ISSN 2321-2152
www.ijmece.com
Vol 13, Issue 1, 2025

IMPLEMENTATION OF HIGH-SPEED BINARY COUNTERS USING SORTING NETWORKS

ALAKUNTLA YOGESH[1] , BEDREI HARSHAVARDHAN RAO[2] , BINGI SRIRAM KARTHIK YADAV[3] , BODEPATI BALAJI[4] , CHILUKALA SEKHARA REDDY[5] , VADLA NAVEEN KUMAR[6], Dr.T.SYED AKHEEL [7]
[123456]UG STUDENTS, DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING, DR.K.V.SUBBA REDDY INSTITUTE OF TECHNOLOGY, KURNOOL,AP, INDIA.
[7]ASSOCIATE PROFESSOR, DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING, DR.K.V.SUBBA REDDY INSTITUTE OF TECHNOLOGY, KURNOOL,AP, INDIA.

ABSTRACT: Efficient summation of multiple operands in parallel is a critical aspect of various digital signal processing units. To accelerate this process, high compression ratio counters and compressors are indispensable. The summation of multiple operands in parallel forms part of the critical path in various digital signal processing units. To speed up the summation, high compression ratio counters and compressors are necessary. In this project, we present a novel method of fast saturated binary counters and exact/approximate (4:2) compressors based on sorting network. The inputs of the counter are asymmetrically divided into two groups, and fed into sorting networks to generate reordered sequences, which can be solely represented by one-hot code sequences. Between the reordered sequence and the one-hot code sequence, three special Boolean equations are established, which can significantly simplify the output Boolean expressions of the counter. Using the above method, we construct and further optimize the (7,3) counter which can performs better in maximum than other designs in delay, area- delay product, power-delay product, respectively. It will be developed using Verilog HDL. Xilinx ISE tool is used to perform the Simulation and Synthesis.

INTRODUCTION

The summation of multiple operands is widely used in various digital signal processing (DSP) units and constitutes a part of the critical path. A basic multiplier circuit add all the partial products up with the Wallace Tree structure, whose performance is the bottleneck of the basic multiplier. Public-key cryptosystems, such as RSA and Elliptic Curve Cryptography (ECC), use big number multiplier based on Toom-Cook or Karatsuba algorithm to construct modular multipliers. Many papers have studied these two algorithms and implemented them with hardware. In many parts of the circuit utilize the summation of multiple operands. Fully homomorphic encryption (FHE) is a post-quantum cryptosystem which provides strong security in cloud computing and it urgently needs Number Theoretic Transform (NTT) [6] to accelerate large number multiplication and polynomial multiplication. In some high radix [6] NTT implementations, the core processing unit is composed of the summation of multiple operands.

The most famous multiple operands summation method is Wallace Tree structure and its improved method Reduced Wallace Tree. These methods use full adders as (3,2) counters to accelerate the summation, resulting in logarithmic time consumption. This type of structure is also called carry save structure.

In several fields, including digital signal processing (DSP), computer vision, multimedia processing, image identification, and artificial intelligence, multipliers are some of the most important arithmetic functional units. These applications often need extensive multiplication, which consumes a lot of power. This high power consumption makes it difficult to execute certain applications, particularly on mobile devices. Because of this, several research have suggested methods to lessen the load on multiplier circuits' power supplies. If the intended applications can tolerate mistake, that is, if they are connected to human senses, then approximating multiplication may be used to lower computer multiplier's power consumption. However, precise computer results are unnecessary because of the limits of the human senses, including the human eye's restricted visual spectrum and indeed the human ear's limited audible range. Mario Donato Marino served as assistant editor and oversaw the manuscript's scrutiny and final approval before publishing. In order to reduce cell space, time delay, & power consumption, approximation multipliers make a compromise on precision.

When the input voltage of such a multiplier gets decreased, the critical path delay lengthens. Therefore, mistakes arise and approximate results are produced if the time route is violated. This second kind involves rethinking the precise circuitry of multipliers like the Wallace Tree Multiplier as well as the Dadda Tree Multiplier in order to alter their useful behaviour. Among the publications proposing new multipliers, the vast majority suggested inexact m-n compressors, which take in m inputs and spit out n outputs. Because the process of compressing partial products required much of the multiplier's energy and produced considerable route delay, these inefficient compressors were utilised to compress all partial products inside multiplication.

