ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

VLSI Oriented Vedic Division with Redundant Number Systems

Goli Manoj Kumar
Department of Electronics & Communication Engineering, HITAM, Hyderabad, Telangana-502401, India.

Vinod Kumar Ahuja
Department of Electronics & Communication Engineering, HITAM, Hyderabad, Telangana-502401, India.

K. Anil Kumar
Department of Electronics & Communication Engineering, HITAM, Hyderabad, Telangana-502401, India.

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 pre-computation 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 [−9, 9] 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.

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 (aka digit recurrence) and multiplicative (aka 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 [1]. Quotient digit selection (QDS) is very simple in conventional binary division algorithms, such as no restoring division scheme, 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.

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. Another cost reducing technique is to perform the comparison only via the few most significant digits, whose number is determined via a fairly complex error analysis [2]. 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

Cite this article as: Goli Manoj Kumar, Vinod Kumar Ahuja & K. Anil Kumar, "VLSI Oriented Vedic Division with Redundant Number Systems", International Journal & Magazine of Engineering, Technology, Management and Research, Volume 6 Issue 1, 2018, Page 60-69.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

recurrence division contribute to extra energy consumption. On the other hand, given the popularity of portable devices and prominence of green computing, the current trend in VLSI design favors low-power products that are often more reliable due to the reduction of die temperature. In addition, the reduction of power dissipation is considerably influential within the arithmetic/logic units, which is often recognized as the hot spot of digital processors.

The commonly used redundant number system that is required for the representation of partial remainders is the binary carry save (CS), which roughly doubles the required number of bits for representing the same value. Although the extra register cost might be affordable, the corresponding extra power dissipation could be in question [3]. On the other hand, there are less costly redundant number systems that represent the same range of numbers via exploiting less number of additional bits. For example, the case of redundant digit floating point arithmetic uses the radix-16 maximally redundant SD (MRSD) representation that requires only 25% extra bits.

Efficient radix-16 MRSD addition and subtraction have been offered. The choice of SD set for the representation of quotient digits is critical, since narrower digit sets (e.g., the minimally redundant digit set $[-2h-1, 2h-1]$) that are desirable for reducing the cost of divisor multiple generation (DMG) tend to require more number of digits for comparing them with partial remainders. Given the aforementioned design space parameters (i.e., the choice of SD or CS for partial remainders and quotient digits), we are motivated to study and explore the low power possibilities in the design and implementation of new binary digit recurrence dividers.

The Ancient Indian Vedic Mathematics comprises of sixteen Sutras and thirteen corollaries. The four elementary operations of a processor, the addition, subtraction, multiplication and the division have been extensively dealt with in the sixteen sutras of Vedic Mathematics. The work in this paper involves the Nikhilam and the Paravartya Sutra which deal with the division [4]. The Nikhilam Sutra also deals with multiplication and some amount of work has been done using another sutra known as the UrdhvaTiryakbhyam Sutra. The novelty of the Vedic division lies in the fact that the procedure incorporates addition and negation operations, both of which are much faster than the traditional successive subtraction methods. The Nikhilam Sutra can be stated as follows:

1.1. The Nikhilam Sutra :

1. This Sutra breaks up the dividend into two parts, one part resembling the Quotient and the other part resembling the Remainder. The number of digits in the Remainder part equals the number of digits in the divisor. For example, if the dividend and divisor are 2002002 and 89998 respectively, then 2002002 is broken up into two parts, 20 (part 1) and 02002 (part 2).

2. The next step in the Sutra adjusts the divisor by complimenting it using the procedure “subtract all from 9 and the last from 10” in which all the digits in the divisor are subtracted from 9 barring the last significant digit which is subtracted from 10 [5].Therefore, the divisor 89998 after adjustment becomes 10002.

3. Next, the first digit of the quotient part (part 1 of the dividend) is divided by the first digit of the actual divisor. This is the only step where division is unavoidable, but in this case also the maximum division is that of a single digit number by a single digit number. In our work a look-up table has been created which stores all the single digit division results in the form of quotient and remainder, and accessed on demand. The first digit of the quotient part ‘2’is divided by ‘8’ (the first digit of the actual divisor) and the remainder ‘2’ is noted down.

