Low Power Divider Using Vedic Mathematics

Dalal Rutwik Kishor
School of Electronics and Communication
VIT Chennai,
Tamilnadu, India
dalal.rutwikkishor2013@vit.ac.in

V.S. Kanchana Bhaaskaran
School of Electronics and Communication
VIT Chennai,
Tamilnadu, India
vskanchana@gmail.com

***Abstract*—Divider is an inevitable and basic hardware module employed in advanced and high speed digital signal processing (DSP) units of high precision. It is widely used in radar technology, communication, industrial control systems and linear predictive coding (LPC) algorithms in speech processing. This paper proposes a fast, low power and cost effective architecture of a divider using the ancient Indian Vedic division algorithm. The merits of the proposed architecture are proved by comparing the gate count, power consumption and delay against the conventional divider architectures. The validation of the proposed architecture has resulted in 52.93 percent reduction in power dissipation against comparison with conventional divider using repeated subtraction. The designs were implemented using industry standard Cadence® software using 45nm technology library. The design has been validated on FPGA Spartan-3E kit. The validation results show appreciable reduction in circuit latency and in Look-Up-Table (LUT) utilization using proposed Vedic divider than the conventional divider.**

Keywords—Vedic divider; Divider Circuit; Arithmetic Circuits; Division by Subtraction

I. Introduction

The high-performance division implementation on the hardware has gained popularity over the years. Division being a sequential type of operation, it incurs increased latency and complex hardware implementation. In the process, it consumes more silicon area. The division algorithms were developed to ease the computational difficulties. The restoring, non-restoring [1] and SRT (Sweeney, Robertson and Tocher) division algorithms [2] [3] are the widely used division algorithms. These algorithms can be divided into slow and fast division types. The slow division methodology produces a single bit of final quotient per iteration [4]. An extra step is required in restoring algorithm to restore the negative residual to the positive one. To avoid this extra step, a non-restoring algorithm was used which operated on negative residuals. The SRT division is logically more advantageous and faster, when the quotient contains consecutive zeroes. The complexity of implementation of these algorithms arises when a quotient digit selection has to be carried out. These drawbacks give rise to exploration for fast division techniques.

The methods followed in the past for a faster division methodology are division by series expansion (Goldschmidt algorithm) and division by convergence (also known as Newton-Raphson method) [1]. The digit recurrence method is based on successive subtraction technique to obtain fixed number of quotient bits per iteration. It necessitates simpler implementation of hardware and requires only smaller area. However, it incurs larger latency during the computation. The fundamental operation of Newton-Raphson and Series Expansion is carrying out the division operation through multiplication. These methods offer a low latency for high precision computations. However, they increase the silicon area requirement. This technique requires normalizing of operands, which is the limitation found in this method. Thus, the Vedic mathematics technique is used to develop a novel divider design.

Vedic mathematics [5] is a unique technique of carrying out mathematical computations and it has its roots in the ancient Indian Mathematics. This paper presents the divider architecture using one of the Vedic mathematics techniques called as *Paravartya-Yojayet*, which means to transpose and apply [1]. The implementation of this method is done using multiplication and addition, resulting in a substantial reduction in the propagation delay and total power consumption.

In this paper, Section 2 gives a brief introduction to Vedic mathematics. Section 3 presents the analysis and optimization of Paravartya sutra. Section 4 provides the design of the architecture. Section 5 illustrates the simulation results and Section 6 summarizes and concludes the paper.

II. Vedic Mathematics

The various Vedic mathematics methods were compiled by *Sri Bharati Krishna Tirthaji* during the years 1911-1918. The word *Vedic* has been derived from *Vedas,* which means a storehouse of knowledge. Vedic mathematics provides single line, simple and super fast method for cross-checking. It gives the mathematical elaboration of 16 *sutras* (simple mathematical formulae from the Vedas) or aphorisms, as algorithms and *upa sutras* or *corollaries* derived from the Sutras [5]. The most striking feature of the Vedic mathematics is its coherence. The whole system has been perfectly interrelated and unified instead of having a series of unrelated techniques. Sutras based on Vedic mathematics have reduced the computational effort in the fields such as algebra, polynomial functions, exponential, addition, subtraction, multiplication and division. It also covers certain high level problems of square, cube, linear and simultaneous equations, factorization, H.C.F and recurring decimals. *Nikhilam Navatascaramam Dasatah* [6] and *Paravartya Yojayet* are the sutras which mainly focus on division. Nikhilam method is effective if the divisor is close to the power of 10. However, the Paravartya method is well suited for division where divisor is larger than the power of 10. Synthetic division which is an easy method to carry out division is a special case of Paravartya sutra.