Both complexity, size, and density for integrated circuits is quickly growing due to the expansion of applications as well as the improvement of semiconductor process technology. This comes at the cost of a sharp rise in power consumption, which then in turn shortens the useful life of gadgets and makes them more prone to malfunction. Fortunately, owing to our limited perceptual abilities, the quality of just what we perceive is unaffected by accuracy loss in a suitable range in so many applications including such multimedia, digital signal processing, or machine learning. That opens the door for approximation computation blocks to take the place of the more expensive full precision ones.

Because in many digital systems, multiplication is indeed an essential process. Multipliers present a wide variety of approximative designs. Larger bit-width multipliers may benefit from hybrid-radix Commonly used is the Booth encoding technique, which relies on an estimate of the production of partial products. Compression trees are employed to apply the approximation for multipliers with lower bit widths, since the partial products for these multipliers are typically created using simple AND gates.

LITERATURE SURVEY

A Reduced Complexity Wallace Multiplier Reduction by R. S. Waters and E. E. Swartzlander

Wallace high-speed multipliers use full adders and half adders in their reduction phase. Half adders do not reduce the number of partial product bits. Therefore, minimizing the number of half adders used in a multiplier reduction will reduce the complexity. A modification to the Wallace reduction is presented that ensures that the delay is the same as for the conventional Wallace reduction. The modified reduction method greatly reduces the number of half adders; producing implementations with 80 percent fewer half adders than standard Wallace multipliers, with a very slight increase in the number of full adders.

High-Speed ECC Processor Over NIST Prime Fields Applied With Toom–Cook Multiplication by J. Ding, S. Li and Z. Gu

In this paper, a high-speed elliptic curve cryptography (ECC) processor specialized for primes recommended by the National Institute of Standards and Technology (NIST) was constructed. Toom-Cook multiplication without division was proposed to implement modular multiplication for NIST primes. Compared with a traditional algorithm, the computation complexity was reduced from 16 base multiplications to 7 in 4-way Toom-Cook multiplication. Moreover, we introduced non-least-positive (NLP) form into our design, so that the carry chain in the large array accumulation was broken down, which greatly shortened the critical path and made parallel processing possible. In order to support NLP form and lazy reduction strategy, conventional fast reduction methods for NIST primes were also modified. In addition, pipeline technique at the level of point multiplication was used, so the latency of modular inverse can be covered. Implemented on the Xilinx Virtex-6 FPGA platform, the ECC processor can perform a point multiplication every 54 µs at the cost of 30.3k LUTs and 48 DSPs. Synthesized with 180nm CMOS technology, the speed achieves 43.7 µs with 466k gate counts. These experimental results show a significantly better performance per area than previous works.

VLSI Design of a Large-Number Multiplier for Fully Homomorphic Encryption by W. Wang, X. Huang, N. Emmart and C. Weems

This paper presents the design of a power- and area-efficient high-speed 768000-bit multiplier, based on fast Fourier transform multiplication for fully homomorphic encryption operations. A memory-based in-place architecture is presented for the FFT processor that performs 64000-point finite-field FFT operations using a radix-16 computing unit and 16 dual-port SRAMs. By adopting a special prime as the base of the finite field, the radix-16 calculations are simplified to requiring only additions and shift operations. A two-stage carry-look-ahead scheme is employed to resolve carries and obtain the multiplication result. The multiplier design is validated by comparing its results with the GNU Multiple Precision (GMP) arithmetic library. The proposed design has been synthesized using 90-nm process technology with an estimated die area of 45.3 mm ^{2} . At 200 MHz, the large-number multiplier offers roughly twice the performance of a previous implementation on an NVIDIA C2050 graphics processor unit and is 29 times faster than the Xeon X5650 CPU, while at the same time consuming a modest 0.97 W.

