Sādhanā (2022)47:74
https://doi.org/10.1007/s12046-022-01843-0

Comparative analysis of Yavadunam Tavadunikrtya Varganca Yojayet Vedic multiplier for embedded DNN

P SANTHOSH KUMAR[1,*] and C GOWRISHANKAR[2]

[1]Department of ECE, Sasurie College of Engineering, Tirupur, India
[2]KSR College of Engineering, Tamil Nadu, Namakkal, India
e-mail: santhosherode1@gmail.com

MS received 12 November 2021; revised 4 February 2022; accepted 6 February 2022

Abstract. Memory and computationally efficient CNNs for mobile and embedded applications have sparked a lot of interest recently. Depth-wise and point-wise convolutions are used in MobileNet. In this paper, a Vedic multiplier based on the Yavadunam sutra is used to reduce hardware resource utilization at all design stages in efficient computational kernels, data pruning, memory compression, and quantization to manage the huge computation and storage difficulties of DNNs. Using the teachings of Vedic math, this multiplier will yield a higher propagation speed, lower computational complexity, reduced propagation delay, and less power consumption than multipliers based on the principles of classical mathematics. A multiplier is a critical component in Digital Signal Processing DSP frameworks, which are key components in the majority of computationally advanced frameworks. As a result, the speed and power utilization of multipliers are two critical parameters. The Vedic multiplier presented in this paper is commonly used to find decimal number squares and cubes. By modifying the existing sutra for binary numbers with the bit reduction technique, the hardware design of the Yavadunam Vedic sutra is proposed. The proposed Vedic multiplier was written in VHDL, which stands for Very High Speed Integrated Circuits Hardware Description Language. On the Spartan FPGA board, the multiplier was implemented using the Xilinx Tool and the system generator instrument. The computational complexity and propagation delay of the Vedic multiplier were lower than those of a regular multiplier.

Keywords. Vedic multiplier; Yavadunam sutra; multiplication; bit reduction technique; Vedic mathematics; binary multiplier.

1. Introduction

Vedic arithmetic is a procedure for performing tedious arithmetic computations in a simple manner. This method was developed in historic India. Vedic mathematics includes unique techniques that are primarily based on 16 sutras or formulae related to mathematics, especially algebra, and geometry. These sutras may be applied to trigonometry, plain and spherical geometry, conics, calculus (differential and critical), and various types of arithmetic. Vedic arithmetic transforms complex calculations into relatively simple ones. Vedic arithmetic is suitable for achieving a low area and computational delay. Consequently, it can be applied to different branches of engineering, including computing and signal processing.

Urdhava Tiryakbhyam is a Sanskrit term that translates as vertically and crosswise. The Urdhava Tiryakbhyam method is suitable for the multiplication of any numbers. The novelty of this approach lies in the generation of partial merchandise. In the aforementioned technique, partial products are generated concurrently through combinational logic gates. The Urdhava Tiryakbhyam method is specifically used in microprocessors to enable functioning at high clock frequencies. An increase in the switching frequency also causes an increase in the switching times. The increase in the switching time enhances the strength and heat dissipation within the system, which results in decreased operating temperatures.

. Another advantage of the Urdhava Tiryakbhyam multiplier is its scalability. The processing electricity can be easily increased by increasing the input and output data bus widths because the aforementioned multiplier has irregular structure. The first author of this paper [1] implemented Anurupyena Vedic multiplier for machine learning applications. In machine learning, DNN plays a vital role. The multipliers and adders are the main components in DNN [2]. Due to its regular structure, the aforementioned multiplier can be easily implemented in VLSI technology with an optimal area. With the increase in the range of enter bits, the gate delay and location of the Urdhava Tiryakbhyam increase very slowly compared with those of other

*For correspondence
Published online: 08 April 2022

multipliers. Therefore, the Urdhava Tiryakbhyam multiplier is time, area, and power efficient.

The author of [3] implemented the Nikhilam sutra multiplier by modifying the 2's complement block and by using encoding techniques in the design. The authors of [4] proposed an iterative multiplication algorithm by using the Nikhilam sutra to remove 0s from the least significant positions and perform bit reduction. The authors of [5] compared the delay and power of the Urdhava and Nikhilam Vedic multipliers with those of an array multiplier. In [6], the authors designed floating-point multipliers by using the divide and conquer method and the pipelining technique.