III. Analysis and Optimization of Paravartya Sutra

Paravartya Sutra helps to minimize computation, achieve process optimization and maintain accuracy even as the numbers of iterations are reduced. The algorithm of Paravartya Sutra is described only for division of decimal numbers in Ancient Mathematics literature. Thus, it is essential to have a basic knowledge of division of decimal numbers using Paravartya Sutra.

A. Decimal Number Division Algorithm

Division algorithm for decimal numbers is based on general human-mind approach. It provides easier and logically simple implementation.

According to Paravartya Sutra, all the digits of the divisor are complemented except the most significant digit. These complemented digits are multiplied to sum of each column of the dividend, followed by successive additions of consecutive columns. The summation of all the columns results in quotient and remainder.

Implementation of the algorithm is illustrated using an example. Assume the dividend is 13567 and the divisor is 123. The number of iterations is 124, if repetitive subtraction method is used. On the other hand, using Vedic methodology, the number of iterations is reduced to 8. Fig. 1. shows the illustration of division of two decimal numbers using Paravartya method [1].

Divisor			Dividend				
1	2	3	1	3	5	6	7
	-2	-3		-2	-3		
					-2	-3	
						0	0
			1	1	0	3	7
			Quotient=110			Remainder=37	

Fig. 1. Vedic Division of decimal numbers

B. Binary Number Division Algorithm

In digital design, there is a wide use of binary numbers, requiring the decimal to binary conversion. Thus, the same algorithm as used for decimal numbers is implemented for the binary numbers in this paper. Paravartya Sutra requires bits to be complemented and it requires the bits, namely, -1, 0 and 1 to represent the divisor and dividend. Fig. 2. shows the division algorithm for binary numbers.

Divisor				Dividend							
1	0	1	1	1	0	0	1	0	0	1	1
	0	-1	-1		0	-1	-1				
						0	0	0			
							0	1	1		
								0	0	0	
									0	-1	-1
				1	0	-1	0	1	1	0	0
				Q=10001-00100							
				Quotient=01101					Remainder= 100		

Fig. 2. Vedic divider for binary numbers

C. Crumb Division Algorithm

The crumb algorithm uses 2 bits to represent a single crumb. The first bit in the crumb is the sign bit and the second being value bit. If the sign bit is 0, the number is considered positive else negative. Table I shows the representation of the sign and value bit of a crumb [7].

TABLE I. Representation of crumb

Dividend or Divisor (D)	Sign bit (Ds)	Value bit (Dv)	Crumb representation
0	0	0	00
1	0	1	01
-1	1	1	11

Vedic divider architecture is thus modified by replacing each bit in Fig. 2. by an equivalent crumb. Fig. 4. shows the modified crumb based Paravartya division algorithm for the division of 8-bit dividend and 4-bit divisor. The 2*2 bit partial multiplication considers two least significant bits as the output in the modified algorithm. The final quotient and remainder are in the encoded crumb format which has to be decoded in binary to obtain proper quotient and remainder.

The intensive algorithm flow description of modified divider architecture is shown in Fig. 4. N and M represents the total number of divisor and dividend bits respectively. Initially, both the divisor and dividend bits are encoded into crumbs. The sign bit Ns is considered as 0 by default. Whenever the value bit Nv of divisor is 1, the Ns bit is complemented. All divisor crumbs are complemented except MSC (most significant crumb).

