Implementation Of Fast Binary Sorting Network Generated By Counters

A.VAISHNAVI[1] , Dr. M. NARAYANA[2]

[1]M.TECH (VLSI SYSTEM DESIGN) ,
[2]PROFESSOR
ANURAG UNIVERSITY, Venkatapur (V), Ghatkesar (M), Medchal Dist-501301.
Email: vaishnaviram1999@gmail.com

ABSTRACT:

Parallel counters play an important role in many circuits that manipulate numbers, especially fast multipliers. Part of the crucial route in many DSP components is the parallel summing of several operands. High ratios of compression counters & compressors are required to speed up the totaling process. A new exact/approximate (4:2) compressor based on a sorting network is being developed for this project. Reordered sequences, which are uniquely portrayed through one-hot code sequences, are generated by feeding the counter's inputs into sorting networks in an asymmetric fashion. The counter's output Boolean expressions may be greatly simplified by using three specific Boolean equations created between the reorganized sequences on the one-hot code sequence. In addition, this undertaking is improved by developing scalable structures based on provided techniques for finding/sorting M biggest values from N inputs utilizing parallel sorting algorithms. The iterative sorting methods were also presented for locating the greatest values. The BUBBLE SORTING algorithm is one example of a parameter-optimized implementation of such an algorithm.

Index Terms –Binary counter, exact/approximate 4:2 compressor, multiplier, one-hot code, sorting network.

INTRODUCTION

Portable electronic devices, such as smartphones, tablets, and other gadgets, have particularly stringent needs for energy efficiency. It is very desirable to achieve this reduction with as little impact on performance as possible. Most sought after among portable components are digital signal processing (DSP) chips for realising diverse multimedia applications. The ALU serves as the computational heart of these blocks, handling the bulk of the block's multiplications and adds. Energy and power consumption might be significant since multiplications are a primary operation in processing components. In many DSP cores, the algorithms for processing pictures and movies culminate in results that can be seen by humans. It makes it easier to use approximations to boost computational speed and power in arithmetic circuits.

This is due to the fact that human beings have relatively restricted visual perception. There are additional uses than the processing of images and videos where the precision of the mathematical procedures is not absolutely necessary. Accuracy, speed, and power/energy usage are all benefits of approximate computing. An approximation multiplier's benefits include a decreased mistake rate and increased processing speed. Time is added to the multiplication process as a whole is needed to do a comparison operation & a memory look up for each operand in order to fix the division problem. Several distinct architectures for approximation arithmetic building blocks like adders and multipliers have been proposed under the category for approximations approaches in function. Different methods and designs for low-power DSP signal processing make use of approximation adders. For a 1212 fixedwidth multiplication, the suggested ETM yields savings of over 50% while keeping the reduced average error of the current technique. Multipliers play a significant role in the design of the multiplier and accumulate unit (MAC) since they are used to construct the fundamental building blocks of arithmetic units. The Adder is the next crucial component in the MAC's architecture. The design gives equal weight to adders. Different types of adders & multipliers have been proposed using the right function approaches. Designers may balance factors like accuracy, speed, energy, and power usage using

approximation computing.

LITERATURE SURVEY

[1] Wenbo Guo and Shuguo Li, Member, IEEE," Fast Binary counters and compressors generated by sorting network".

High compression ratio counters & compressors are required to speed up the totaling process. New exact/approximate (4:2) compressor based on the sorting network and fast saturated (7,3) & (15,4) binary counters. Reordered sequences, which are uniquely portrayed through one-hot code sequences, are generated by feeding the counter's inputs into sorting networks in an asymmetric fashion. The ADP and PDP of the 73 counter are shorter. In a similar the (15,4) counters is built and achieves about shorter latency with much lower power and space requirements. The exact/approximate (4:2) compressor built on a sorting network perform better when incorporated in an 8 8 bit approximate multiplier for both the ADP and PDP.

[2] C. Fritz and A. T. Fam, "Fast binary counters based on symmetric stacking".

There is a suggestion for a new kind of binary counter. The "1" bits are stacked in groups of three in 3-bit stacking circuits, and then the circuits are combined using a unique symmetric way to form 6-bit stacks. The critical route of the 6:3 counting circuits does not include any xor gates. By excluding xor gates, designs may save time and resources without sacrificing performance. The suggested counters are 30% quicker in VLSI simulations than conventional parallel counters while using less electricity. Latency and power consumption for 64-bit and 128-bit multipliers are both decreased when the suggested counters are used in current counter-based Wallace tree multiplication designs.

