i.e., binary64 floating point) are assumed for all the proposed architectures as is the case for the main reference work. 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. The 5-bit representation of such digits is the same as the MRSD of Fig. 1. The primary choice would be the minimally redundant [−8,8] digit set that requires the minimum number of divisor multiples, while the other extreme choice would be the maximally redundant [−15,15]digit set. However, another influential measure is the number of fractional digits that are sufficient for truncated comparison of partial remainders with divisor multiples. This is later shown to be 2 in the case of digit sets $[-\alpha,\alpha](\alpha \in [9,15])$, and 3 for $\alpha = 8$.

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 qj+1D. However, besides implementing the latter conventional method, we propose the following method that uses a four-way multiplexer. In the initialization cycle, we generate only {2,3,6}D, and dynamically obtain ±{4,5,7,8,9}D, as ±{6D−2D,6D∓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. Therefore, our other choice is to use higher radix redundant number systems for partial remainder representation. For example, radix-16 maximally redundant number system (MRSD) [5] is a viable choice, with only 25% extra representation storage.

A. Quotient Digit Selection

The quotient digit set of our choice is [−9,9]. In general, the next quotient digit qj+1 should be selected from [−α,α], such that the convergence condition that is described by (3) (above) holds [1], where in this case (i.e., $\alpha = 9$), $\rho = (a/(16-1)) = (9/15) = 0.6$.

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, forqj+1 =− 1) in order to decide the value of the next quotient digit. However, for some W[j] values (e.g., 16W[j]=− .5 D), there may be more than one valid qj+1. For example, see the overlapped zone between the dashed lines for qj+1 =−1 and qj+1 =0.

To ease the QDS process and reduce the number of comparisons, it is common to pre compute a set of comparison constants, as fixed multiples of divisor [as in (2)], to be used for exact QDS. As such, (4) provides the proper range of Mk, for $k \in [-9,9]$. 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 qj+1 =− 1) falls between M−2 =− 1.5D and M−1 =− .5D

International Journal of Engineering Innovation & Research
Volume 6, Issue 5, (September) ICEMS-2017, ISSN (Online) : 2277 – 5668

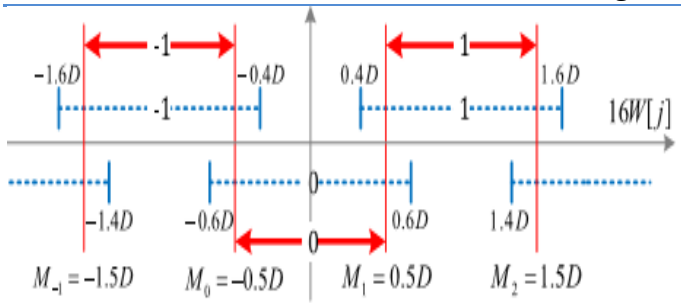

Fig. 3. Overlapped zones for Quotient Digit values

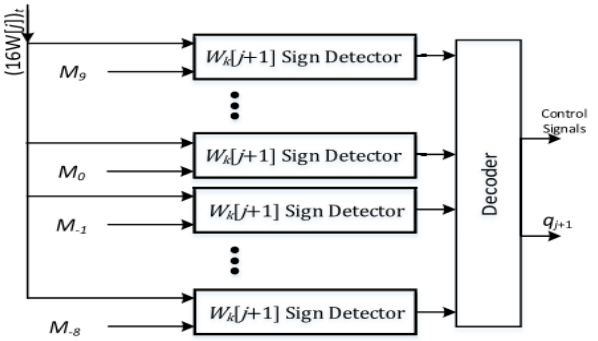

Fig. 4. QDS Architecture

$D(k+0.4) \leq Mk \leq D(k+0.6)$ (4)

the comparison of an SD or CS number (i.e., partial remainder) with a nonredundant one (i.e., comparison constants) cannot be trusted to a digit by digit comparator (most significant digits first). For example, consider the first two digits of a hexadecimal MRSD operand as X = x1x0, and non redundant number Y = y1y0, and assume x1 =8, x0 =−2, y1 =7, and y0 =15. A normal comparator rules X > Y, sincex1 > y1. However, it turns out that X can be equivalently represented as x1 = 7 andx0 = 14. Therefore, the same trivial comparison scheme leads to X < Y, since x0 = 14 < 15 = y0. A straightforward way out of this problem is to convert the SD or CS operand into their two's complement equivalents and use a conventional comparator, which is of course counterproductive with respect to losing the carry-free advantage of SD or CS number systems. However, it is possible to undertake the required comparisons based on the few most significant fractional digits, such that the convergence condition (3) holds. It turns out, as is proved in Section III-A1 that t =2 fractional digits are sufficient, where for the sake of similar treatment of SD and CS representation, we consider four CS digits as one radix-16 fractional digit. The overall QDS architecture is shown in Fig. 4, where $W_k[j+1]=(16W[j])t - (Mk)t$, and the index t denotes the fractional digits. 1) Proof of t =2: Assuming t fractional digits, for Mk and W, let the truncated operands be denoted as $Mk - 16^{-t} < (Mk)t \leq Mk$ and $16W[j]-16^{-t} < (16W[j])t < 16W[j]+16^{-t}$, respectively. Note that the discarded digits in the latter case can assume both positive and negative values due to SD nature of the partial remainders. We follow a similar procedure as in the decimal division schemes and adopt it for radix-16 division. Replacing the operands of (2) with the corresponding truncated operands, we get at (5) and (6), respectively.