In [7–11], bit reduction principle is used in Vedic multipliers. Nikhilam and Karatsuba Vedic sutra were modified to bring the generalized Nikhilam Vedic multiplier. They used N-1 bit and N-2 bit multipliers for N-bit multiplication. The method is effective for higher order bits and not much different in lower order bits.

This research focuses on the design and implementation of a high-speed Vedic multiplier by using the Yavadunam sutra based on weight reduction in remainder calculation. The strength of the multiplication is reduced by calculating remainders with the bit reduction technique. The remainder is calculated such that a value with N-1 bits, instead of N bits, is obtained. Moreover, the number of bits for the remainder is maintained constant for any range of inputs. The conventional Yavadunam sutra is designed for determining the square of a decimal number. In this study, the aforementioned sutra was modified marginally to obtain the product of two binary numbers through hardware implementation. The limitation of the Vedicsutra lies in the usage of specific ranges of numbers. However, the modified sutra can produce a result for any input bit numbers. The algorithm based on the Yavadunam sutra depends on the MSB of the multiplicands. This study aimed to increase the speed and reduce the design complexity of multiplier algorithms. Weight reduction was performed in the remainder calculation to reduce the design complexity. Therefore, for remainder multiplication, only the (N–1) bit multiplier is required. (NxN)-bit multiplication is performed through (N-1) x (N-1) bit multiplication.

In Rasoulinezhad *et al* [12] adapted the concept to the Xilinx DSP48E2 27 × 18 multiplier, which generates two partial results (the subsequent ALU is in charge of summing these two outputs). A and B are chopped into I = 3 and j = 2, 9-bit portions to make a 27 × 18C32D2 configuration. Because each smaller multiplication is a signed/unsigned 9-bit multiplication, recursive decomposition with depth k = 2 is utilized to alter the 9 × 9 signed/unsigned multiplier to accommodate two 4 × 4 or four 2 × 2 multiplications in addition. Extra bits are included is shown in figure 1 to ensure that precision is maintained. The bit-level carry propagation is organized from one column to the next. Multi-precision MAC procedures can be computed without accuracy loss by combining the six 9 × 9 multipliers used in the architecture of the proposed Yavadunam multiplier.

2. Proposed Yavadunam sutra for multiplication

The conventional Yavadunam sutra comprises the following steps to determine the square of a number:

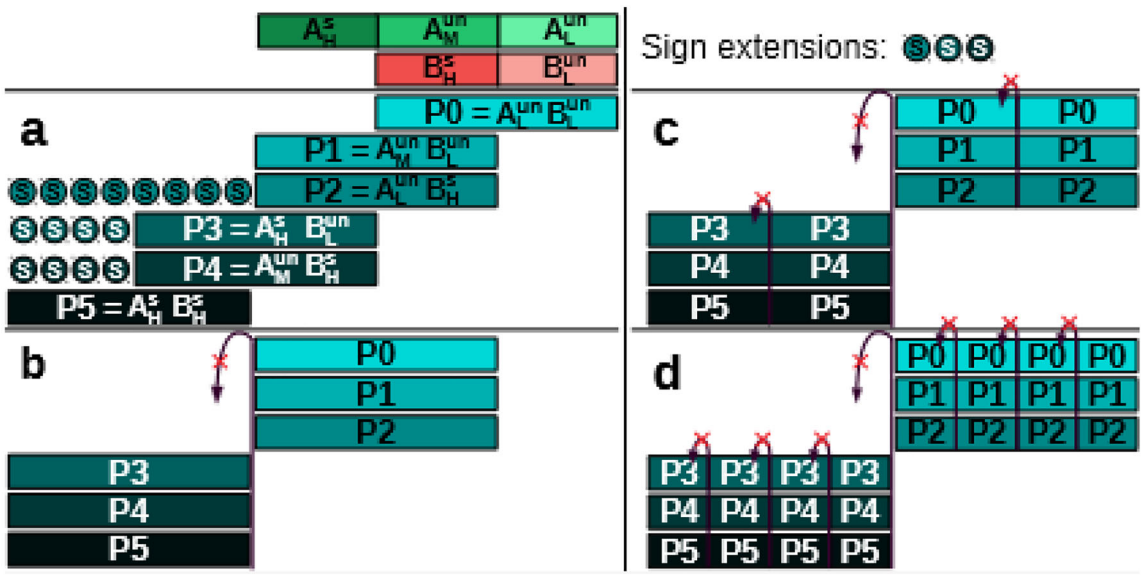

Figure 1. Multiplier based on high level presentation of chopping (a and b), and divide and conquer techniques (c and d) used in [12].