In the third step of the sutra, a single digit division has been done that can be performed using a look-up table. The division process has been performed with multiplication process in subsequent steps and addition. Multiplication is a relatively faster and cheaper operation than division [6]. Also the largest multiplication that may be required is multiplying 9 by 9.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

1.2 The Paravartya Sutra :

The Paravartya Sutra is suitable for divisions including large as well as small divisors. The sutra is actually known as "Paravartya Yojayet" which means "Transpose and Apply". The Paravartya Sutra can be easily explained using the famous Remainder Theorem as follows:

1. If E = Dividend, D = Divisor, Q = Quotient and R = Remainder and if the divisor is taken to be (x-p), then a relationship can be stated as follows: E = D.Q + R, or E = Q.(x-p) + R.
2. Now, if 'x' is substituted by 'p' then the identity becomes E = R, thus the expression E automatically becomes the remainder as 'p' is achieved by equating x-p to zero. Hence, actually the sign of 'p' is reversed.

II. BACKGROUND AND THE RELATED WORKS

The balance w[j] is stored in carry-save representation (wS and wC) [7]. The signed- digit representation of the caliber is adapted to accepted two's accompaniment representation and angled by the on-the-fly convert-and-round unit. The accomplishing of the accepted divider, optimized for beeline latency, is apparent in Figure 2. The ceremony is implemented with the alternative action (SEL), two assorted generators (MULT), two carry-save adders (CSA)2 and two registers (REG) to abundance the carry-save representation of the residual. Because of the carry-save representation of the residual, the alternative action (SEL R-4) in Figure 1 is composed by a 7-bit carry-propagate adder and a action implemented with argumentation gates.

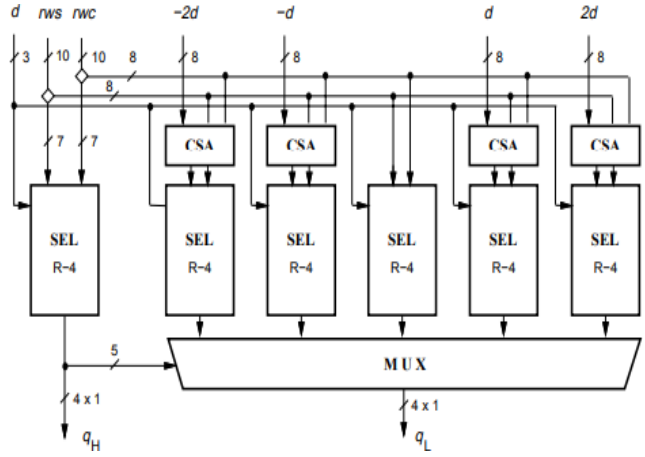

Figure 1: Selection function for radix-16

The advocate performs the about-face and the rounding according to the assurance of th -16 nal balance and the arresting that detects if it is zero, which are produced by a sign-zero-detector (SZD). Table 1 shows the adjournment through the two locations of SEL. Note that the beyond adjournment of SELqL is compensated by the added CSA that exists in the aisle from SELqH [8] . The analytical aisle post-layout is 9.2 ns and 16 iterations are appropriate to complete the operation, agnate to a cessation of 15ns.

	path		delay [ns]
q_L	SEL_{q_L} - MULT - HA - REG 5.7 + 1.4 + 0.6 + 1.5	=	9.2
q_H	SEL_{q_H} - MULT - HA - FA - REG 4.0 + 1.4 + 0.6 + 1.1 + 1.5	=	8.6

