APPLICATION OF VEDIC MATHEMATICS IN HIGH SPEED SYSTEM – A SURVEY

Raymond Austn

ABSTRACT:

A system's performance is determined by the performance of a multiplier. It is a key component of many high-performance systems like microprocessors, digital signal processors etc. It is the slowest element in the system, and consumes a major storage area. Hence, optimizing the speed and the storage area of the multiplier is critical.

Multiplication is the most basic and frequently used operation in the CPU, and forms a basis for other complex operations such as convolution, discrete Fourier transform, fast Fourier transforms etc. It becomes imperative to have faster arithmetic unit to match the increasing need for faster clock frequency. Traditional methods like booth, array method, carry save, Wallace tree etc. used for arithmetic operations takes longer time for processing. Multiplier architecture based on these methods are not very efficient in terms of speed, power and area.

Software Quality Indexes [1] is a part of requirement document to achieve software quality. McCall's quality factors model deals with the requirements that directly affect the daily operation of the software.

Vedic Mathematics [2] is a system or a set of strategies to solve a range of mathematical problems covering almost all branches of mathematics and is considered as India's gift to the world. The Vedic Sutras can be applied to multiplex problems involving many mathematical operations. The application of sutras saves a lot of time and effort in solving problems, when compared to conventional methods.

The study presents an extensive survey on features of the Vedic multipliers based on the research papers. Furthermore, association of these features to the software quality indexes are also presented. Implementation of appropriate sutras results in speed improvement, accuracy, reduction of power consumption, complexity and area.

INTRODUCTION:

Computational arithmetic operations like multiplication, division, square, square root, cubing, reciprocal and other basic operations play a vital role in the field if digital signal processing (DSP), image processing, computer graphics, cryptography etc. All these operations are usually implemented in software, using hardware. With the advancement of technology, computers keep getting faster; there are always new applications that need more processing speed than before. Examples of applications include real-time video stream encoding and decoding, real-time biometric (face, retina, finger print) recognition, military aerial and satellite surveillance. To meet the demand of these new applications, there is a need to develop algorithms for accelerating applications on commercial hardware.

A system's performance is determined by the performance of a key component multiplier due to the fact of being the slowest element in the system. Furthermore, it is also the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. Multiplication is the most basic and frequently used operation in the CPU, forms a basis for other complex operations such as

convolution, discrete Fourier transform, fast Fourier transforms etc. A CPU devotes considerable amount of processing time in performing arithmetic operations. Two important parameters associated with multiplication algorithms are latency and throughput. Latency is the "real delay of computing a function". Throughput is a measure of "how many computations can be performed in a given period of time". It becomes imperative to have faster arithmetic unit to match the increasing need for faster clock frequency. Traditional method used for arithmetic operations takes longer time for processing. These methods include booth, array method, carry save, Wallace tree etc. Multiplier architecture based on these methods are not very efficient in terms of speed, power and area.

Vedic Mathematics is a system or a set of strategies to solve a range of mathematical problems covering almost all branches of mathematics and is considered as India's gift to the world. Vedic Mathematics was compiled by Sri Bharati Krsna Tirthaji, Jagadguru Sankaracharya of Puri (March 1884- February 1960). It includes sixteen main formulae (sutras) and fourteen corollaries (up-sutras or sub-sutras) from the Atharva Veda. It is important to note that these formulae are not found in the present editions of Atharva Veda. The Vedas, the scriptures are the sources of all knowledge and of all sciences. The vedic sutras can be applied to multiplex problems involving a large number of mathematical operations. The application of sutras saves a lot of time and effort in solving problems, when compared to formal methods.

Software Quality Indexes is a part of requirement document to achieve software quality. McCall's quality factors model deals with the requirements that directly affect the daily operation of the software. The McCall's factor model provides a practical, up-to-date method for classifying software requirements.

The study presents an extensive survey on features of the Vedic multipliers based on the research papers. Furthermore, association of these features to the software quality indexes are also presented. Implementation of appropriate sutras results in speed improvement, accuracy, reduction of power consumption, complexity and area. The study is organized as vedic sutras, vedic multiplier design, McCall's factor model, surveyed designs associating each of their features to software quality factors and the conclusions.

VEDIC SUTRAS:

The lexical meaning of Sutra is "thread", "string", "a key" or "a formula". Here the last two meanings are applicable. The sutras are single line phrases in Sanskrit. The knowledge of Sanskrit language is not required since they are well translated and are easy to understand and remember.