S	Divisor	Dividend	
	01 00 01 01	01 00 00 01 00 00 01 01	
S1	00 11 11	00 11 11	
S2		00 00 00	
S3		00 01 01	
S4		00 00 00	
S5		00 11 11	
		01 00 11 00 01	01 00 00
		Q=10001-00100	
		Quotient=01101	Remainder= 100

Fig. 3. Modified algorithm for Vedic division

The modified Vedic division algorithm is elaborated through its different stages(S) as shown in Fig.3 and the process is generalized using a flowchart provided in Fig.4.The proposed modified Vedic algorithm can be categorized into the following group of operations [7].

1. Binary to Crumb Encoding - Divisor and dividend bits are encoded into crumbs. Crumb is the group of two bits, which represents the unsigned values from 0 to 3 and the signed values from -1 to 1.
2. Vedic Division Operation - Division operation is performed through successive partial multiplication and addition process.
3. Crumb to Binary Decoding - The quotient is extracted from the division operation result by crumb to binary decoding method.

The partial multiplication of complemented divisor crumbs with MSC of the dividend (Dd [M-1]) is carried out at the initial stage S1 of the algorithm. The output of the multiplication is temporarily saved in TEMP array. The elements of TEMP array are successively added with the dividend array elements starting from Dd [M-1] to Dd [M-N+2]. The subsequent element Dd [6] is then replaced by previous addition result of Dd and TEMP array, and used for the partial multiplication of the next stage. The total number of stages incurred in M/N division is M-N+1. After completion of all stages, the output of most significant M-N+1 crumb elements of Dd and TEMP array addition gives the crumb encoded quotient. The remaining N-1 results give the crumb encoded remainder.

Fig. 3 shows the example of the division of an eight bit dividend by a four bit divisor using the proposed modified algorithm. In this example, the least significant three divisor crumbs are complemented and successively partially multiplied with dividend. As number of divisor array elements complemented is equal to the number of addition result elements reserved for the remainder, the quotient is determined by five MSCs (most significant crumb) and the remainder is extracted from the remaining three crumbs of the final result. The exact binary quotient and remainder values are then obtained by decoding the crumb into the bit format.

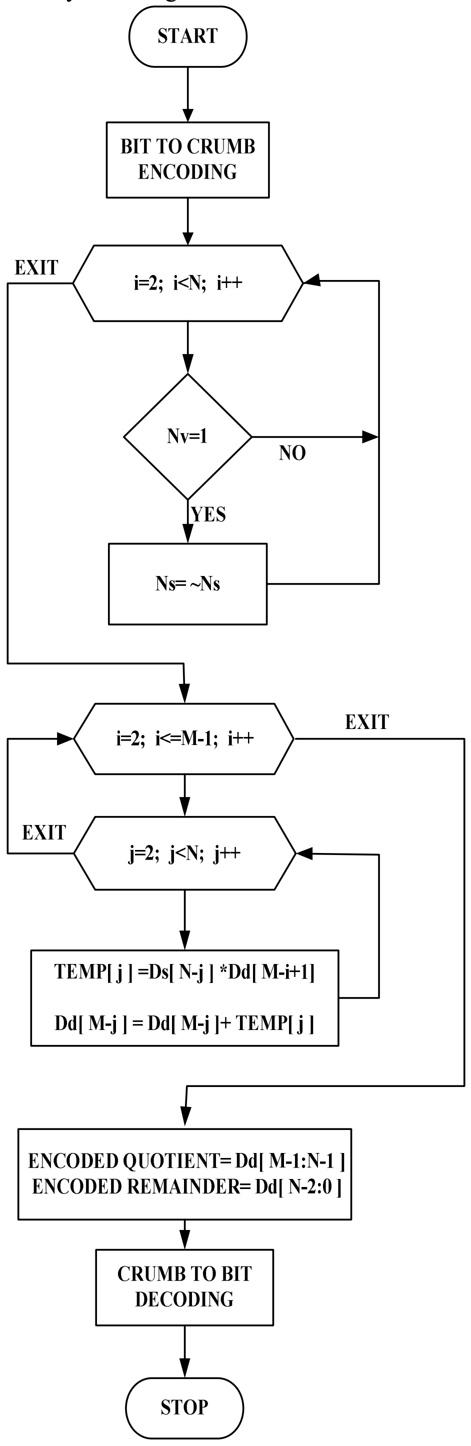
START
BIT TO CRUMB ENCODING
EXIT
i=2; i<N; i++
Nv=1
NO
YES
Ns= ~Ns
EXIT
i=2; i<=M-1; i++
EXIT
j=2; j<N; j++
TEMP[j] =Ds[N-j] *Dd[M-i+1]
Dd[M-j] = Dd[M-j]+ TEMP[j]
ENCODED QUOTIENT= Dd[M-1:N-1]
ENCODED REMAINDER= Dd[N-2:0]
CRUMB TO BIT DECODING
STOP