[3] A. Fathi, B. Mashoufi, and S. Azizian, "Very fast, high-performance 5-2 and 7-2 compressors in CMOS process for rapid parallel accumulations".

We suggest ultrafast 5-2 and 7-2 compressors. When compared to earlier designs, the gate-level latency has been drastically cut down. Speed performance is enhanced by 32% and 30%, respectively, when applying the technology established for a 5-2 compressor to a 7-2 design.

The effectiveness of the planned compressor blocks has been tested by implementing a standard 16 16-bit multiplier. The findings show that the delay for the 5-2 compressors has been attained at 303 ps, whereas the delay for the 7-2 compressor, as intended, has been measured at 464 ps.

EXISTING SYSTEM

The ability to efficiently and rapidly expand a variety of operands is a cornerstone of every computing device. Multiplier circuits' speed and power efficiency are fundamental to the chip's overall operation. An integral feature of any sophisticated flag processor architecture used for sorting and convolution is the multiplier circuit. A variety of methods, such as the prominent column pressure techniques in the Wallace tree and Dadda tree, or the improved design, have been presented to improve the efficiency of the imperfect item summation. One such method employs complete adders functioning as counters to parallelly reduce groups of three identical loads to two different loads using a carry spare viper tree. When dealing with larger data sets, the underlying approach is the same: bits in a single sector are counted, resulting in fewer bits of varied loads. A 7:3 counter circuit, for instance, uses 7 bits and checks for the presence of "1" bits to determine their corresponding weight. This sum is then obtained by applying 3 increasing weight bits. The "1" bits are stacked in groups of three in 3-bit stacking circuits, and then the circuits are combined using a unique symmetric way to form 6-bit stacks. This process of converting bit stacks to binary counts yields 6:3 counter circuits without any XOR gates on the crucial route.

As a result, in the suggested method, we develop a 128x128 Vedic Wallace stacker and an 8x8 Wallace branch multiplier stacker. When compared to the 6 TO 3 Bit stacker and the 7 TO 3 Bit stacker, the suggested techniques provide a significant

improvement in execution speed. In the current method, the "1" bits are grouped together in clusters using 3-bit stacking circuits, and then a unique symmetric mechanism is used to combine multiple 3-bit stacks into a 6-bit stack. To get 6:3 counter circuits without XOR gates, the bit stacks are converted to paired checks. Proposed counts are 30% faster than current parallel counters and use less power than other high-request counters in VLSI simulations.

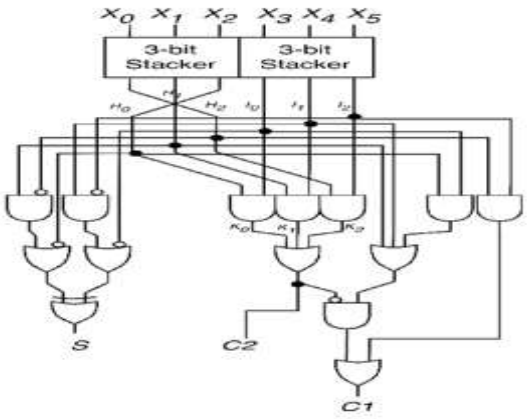


Fig: 1. counter based on symmetric stacking

IMPLEMENTATION OF (7, 3) COUNTER:

In fig, because "1" is greater than "0," all "1" s are at the top of the series if there are "1" s, and all "0" s were at the bottom of the order if there exist "0" s. There has to be a point in the rearranged sequence where a "1" and a "0" meet if there are both ones and zeroes. In the case of a binary sequence consisting entirely of "1"s or "0"s, the sequence may be controlled by inserting a fixed "1" at the beginning of any reordered sequence and a fixed "0" at the end in order to guarantee that the 0,1-junction always resolves to the "1" state.