Vedic Mathematics has sixteen main formulae (sutras) and fourteen corollaries (up-sutras or sub-sutras). They are listed in the tables below:

SUTRAS:

S.No.	सूत्रा	Transliteration	Translation
01	एकाधिकेन पूर्वेण	Ekadhikena purvena	By one more than the previous one
02	निखिलं नवत: चरमं दशत:	nikhilam navatascharamam dasataḥ	All from 9 and the last from 10
03	ऊर्ध्वतिर्यग्भ्याम्	Ūrdhvatiryagbhayām	Vertically and Crosswise
04	परावर्त्य योजयेत्	parāvartya yojayet	Transpose and Apply
05.	शून्यं साम्यसमुच्चये	shūnyam samyasamuccaye	If the sum is same, that sum is zero
06.	आनुरूप्ये शून्यमन्यत्	Ānurūpye shūnyamanyat	If one is in the ratio, the other is zero
07.	संकलनव्यवकलनाभ्याम्	SaṅKalanavyavakalanābhyām	By addition and by subtraction
08.	पूरण।पूरण।भ्याम्	Pūraṇapūraṇabhyām	By the completion and non-completion
09.	चलनकलनाभ्याम्	Calanakalanabhyām	By integration and differentiation
10.	यावदूनम्	Yāvadūnam	Whatever is less
11.	व्यष्टीसमष्टी:	Vyaṣṭisamsṭiḥ	Specific and general
12.	शेषा्न् अंकेन चरमेण	Sheṣāṅkena carameṇa	The remainder by the last digit
13.	सोपान्त्यद्वयमन्त्यम्	Sopāntyadvayamantyam	The ultimate and twice the penultimate
14.	एकन्यूनेन पूर्वेण	ekanyūnena pūrvena	By one less than the previous
15.	गुणीतसमुच्चय:	gunṇitsamuccayaḥ	The product of the sum
16.	गुणकसमुच्चय:	guṇakasamuccayaḥ	All the multipliers

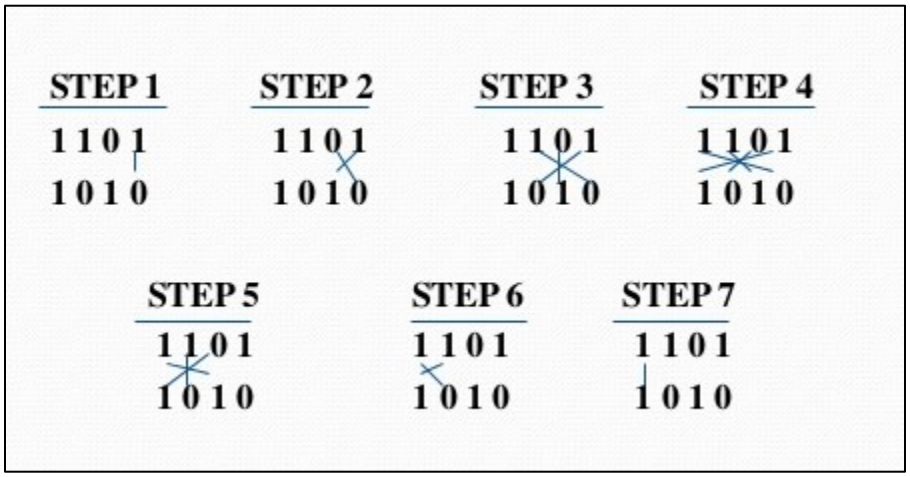
SUB-SUTRAS:

S.No.	उपसूत्रा	Transliteration	Translation
01	आनुरूप्येण	Ānurūṇpyena	Proportionately
02	शिषयते शेषसंज्ञ:	Shiṣyate sheṣamjñaḥ	The remainder remains constant
03	आद्यमाद्येनान्त्यमन्त्येन	Ādhyam ādyen antyam antyena	First by the First and Last by the Last
04	केवले: सप्तकं गुणयात्	Kevalaiḥ saptakam guṇiyat	For seven the multiplicand is 143
05.	वेषटनम्	Veṣṭanam	By osculation
06.	यावदूनम् तावदूनम्	yāvadūnam tāvadūnam	Lessen the deficiency
07.	यावदूनम् तावदूनीकृत्य वर्गम् च योजयेत्	yāvadūnam tāvadūnikṛtya vargam ca yojayet	Whatever the deficiency lessen by that and set the square
08.	अन्त्ययो: दशकेऽपि	Antyayordashake'pi	Last summing ten or its higher power
09.	अन्त्ययोरेव	Antyayoreva	Only the last two
10.	समुच्चय गुणत:	samuccaya guṇitaḥ	the sum of the product
11.	लोपनस्थापनाभ्याम्	Lopanasthāpanābhyām	by alternate elimination and retention
12.	विलोकनम्	Vilokanam	by mere observation
13.	गुणीतसमुच्चय: समुच्चय गुणत:	guṇitasamuccayaḥ	The product of the sum of the co-efficient is equal to the sum of the co- efficient in the products

