AN EFFECTIVE ALGORITHM IMPLEMENTATION FOR IMPROVING BINARY COUNTER USING ITERATIVE SORTING METHODS

[1]A. Venkat Reddy, [2]M. Kranthi Kumar, [3]T. Lokesh Mudiraj
[1,2]Assistant Professor, [3]Student, [1,2,3]Dept. Of Electronics and Communication Engineering.
Vaageswari College of Engineering, Karimnagar, Telangana.
E-Mail: Venkat7641h@gmail.com

ABSTRACT
Many number juggling circuits use parallel counters as inputs, especially rapid multipliers. In various digital signal processing devices, the critical route includes the parallel summing of multiple operands. High compression ratio counters and compressors are required to accelerate the summing. In this article, we offer a unique sorting network-based technique for fast saturated binary counters and exact/approximate (4:2) compressors. In order to create reordered sequences that can only be represented by one-hot code sequences, the counter's inputs are asymmetrically split into two groups and fed into sorting networks. Three unique Boolean equations are constructed between the reordered sequence and the one-hot code sequence, which can greatly simplify the output Boolean expressions of the counter. Further, this project is enhanced by using parallel sorting algorithms for finding/ sorting M largest values from N inputs and then design scalable architectures based on proposed algorithms. For finding the largest values the iterative sorting techniques also proposed.Bitonic Sorting Is one type of efficient such algorithm for implementing with optimised parameters.
Keywords: Approximate (4:2) compressors, Sorting, asymmetric, Digital signal processing, Arithmetic Logic Unit.

INTRODUCTION

Most electronic systems need to use as little energy as possible, especially portable ones like smart phones, tablets, and other devices. Achieving this minimization with the least amount of performance (speed) penalty is highly sought [1]. The most desired digital signal processing (DSP) building blocks for portable components are those that enable various multimedia applications. The ALU, which is at the centre of these blocks' computations, primarily performs ads and multiplications [6]. The primary operation in the processing elements is multiplication, which might result in significant energy and power consumption. Many DSP cores employ image and video processing algorithms, the results of which are either human-readable images or videos. It makes it easier to use estimates to increase speed and energy in the arithmetic circuits. This originates from the limited perceptual abilities in observing an image or a video for human beings. In addition to the image and video processing applications, there are other areas where the exactness of the arithmetic operations is not critical to the functionality of the system (see [2],[3]). Approximate computing provides an accuracy, speed and power/energy consumption. The advantage of approximate multiplier reduces the error rate and gain high speed. For correcting the division error compare operation and a memory look up is required for the each operand is required which increases the time delay for entire multiplication process [4]. At various level of abstraction including circuit, logic and architecture levels the approximation is processed [5]. In the category for approximation methods in function, a number of approximating arithmetic building blocks, such as adders and multipliers, at different design levels have been suggested in various structures [6],[7]. Broken array multiplier was designed for efficient VLSI implementation [8].

LITERATURE SURVEY

A traditional method to reduce the aging effects is overdesign which includes techniques like guard-banding ad gate oversizing. This approach can be area and power inefficient [8]. To avoid this problem, an NBTI-aware technology mapping technique was proposed in [7] which guarantee the performance of the circuit during its lifetime. Another technique was an NBTI- aware sleep transistor in [3] which improve the lifetime stability of the power gated circuits under considerations. A joint logic restructuring and pin reordering method in [6] is based on detecting functional symmetries and transistor stacking effects. This approach is an NBTI optimization method that considered path sensitization. Dynamic voltage scaling and bogy-biasing techniques were proposed in [4] and [5] to reduce power or extend circuit life. These techniques require circuit modification or do not provide optimization of specific circuits. Every gate in any VLSI circuit has its own delay which reduces the performance of the chip. Traditional circuits use critical path delays the overall circuit clock cycle in order to perform correctly. However, in many worst-case designs, the pMRSAbility that the critical path delay is activated is low. In such cases, the strategy of minimizing the worst-case conditions may lead to inefficient designs. For noncritical path, using the critical path delay as the overall cycle period will result in significant timing waste. Hence, the variable latency design was proposed to reduce the timing waste of traditional circuits. A short path activation function algorithm was proposed in [16] to improve the accuracy of the hold logic and to optimize the performance of the variable-latency circuit. An instruction scheduling algorithm was proposed in [17] to schedule the operations on nonuniform latency functional units and improve the performance of Very Long Instruction Word processors. In [18], variable-latency pipelined multiplier architecture with a Booth algorithm was proposed. In [19], process-variation tolerant architecture for arithmetic units was proposed, where the effect of process-variation is considered to increase the circuit yield. In addition, the critical paths are divided into two shorter paths that could be unequal and the clock cycle is set to the delay of the longer one. These research designs were able to reduce the timing waste of traditional circuits to improve performance, but they did not consider the aging effect and could not adjust themselves during the runtime.

