Sādhanā (2020)45:272
https://doi.org/10.1007/s12046-020-01500-4

Design and investigation of low-complexity Anurupyena Vedic multiplier for machine learning applications

SANTHOSH KUMAR PARAMESWARAN[1,*] and GOWRISHANKAR CHINNUSAMY[2]

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

MS received 7 May 2020; revised 19 August 2020; accepted 16 September 2020

Abstract. The current world of computers is based on machine leaning and profound learning towards artificial intelligence. In recent investigations, parallelisms are used to solve difficult problems. For the implementation of the FPGA, new architectures have been built using design techniques VLSI and parallel computing technologies. Research on reconfigurable computing, machine learning and signal processing should be constantly monitored in the development of artificial intelligence. Energy-restricted computer devices should be continuously developed to support algorithms in the machine learning process. In machine learning algorithms, multipliers and adders play a significant role. In ALU, Convolutionary Neural Network (CNN) and Deep Neural Networks (DNN), the multiplier is an energy-consuming factor of signal processing. In this project, for the DNN, the high-speed Vedic multiplier has been introduced. The versions of the parallel–parallel (PP), serial–parallel (SP) and two-speed (TS) multipliers are compared to the standard 64-, 32- and 16-bit models. The results are obtained for an Intel Cyclone V 5CSEMA5U23C6 FPGA, using the Intel Quartus 17.0 software suite.

Keywords. Binary multiplication; Vedic mathematics; Nikhilam Navatashcaramam Dashatah; Urdhava Triyagbhyam; Anurupyena sutra.

1. Introduction

A technique that gives a simple solution to complex problems is Vedic mathematics. In [1], Moss has introduced a serial–parallel two–speed radix-4 multiplier to speed up applications such as digital filters, artificial neural networks and other algorithms for machine learning. For zero and non-zero recoveries, a two-speed multiplier is used with different critical paths. In [2] the author deals with AddNet, which uses shifted values at k levels in neural networks as multipliers for weight modification. In the case of a neural network architecture and ternary weight data, this paper [3] shows that the extreme performance implementation of the neural network inference can be achieved by customizing the data path and routing to eliminate unnecessary computations and data movement.

2. Existing Anurupyena algorithm

Anurupyena is a Nikhilam sutra's sub-sutra. The shortcut methods for multiplying numbers are provided [4]. A new Vedic hybrid multiplier combining Karatsuba and Urdhava Tiryagbhyam algorithms has been developed in [5]. A new algorithm based on Nikhilam sutra was suggested with the bit reduction technique in [5–7]. A generic architecture was generally proposed for the Nikhilam sutra [8, 9]; the Anurupyena method is used only when the multiplicands are in the same range in mental calculations.

3. Proposed Anurupyena algorithm

The Anurupyena algorithm is modified in this analysis, so that decimal numbers can be multiplied in any range even if they are not near the basic value. The amendment was also made to incorporate the hardware of the Anurupyena algorithm.

X and Y are two numbers defined by the base numbers. B is the base number, X_d is shortfall of X and Y is shortfall of Y. The deficiencies (shortfall) can be positive or negative. If the number is greater than the base value the shortfall is positive, while the shortfall is negative if the number is less than the basic value. For a 4-bit combination, except for "10," the deficiency is negative. This combination is presented with examples in table 1.

The provided N-bit numbers are divided according to their fundamental value into four groups. If the number starts with "00" it is in the first group, and the common base value k is 01. If the number begins with "01" or "10"

*For correspondence
Published online: 03 November 2020

Table 1. Calculation using proposed Anurupyena sutra for binary numbers.