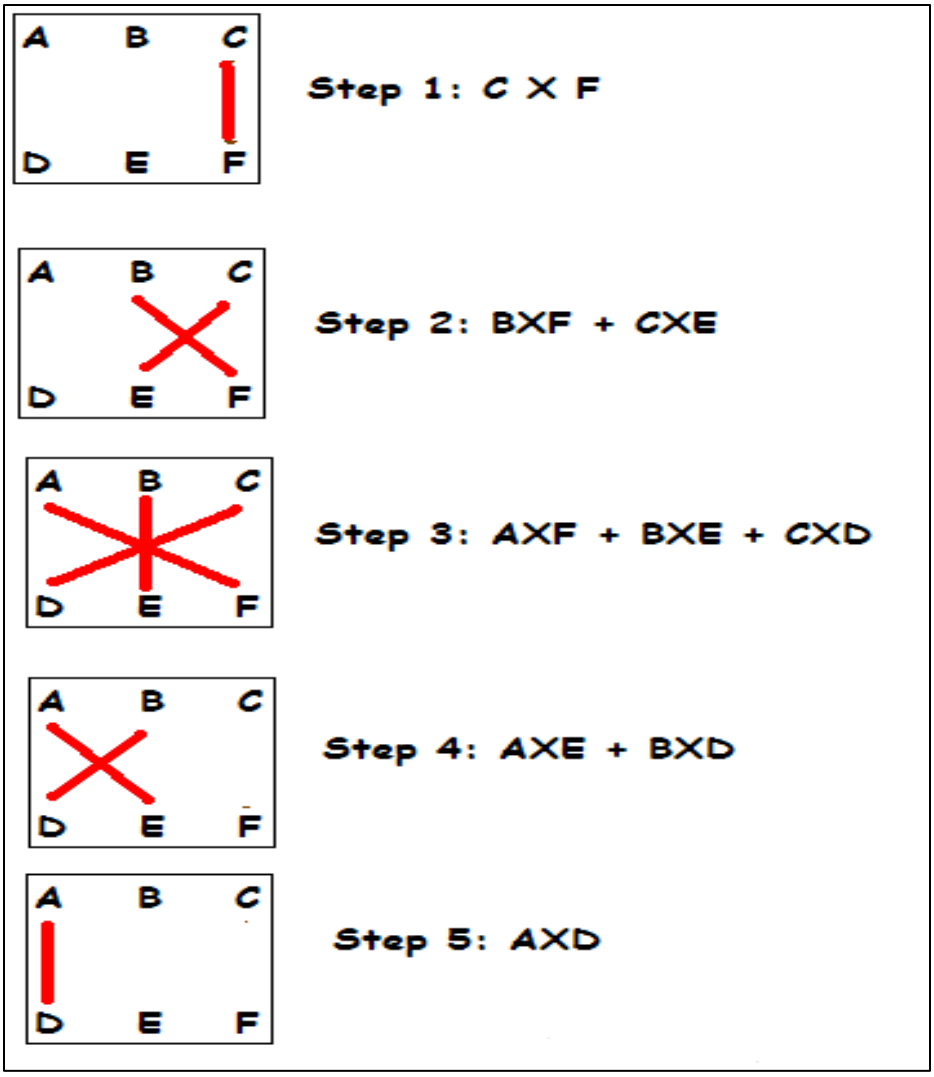
VEDIC MULTIPLIER DESIGN:

Urdhvatiryagbhyam Sutra: The Sanskrit term means “Vertically and Crosswise”. This is a general formula applicable to all types of multiplication and also of division of a large number by another large number. The uniqueness of this sutra is that partial product generation and addition can be done simultaneously at the same time. Since there is a parallel generation of partial products and their sums, the processor becomes independent of the clock frequency. The advantage here is that parallelism reduces the need processors to operate at increasingly high clock frequencies. A high clock frequency will result in increased processing power and will lead to increased power dissipation resulting in higher device operating temperature. All the demerits associated with the increase in power dissipation can be negotiated by employing vedic multiplier. Since it is faster and efficient its layout has a quite regular structure. Due to its regular structure it can be realized easily in a silicon chip.

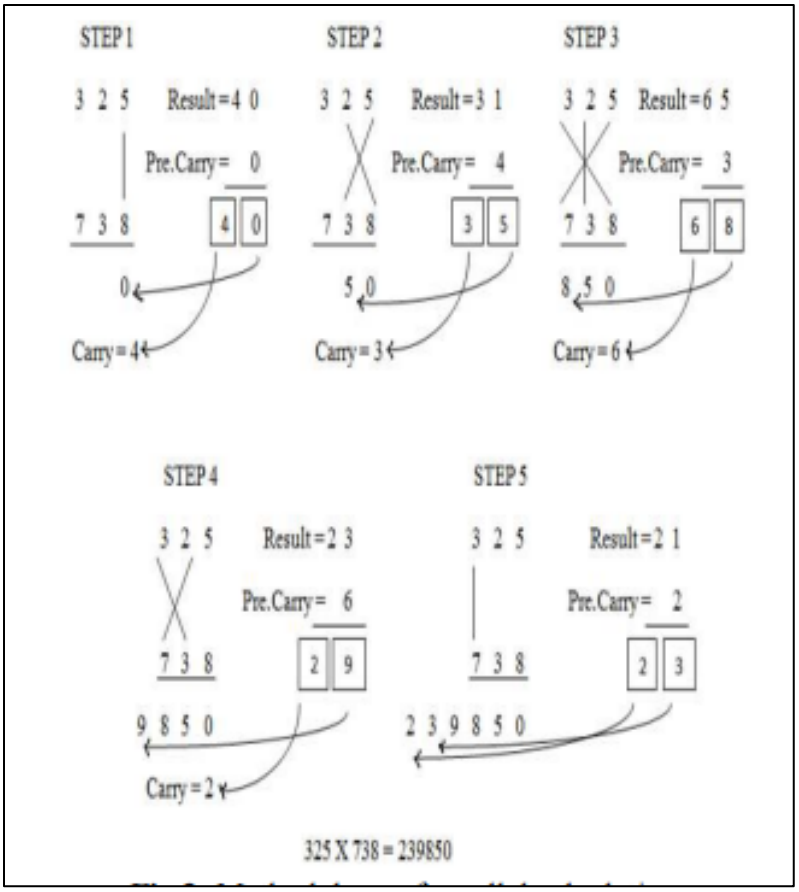
The multiplication scheme can be explained by the following example as shown below:



Line diagram for multiplication of two 4-bit numbers

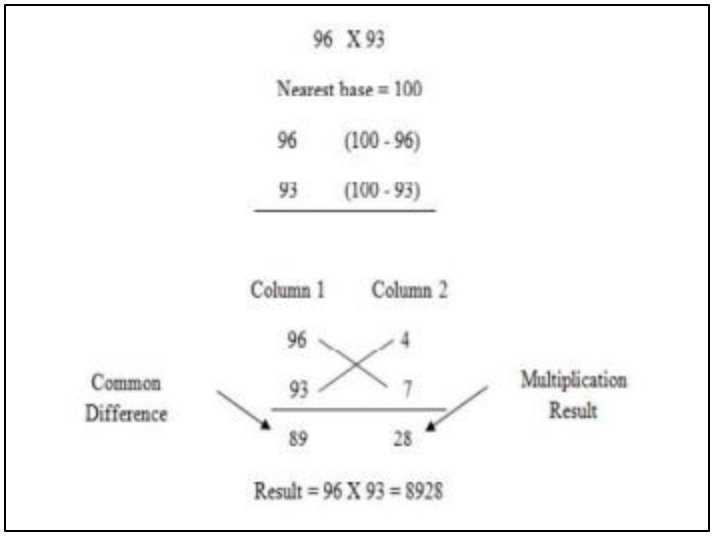


Line diagram for multiplication



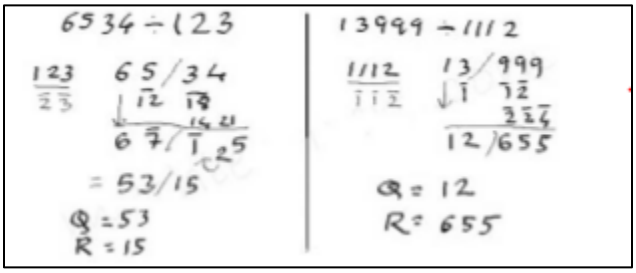
Multiplication of two numbers using Urdhvatiryagbhyam Sutra

Nikhilam navatasscaramam dastah Sutra: The Sanskrit term means “All from 9 and the last from 10”. The sutra can be efficaciously applied in multiplication of numbers, which are nearer to base like 10, 100,1000 etc. The numbers to be multiplied can be either less or more than the base considered (one of them of both), however both the numbers should have one common base. The difference between the number and the base is the deviation which be either positive or negative. The multiplication procedure using nikhilam sutra involves minimum number of steps in computation, reducing the space area, saving more time for computation. An example of nikhilam sutra is illustrated below.



Nikhilam sutra illustration

Paravartya Yojayet Sutra: The Sanskrit term translated into English, says “Transpose and Apply”. This method is related to the Chinese remainder theorem and the Horner’s rule of the synthetic division. An example of Paravartya Yojayet sutra is illustrated below.



Paravartya Yojayet sutra illustration

Gunakasamuchayah Sutra: The Sanskrit term translated into English, says “All the multipliers”. The sutra means the factors of the sum is equal to the sum of the factors. It means the product of the sum is equal to the sum of the coefficients in the product. i.e. Sc of the product= Product of the Sc (in factors). This sutra is useful in verifying the correctness of the obtained answes in multiplication, division and factorization. This rule holds good for cases of cubics, biquadratics etc.

Example:

$(x+1)(x+2)(x+3) = x^3 + 6x^2 + 11x + 6$

$2*3*4 = 1 + 6 + 11 + 6$

$= 24$. Thus verified.

McCall's FACTOR MODEL:

Software Quality Factors (Indexes) define the broad spectrum of software quality requirements and is a part of requirement document to achieve software quality. In order to achieve satisfaction of the users, it is expected that individuals defining software requirements refer to each factor and, accordingly, examine the need to in cooperate the respective requirements in their requirement documents. Due to differences among software projects, not all the factors can be represented in all the requirements documents. Besides, unable to develop software that meets simultaneously all indexes of quality. This is due to problems of cost and time required to develop software that meets every quality index and because some of the indexes contradict each other. E.g. Efficiency, Flexibility and Portability reduce software reliability. Flexibility, Maintainability may decrease efficiency.