1. Consider the nearest power of 10 as the base.
2. Find the deficiency, by subtracting the Least Significant Digit from the base.
3. Subtract the deficiency from the given number.
4. Setup as many 0s as possible as the base.
5. Square the deficiency, and include the answer as a right part.

Consider the number 96. The nearest base is 100, and the deficiency is 100–96 = 4. The right part is derived by obtaining the square of the deficiency (i.e., 16). The left part is calculated by subtracting the deficiency from the given number (96–4 = 92). The final product is derived by concatenating the left and right parts (96^2 = 9216).

This sutra is modified to derive the product of two binary numbers. The steps in the proposed multiplier are as follows:

Step 1 Calculate the remainders (or deficiencies) Ar and Br of the inputs A and B, respectively, by using the bit reduction technique.

$$\begin{aligned} \mathrm{Ar} &= \mathrm{A_{N-2}A_{N-3}}\ldots\mathrm{A_0} \quad \text{if} \quad \mathrm{A_{N-1}} \\ &= 1(\text{for a positive remainder}) \\ &= \text{Complement}\ (\mathrm{A_{N-2}A_{N-3}}\ldots\mathrm{A_0})\ \text{if}\ \mathrm{A_{N-1}} \\ &= 0\ (\text{for a negative remainder}) \end{aligned}$$

Step 2 Determine the product of Ar and Br and consider the least significant N -2 bits as the right-hand side (RHS) of the product.

Step 3 Add or subtract the remainders with the inputs according to the type of remainders.

Step 4 Add or subtract the value derived in Step 3.

Step 5 For a positive remainder, the result determined in Step 4 is added with the bits from the RHS. For a negative remainder, the bits from the RHS are subtracted from the result obtained in Step 4.

The proposed multiplier is explained with the following example. Consider A= 1001 and B= 0010. The first number has a positive remainder because the MSB is 1. Therefore, the remainder of A is 001. The second number has a negative remainder because the MSB is 0. Therefore, the remainder of the second number is calculated by taking complement of the number (i.e., 110; Ar= 001 and Br= 110). The product of these remainders is 000110. The least significant N- 2 bits (10) are considered, and the complement is taken because the product is negative. To derive the left part of the final product, A+Ar is calculated as 1010 (A has a positive remainder) and B Br is derived as 0100 (B has a negative remainder). The left part is calculated by adding (A+Ar) or subtracting (B–Br) according to the control logic. For the considered example, the result is 0100. By concatenating the left part and right part, the final result is derived (00010010). The sample calculation is given in table 1.

Table 1. Calculation based on the proposed method.

				S1		S2		S3						
A	B	Ar	Br	A+Ar	A-Ar	B+Br	B-Br	S1+S2	S1-S2	Sg	S4	ArBr	LHS of the product	RHS of the product
15=1111	3=0011	0111	0101	10110	–	–	00010	–	10100	1	1001	100011	001011	01
9=1001	13=1101	001	101	1010	–	10010	–	11100	–	0	0001	000101	011101	01
7=0111	14=1110	001	110	–	00110	10100	–	11010	–	1	0010	000110	011000	10
31=11111	15=01111	1111	0001	101110	–	–	001110	111100	–	1	00010	00001111	0111010	001