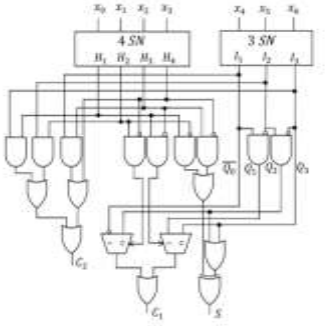


Fig: 2. Overall (7, 3) Counter

$$Q_0|Q_1|Q_2|Q_3|Q_4 = \overline{Q_0|Q_1|Q_2|Q_3|Q_4}$$

$$P_0|P_1|P_2|P_3|P_4|P_5|P_6 = \overline{P_0|P_1|P_2|P_3|P_4|P_5|P_6} \quad (17)$$

$$I_i = I_i|I_{i+j},$$

$$(i = 0, 1, \ldots, 6, 7; j \geq 0; i + j \leq 8)$$

Second, there is no change in the

overall number of "1" s and "0" s between the original and rearranged sequences. We disregard the padded "1" while counting since its presence would change the overall amount of "1" s in the padding sequence, but this value is fixed.

Find 0,1-Junction and One-Hot Code Sequence: With the assurance of the stretched fixed "0" and "1," the 0,1-junction is the only place where the rearranged sequence may be represented. Keep in mind that the left-to-right orientation of the 1, 0 connection must be 1, 0 for this structure to produce the desired P0-P4 sequence through Boolean expression. Sequence P0-P4 has just one "1" because the rearranged and lengthened sequence has exactly one 0, 1-junction. Given that ("|" represents "OR" and "&" represents "AND"), the sequence P0-P4 is a one-hot coding that satisfies.

IMPLEMENTATION OF (15, 4) COUNTER:

Eight-way filtering system. Six layers of simple logic gates are used in this sorting network. The eight-way sort network may be reduced to a seven-way sorting network using only six levels of basic logic gates if only one bit is removed.

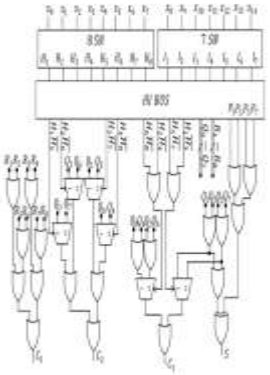


Fig. 3. Overall [15, 4] counter circuit

PROPOSED METHOD

BUBBLE SORTING:

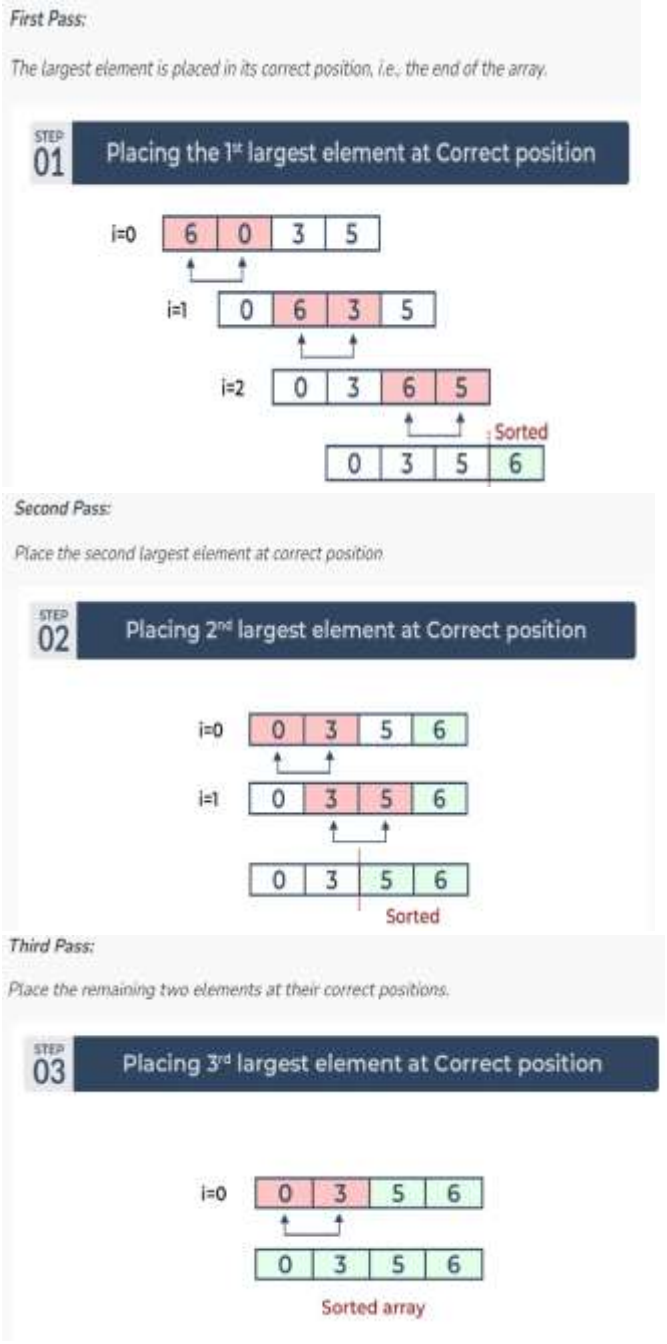
Bubble sort, also known as sink sort, is a straightforward method of sorting that iteratively compares each member of the input list to the next, switching values if necessary. Iteratively performing these sorts on the list until no swaps are required during any iteration results in a sorted list. Named

after how bigger items "bubble" to the top of the list, the method is a kind of comparison sort.