McCall's quality factors model deals with the requirements that directly affect the daily operation of the software. The McCall's factor model provides a practical, up-to-date method for classifying software requirements. McCall's factor model classifies all software requirements into 11 software quality factors. The 11 factors are grouped into three categories – product operation, product revision and product transition- as follows:

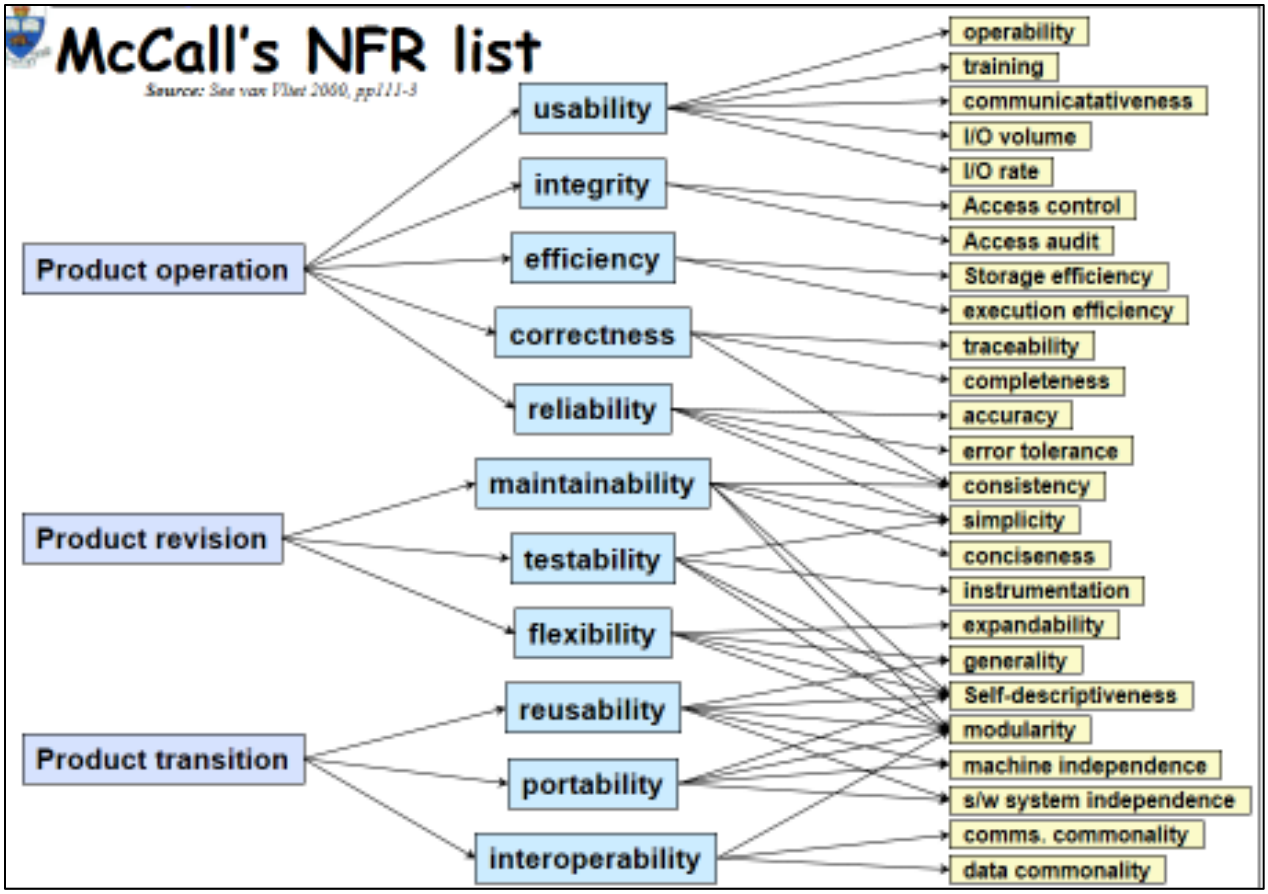
- **Product operation factors:** Correctness, Reliability, Efficiency, Integrity, Usability.
- **Product revision factor**s: Maintainability, Flexibility, Testability.
- **Product transition factors:** Portability, Reusability, Interoperability.

According to McCall's model, five software quality factors are included in the product operation category, all of which deal with requirements that directly affect the daily operation of the software. The table presents factors, sub-factors and definitions suggested by McCall's model:

McCall's Quality Categories	**S/W Quality Factors**	**S/W Sub-Factors**	**Definition**
Product Operation	Correctness	Accuracy Completeness Up-to-dateness Availability (response time) Coding and documentation guidelines compliance (consistency)	Extent to which a program satisfies its specification and fulfills the users objectives.
			The ability of the software to produce output given an input and the extent to which it meets the defined requirements.
	Reliability	System reliability Application reliability Computational failure recovery Hardware failure recovery	Extent to which a program can be expected to perform its intended function with required precision.
			Failures to provide service. Maximum allowed software failure rate. Refer to entire system or one or more of its separate functions.