X	Y	k	d	X_d	Y_d	LHS (I level) Cross add	RHS (I level) X_dY_d	LHS (II level) LHS×k	RHS (II level)	LHS (II level)	RHS (II level)	Product (P)
15 = 16 − 1	3 = 4 − 1	1	3	− 01	− 01	1111 − 01= 1110	0001	1110	0001	1110 + (− 01×11) = 1011 + 00 =1011	01	101101
1111	0011											
9 = 8 + 1=	13= 16 − 3= 1101	2	2	01	− 11	1101 + 01= 1110	− 11	11101	01	11101 + (01×10) = 11110 − 01= 11101	01	11110101
1001												
7 = 8 − 1	14 = 16 − 2	2	2	− 01	− 10	1110 − 01= 1101	0010	11011	0010	11011 + (− 01×10)= 11000	10	1100010
0111	1110											
31 = 32 − 1	15 = 16 − 1	2	2	− 01	− 01	11111 − 01= 11110	000001	111100	000001	111100 + (− 01×10)= 111010 + 000	001	111010001
11111	01111											

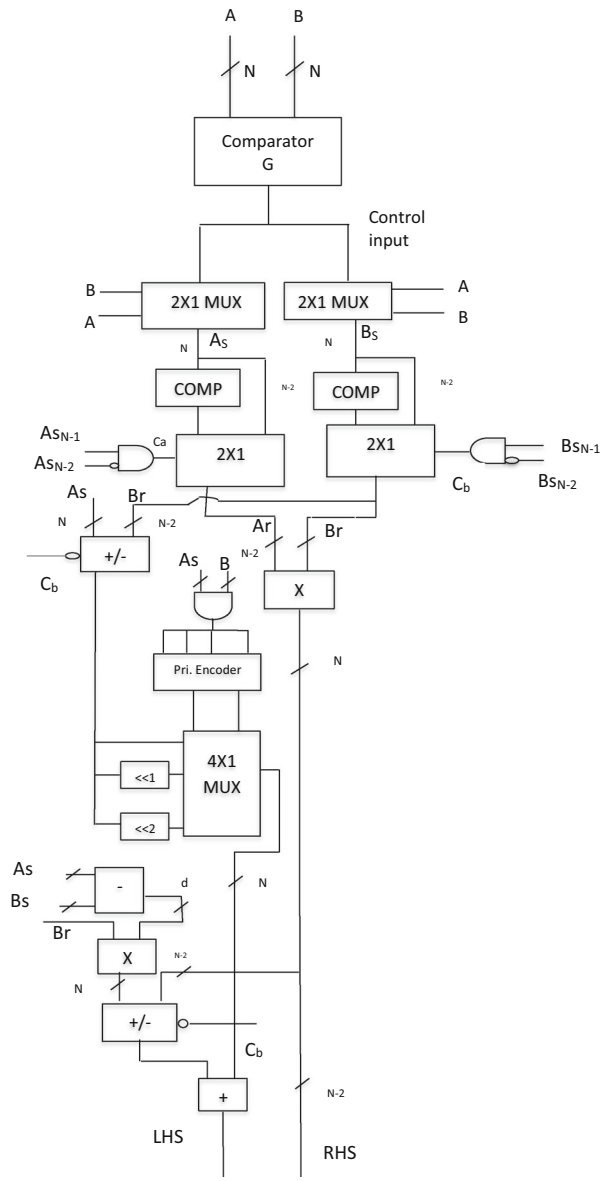

Figure 1. Architecture of the proposed Anurupyena Vedic multiplier.

it belongs to the second group, and the k value for this group is 2 (10). The common base value is 3 (11) if the number starts with "11." The other steps are the same as those followed in the decimal calculation.

In the first example set out in Table 1, the first two MSBs are "11" and "00," and the deficiency is negative for both numbers. The difference is 3 (11–00) between the base values d. The common base value k is calculated by the group in which the smallest number of the MSB is located. Mainly the first two bits are considered for the calculation of the deficiency and the common base value.