SORTING

A sorting network is a collection of interconnected compare-and-exchange (CAE) block. Sorting networks consist of comparators, which are in fact hardware implementations of the CE operation. A comparator has two inputs and two outputs. Depending on the sense of ordering, comparators can be of two kinds as shown in figure1 (a), (b).

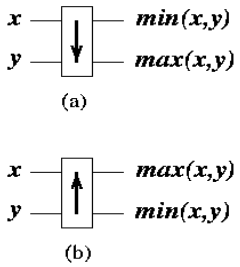

Fig 1: Basic Sorting Unit

Sorting is any process of arranging items according to a certain sequence or in different sets, and therefore, it has two common, yet distinct meanings:

PARALLEL SORTING NETWORKS

A sorting network is a collection of interconnected compare-and-exchange (CAE) blocks that guides a parallel set of inputs to a parallel set of outputs in sorted order. Each CAE block has two inputs and two outputs. If the input values are already in order, they are directed to the corresponding outputs; otherwise, they are swapped. There are two types of CAE blocks, called increasing and decreasing CAE blocks, used in hardware-based sorting

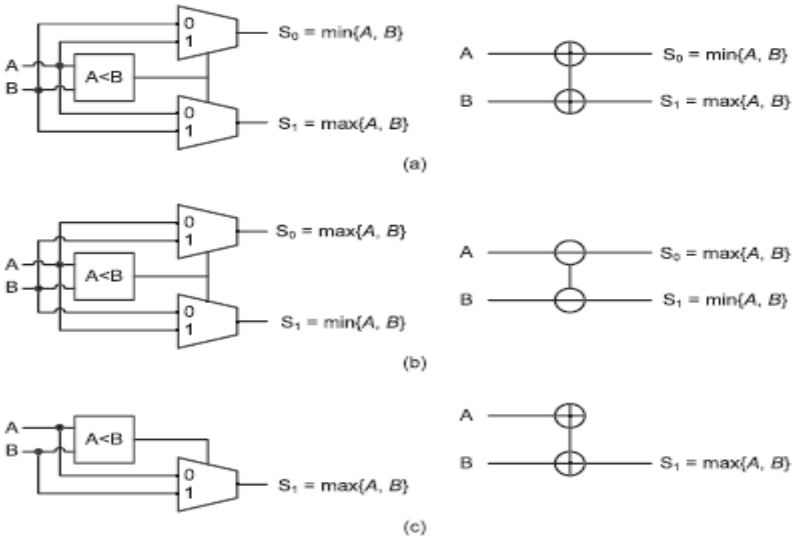

Fig: 2 parallel sorting units

units. Fig.2 shows the high-level implementations (left) and schematic symbols (right) for three building blocks used in previous sorting units and in our designs. A decreasing CAE block, shown in Fig. 2b, outputs its inputs in descending order. Decreasing and increasing CAE blocks are identical, except for their wiring. Each CAE block contains a comparator and two multiplexers. We also define Max units which are used in our designs. A Max unit, shown in Fig. 1c, takes two inputs and returns the larger input. Note that the _ and _ symbols determine the type of the block in Fig. 2. A sorting network usually consists of a series of stages in which each stage contains a number of CAE blocks that operate in parallel. The latency of a sorting network is proportional to its depth (the number of consecutive CAE blocks). Two popular parallel sorting networks that currently have the lowest known latency for hardware implementation are the bitonic and odd-even merge sorting networks proposed by Batcher [25]. The structure of a sorting network is fixed, regardless of the value of the numbers being sorted and the results of the comparisons. Sorting networks are a common solution for hardware based sorting. Their parallel nature allows them to perform sorting much faster than the OðN _ logðNÞÞ time achievable by the fastest sequential software-based sorting algorithms. A sorting network may also be pipelined to further increase throughput.