	Efficiency	Efficiency of processing Efficiency of storage Efficiency of communication Efficiency of power usage (for portable units)	The amount of computing resources and code required by a program to perform a function.
			Deal with the hardware resources needed to perform all the functions of the software system in conformance to all other requirements.
	Integrity	Access control Access audit	Extent to which access to software or data by unauthorized persons can be controlled.
			Deal with the software system security-requirements to prevent access to unauthorized persons.
	Usability	Operability Training	Effort required to learn, operate, prepare input, and interpret output of a program.
			The extent to which the program is easy to use and its tolerance level of user errors. Deal with the scope of staff resources needed to train a new employee and to operate the software system.
Product Revision	Maintainability	Simplicity Modularity Self-descriptiveness Coding and documentation guidelines compliance (consistency) Document accessibility	Effort required to locate and fix an error in an operational program.
			The efforts needed by the users and maintenance personnel to identify the reasons for software failures, to correct the failures and to verify the success of the corrections.
	Flexibility	Modularity Generality Simplicity Self-descriptiveness	Effort required to modify an operational program.
			The capabilities and efforts required to support adaptive maintenance activities are covered by the flexibility requirements.

	Testability	User testability Failure maintenance testability Traceability	Effort required to test a program to insure it performs its intended function.
			Deal with the testing of an information system as well as its operation.
Product Transition	Portability	Software system independence Modularity Self descriptive	Effort required to transfer a program from one hardware configuration and/or software environment to another.
			Adaption of a software system to other environments consisting of different hardware, different operating systems, and so forth.
	Reusability	Modularity Document accessibility Software system independence Application independence Self descriptive Generality Simplicity	Extent to which a program can be used in other applications – related to the packaging and scope of the functions that programs perform.
			Deal with the use of software modules originally designed for one project in a new software project under development.
	Interoperability	Commonality System compatibility Software system independence Modularity	Effort required to couple one system with another.
			Focus on creating interfaces with other software systems or with other equipment firmware.



SURVEYED DESIGNS:

Honey Durga Tiwari et.al [3] developed a multiplier and square architecture for low power and high speed applications. They used Urdhva Tiryakbhyam and Nikhilam sutras efficiently for multiplication of two large numbers by reducing it to the multiplication of two small numbers. The FPGA implementation of the proposed design is more efficient in terms of space (area) and delay time compared to the conventional booth and array multipliers design.

M Ramalatha et.al [4] designed high speed energy efficient ALU using Urdhva Tiryakbhyam sutra. The designed high speed multiplier helps coprocessor which reduces load of processor. They showed that application of Urdhva Tiryakbhyam sutra reduced unwanted multiplication and produces intermediate results parallely. The optimized vedic multiplier designed was efficient having advantages – high speed, less complexity, decreased delay and consuming less area.

Devika Jaina et.al [5] proposed a design for multiplier accumulator unit (MAC) applying Urdhva Tiryakbhyam sutra. The algorithm was implemented in VHDL. Their approach was compared with the modified booth Wallace multiplier and high speed vedic multiplier. Their approach was found to be highly efficient in terms of speed (reduced delay). It can be realized easily on silicon due to regular and parallel structure.

Akhalesh K. Itawadiya et.al [6] proposed designing Digital Signal Processing (DSP) operations using multiplication in these operations e.g. convolution, correlation. A fast computation of DSP operations of two finite length sequence was implemented using Urdhva Tiryakbhyam sutra for multiplication. They implemented these operations in MATLAB with single GUI window. Their approach requires less processing time (average time in micro seconds) as compared to inbuilt functions of MATLAB (average time in mili seconds).

Sushma R Huddar et.al [7] identified the need of high speed cryptographic algorithm used in secure transactions. In order to meet this requirement, they developed an efficient architecture for performing the mix columns and inverse mix columns operation, considered as a major operation in the Advanced Encryption Standard (AEC) method of cryptography. The Urdhva Tiryakbhyam sutra was used. The designed cryptographic unit involving mix columns and inverse mix columns for AES was implemented on a Xilinx of FPGA. A 100% area efficiency and a two times speed increase was achieved as compared to other implementations – Splitting Approach & Look-up Table Approach.

Diganata Sengupta et.al [8] proposed an algorithm for fast BCD division based on Nikhilam and Parvartya sutras. They showed that execution time does not depend on the size of the dividend or the divisor, but on the number of remainders normalizations required. The Vedic Division Algorithm exhibited remarkable results with respect to conventional division algorithms. VLSI implementation of this algorithm was not tested.

Pavan Kumar et.al [9] designed an 8-bit Vedic multiplier using barrel shifter which requires only one clock cycle for “n” no. of shifts using Nikhilam sutra. The design was implemented and verified using FPGA and ISE simulator. The proposed design exhibited reduced propagation delay (an improvement of 45%) when compared to array multipier, booth multiplier and conventional vedic multiplier implemented on FPGA. The high speed implementation of such multiplier has a wide range of applications in image processing, arithmetic logic unit and VLSI signal processing.