Analysis of different architectures of counter based Wallace multipliers, by S. Asif and Y. Kong

Multiplication is one of the most commonly used operations in the signal processing algorithms. Multipliers based on Wallace reduction tree provide an area-efficient strategy for high speed multiplication. A number of modifications are proposed in the literature to optimize the speed and area of the Wallace multiplier. Counter based Wallace multipliers are proved to provide faster operation as compared to the traditional Wallace multipliers. This work proposes a number of architectures for the counter based Wallace multipliers to analyse their performance for various bit lengths. Designs are synthesized using Synopsys Design Compiler in 90 nm process technology and the post synthesis delay and power results are obtained by using Synopsys Prime Time. The proposed counter based Wallace multipliers are also compared with traditional Wallace multiplier to evaluate the energy per operation of both designs. The synthesis results shows that the Power-Delay Product of the counter based Wallace multiplier is up to 17% lower as compared to the traditional Wallace multiplier.

Low-power and high-speed 4-2 compressor by A. Najafi, B. Mazloom-nezhad and A. Najafi

Compressors are the most important component of multipliers. Multipliers themselves are important components that dictate the overall performance of arithmetic circuits. In this paper, a new 4-2 compressor architecture is proposed. This architecture uses Carry Generator Module (CGEN) which has been used for 5-2 and 7-2 compressors already. The proposed architecture is compared with the best architecture presented in the literature in terms of power and delay. Interestingly, analysis shows that the proposed compressor architecture outperforms the best existing architecture both in terms of power and delay.

EXISTING SYSTEM

A compressor adder provides reduced delay over conventional adders using full adders and half adders. It is represented as N-r, where N represents the number of bits and r represents the total count of 1s present in N bits. It is termed as compressor so that it reduces the gate count and delay compared to other adder circuits. Studies are taken place to improve circuits of lower order compressors.

The compressor circuits which can be used for multiplication process such as 5-3, 10-4, 15-4 and 20-5 are explained in next subsection. The schematic view of 5-3 compressor adder. In this compressor adder maximum of five bits can be added and a result of three bit is obtained .Maximum possible value obtained is 101, which is the three bit binary of decimal 5.

In the figure n2, n3 and n4 are 4:1 multiplexers which allow only one output to be high in an instant, which results in lowering the delay and consumes low power. In Fig. 2 schematic view of 10-4 compressor adder obtained using incisive simulator in cadence is showed. In this4 compressor adder maximum of ten bits can be added and a result of four bit is obtained. Maximum possible value obtained is 1010, which is the four bit binary of decimal 10.

To expedite the compacting of partial products, high-speed parallel multipliers are using the 4-2 Carry Save Adder. Traditional implementations of such a 4-2 Carry Save Adder use a cascade of two complete adders, as illustrated in Fig.4.6.

To do the summation of multiple operands in various digital signal processing units, we need to improve the properties of the processing units, by speeding up the process of counters used as the

multiple operands. The Wallace tree structure is the best method for summation of multiple operands.Wallace tree structure method use (3,2) counters filled with full adders to speed up the summation and it gives logarithmic time consumption.

So, to construct a structure with timeefficient to speed up the summation process many papers

have discussed. Now we have a basic (7,3) counter designed by the symmetric bit stacking. The

counters are used to count the number of 1's bit in the given inputs. So, the counters that are used in digital processing units are used to countthe number of 1's in input[1]. The limit of compression efficiency can be achieved bythe counter, if the counter is a saturated counter. The designs which are unsaturated counters will use too much area space and speed. This counter designed by symmetric stacking also consumes more power. The (7,3) counter designed using symmetric stacking is unsaturated but it is very fast compared to the other counter designs. These counters are unsaturated counters[1]thus, Fritz and Fam[4] proposed symmetric stacking method to make the(7,3) counter a saturated counter

ISSN 2321-2152
www.ijmece.com
Vol 13, Issue 1, 2025

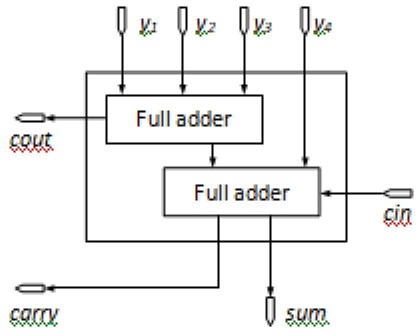

Fig. 1 . Implementation of a conventional 4-2 carry save adder**.**

It uses 5 inputs (y1, y2, y3, y4, and cin) and 3 outputs (y0, y1, and cin) with each 4-2 carry save adder (sum, cout, and carry).