$(Mk)t \leq (16W[j])t$, for $qj+1 = k$ (5)

$(16W[j])t < (Mk)t$, for $qj+1 = k-1$ (6)

After some elaborations on (5) and (6), as follows, we get at $Mk - 2\times16^{-t} - kD < W[j+1]$ and $Mk + 16^{-t} - (k-1)D > w[j+1]$, respectively. Applying these results on the convergence condition in (3) (i.e., $-\rho D < W[j+1]$ and $W[j+1] < \rho D$) leads to (7) and (8), respectively.

(5) $\Rightarrow Mk - 16^{-t} < (Mk)t \leq (16W[j])t < 16W[j]+16^{-t}$

$\Rightarrow Mk - 2\times16^{-t} - kD < 16W[j] - kD = W[j+1]$

(6) $\Rightarrow 16W[j] - 16^{-t} < (16W[j])t < (Mk)t \leq Mk$

$\Rightarrow Mk + 16^{-t} - (k-1)D > 16W[j] - (k-1)D = W[j+1] - \rho D < Mk - 2\times16^{-t} - kD$

$\Rightarrow 2\times16^{-t} + kD\rho D < Mk$ (7)

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

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

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

$\rho D + 2\times16^{-t} + kD < \rho D - 16^{-t} + (k-1)D$

$\Rightarrow 3\times16^{-t} < (2\rho - 1)D = 0.2\times D \Rightarrow 16t > 15/D$.

B. Partial Remainder Computation

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

1) 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]. The multiplexer selects one precomputed divisor multiple from [0,9]×D, which are obtained in the initialization phase. Fig. 5 shows the required architecture, for designs CS-10 and MRSD-10, where a digit is meant to denote either an MRSD or four CS digits, and QDS box represents Fig. 4, whose output control signals are shown as "MUX Selector. and "ADD/SUB." The ten-way multiplexing is expected to be highly influential in prolonging the latency. Recalling the PRC, as $W[j+1]=16W[j]-qj+1D$, note that W[j] is maintained either in radix-2 CS or radix-16 MRSD, and the multiples of D are represented in binary.

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 $qj+1 = 6\alpha + \beta$, where$\alpha \in$

[− 1,1] andβ ∈[0,3]. The overlapped PRC computes W[j + 1] =16 W[j]+6αD via a CFA and a CFS, since W[j]is represented in binary CS or radix-16 MRSD formats. The operation of this part overlaps with the QDS. The rest of PRC is carried out after QDS, where βD is selected via the four-input multiplexer, followed by computing W[j + 1]=W[j + 1] + βD, via another CFA.

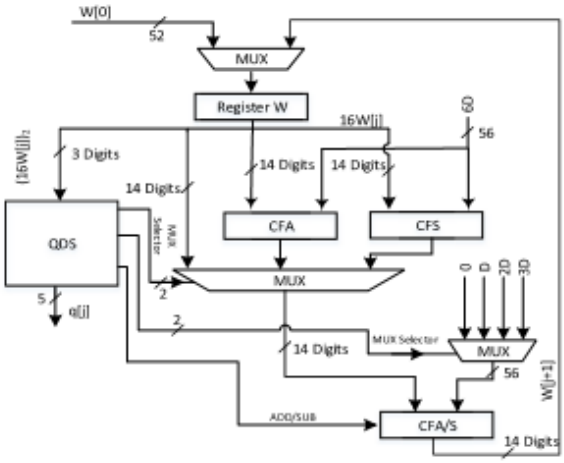

Fig. 5. PRC for designs CS-4 and MRSD-4

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. However, in MRSD cases, a zero-valued negabit (with logical status 1) is added per each four bits of X to lead to the radix-16 MRSD representation of W[0].

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,5 D = 4D + D,7 D =8D−D, and 9D =8D+D).

3) Parallel Precomputation of the Truncated Comparison Constants: This is done, as in the following expressions for M0 to M8.

CS-4 and MRSD-4 Designs:

M1 = D/2, M2 =3D/2, M3 =2D+M1
M4 = 3D+M1, M5 =4D+M1
M6 = 4D+M2, M7 =6D+M1
M8 = 6D+M2, M9 =8D+M1.

The terminal cycle is responsible for the conversion of the obtained quotient and remainder to nonredundant binary format. Such conversionentails word wide binary addition that is undertaken via a parallel prefix binary adder in both CS and MRSD cases, which also takes care of the required rounding addition.