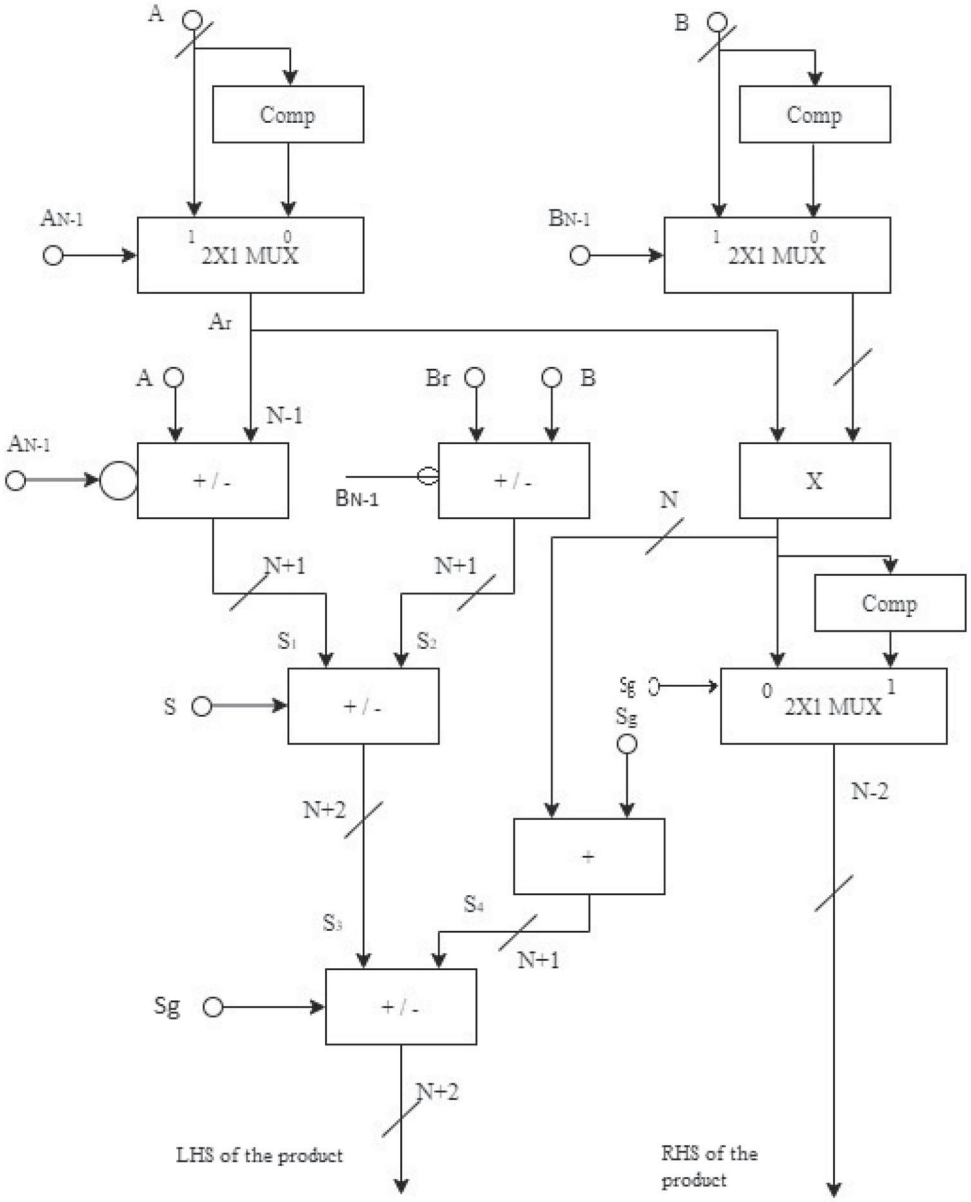

Figure 2. Architecture of the proposed Yavadunam multiplier.

The architecture of the proposed multiplier is shown in figure 2. Input multiplexers are used to determine the remainders of inputs A and B. The obtained value is left unchanged for a positive remainder; however, for a negative remainder, the complement is considered. According to the control input, addition or subtraction of the remainders and inputs is performed. S1=A+Ar when Ar is positive otherwise S1=A-Ar. In a similar way, S2 is calculated based on Input B. Sg is 1 when either Ar or Br is negative otherwise Sg=0. The addition or subtraction of S1 and S2 is decided by the control input S. S=0 when first two MSB bits of inputs A or B are not zero. The control input S can be written $S = \overline{A_{N-1}A_{N-2}} + \overline{B_{N-1}B_{N-2}}$.

3. Results

Different bit values were considered for comparing the proposed multiplier with conventional methods. The proposed method and other methods were functionally verified using the Mentor Graphics tool. The VHDL codes were implemented in Xilinx Spartan 3e and a Virtex FPGA. The Mentor Graphics tool was also used to perform power analysis. The array multiplier is widely used due to its linear structure. It has identical cells that simultaneously generate and accumulate partial products. Pipelining can be performed easily at each level. In the aforementioned multiplier, the delay is logarithmically proportional to the

Table 2. Comparison of various multipliers.

Bit size	Type	Area (LEs)	Delay (nS)	Latency (Cycles)	Power (mW)
16	Parallel	319	6.8	1	0.94
	Pipelined	368	3.2	4	3.49
	Booth SP	81	2.72	9	1.67
	Two Speed	87	1.52	14	4.35
	Vedic Multiplier [1]	96	3.4	6	2.8
	Proposed	89	3.3	6	2.7
32	Parallel	1255	10.2	1	1.33
	Pipelined	1232	4.6	4	5.07
	Booth SP	156	3.8	17	1.78
	Two Speed	159	1.76	25.6	3.18
	Vedic Multiplier [1]	174	4.3	15	3.4
	Proposed	158	2.8	12	2.5
64	Parallel	5104	14.7	1	2.23
	Pipelined	4695	6.99	4	9.62
	Booth SP	292	3.9	33	2.23
	Two Speed	304	1.83	45.2	5.2
	Vedic Multiplier [1]	321	4.8	29	4.6
	Proposed	300	3.5	26	3.1

bit size of the multiplicands. However, the multiplier requires a large number of gates. The array multiplier is efficient for a low number of bits. The proposed multiplier is compared with four conventional multipliers as shown in table 2.