PROPOSED SYSTEM

Sorting network is an efficient parallel hardware network utilized for data sorting. The famous 0,1 principle shows that if a sorting network can sort a group of data whose elements are all 1-bit numbers, it can sort all types of numbers. In this paper, we only adopt it for 1-bit data sorting.

A. Sorting Network Working Principle

The typical 3-way and 4-way sorting networks are shown in Fig.2. Each vertical line represents a sorter which has two data inputs and two data outputs, and all data are 1 bit numbers. The sorter always puts the larger input up, the smaller one down. In Fig.2, we give an input example: sequence [0, 1, 1, 1] represents the input of 4-way sorting network (4 SN), and sequence [0, 1, 1] represents the input of 3-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 3 layers of sorter.

As mentioned above, the sorter reorders two inputs according to numerical magnitudes. As for two 1-bit data, the logical circuit illustrated in Fig.3 can sort them easily. This means that

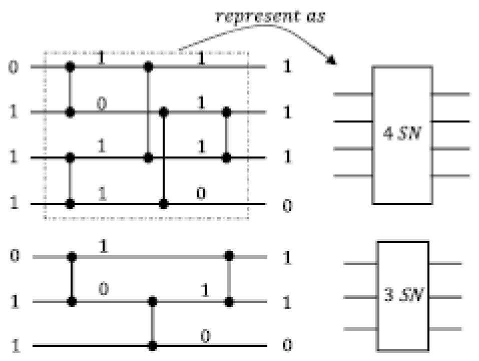

Fig. 2. 3-way and 4-way sorting networks.

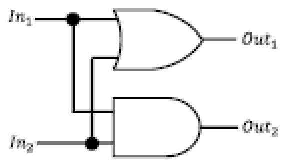

Fig. 3. Tow inputs binary sorter.

a sorter consumes one layer of 2-input basic logic gate, and the 3-way and the 4-way sorting networks both consume 3 layers of 2-input basic logic gates.

PROPOSED (7,3) COUNTER

In this section, we construct an efficient (7,3) counter. As the main comparison object, we first briefly review the design in [11]. [11] proposed a very fast (6,3) counter with a symmetric stacking structure, and they constructed a (7,3) saturated counter on the basis of this (6,3) counter. Although it is the fastest compared to other (7,3) counter designs, its delay performamce is worse because of simply introducing a MUX on the critical path without any optimization. To solve the problem in [11], we propose this method of directly construct a (7,3) counter. Unlike the symmetric stacking structure, we start with two sorting networks asymmetrically as illustrated in Fig.2. By generating one-hot code sequences, we establish three special Boolean equations (equation (2), (13) and (15)) which significantly simplify the Boolean expressions related to outputs.

A. Some Characteristics of Sorting Network

According to the review in the previous section, we summarize two characteristics of sorting networks.

ISSN 2321-2152

www.ijmece.com

Vol 13, Issue 1, 2025

First, as shown in Fig.4, 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, 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 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.

1) Asymmetric pre-reorder: As illustrated in Fig.2, both 3- way and 4-way sorting networks require 3 layers of binary sorter (the two binary sorters on the same layer in 4-way sorting network can be calculated in parallel). Each layer of binary sorters consumes one basic two input logical gate layer, text colorredas shown in Fig.3. This means that the time consumed by the 3-way and 4-way sorting networks is almost the same.

Based on this, we divide the 7 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: As shown in Fig.4, 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 utilizing the 4-way sorting network as an example and then we have the structure in Fig.5.

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 which satisfies (“j” represents “OR” and “&” represents “AND”)

P0jP1jP2jP3jP4 = 1 (1)

If the sequence elements (P0-P4) are randomly divided into two groups, such as P0; P2; P4 as group 1, P1; P3 as group 2, then because of one and only one “1” in the sequence, we have

P0jP2jP4 = P3jP4 (2)

All results of random separation satidfy this rule. We also apply the same method on 3-way sorting network’s output sequence, and get the one-hot code sequence Q0-Q3. This sequence also satisfies the rule above.