EXISTING TECHNIQUE

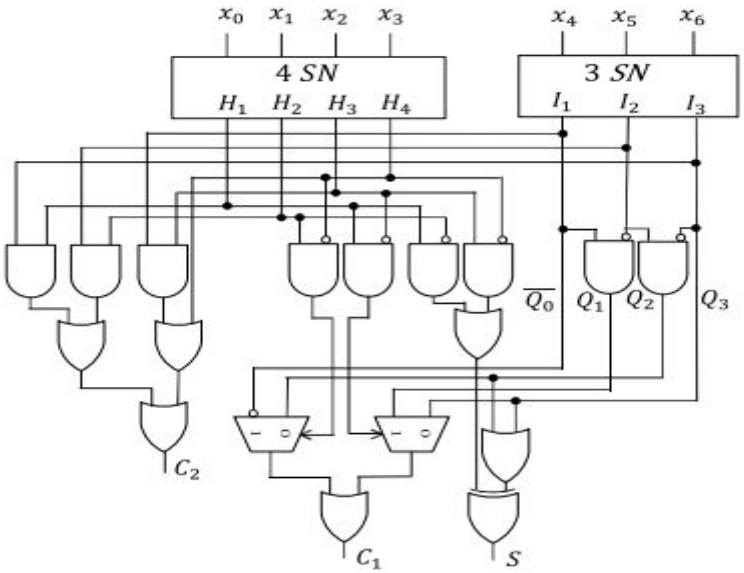

Fig:3 Overall (7,3) counter circuit

The whole architecture is shown in above Fig. It is obvious that the paths from sequences H and I to C2, C1, and S are almost independent. Increases the parallelism of the circuit. However, as will be discussed later, the area of the proposed design will not increase with the parallelism.

IMPLEMENTATION

First, as shown in above Fig. due to the fact that "1" is bigger than "0," all the "1"s are at the top of the sequence if there exist "1"s, and all the "0"s are at the bottom of the sequence if there exist "0"s. If there exist both "1"s and "0"s, there must be a position in the reordered sequence where there is the junction of "1" and "0." If there are only "1"s or "0"s, we can manage the sequence by padding fixed one bit "1" at the top and one bit "0" at the bottom of the reordered sequence to make sure that 0,1-junction always exits. Second, the reordered sequence has the same total number of "1"s and "0"s as the original sequence (the inputs of two sorting networks). Although the padded "1" would influence the total number of "1"s in the padded sequence, it is fixed, so we ignore it while counting.

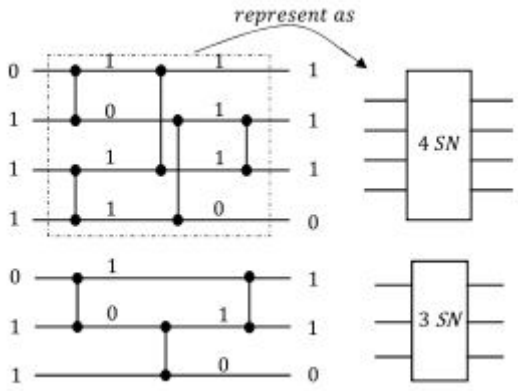

Fig.4 Three- and four-way sorting networks.

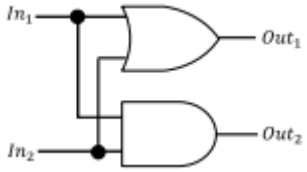

Fig.5 Two-input binary sorter

We give an input example: sequence [0, 1, 1, 1] represents the input of four-way sorting network (4 SN), and sequence [0, 1, 1] represents the input of three-way sorting network (3 SN). For both 4 SN and 3 SN, the input sequences are reordered in the form of the larger number at the top and the smaller number at the bottom after three layers of sorter.