This basic algorithm is mostly utilised in the classroom because of its poor performance in the actual world. Python and Java both provide built-in sorting libraries that make advantage of more modern and efficient algorithms like quicksort, timsort, and merge sort. While insertion sort and selection sort both benefit from parallel processing, they do not combine as well as bubble sort, which sorts in O(n) time.

The Bubble sort algorithm was first described as a "Sorting exchange algorithm" in a 1956 paper titled "Sorting on electronic computer systems" by mathematical expert and actuary Edward Harry Friend[4] in the fourth issue of the second installment of the Journal of the Association for Computing Machinery (ACM). Although Friend's research was mostly ignored at the time of its publication, numerous computer scientists—including the man responsible for the algorithm's present name, Kenneth E. Iverson—rediscovered it years later.

If elements are not in the correct order, Bubble Sort will continually swap them with their neighbours. Due to its high average & worst-case time complexity, this approach is unsuitable for use with huge data sets.
The biggest piece is first moved to its rightmost end by starting from the left and comparing neighbouring pieces to determine which is superior.
This procedure is repeated to identify the next biggest item and assign it a location, and so on, until all the information has been sorted.



SIMULATION RESULTS

Currently, the Counter with n inputs is split symmetrically into two halves (6,3) and (7,3). There are both full adders & half adders in use here. More XOR gates in a circuit increase its latency and the amount of real estate it needs. Due to the increased number of XOR gates, the hardware complexities of

the circuit is likewise significant, as is the power consumption of the current design. The suggested counters use a sorting network to arrange the (7,3) & (15,4) Counters in a symmetrical fashion. The suggested design makes use of reduced full and half adders. Saturated counters, with reduced latency and a lower footprint because to the absence of XOR gates. There is little energy waste.