Table 1. Critical path through qL and qH

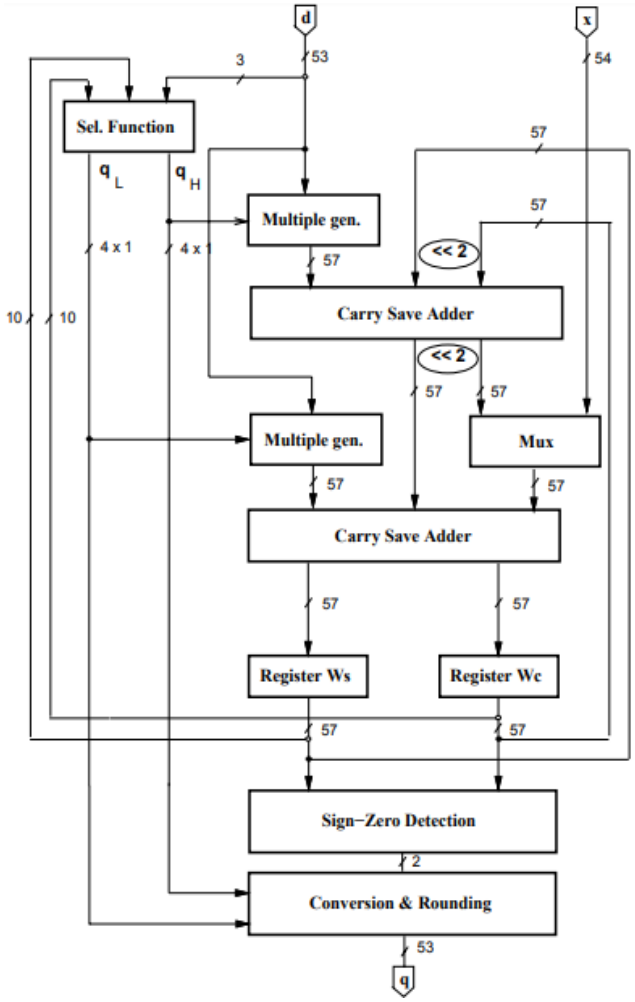

Figure 2. Implementation radix-16

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

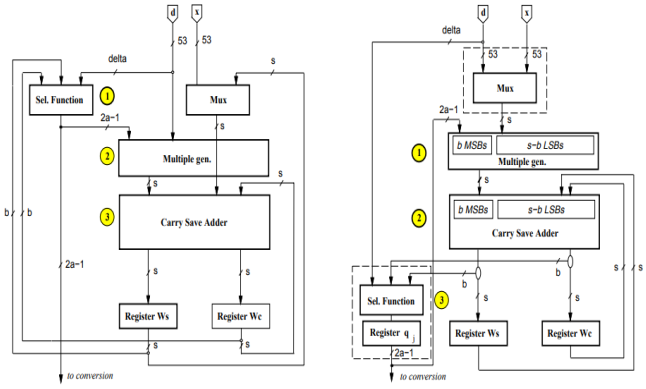

Figure 3. Retiming the reccurence of Low power Implementing

Low-power architecture techniques are activated to the accepted accomplishing of the affiliate to abate the activity burning after chastening the latency. The techniques, advised below, were originally developed for radix-4 [7] and are actuality acclimatized to the radix-16 case.

Retiming of the recurrence:

The retiming of the ceremony is done by affective the alternative action from the aboriginal allotment of the aeon to the endure allotment of the antecedent cycle. This is apparent in Figure 3 for a radix-r divider. A new annals is bare to abundance the caliber digit. This retiming has the advantage of attachedtheanalytical aisle to the b most-significant $.25 of the ceremony (b = 10 for radix- 16), so that the blow can be redesigned for low power. For instance, for the aisle through SELqL, the two paths are as adumbrated in the antecedent section, and apparent in Figure 4a, the beyond adjournment of SELqL is compensated by the added carry-save adder in the aisle of SELqH. This advantage is alone by the retiming, as apparent in Figure 4b. Consequently, in adjustment not to access the aeon delay, the alarm of annals qLis skewed, consistent in the paths of Figure 4c. The alarm can be skewed by abacus the adapted adjournment (e.g. some buffers) in the alarm administrationtree.