ONE-HOT CODE GENERATION

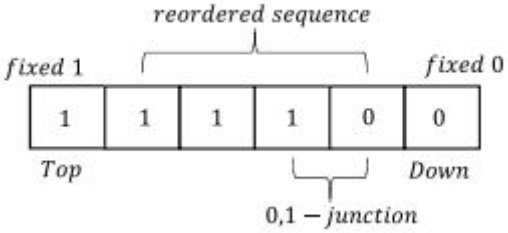

Fig.6 Definition of a sequence

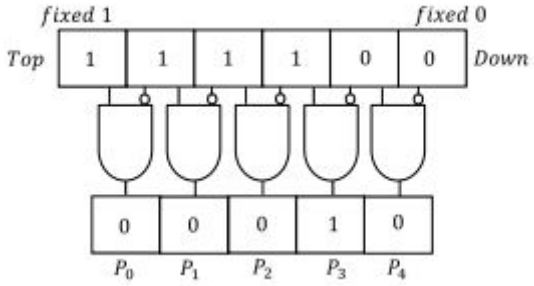

Fig. 7 One-hot code generation circuit.

1) Asymmetric Pre reorder: Both three- and four-way sorting networks require three layers of binary sorter (the two binary sorters on the same layer in four way sorting network can be calculated in parallel). Each layer of binary sorters consumes one basic two-input logical gate layer, as shown in Fig. 3. This means that the time consumed by the three- and four-way sorting networks is almost the same. Based on this, we divide the seven inputs of a (7,3) counter into two parts. One part contains 4 bits, while the other contains 3 bits.

2) Find 0,1-Junction and One-Hot Code Sequence: The 0,1-junction can solely represent the reordered sequence under the promise of the extended fixed "0" and "1." Notice that the position of the 0,1-junction must be 1,0 from left to right. Therefore, we still utilize the four way sorting network as an example, and then, This structure uses a Boolean expression (AB) to obtain a new sequence P0–P4. Because there is one and only one 0,1-junction in the reordered and extended sequence, there is one and only one "1" in sequence P0–P4. This means that sequence P0–P4 is one-hot code that satisfies ("|" represents

TRUTH TABLE OF (7,3) COUNTER OUTPUTS

Num	C_2	C_1	S	Num	C_2	C_1	S
0	0	0	0	4	1	0	0
1	0	0	1	5	1	0	1
2	0	1	0	6	1	1	0
3	0	1	1	7	1	1	1

"OR" and "&" represents "AND") P0|P1|P2|P3|P4 = 1. (1) If the sequence elements (P0–P4) are randomly divided into two groups, such as P0, P2, and P4 as group 1 and P1 and P3 as group 2, then, because of one and only one "1" in the sequence, we have P0|P2|P4 = P3|P4. (2) All results of random separation satisfy this rule. We also apply the same method on the three-way sorting network's output sequence and get the one-hot code sequence Q0–Q3. This sequence also satisfies the rule above.

we have two sequences P and Q. P0 = 1 means that there is no "1" in the input sequence of four-way sorting network, P1 = 1 represents one "1" in it, and Pi = 1 represents i "1"s in it and so is the sequence Q. Here are some symbol conventions. The outputs of (7,3) counter are denoted as C2, C1, and S, and C2 has the most significant weight, while S has the lowest weight. Table I shows the total numbers of "1"s ("Num" column in the table) in the input 7 bits corresponding to outputs, i.e., Num = 22C2 + 21C1 + 20 S. The sequence output from four-way sorting network is denoted as sequence H, including H1–H4 from left to right in Fig. 4. The sequence output from three-way sorting network is denoted as sequence I, including I1–I3 from left to right. According to Table I, we know that at least four "1"s are in the input sequence of the (7,3) counter when C2 = 1. As discussed before, P4 = 1 means that four "1"s are in sequence H (also in the input sequence of 4 SN because sorting network has no effect on total number of "1"s), and Q0 = 1 means that no "1" is in sequence I. Thus, P4&Q0 = 1 means that there are totally 4 + 0 = 4 "1"s in the input 7 bits. As a result of this type of representation, C2 is equal to 1 when the summation of subscripts of P and Q is no less than 4. In this way, C2 can be expressed as

$$C_2 = (P_4\&(Q_0|Q_1|Q_2|Q_3))|(P_3\&(Q_1|Q_2|Q_3))| \\ (P_2\&(Q_2|Q_3))|(P_1\&Q_3). \quad (3)$$