Rakshith Saligram et.al [10] aimed to enhance the performance of pervious reversible logic designs. The Total Reversible Logic Implementation Cost (TRLIC) defined as sum of all cost metrics, was used as an aid to evaluate the proposed design. Urdhav tiryakbhayam sutra was used for designing multiplier. The efficiency of a reversible logic circuit is characterized in terms of parameters such as quantum cost, number of constant inputs, garbage outputs and number of gates utilized to realize the logic implementation. Lower the value of these parameters more efficient is the design. The quantum cost is the parameter that directly reflects the delay of quantum circuit. The proposed optimized design exhibited by lower TRLIC, minimum gate count, better number of constant inputs as well as minimum garbage outputs as compared to the pervious reversible logic designs. Lower TRLIC implicitly means lower quantum cost, hence lower delay and vice versa. The design was verified using MODELSIM .This multiplier can be efficiently adapted in designing Fast Fourier Transforms (FFTs) Filters and other applications of DSP like imaging, software defined radios, wireless communications.

Kavita et.al [11] designed a vedic multiplier implemented on FPGA using Gunakasamuchayah Sutra. The implemented design is more efficient and fast as compared with other multipliers and the look up tables required to implement the multiplier is also less when compared with other multipliers.

Surabhi Jain et.al [12] implemented an optimized binary division architecture for calculating deconvolution using Nikhilam and Parvartya sutras. The simulation results showed that time delay of vedic divider architecture is reduced by approximately 19% than the conventional method. They also introduced a straightforward approach for performing deconvolution. The application of vedic division in this approach of deconvolution has less delay of 31% than the conventional method. The algorithm was coded in Verilog, synthesized and simulated using Xilinx ISE.

Below is the table for surveys of different multiplier design using Vedic Mathematics.

Sr. No	**Title of Paper**	**Publisher and Year**	**Method/Sutra from Veda**	**Features**	**Language or Tool**
1	Multiplier Design Based on ancient Indian Vedic Mathematics	IEEE-2008	Urdhva Tiryakbhayam , Nikhilam sutra	Faster multiplier and square architecture, delay and design area less	ALTERA Cyclone –II FPGA
2	High Speed Energy Efficient ALU Design using Vedic Multiplication techniques	IEEE-2009	Urdhva Tiryakbhayam	Parallel generation of intermediate product, delay and less design area	-
3	Vedic Mathematics Based Multiply Accumulate Unit	IEEE-2011	Urdhva Tiryakbhayam	Binary number multiplication, Realized easily on silicon due to regular and parallel structure	VHDL and Xilinx ISE
4	Design a DSP Operation using Vedic Mathematics	IEEE-2013	Urdhva Tiryakbhayam	Vedic mathematics based DSP requires less processing time than inbuilt MATLAB function, Gives better results	MATLAB
5	Novel Architecture for Inverse Mix Column for AES using Vedic Multiplication on FPGA	IEEE-2013	Advance Encryption Standard (AES), Urdhva Tiryakbhayam	High Speed and Low on chip area	FPGA

6	A New Paradigm In Fast BC D Division Using Ancient Indian Vedic Mathematics Sutras	ICCSEA-2013	Nikhilam , Parvartya sutra	The computation time required by the Vedic Division Algorithm is approximately constant irrespective of size of the dividend	-
7	FPGA Implementation of High Speed 8-bit Vedic multiplier using barrel shifter	IEEE 2013	Nikhilam Sutra	Barrel shifter were used to reduce the delay , improvement in speed	FPG, ISE
8	Optimized Reversible Vedic Multiplier for High Speed Low Power Operations	IEEE 2013	Urdhav tiryakbhayam sutra	Decreasing Total Reversible Logic Implementation Cost (TRLIC) , delay was reduced, Useful in applications like wireless communication, image processing and software defined radios	MODELSIM
9	FPGA Implementation of Vedic Multiplier	IJARET 2013	Gunakasamuchayah Sutra	Reduced delay, low power, improvement in speed, area efficient, reduced design complexity, increased modularity	Xilinx
10	Binary Division Algorithm and High speed Deconvolution Algorithm (based on Ancient Indian Vedic Mathematics)	IEEE-2014	Nikhilam, Parvartya sutra	Applied for calculating deconvolution, reduced time delay and complexity.	Verilog and Xilinx ISE

All the surveyed designs were aimed for reducing computation time, design area, power consumption and complexity. The quality factors associated to the outputs of the surveyed designs are very limited. The quality factor “Efficiency” can be linked to the above stated survey outputs. Another important Quality factor “Performance” is defined as “The degree to which a system or component accomplishes its designated functions within given constraints”. Performance usually manifests itself in the three measures: Throughput, response time and Deadlines. Quality factor “Performance”, not included in McCall’s quality factors model, can also be associated.