Fig. 4. Flowchart representation of modified Vedic divider architecture

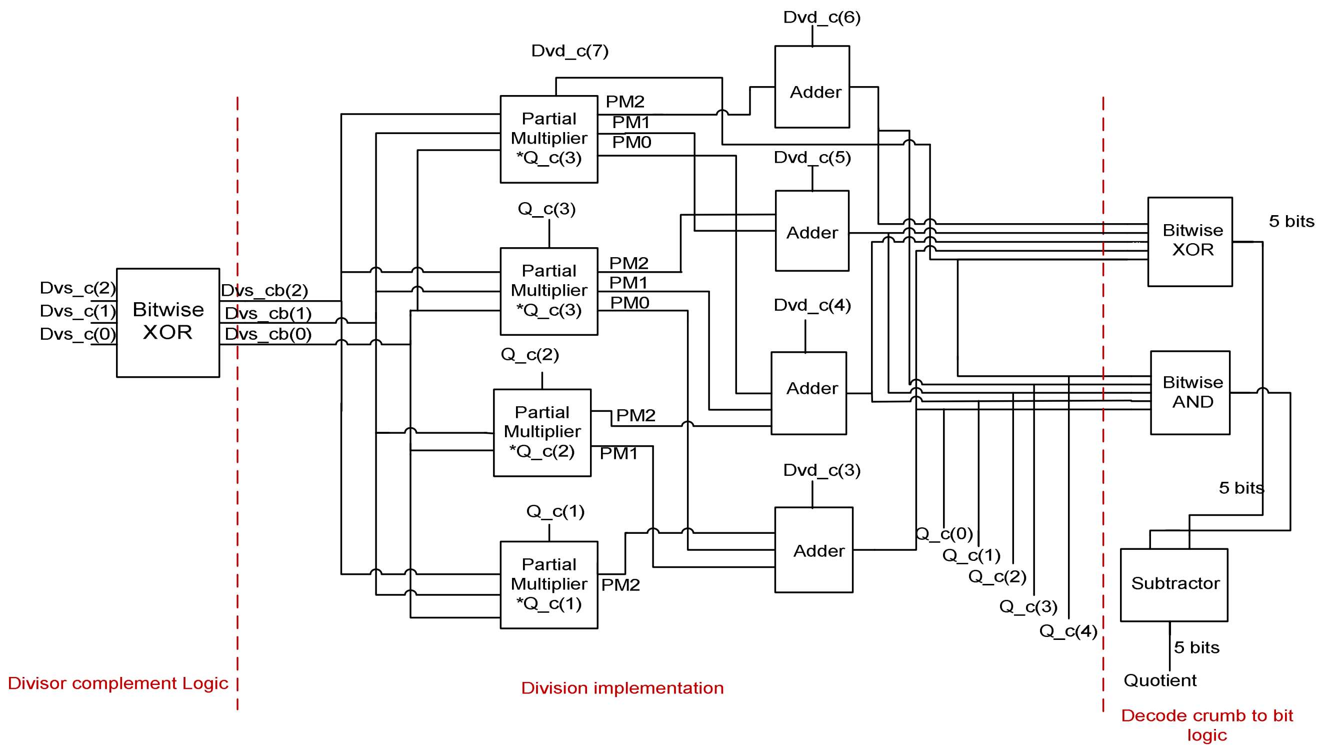

Fig. 5. Architecture of Modified Vedic Divider for 8 bit dividend and 4 bit divisor

IV. Design of the Vedic Divider Architecture

The hardware block diagram of modified Vedic divider is shown in Fig. 5. The execution of proposed divider algorithm is divided into three steps. First, the divisor crumb is complemented for further partial multiplication. Second, the division algorithm is implemented through successive addition and partial multiplication. Lastly, the final quotient in the crumbs is decoded into bits.