Notice that the sequence Q, with the same method in (2), satisfies

$$Q_0|Q_1|Q_2|Q_3 = 1 \quad (4)$$

$$Q_1|Q_2|Q_3 = \overline{Q_0}. \quad (5)$$

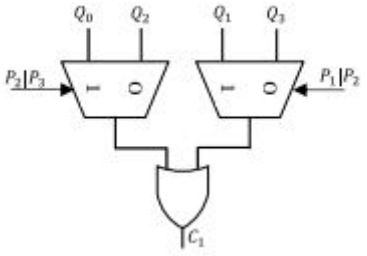

Fig. 8 C1 generating circuit

HIGHER COMPRESSION RATIO COUNTERS:
CONSTRUCT (15,4) COUNTER

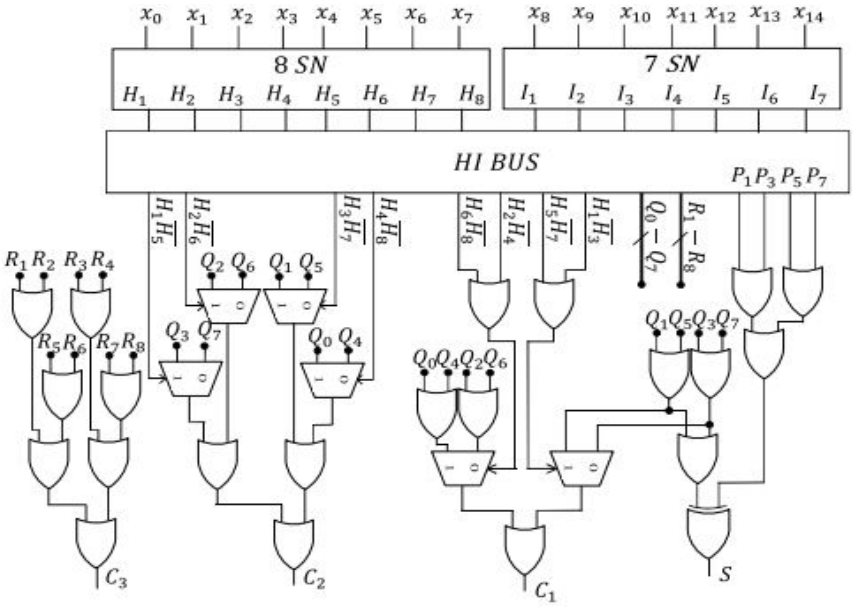

Fig.9 Overall (15,4) counter circuit

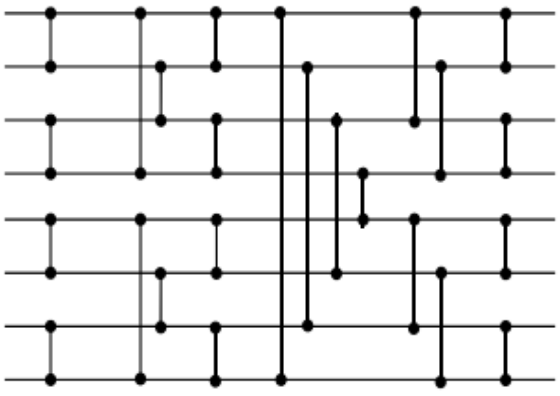

Fig 10 Eight-way sorting network

Seven- and Eight-Way Sorting Networks: Fig. shows an eight-way sorting network [14]. This sorting network consumes six layers of basic logic gates to output the result. Removing one bit from the eight-way sorting network can obtain a seven-way sorting network that also consumes six layers of basic logic gates.

$$Q_0|Q_2|Q_4|Q_6 = \overline{Q_1|Q_3|Q_5|Q_7}$$
$$P_0|P_2|P_4|P_6|P_8 = \overline{P_1|P_3|P_4|P_5|P_7}$$
$$I_i = I_i|I_{i+j},$$
$$(i = 0, 1, \ldots, 6, 7;\ j \geq 0;\ i + j \leq 8)$$

TRUTH TABLE OF (15,4) COUNTER OUTPUT

TRUTH TABLE OF (15,4) COUNTER OUTPUTS

Num	C_3	C_2	C_1	S	Num	C_3	C_2	C_1	S
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1

PROPOSED TECHNIQUE

BITONIC SORTING

Bitonic sort is one of the fastest sorting networks. A sorting networkis a special kind of sorting algorithm, where the sequence of comparisons is not data-dependent.