Since in the retimed accomplishing the alternative action is placed afterwards the additional CSA, instead of anon afterwards the registers, there is a ample access in the amount of glitches, which are amenable for the added amusement of the alternative function. One way to clarify those glitches is to absorber the alternative action with multiplexers acting as latches, as declared in Figure 5.

The baddest arresting is apprenticed by a altered alarm (same period, altered phase) that enables the muxes to address the amount from the CSA if it is stable, and authority the accepted amount otherwise. However, in this case the adjournment of the mux affects the analytical path. For radix-16 the activity blown in the alternative action is halved, but the analytical aisle is added by about 5%. For this reason, the amount is not included in Table 2, which summarizes the activity reductions.

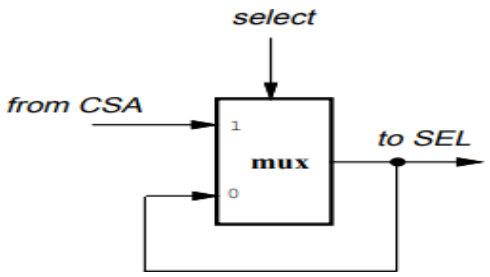

Figure 4. Multiplexers to filter glitches in selection function

III. PROPOSED MODEL

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) is assumed for all the proposed architectures as is the case for the main reference work.

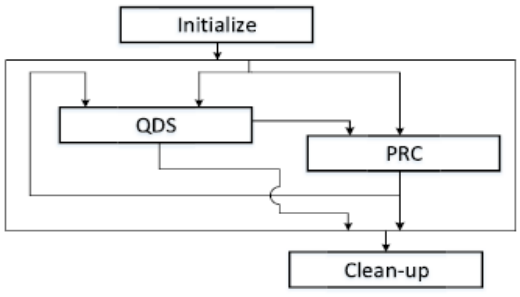

Figure 5.General division architecture

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Within the framework of Fig. 2, static and semidynamic 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) Semidynamic 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 $q_{j+1}D$. 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 $\bullet\}\{4, 5, 7, 8, 9\}D$, as $\bullet\}\{6D - 2D, 6D \mp D, 6D + 2D, 6D + 3D\}$, 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 semidynamic 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) is a viable choice, with only 25% extra representation storage.

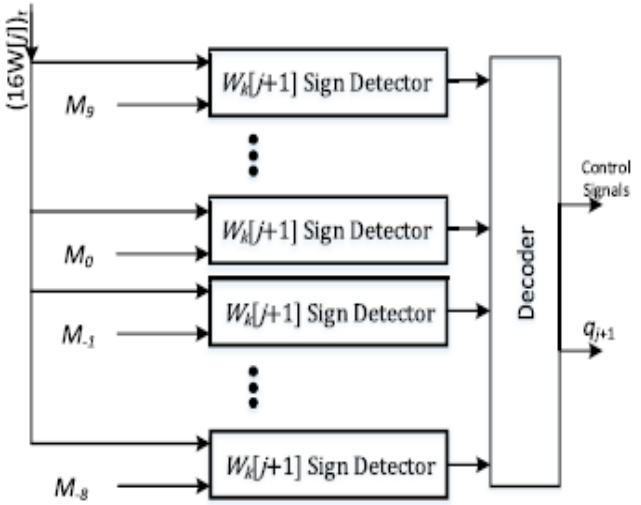

Figure 6. QDS architecture

Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya- Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda. It covers explanation of several modern mathematical terms including arithmetic, geometry (plane, co-ordinate), trigonometry, quadratic equations, factorization and even calculus. His Holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884-1960) comprised all this work together and gave its mathematical explanation while discussing it for various applications. Swahiji constructed 16 sutras (formulae) and 16 Upa sutras (sub formulae) after extensive research in Atharva Veda. Obviously these formulae are not to be found in present text of Atharva Veda because these formulae were constructed by Swamiji himself. Vedic mathematics is not only a mathematical wonder but also it is logical. That‟s why VM has such a degree of eminence which cannot be disapproved. Due these phenomenal characteristic, VM has already crossed the boundaries of India and has become a leading topic of research abroad. VM deals with several basic as well as complex mathematical operations. Especially, methods of basic arithmetic are extremely simple and powerful.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The word „Vedic" is derived from the word „veda" which means the store-house of all knowledge. Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc. These Sutras along with their brief meanings are enlisted below alphabetically.