C. Output Generation

1) Basic output generation: Now we have two sequences P and Q. P0 = 1 means that There is no “1” in the input sequence of 4-way sorting network, and P1 = 1 represents one “1” in it, and Pi = 1 represents i “1”s in it. So is sequence Q.

Here are some symbol conventions. The outputs of (7,3) counter are denoted as C2;C1; S, and C2 has the most significant weight while S has the lowest weight. The total numbers of “1”s (“Num” column in the table) in the input 7 bits corresponding to outputs, i.e. Num = 22C2 +21C1 +20S. The sequence output from 4-way sorting network is denoted as sequence H, including H1-H4 from left to right in Fig.4. The sequence output from 3-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 affect on total number of “1”s), and Q0 = 1 means that no “1” is in sequence I. So 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

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

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

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

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

Put equation (4) and (5) into equation (3) we get
C2 = P4j(P3&Q0)j(P2&(Q2jQ3))j(P1&Q3) (6)
As for C1, the sum of subscripts of sequences P and Q equals 2,3,6,7, then C1 = 1. So we get equation (7).

$$C_1 = (Q_0\&(P_2|P_3))|(Q_1\&(P_1|P_2))|(Q_2\&(\overline{P_2|P_3}))|(Q_3\&(\overline{P_1|P_2})) \qquad (7)$$

Note that

$$C_1 = (Q_0\&(P_2|P_3))|(Q_1\&(P_1|P_2))|(Q_2\&(\overline{P_2|P_3}))|(Q_3\&(\overline{P_1|P_2})) \qquad (10)$$

In equation (10), P2jP3, P2jP3, and P1jP2, P1jP2, construct two Multichannel selection constructions. And via the circuit in Fig.6, C1 can be calculated time efficiently.
As for S, it can be easily obtained by equation (11), where denots “XOR”.

$$S = (P_1|P_3) \oplus (Q_1|Q_3) \qquad (11)$$

2) Further Optimization: In the last subsection, we got two sequences H1-H4 and I1-I3. Here we extend sequence H1-H4 by H0 (denotes the fixed “1” in Fig.4) and H5 (denotes the fixed “0”). Do the same for sequence I. I0 denotes the fixed “1” and I4 denotes the fixed “0”. Thus we have equation (12).

$$P_i = H_i\&\overline{H_{i+1}}, i = 0,1,2,3,4$$
$$Q_i = I_i\&\overline{I_{i+1}}, i = 0,1,2,3 \qquad (12)$$

In addition, we notice that, when subsequences selected from the sequence Q or P are given, if their subscripts are successive (P1; P2; P3 for example), the result of “OR” them up can be easily expressed by sequence I or H (P1jP2jP3 =H1&H4 for example). So Boolean equation (12) is generalized as equation (13). “P” in equation (13) represents continuous “OR”.

$$\sum_{n=i}^{n=j} P_n = H_i\&\overline{H_{j+1}}, (0 \le i \le j \le 4)$$
$$\sum_{n=i}^{n=j} Q_n = I_i\&\overline{I_{j+1}}, (0 \le i \le j \le 3)$$
(13)

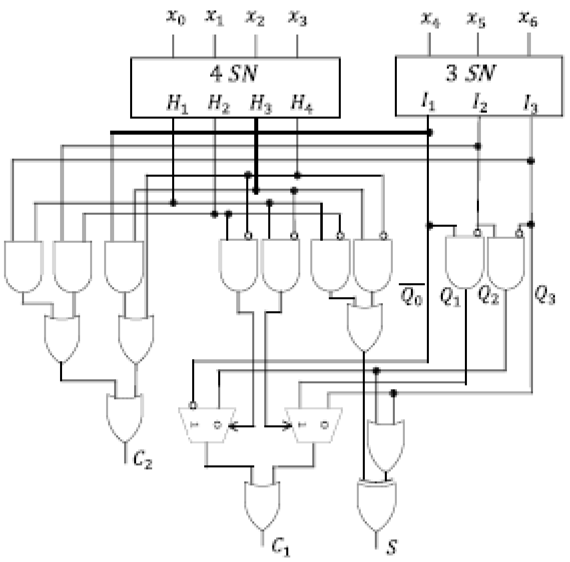

Fig. 4. Overall (7,3) counter circuit.
Based on this, equation (10) is simplified to equation (14) which can also be calculated from the circuit in Fig.6.
C1
=(Q0&(P2jP3))j(Q1&(P1jP2))j(Q2&(P2jP3)) j(Q3&(P1jP2))
=(Q0&(H2jH4))j(Q1&(H1jH3))j(Q2&(H2j H4))j(Q3&(H1jH3)) (14)