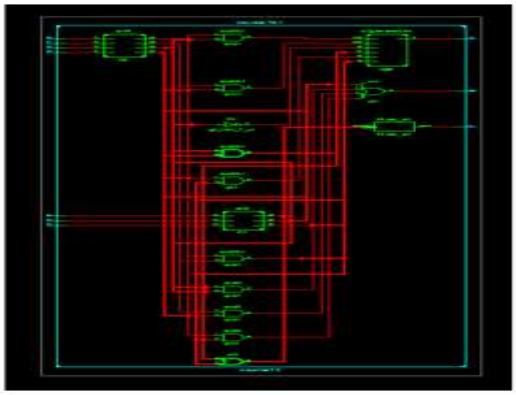


Fig: 4. Schematic diagram of (7, 3) Counter

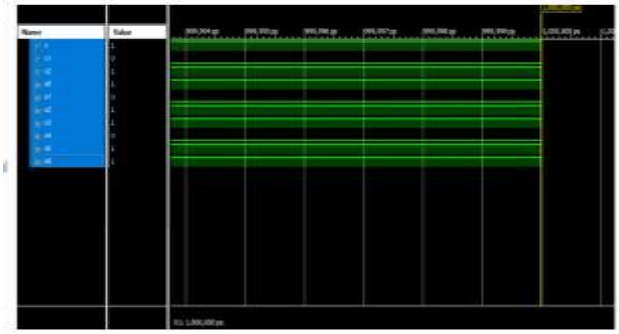


Fig: 5. (7, 3) Output waveform

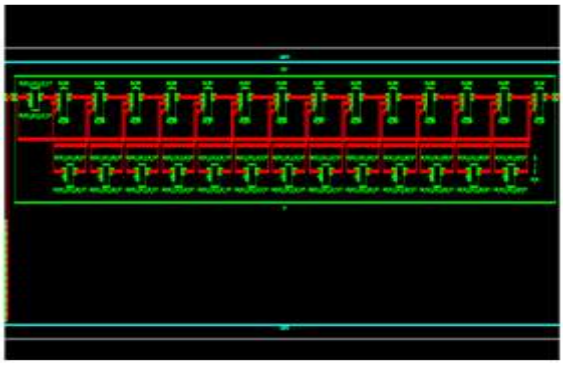


Fig: 6. Schematic diagram of (15, 4) Counter

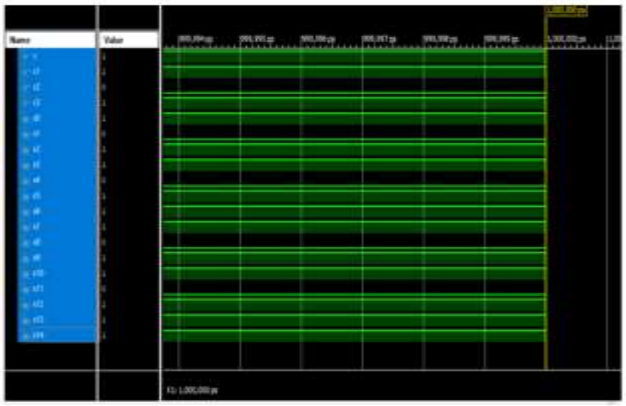


Fig: 7. (15, 4) counter Simulation result

COMPARISION TABLE

METHOD	TYPE OF COUNTERS	DELAY (ns)	AREA (LUT)
EXISTING METHOD	7,3	2.789ns	12
	15,4	6.114ns	67
PROPOSED METHOD	7,3	1.696ns	6
	15,4	5.839ns	28

Table 1: Comparison Table of Existing and Proposed Methods.

CONCLUSION

The goal of this research is to develop and replicate an effective, regionally feasible counter strategy. In this procedure, we build (7, 3), (15, 4), and propose a novel counter

design approach based on an order network. By achieving reduced delay when speed is necessary and outperforming present designs in ADP, the proposed counters were more adaptable than those now in use. This new method of counter building is very helpful for constructing BUBBLE sort-based counters like the (7,3) or (15,4) etc. Compared to conventional counter designs, the suggested one is more flexible thanks to its lower latency and greater efficiency in area-delay product (ADP). Using an FPGA Zed board, the FPGA was also able to build counters for the numbers 7, 3, and 15.

FUTURE SCOPE:

Since we have already constructed multiplier designs based on the suggested binary counters, we may use them to create higher bit multipliers in a manner that optimizes the higher bit multiplier's total latency.

REFERENCES

(1) Wenbo Guo and Shuguo Li, “Fast binary counters and compressors generated by sorting network” IEEE TRANSACTIONS ON VERY LARGE-SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 29, NO. 6, JUNE 2021.

(2) “Design using multi-input counter and compressor circuits,” in Proc. 10th IEEE Symp. Comput. Arithmetic, Grenoble, France, Jun. 1991, pp. 43–50.

(3) A. Fathi, B. Mashoufi, and S. Azizian, “Very fast, high-performance 5-2 and 7-2 compressors in CMOS process for rapid parallel accumulations,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 6, pp. 1403–1412, Jun. 2020.

(4) T. Satish and K. S. Pande, “Multiplier using NAND based compressors,” in Proc. 3rd Int. Conf. Electron., Mater. Eng. NanoTechnol. (IEMENTech), Kolkata, India, Aug. 2019, pp. 1–6

(5) M. H. Najafi, D. J. Lilja, M. D. Riedel, and K. Bazargan, “Low-cost sorting network circuits using unary processing,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 8, pp. 1471–1480, Aug. 2018.

(6) Q. Jiang and S. Li, “A design of manually optimized (15, 4) parallel counter,” in Proc. Int. Conf. Electron Devices Solid State Circuits (EDSSC), Hsinchu, Taiwan, Oct. 2017, pp. 1–2

(7) C. S. Wallace, “A suggestion for a fast multiplier,” IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

(8) M. Mehta, V. Parmar, and E. Swartz lander, “High-speed

multiplier design using multi-input counter and compressor circuits," in Proc. 10th IEEE Symp. Comput. Arithmetic, Grenoble, France, Jun. 1991, pp. 43–50.

(9) S. Asif and Y. Kong, "Design of an algorithmic Wallace multiplier using high speed counters," in Proc. 10th Int. Conf. Comput. Eng. Syst. (ICCES), Cairo, Egypt, Dec. 2015, pp. 133–138.