In the first step, the bitwise XOR operation is carried out for all the divisor crumbs except the MSC. The XOR output is replicated to obtain the output in the crumb format. Hence, using the XOR logic, 01 is converted into 11 while 00 remains the same. MSC is not complemented, since it is not required for further calculations. In the second step, two logic blocks namely, i) 2*2 Partial Multiplier and ii) 2 bit adder are used.

$$\mathbf{PM0 = A0 * B0}$$
$$\mathbf{PM1 = A1 * B0 + A0 * B1}$$

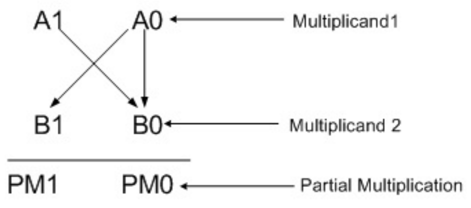

Fig. 6. Partial multiplier

The partial multiplication process is shown in Fig. 6. [8] Urdhva Tiryabhyam algorithm of the Vedic mathematics is used for partial multiplication. {A1, A0} and {B1, B0} are the multiplicands which give the resultant output {PM1, PM0}. Firstly, the LSBs of the two multiplicands are multiplied to give PM0. The addition of cross multiplication of (A1*B0) and (B1*A0) is carried out which becomes PM1. In partial multiplication, only the least significant two bits are considered as the output.

The complemented output of step 1 is partially multiplied with the MSC of the dividend and successively added with the next dividend crumb. This result of successive addition is used as one of the multiplicand for the next partial multiplication. The number of inputs for the first adder is 2, that for the second adder is 3, and it successively increases upto 4 for the 8-bit by 4-bit division process. In other words, for an M-bit by N-bit divider, adders with inputs from 2 to successively increasing inputs upto N would be employed. The output of addition gives the quotient which is encrypted in the crumb format. Thus, the quotient crumbs are decoded into bits in step three. In step three, the bitwise AND and bitwise XOR operations are performed on the divisor and passed through the subtractor. The output of the subtractor gives the final quotient in bit format.

It may be noted that the conventional Non-restoring division hardware requires numerous comparators which increases the design complexity. On the other hand, in the Vedic divider, no comparator is required thus making the hardware simpler and the operation faster. As MSB of 2*2 multiplication output is not necessary for the Vedic division, partial multiplier terminates the operation after two bits. This eliminates unnecessary logic cells from design hardware.

V. Simulation Results

A. Verification of the Algorithm

The algorithm has been verified using the Spartan3E-xc3s250e-4tq144 family of FPGA (Field Programmable gate array). This enables the verification in the form of simulation and synthesis of the developed code. Generic dividers require several comparators to obtain the final result. However, in the Vedic divider, comparators are eliminated resulting in reduction of the power consumption and delay. The delay reduction realized through the proposed architecture is shown in Table II. There is a 34 percent reduction of latency in the Vedic divider as compared with the conventional divider.

TABLE II. Delay comparison of conventional and Vedic divider Styles

Specification	Conventional Divider	Vedic Divider
Family	Xilinx-Spartan 3E	Xilinx-Spartan 3E
Selected Device	xc3s250e-4tq144	xc3s250e-4tq144
LUT Utilization (Number of slices)	1.5% (69/2448)	1% (38/2448)
Total Delay	41.5ns	27.393ns

The Look Up Table (LUT) utilization in the configurable logic blocks (CLB) of FPGA is reduced to 1 % in the Vedic divider against the 1.5% usage for the conventional divider structure.

B. Circuit Implementation

The divider was designed on a 45nm CMOS technology node with a supply voltage of 1.2V using the industry standard Cadence tool. The total power dissipation in the Vedic divider is 9.227uW. The reduction in power is 52.93 % as compared to conventional divider. Fig. 7 illustrates the comparison of average power for conventional and Vedic divider.