The hardware implementation of the proposed Vedic multiplier is shown in Figure 1. Originally, a comparator is used for modifying the numbers to refer to the larger

Table 2. Implementation results of 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
	Proposed	96	3.4	6	2.8
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
	Proposed	174	4.3	15	3.4
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
	Proposed	321	4.8	29	4.6

number as A and the smaller number as B. In the second stage, the deficiency according to the MSB values is found by means of a 2–1 multiplexer. Due to the rise in numerical deficits, the RHS of the commodity is extracted. In figure 1, the multiplier x receives a number of two $(N-2)$ bits. Crossing A and B deficiency was used to measure the LHS of the drug initially. The sum obtained will then be multiplied by k (both numbers widely used). A priority encoder is used to pick the multiplicand for multiplication with k. Once k has been multiplied, the partial product is applied to the B and d product (difference in the base value range) or is subtracted from it. In addition, for hardware implementation an algorithm must operate correctly with any number of inputs. This method can be substituted with shifting action. Thus both limitations of the existing Anurupyena algorithm are overcome with the proposed method.

4. Results and discussion

This section presents the results of the Anurupyena Vedic multiplier being proposed for implementation; the area and delay of different multiplier instantiations are given.

The program performs static timing analysis over four separate PVT corners during position and path while keeping voltage steady; specifically: (1) 1100 mV 0C rapid; (2) 85C quick 1100 mV; (3) 1100 mV 0C slow and (4) 1100 mV 85C slow. The proposed multiplier is "placed and routed" using the methodology based on timing constraints, and all recorded frequencies for each multiplier reflect the upper limit for each one considered as a standalone element. However, the proposed Vedic multiplier is designed to take advantage.

Table 2 compares different multiplier designs with six major factors: Area, Time, Power, Area Time, Time Power and Area Time Power, often dictating the most appropriate speed application. Trade-offs are usually evaluated and the version with the best value is selected. TSM or Anurupyena is the best option for the Area, since it has the lowest footprint. To the best of our knowledge there are only three recent publications in the field of multiplier optimization of FPGA microarchitecture, aimed at Booth algorithm SP computation. All these works were installed on 90-nm FPGAs, making a direct comparison difficult as they were not only slower and consumed higher power but also due to technology their architecture was also different, e.g. they used four-input lookup tables, and the performance of the larger multipliers, including 32 and 64 bits, was not recorded. Hence a fair comparison is impossible but the reported results are listed at the bottom of Table 2, and we note that the TSM improved Area Time and Area Time Power by an order of magnitude for the 16- and 32-bit cases. Both the parallel (combinatorial) and parallel (pipeline) multipliers are taken from libraries that implement the latest optimizations for multipliers and serve as a good comparison between our work and industry standards.

5. Conclusion

In this paper, we presented Anurupyema Vedic sutra's generic architecture, which is modified to be used for binary numbers in any range. In real time, this multiplier's efficiency can be improved only on bit representation distribution. We showed that traditional compute sets, such as uniform and Gaussian and neural networks, can expect significant improvements of 3 and 3.56 using normal learning and sparse techniques, respectively. Costs associated with handling lower-bit-width representations, such as Gaussian-8 on a 64-bit multiplier, are mitigated and display a gain of 3.64 compared with the standard parallel multiplier.

References

[1] Moss D J M, Boland D and Leong P H W 2019 A two-speed, Radix-4, serial–parallel multiplier. *IEEE Trans. Very Large Scale Integr. (VLSI) Systems* 27: 769–777

[2] Krizhevsky A, Sutskever I and Hinton G E 2012 ImageNet classification with deep convolutional neural networks. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*, pp. 1097–1105

[3] Lecun Y, Bottou L, Bengio L and Haffner P 1998 Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 2278–2324

[4] Han S, Pool J, Tran J and Dally D J 2015 Learning both weights and connections for efficient neural networks. In: *Proceedings of the. 28th International Conference on Neural Information Processing Systems*, pp. 1135–1143

[5] 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

[6] 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

[7] 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

[8] 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

[9] 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. *Trans. Rev.* 7: 982–992