The computational delay for the compared multipliers is listed in table 2. The proposed Vedic multiplier has resulted in the optimal area and delay as compared to other multipliers used for Embedded DNNs. The proposed Vedic multiplier is derived from the Vedic sutra which is used for squaring the numbers in an easier way. Vedic multipliers can be used for high-speed applications. The proposed multiplier uses simple components like multiplexer and adder/subtractor. The multiplexer is used to derive the derivatives (remainders) of the multiplicand and the multiplier. The remainder may be positive or negative. According to the sign of the remainders, the addition or subtraction operations are carried out.

4. Conclusion

This paper has described a Vedic multiplier based on the Yavadunamsutra. The binary multiplier's structure is changed and the bit reduction approach is used to calculate the remainder. Thus the number of bits required to the proposed multiplier is lowered to N-1. As a result, the propagation delay and computation time are significantly reduced. The modified Vedic multiplier can be used to improve computation speed and reduce implementation. The updated multiplier's performance is solely dependent on addition and subtraction operations. A Xilinx Spartan3e kit was used to implement the proposed algorithm. The area and delay were optimized among others. This algorithm works well in high-speed applications with a large number of bits. In the future, this multiplier can be employed in PIR-DSP [12] for comparative analysis.

References

[1] Santhosh Kumar P and Gowrishankar C 2020 Design and investigation of low-complexity Anurupyena Vedic multiplier for machine learning applications. *Sadhana. Proc. Indian Acad. Sci.* 45: 272. https://doi.org/10.1007/s12046-020-01500-4

[2] Choudhary R 2014 FPGA based multiplication using MUX and Vedic multiplier. Thesis. Thapar University. Patiala, Punjab

[3] Nivas S and Kayalvizhi N 2012 Implementation of power efficient Vedic multiplier. *Int. J. Comput. Appl.* 43(16): 21–24. https://doi.org/10.5120/6188-8673

[4] Vedavathi K and Lakshmi T 2012 An Iterative binary multiplication algorithm using Nikhilam Sutra. *Int. J. Curr. Res.* 4(10): 187–196

[5] Kumar H and Kumar Ch 2013 Implementation and analysis of power, area, delay of Array, Urdhava, Nikhilam Vedic multipliers. *Int. J. Sci. Res. Publ.* 3(1): 1–5

[6] Singla V 2013 *Design and Implementation of single precision floating point multiplier using divide and conquer algorithm*. Patiala, Punjab

[7] Manikandan S K and Palanisamy C 2016 Design of an efficient binary Vedic multiplier for high speed applications

using Vedic mathematics with bit reduction techniques. *Circuits Syst.* 07: 2593–2602

[8] Manikandan S K and Palanisamy C 2016 Design of high speed vedic multiplier based on nikhilam sutra algorithm using successive approximation. *Asian J. Res. Soc. Sci. Hum.* 6: 608–608

[9] Nisha Angeline M and Valarmathy R S 2016 Implementation of hybrid Vedic multiplier using Nikhilam sutra and Karatsuba algorithm for N-bit multiplier using successive approximation of N-1 bit multiplier. *Asian J. Inf. Technol.* 15: 3598–3604

[10] Nisha Angeline M and Valarmathy R S 2016 Implementation of N-bit Binary multiplication using N-1 bit multiplication based on Nikhilam Sutra principles and bit reduction. *Transylvanian Rev.* 7: 982–992

[11] Nisha Angeline M and Valarmathy R S 2016 Implementation of N-bit binary multiplication using N-1 bit multiplication based on Nikhilam sutra and Karatsuba principles using complement method. *Circuits Syst.* 7: 2332–2338

[12] Rasoulinezhad S, Zhou H, Wang L and Leong P H W 2019 PIR-DSP: An FPGA DSP Block Architecture for Multi-precision Deep Neural Networks. In: *IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM):* 35-44, https://doi.org/10.1109/FCCM.2019.00015