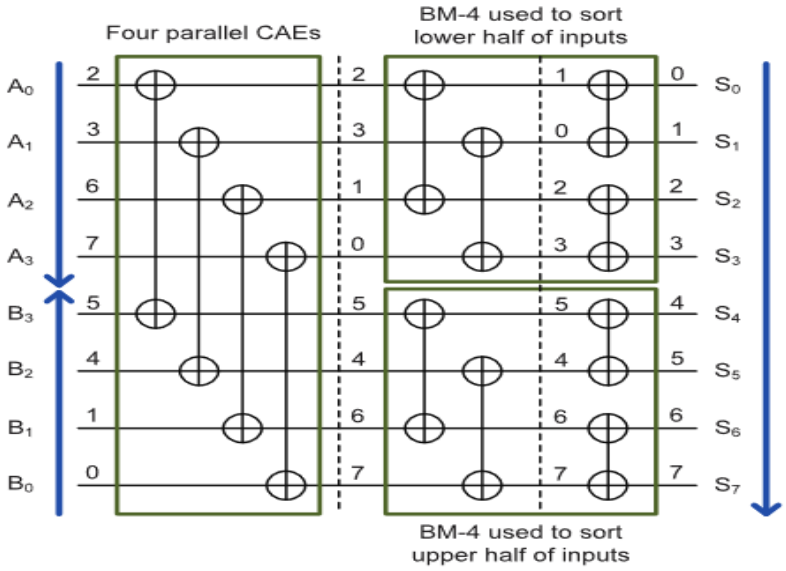

Fig.11 An increasing 8-input bionic merging unit

Each box performs the same operation as a blue box, but with the sort in the opposite direction. So, each green box could be replaced by a blue box followed by a crossover where all the wires move to the opposite position. This would allow all the arrows to point the same direction, but would prevent the horizontal lines from being straight. However, a similar crossover could be placed to the right of the bottom half of the outputs from any red block, and the sort would still work correctly, because the reverse of a bitonic sequence is still bitonic. If a red box then has a crossover before and after it, it can be rearranged internally so the two crossovers cancel, so the wires become straight again. Therefore, the following diagram is equivalent to the one above, where each green box has become a blue plus a crossover, and each orange box is a red box that absorbed two such crossovers.