1) (Anurupye) Shunyamanyat – If one is in ratio, the other is zero.
2) Chalana-Kalanabyham – Differences and Similarities.
3) EkadhikinaPurvena – By one more than the previous One.
4) EkanyunenaPurvena – By one less than the previous one.
5) Gunakasamuchyah – The factors of the sum is equal to the sum of the factors.
6) Gunitasamuchyah – The product of the sum is equal to the sum of the product.
7) Nikhilam NavatashcaramamDashatah – All from 9 and last from 10.
8) Paraavartya Yojayet – Transpose and adjust.
9) Puranapuranabyham – By the completion or noncompletion.
10) Sankalana- vyavakalanabhyam – By addition and by subtraction.
11) ShesanyankenaCharamena – The remainders by the last digit.
12) ShunyamSaamyasamuccaye – When the sum is the same that sum is zero.
13) Sopaantyadvayamantyam – The ultimate and twice the penultimate.
14) Urdhva-tiryakbhyam – Vertically and crosswise.
15) Vyashtisamanstih – Part and Whole.
16) Yaavadunam – Whatever the extent of its deficiency.

The beauty of Vedic mathematics lies in the fact that it reduces the otherwise cumbersome-looking calculations in conventional mathematics to a very simple one. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing and digital signal processing.

The multiplier architecture can be generally classified into three categories. First is the serial multiplier which emphasizes on hardware and minimum amount of chip area. Second is parallel multiplier (array and tree) which carries out high speed mathematical operations. But the drawback is the relatively larger chip area consumption. Third is serial- parallel multiplier which serves as a good trade-off between the times consuming serial multiplier and the area consuming parallel multipliers..

The proposed division algorithm in this paper is a combination of the earlier discussed two sutras, the Nikhilam Sutra and the Paravartya Sutra, with slight modification so as to obtain a generalized algorithm for all the possible divisors. Presently, numerous division algorithms are used depending upon system or application requirements such as the Restore Type Division Algorithm, SRT Division Algorithm, and the Non Restore Type Division Algorithm [10-12], the latter being the fastest and economic. It has been statistically proven further in our work that the proposed algorithm performs better with respect to the Non Restore Type Division Algorithm in terms of speed and memory requirement.

The proposed algorithm performs the calculations on the number of digits in the divisor and the dividend rather than on the number of bits representing them. In Non Restore Type Division Algorithm, the time estimate of the division is proportional to the number of bits. But in the Vedic Division Algorithm, the time requirement is based mainly on the number of normalizations (illustrated further) of the intermediate remainders. Hence, the algorithm exhibits remarkable results on divisions involving big numbers. The novelty of the algorithm lies in the fact that since the computation is done on digits rather than on the bits, very large numbers, having size up to 38 digits (127 bits), can be divided in the present form and if modified, it can divide even larger numbers.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Step-By-Step Algorithm description :

1. In the first step the divisor is adjusted using a combined logic of both the Nikhilam Sutra and the Paravartya Sutra. All the digits that are less than or equal to 5 are negated. For all those digits which have values more than 5, 10's compliment of the digit is taken and 1 is added to the next higher digit. If the divisor is 47483647, then a close observation reveals that all the consecutive digits are alternately less than and greater than 5. The divisor adjustment starts from the Least Significant Digit. As 7 > 5, hence 10's compliment of 7 is taken and 1 is added to the next higher digit '4',replacing 7 by 3 and 4 by 5. Now this 5 is adjusted by -5. Hence the adjustment of the divisor is shown below.