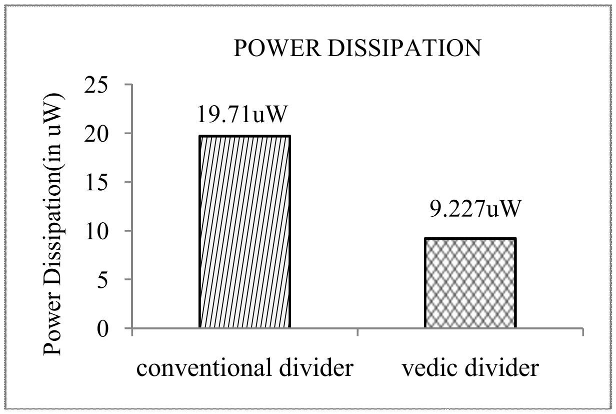

Fig. 7. Average power (in uW) for Conventional and Vedic divider

The comparison of the logic cells as used for the circuit implementation of 8-bit by 4-bit divider in both the conventional and the Vedic methods has been made as shown in Table III. The total number of logic cells employed in Vedic divider is 54.71% lesser than that of the conventional divider design.

TABLE III. Comparison of Conventional and Vedic division on the basis of logic cells used

Logic Cells	Conventional Division	Vedic Division
Adders/Subtractor	27	11
Comparators	11	-
Multiplexers	07	-
2x2 multiplier	-	09
XOR gates	08	04
Total	53	24

VI. Conclusion

The proposed Vedic divider architecture is implemented and is found to realize 52.93% less power reduction as compared to the conventional divider. 45nm CMOS technology files have been used in the process. The delay incurred by the proposed structure is only 34% of that incurred by the conventional divider. The Vedic divider employs 38 numbers of slices as against 69 slices used by the conventional divider circuit. Hence, the Vedic divider is validated for faster and more power efficient operating characteristics. Furthermore, the number of logic cells used in the Vedic divider is reduced by 54.71% as compared to that of conventional divider. These results facilitate the usage of Vedic divider for high performance applications like digital signal processors (DSP), digital calculators and radar technology.

The future work includes the extension of proposed Vedic divider architecture for 16 bits and 32 bits applications which will incorporate the comparative study of power consumption and performance. In this paper, the proposed Vedic divider is compared with repetitive subtraction technique of divider against power and delay incurred. The proposed architecture can be compared with other divider technique like SRT. The proposed Vedic methodology employed to determine the quotient can be extended to obtain the remainder.

References

[1] R. Senapati, B.K. Bhoi and M. Pradhan, "Novel binary divider architecture for high speed vlsi applications," Proceedings of IEEE Conference on Information and Communication Technologies, 2013, pp. 675-679.

[2] Wey and Wang, "Design of a fast radix-4 SRT divider and its VLSI implementation," Proceedings of IEEE Conference on Computers and Digital techniques, 2013, pp. 675-679.

[3] Kornerup, "Revisiting SRT quotient digit selection," Proceedings of IEEE Symposium on Computer Arithmetic, 2003, pp. 38-45.

[4] Derek Wong and Michael Flynn, "Fast division using accurate quotient approximations to reduce the number of iterations," IEEE Transactions On Computers, Vol. 41. No. 8, August 1992, pp. 981-995.

[5] Tirthji Maharaja JSSBK "Vedic mathematics," Motilal Banarsidas, Varanasi, India, 1986.

[6] Sengupta, Sultana and Chaudhuri, “An algorithm facilitating fast BCD division on low end processors using Ancient Indian Vedic Mathematics Sutras,” Proceedings of International Conference on Communications, Devices and Intelligent Systems, 2012, pp. 373-376.

[7] Nazeih M. Botros, “HDL programming,” Dreamtech Press,2009.

[8] B. R. Appasaheb and V S Kanchana Bhaaskaran, “Design and implementation of an efficient multiplier using vedic mathematics and charge recovery logic,” Proceeding of International Conf. on VLSI, Springer 2013, pp. 103-108.