There is another trick for sequences H and I. Because H0-H5 are all in order, this means that if Hi = 1(i = 0; 1; 2; 3; 4; 5) then for every j < i, Hj = 1 always holds. So is the sequence Q. Then we get equation (15).
Ii = IijIi+j ; (i = 0; 1; 2; 3; j _ 0; i + j _ 4)
Hi = HijHi+j ; (i = 0; 1; 2; 3; 4; j _ 0; i + j _ 5)
(15)
Note that H0 = I0 = 1 and H5 = I4 = 0 always holds, we can simplify equation (3) as (using a trick:Aj(A&B) = AjB)

$$
\begin{aligned}
C_2 =&(P_4\&(Q_0|Q_1|Q_2|Q_3))|(P_3\&(Q_1|Q_2|Q_3))|\\
&(P_2\&(Q_2|Q_3))|(P_1\&Q_3)\\
=&H_4|(H_3\&\overline{H_4}\&I_1)|(H_2\&\overline{H_3}\&I_2)|\\
&(H_1\&\overline{H_2}\&I_3)\\
=&H_4|(H_3\&I_1)|(H_2\&\overline{H_3}\&I_2)|(H_1\&\overline{H_2}\&I_3)\\
=&H_4|(H_3\&(I_1|I_2))|(H_2\&\overline{H_3}\&I_2)|(H_1\&\overline{H_2}\&I_3)\\
=&H_4|(H_3\&I_1)|((H_2\&\overline{H_3}|H_3)\&I_2)|(H_1\&\overline{H_2}\&I_3)\\
=&H_4|(H_3\&I_1)|((H_2|H_3)\&I_2)|(H_1\&\overline{H_2}\&I_3)\\
=&H_4|(H_3\&(I_1|I_2))|(H_2\&I_2)|(H_1\&\overline{H_2}\&I_3)\\
=&H_4|(H_3\&(I_1))|(H_2\&I_2)|(H_1\&\overline{H_2}\&I_3)\\
=&H_4|(H_3\&(I_1))(H_2\&(I_2|I_3))|(H_1\&\overline{H_2}\&I_3)\\
=&H_4|(H_3\&I_1)|(H_1\&I_3)|(H_2\&I_2)
\end{aligned}
$$

(16)

EXACT/APPROXIMATE (4:2) COMPRESSORS

A. Exact (4:2) Compressor

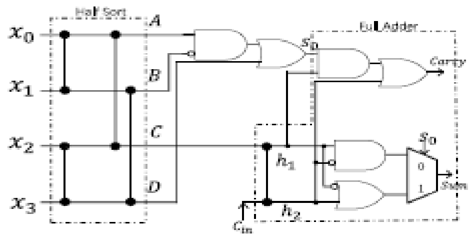

Fig.5. Proposed exact (4:2) compressor.

A (4:2) compressor has the same logical function as Fig.10 shows. To construct a high speed (4:2) compressor, we also introduce sorting networks. The 4-way sorting network, as shown in Fig.2, needs three stages to sort four inputs, and we observed that the last stage of the 4-SN just sorts the two data in the middle, which means that the data at the top and the data at the bottom are the maximum and the minimum of the four data, respectively. We redisplayed the first two stages of a 4-SN in Fig.5 as "Half Sort", and the results of the "Half Sort" are denoted as A, B, C and D. Since A and D are the maximum and minimum data, respectively, the sequence [A,B, D] is already sorted completely. Then the summation of A, B and D can be calculated with equation (29).
s0 = (A&B)jD
Cout = B (29)
The summation of s0, Cin and C is calculated with a "Full Adder" (as shown in Fig.11) which has been modified. Equation (30) describes the "Full Adder".
h1 = CjCin
h2 = C&Cin
Carry = (s0&h1)jh2
Sum = s0?(h1jh2) : (h1&h2) (30)

B. Approximate (4:2) compressors

We also use the name "Yang1", "Yang2" to represent the approximate (4:2) compressors, which have 1 and 2 errors, respectively, proposed in [19]. And we name the approximate (4:2) compressors proposed in [18] that with 1 and 2 errors as "Strollo1" and "Strollo2", respectively.