CONCLUSION:

This paper reviews several Vedic sutras that are implemented in different designs of multipliers for reducing computation time, design area, power consumption and complexity. Use of sutras are useful in different applications like digital signal processing, image processing and computation of heavy calculations. The quality factors “Efficiency” and “Performance” were found associated to the surveyed design outputs.

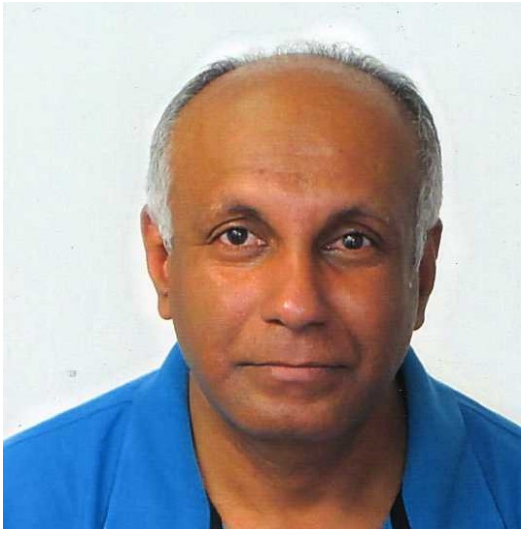
ACKNOWLEDGMENT:

The author would like to thank to the below stated individuals for their contribution of this work:
Swati Dave of "The Institute for the Advancement of Vedic Mathematics" (IAVM), UK.
Prof. Amos Notea , Holon Institute of Technology Israel, Faculty of Technology Management- Head of Quality Assurance and Reliability Program.
Dr. Malki Grossman, Technion – Israel Institute of Technology.

REFERENCES:

1. Vedic Mathematics, Bharati Krsna Tirthaji, Jagadguru Sankaracharya of Puri. Motilal Banarsidass Publishers Private Limited.
2. Software Quality Assurance – From theory to implementation, Daniel Gain. Pearson Education Limited.
3. Honey Durga Tiwari, Ganzorig Gankhuyag, Chan Mo Kim, Yong Beom Cho, "Multiplier design based on ancient Indian Vedic Mathematics", 978-1-4244-2599-0/08/ ©2008 IEEE.
4. M. Ramalatha, K. Deena Dayalan, P. Dharani, S. Deborah Priya, "High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques", 978-1-4244-3834-1/09/ © 2009 IEEE.
5. Devika Jaina, Kabiraj Sethi and Rutuparna Panda, "Vedic Mathematics Based Multiply Accumulate Unit", 978-0-7695-4587-5/11 © 2011 IEEE.
6. Akhalesh K. Itawadiya, Rajesh Mahle, Vivek Patel, Dadan Kumar, "Design a DSP Operations using Vedic Mathematics",978-1-4673-4866-9/13/ © 2013 IEEE.
7. Sushma R Huddar, Sudhir Rao Rupanagudi, Ramya Ravi, Shikha Yadav & Sanjay Jain, "Novel Architecture for Inverse Mix Columns for AES using Ancient Vedic Mathematics on FPGA", 978-1-4673-6217-7/13/ 2013 IEEE.
8. Diganata Sengupta, Mahamuda Sultana, Atal Chaudhuri, "A New Paradigm In Fast BCD Division Using Ancient Indian Vedic Mathematics Sutras", David C. Wyld (Eds): ICCSEA, SPPR, CSIA, WimoA-2013.
9. Pavan Kumar, A Radhika, "FPGA Implementation of high speed 8-bit Vedic multiplier using barrel shifter", 978-1-4673-6150-7/13/$31.00 ©2013 IEEE.
10. Rakshith Saligram, Rakshith T.R, "Optimized Reversible Vedic Multiplier for High Speed Low Power Operations" Proceedings of 2013 IEEE Conference on ICT 2013.
11. Kavita, Umesh Goyal, "FPGA Implementation of Vedic Multiplier", International Journal of Advanced Research in Engineering and Technology (IJARET), Vol 4, Issue 4, May-June 2013, pp. 150-158.
12. Surabhi Jain, Mukul Pancholi, Harsh Garg, Sandeep Saini, "Binary Division Algorithm and High speed Deconvolution Algorithm (based on Ancient Indian Vedic Mathematics)" , 978-1-4799-2993-1/14/©2014 IEEE.

ABOUT THE AUTHOR:



Raymond Austin is a quality assurance engineer at Medical Electronic Systems, Israel. He earned a bachelor's degree in Electrical and Electronics Engineering from Coventry University (UK) Programme in Ruppin Academic Centre, Israel and in Quality & Reliability Engineering from Kinneret College on the Sea of Galilee, Israel. Raymond is an American Society for Quality (ASQ) and Israel Society for Quality (ISQ) certified quality engineer. He is also a certified Vedic Math trainer and a member of "The Institute for the Advancement of Vedic Mathematics" (IAVM), UK. IAVM is a charity established to promote, disseminate, research and support the system of Vedic Mathematics internationally.