RESULTS

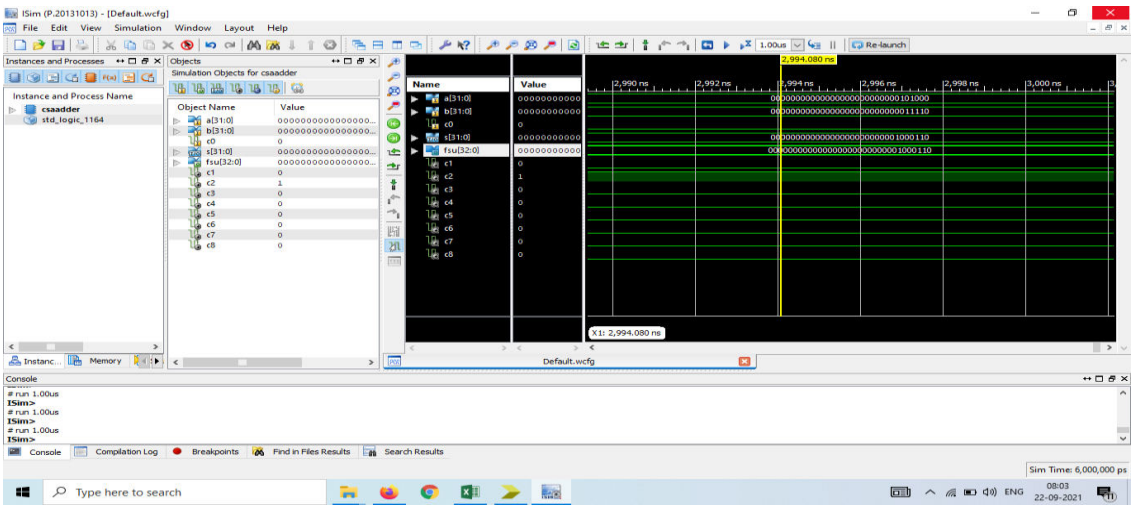

Fig: 12 Existing simulation output

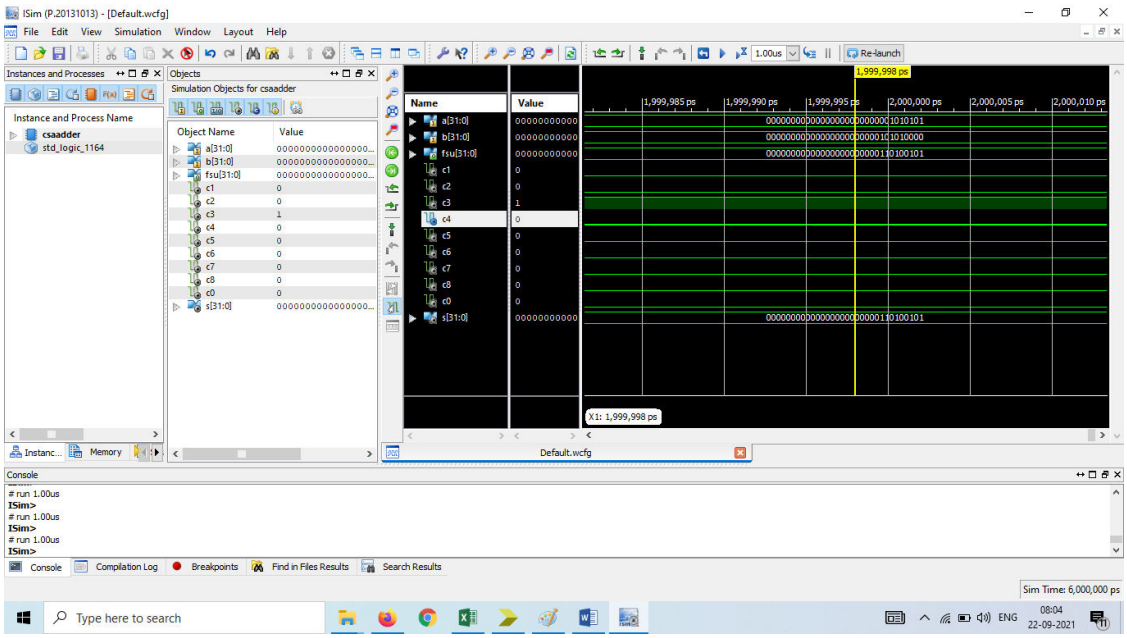

Fig: 13 proposed simulation output

CONCLUSION AND FUTURE SCOPE

This paper plans and replicates a strategy for a swift and effective counterattack. A double counter with reference to a unique symmetric piece the proposed calculation method uses aggregate and convey. We build counters (7,3) and (15,4) based on a new counter design method that is based on a sorting network that is proposed during this procedure. Because proposed counters accomplish less latency where speed is a factor and outperform previous designs in ADP, they are more adaptable than present designs. Using rough multipliers, the image, videos, and video clarity may be examined. And that are worn out to match the precise multipliers' clarity. We have used Brent Kung Adder to reduce area and speed up multiplication to increase speed we can use Kogge Stone Adder.

REFERENCES

[1] Wen-Chang Yeh and Chein-Wei Jen, “High-speed Booth encoded parallel multiplier design,” IEEE Trans. on Computers, vol. 49, isseu 7, pp. 692-701, July 2000.

[2] Jung-Yup Kang and Jean-Luc Gaudiot, “A simple high-speed multiplier design,” IEEE Trans. on Computers, vol. 55, issue 10, Oct. pp. 1253-1258, 2006.

[3] Shiann-RongKuang, Jiun-Ping Wang and Cang-Yuan Guo, “Modified Booth multipliers with a regular partial product array,” IEEE Trans. onCircuit and Systems, vol.56, Issue 5, pp. 404-408, May 2009.