Table 1: Power Dissipation(Mw) At Maximum Frequency

Component↓	CS-10	CS-4	MRSD-10	MRSD-4
Initialization	2.11	1.86	2.22	1.90
QDS	3.05	3.17	2.33	2.84
PRC	3.27	4.48	2.62	3.60
Clean-up	2.15	2.47	2.33	2.21
Total	10.59	11.97	9.49	10.55
Ratio	1.12	1.26	1.00	1.11

IV. Evaluation and Comparison

We have described all components of the proposed architectures with hardware description language (HDL) code, tested them for correctness with 30000 random input data, and synthesized the corresponding circuits with Synopsys Design Compiler. In addition, Synopsys Power Compiler has been exploited to extract the power consumption of the designs under the tested random input data. The STM 90-nm technology was used in order to compare the figures of merit with those of the main reference work. On the other hand, comparison of synthesis results requires the use of the same technology and the corresponding libraries. Tables I contain the power (at maximum frequency), area, and delay figures for the components of the proposed architectures, respectively, where XFF (Setup) refers to the clock-to-Q (register) time.

Table 2: Area Usage (mm2) For The Four Proposed Dividers

Component↓	CS-10	CS-4	MRSD-10	MRSD-4
Initialization	31322	17528	30453	18578
QDS	15308	15838	13070	13244
PRC	10333	12452	9529	12711
Clean-up	5616	5989	5627	6250
Total	62578	51808	58680	50783
Ratio	1.23	1.02	1.16	1.00

Table 3: Delay(ps) of components in critical path

Component↓	CS-10	CS-4	MRSD-10	MRSD-4
X_{FF}	113	109	103	104
QDS	564	617	573	577
MUX	300	144	270	135
ADD/SUB	319	336	370	400
X_{setup}	64	64	64	64
Delay	1360	1270	1380	1280
Ratio	1.07	1.00	1.09	1.01

1) The MRSD designs, which use smaller registers, show less power figures than the CS ones (10%–12%). In addition, the QDS of the MRSD-10 (-4), dissipate 23% (10%) less power, where the similar reduction measure for the PRC is around 20%.

2) The CS designs are slightly faster than their MRSD counterparts, while the latter is less area consuming.

3) The CS-4 and MRSD-4 designs exhibit 7% and 8% lower delay figures, as compared with CS-10 and MRSD-10 designs, while those of the latter two dissipate 12% and 10% less power, respectively.

4) The CS-4 and MRSD-4 designs are less area consuming (16%), which is mainly due 41% less area consumption of the initialization phase. On the other hand, area figures of the active phases (i.e., QDS and PRC) of CS-10 and MRSD-10 designs consume less area, which explains their less power dissipation, despite that their overall area figures are more.

The total figures of merits on delay, power, energy, energy delay-product (EDP), total area, area without that of the initialize stage, E × area, and EDP × area for the proposed designs and that of the reference work are compiled in Table V, where the area of a NAND2 is equal to 4.4μm2, and energy is obtained as the power × 16 × delay in maximum frequency. The power measure at maximum frequency for [6] is estimated to be28 mW, based on the reported average power at 100MHZ (i.e., 2.73 mW per Table VI of [6]). This estimation result seems to be viable, since the reported power measure in 1.2-ns delay is 27.47 mW.

The improvement in power and EDP measures also follow the following.

1. The power dissipation in comparison to the average of the power measures of the four proposed designs is about 62% more.
2. The area consumption is, on the average, 35% less than that of the proposed designs. However, that of the former is more when excluding the area measure of the initialization stages.
3. The least achieved delay is, on the average, 26% less than that the proposed designs
4. The energy figures of the proposed designs are, on the average, 48.5% less
5. The EDP of the proposed designs is, on the average, about 30% less
6. The EDP of the MRSD-4 design is, on the average, 15.5% less than that of our other three designs. The energy-per division of the MRSD designs are, on the average, 10% less than those of the CS designs.
7. The E × Area of all the proposed designs and the EDP × Area of MRSD-4 are less.

V. Conclusion

We studied 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] ×D around the pre-computed 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. For better evaluation of the above options, we explored a design space containing four architectures based on pre-computation of all or part of divisor multiples and CS or MRSD representation of partial remainders. The HDL simulations and synthesis showed low-power and low-energy advantages of all the four designs as compared with the best previous work. However, while our designs do not operate as fast as the reference one, the EDP of the proposed designs is 26%–35% less than that of the reference work.

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. 6, no. 8, Aug. 2012.

[7] J. Ebergen and N. Jamadagni, “Radix-2 division algorithms with an overredundant digit set,” IEEETrans. Comput., vol. 64, no. 9, pp. 2652–2663, Sep. 2015.

[8] A. Nannarelli and T. Lang, “Low-power divider,” IEEE Trans. Comput., vol. 48, no. 1, pp. 2–14, Jan. 1999.

[9] A. Nannarelli, “Performance/power space exploration for binary64 division units,” IEEE Trans. Comput., vol. 65, no. 5,

[10] E. Antelo, T. Lang, P. Montuschi, and A. Nannarelli, “Digit-recurrence dividers with reduced logical depth,” IEEE Trans. Comput., vol. 54, no. 7, pp. 837–851, Jul. 2005.