4 7 4 8 3 6 4 7	becomes
-5 3 -5 2 -4 4 -5 3	(Adjusted divisor)

2. Let us take an example in which the dividend is 99999 and the divisor is 456. Therefore, the divisor 456 after adjustment becomes -5 4 4. It may also be observed that the first digit of an adjusted divisor will always be a negative digit.

3. Next, the first digit of the dividend is divided by the magnitude of the first digit of the adjusted divisor to obtain the quotient. In the example, the first digit of the dividend (99999) is 9 which is divided by magnitude of the first digit of the adjusted divisor (-5 4 4) which is 5 to obtain the quotient 1. This is the only step where division is unavoidable but it has been accomplished with the aid of two look – up tables as shown in Table 1 and Table 2. The use of these tables saves the division time otherwise required for obtaining the quotient and the remainder.

Q : Quotient Table										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	2	3	4	5	6	7	8	9
2	0	0	1	1	2	2	3	3	4	4
3	0	0	0	1	1	1	2	2	2	3
4	0	0	0	0	1	1	1	1	2	2
5	0	0	0	0	0	1	1	1	1	1

Table 2. Look – up Table for Quotient

R : Remainder Table										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	0	0	0	0	0	0	0	0	0
2	0	1	0	1	0	1	0	1	0	1
3	0	1	2	0	1	2	0	1	2	0
4	0	1	2	3	0	1	2	3	0	1
5	0	1	2	3	4	0	1	2	3	4

Table 3. Look – up Table for Reminder

The second row and the first column in Table 1 and Table 2 are shown in bold and represent the array indices. The row indices have been started from 1. Column 1 represents the denominator values and Row 2 represents the numerator values. Suppose we want to divide 'a' by 'b'. The quotient is obtained from the cell Q[b,a] of Table 1 where the maximum value of b can be 5 and not equal to 0.

The remainder is obtained from Table 2 in the same manner, that is, the value of cell R[b,a]. The new quotient is then multiplied by all the digits of the adjusted divisor and placed below the dividend at the exact positions and then added to get the new remainder

1. This step normalizes the remainder. Normalization means replacing a multiple digit value by a single digit value at each position. The procedure is started from the Least Significant Number of the remainder and followed to the Most Significant Number. The Least Significant Digit of the multiple digit number is kept at the position and the rest is added to the next Higher Significant Digit.

2. The next step checks for a '0' at the Most Significant Digit of the normalized remainder. If it is not '0', as in this case, then the normalized remainder is again divided by the adjusted divisor and the new quotient is added to the previous quotient. Else if it is '0', then the previous procedures are repeated till completion.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The Algorithm:

Initialization Part:

1. The dividend and the divisor are held in two arrays – Array 'a' and Array 'b' having index 'm' and 'n' respectively. Array 'b' finally stores the remainder. A temporary array 'temp' has been used to hold the quotient having index't'. Another array 'c' is used to hold the copy of the divisor. All the data has been stored in Little Endean Formats and initialized to 0.The length of Array 'b' is 'n' and Array 'a' is 'm+1', Array 'c' is 'm' and 'temp' is 'n' respectively.
2. The divisor is adjusted so that no digit in the divisor is more than 5.
3. 'm' is updated if adjusting the divisor causes increase in length of the divisor.
4. Inverse each digit of the divisor (including the most significant digit).

Division Part:

5. A new variable 'j' is taken for holding the value of the possible number of iterations and its limits are set from 1 to 'n-m+1'.
6. Compare the most significant digits of divisor and dividend.
7. If MSD (Most Significant Digit) of divisor >MSD of dividend, club the most significant pair of digits of dividend. Update (decrement) n and t.
8. If clubbing results in decrease of the number of digits of dividend below that of divisor, break from the 'For Loop using 'j'' and go to step 13.
9. Quo= MSD of dividend/ MSD of divisor. Calculate the remainder (dividend).
10. If MSD of dividend not divisible by that of divisor, then break from the "for loop". A fresh iteration is being started. Go to step 13.
11. Decrement n and t, go to step 8.
12. The remainder is then normalized. Increment 'n' and't' if normalization increments the number of digits in remainder.
13. If (nn-m+1) in the previous iteration, no further iterations are possible and 2nd condition means we have already tried to divide once more but failed, go to step 16.
14. Else division is possible, go to step 7.
15. Check the remainder. If the remainder is negative, the quotient is decremented once. Else if it is positive, it is the 'normalized version of divisor' and cannot be divided again. So check if original form of divisor (stored in array 'c') can divide again. This division can result into incrementing the quotient by 1.
16. The quotient is stored in array 'temp' with the leading and trailing 0'sand remainder in array 'b'. Quotient can further be normalized.

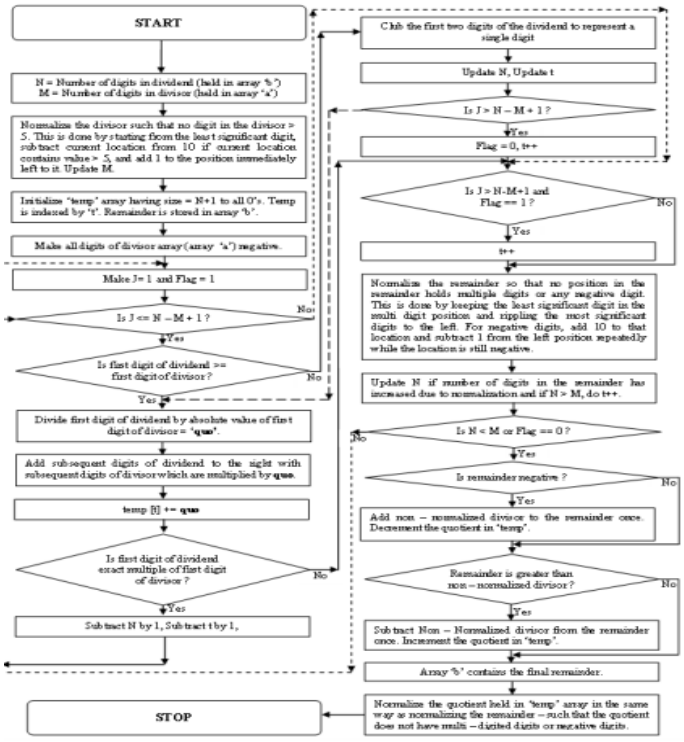

Figure 7. Flowchart of the Vedic Division Algorithm

IV. SIMULATION RESULTS

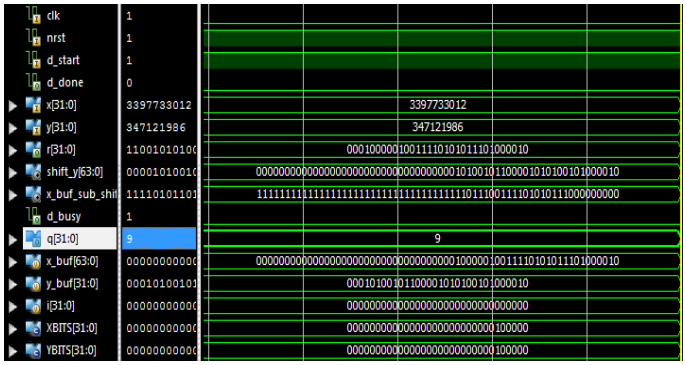

Figure 8. Simulation Results for binary division(32-bit)

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

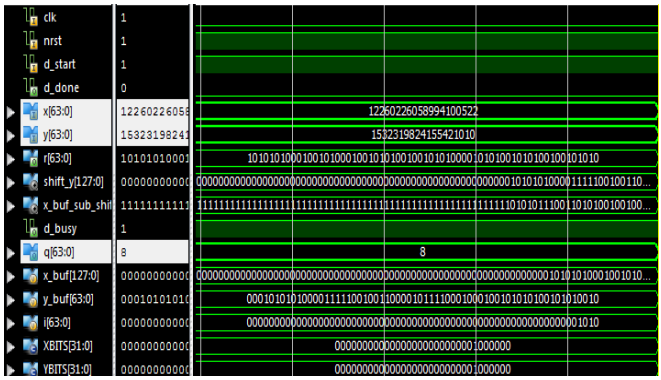

Figure 9: Simulation Results for binary division(64-bit)