[4] Li-rong Wang, Shyh-JyeJou and Chung-Len Lee, “A well-structured modified Booth multiplier design,” Proc. of IEEE VLSI-DAT, pp. 85-88, April 2008.
[5] A. A. Khatibzadeh, K. Raahemifar and M. Ahmadi, “A 1.8V 1.1GHz Novel Digital Multiplier,” Proc. of IEEE CCECE, pp. 686-689, May 2005.
[6] S. Hus, V. Venkatraman, S. Mathew, H. Kaul, M. Anders, S. Dighe, W. Burleson and R. Krishnamurthy, “A 2GHZ 13.6mW 12x9b mutiplier for energy efficient FFT accelerators,” Proc. of IEEE ESSCIRC, pp. 199-202, Sept. 2005.
[7] Hwang-Cherng Chow and I-Chyn Wey, “A 3.3V 1GHz high speed pipelined Booth multiplier,” Proc. of IEEE ISCAS, vol. 1, pp. 457-460, May 2002.
[8] M. Aguirre-Hernandez and M. Linarse-Aranda, “Energy-efficient high-speed CMOS pipelined multiplier,” Proc. of IEEE CCE, pp. 460-464, Nov. 2008.
[9] Yung-chin Liang, Ching-ji Huang and Wei-bin Yang, “A 320-MHz 8bit x 8bit pipelined multiplier in ultra-low supply voltage,” Proc. of IEEE A-SSCC, pp. 73-76, Nov. 2008.
[10] S. B. Tatapudi and J. G. Delgado-Frias, “Designing pipelined systems with a clock period approaching pipeline register delay,” Proc. of IEEE MWSCAS, vol. 1, pp. 871-874, Aug. 2005.
[11] A. D. Booth, “A signed binary multiplication technique,” Quarterly J. Mechanical and Applied Math, vol. 4, pp.236-240, 1951.
[12] M. D. Ercegovac and T. Lang, *Digital Arithmetic*, Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 2003.
[13] C. S. Wallace, “A suggestion for a fast multiplier,” IEEE Trans. On Computers, vol. BC13, pp. 14-17, Feb. 1964.
[14] M.D. Ercegovac et al., ªFast Multiplication without Carry- Propagate Addition,º IEEE Trans. Computers, vol. 39, no. 11, Nov. 1990.
[15] R.K. Kolagotla et al., ªVLSI Implementation of a 200-Mhz 16 _ 16 Left-to-Right Carry-Free Multiplier in 0.35_m CMOS Technology for Next-Generation DSPs,º Proc. IEEE 1997 Custom Integrated Circuits Conf., pp. 469-472, 1997.
[16] P.F. Stelling and V.G. Oklobdzija, ªOptimal Designs for Multipliers and Multiply-Accumulators,º Proc. 15th IMACS WorldCongress Scientific Computation, Modeling, and Applied Math., A. Sydow, ed., pp. 739-744, Aug. 1997.
[17] Passport 0.35 micron, 3.3 volt, Optimum Silicon SC Library, CB35OS142, Avant! Corporation, Mar. 1998.
[18] G. Goto et al., ªA 4.1ns compact 54 _ 54-b Multiplier UtilizingSign-Select Booth Encoders,º IEEE J. Solid-State Circuits, vol. 32, no. 11, pp. 1,676-1,682, Nov. 1997.
[19] G. Goto et al., ªA 54 _ 54-b Regularly Structured Tree Multiplier,º IEEE J. Solid-State Circuits, vol. 27, no. 9, Sept. 1992.
[20] R. Fried, ªMinimizing Energy Dissipation in High-Speed Multipliers, º Proc. 1997 Int'l Symp. Low Power Electronics and Design, pp. 214-219, 1997.
[21] N.H.E. Weste and K. Eshraghian, Principles of CMOS VLSI Design: A Systems Perspective, second ed., chapter 8, p. 520. Addison Wesley, 1993.
[22] J. Fadavi-Ardekani, ªM_N Booth Encoded Multiplier Generator Using Optimized Wallace Trees,º IEEE Trans. VLSI Systems, vol. 1, no. 2, June 1993.
[23] A.A. Farooqui et al., ªGeneral Data-Path Organization of a MAC Unit for VLSI Implementation of DSP Processors,º Proc. 1998 IEEE Int'l Symp. Circuits and Systems, vol. 2, pp. 260-263, 1998.
[24] K. Hwang, Computer Arithmetic: Principles, Architecture, and Design, chapter 3, p. 81. John Wiley & Sons, 1976.
[25] K.H. Cheng et al., ªThe Improvement of Conditional Sum Adder for Low Power Applications,º Proc. 11th Ann. IEEE Int'l ASICConf., pp. 131-134, 1998.