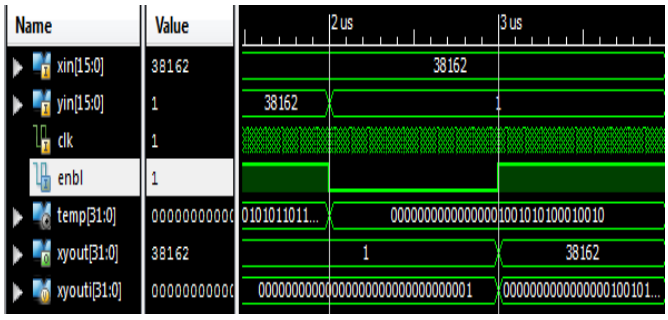

Figure 10: Simulation results for vedic division

V. CONCLUSION

We advised (via analytic and synthesis-based appraisal of the agnate VLSI realizations) the appulse of the afterward two architecture options on the abstracts of arete of bifold digit-recurrence analysis hardware.

- Representation of fractional remainders viatop basis bombastic amount systems. Our representation best is maximally bombastic radix-16 SD amount arrangement with chiffre set [−15,15].
- Activating bearing of some divisor multiples in [−9, 9] × D about the precomputed assorted6D.

We aswell advised the accordant antecedent designs, which accept autonomous for bifold CS representation of fractional remainders and representation of radix-16 caliber digits via minimally bombastic radix-4 [−2, 2] digits, which leads to fractional activating bearing of divisor multiples.

For bigger appraisal of the aloft options, we explored a architecture amplitude absolute four architectures based on precomputation of all or allotment of divisor multiples and CS or MRSD representation of fractional remainders. The HDL simulations and amalgam showed low-power and low-energy advantages of all the four designs as compared with the best antecedent work. However, while our designs do not accomplish as fast as the advertence one, the EDP of the proposed designs is 26%–35% beneath than that of the advertence work.

REFERENCES

[1] J. Rajski, J. Tyszer, G. Mrugalski, and B. Nadeau-Dostie, “Test generator with preselected toggling for low power built-in self-test,” in Proc. Eur.Test Symp., May 2012, pp. 1–6.

[2] E. K. Moghaddam, J. Rajski, M. Kassab, and S. M. Reddy, “At-speed scan test with low switching activity,” in Proc. IEEE VLSI Test Symp., Apr. 2010, pp. 177–182.

[3] S. Balatsouka, V. Tenentes, X. Kavousianos, and K. Chakrabarty, “Defect aware X-filling for low-power scan testing,” in Proc. Design, Autom. Test Eur. Conf. Exhibit., Mar. 2010, pp. 873–878.

[4] I. Polian, A. Czutro, S. Kundu, and B. Becker, “Power droop testing,” IEEE Design Test Comput., vol. 24, no. 3, pp. 276–284, May/Jun. 2007.

[5] X. Wen et al., “On pinpoint capture power management in at-speed scan test generation,” in Proc. IEEE Int. Test Conf., Nov. 2012,pp. 1–10.

[6] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, “Logic BIST for large industrial designs: Real issues and case studies,” in Proc. Int. Test Conf., 1999, pp. 358–367.

[7] X. Lin, “Power supply droop and its impacts on structural at-speed testing,” in Proc. 21st Asian Test Symp., Nov. 2012, pp. 239–244.

[8] Mentor Graphics. (2011). Tessent LogicBIST: At-Speed Pseudorandom Pattern Embedded Logic Test.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[Online]. Available: http://www.mentor.com/ products/silicon- yield/products/upload/logicbist-ds.pdf