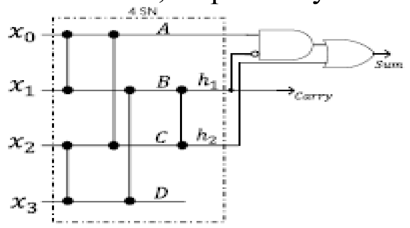

Fig 6. Proposed approximate (4:2) compressor with 1 error

In Fig 6, we construct an approximate (4:2) compressor based on sorting network. D is one of the outputs of 4SN, and it is the minimum one of the inputs. By simply discarding D, the structure is constructed and it has the same logical function as that "Yang1" and "Strollo1" have.

IJMECE International Journal of modern electronics and communication engineering
ISSN 2321-2152
www.ijmece.com
Vol 13, Issue 1, 2025

Carry = A&h1
Sum = h2j(A ^ h2) (31)

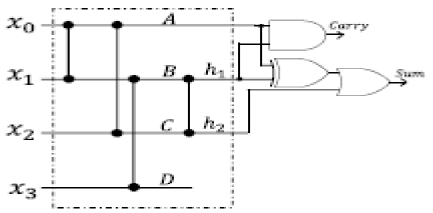

Fig7. Proposed approximate (4:2) compressor with 1 error

To construct a faster approximate (4:2) compressor, a sorter is discarded in 4 SN, as shown in Fig.7. Although it is uncertain that the sequence [A, h1, h2] is sorted completely, we assume that the sequence is sorted completely. In order to correct the deviation introduced by incomplete sorting, the output expressions are modified to equation (31).

16_16 bit multiplier: To show the performance of the proposed counters, we construct 16_16 bit multipliers as application platforms. The proposed (31,5) counter has poor area efficiency, so we do not utilize it in this subsection. The 16_16 bit multipliers' architectures are illustrated in Fig.14(a) and Fig.14(b) for (7,3) counter and (15,4) counter, respectively.

Multipliers with the proposed counters has better ADP and PDP than other designs, and they can also reach lower delay than other designs. So, in some high-performance cases, it will be very suitable, and it also performs well in low-power and area-efficient cases.

2) 8_8 bit approximate multiplier: To make a compression between the proposed and the other exact/approximate (4:2) compressors, we construct a 8_8 bit approximate multiplier which is proposed

RESULTS

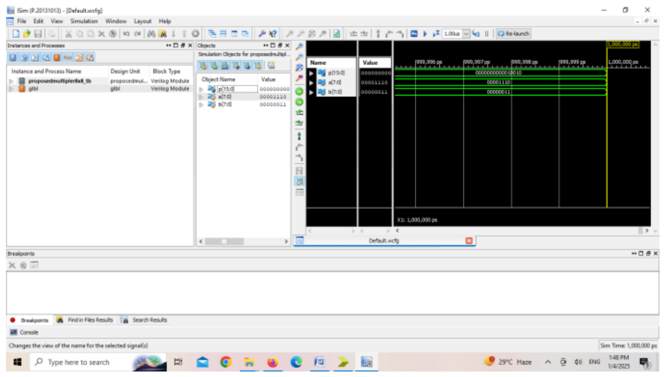

Fig 7.1 Simulation Result

COMPRESSION TABLE

	EXISTING SYSTEM	PROPOSED SYSTEM
AREA	143	124
DELAY	32.681 ns	21.333 ns
SPEED	30.59 Mhz	54.43 Mhz

CONCLUSION AND FUTURE WORK

In this paper, a new counter design method based on sorting network is proposed, and we construct (7,3), (15,4) and (31,5) counters based on this method. The (7,3) counter has 8.1% 27.0% less delay than other designs, and also consumes less area and power. The (15,4) counter is more flexible than existing designs because it achieves 14.9%-35.2% less delay when the speed is critical, and performs 14.7%-49.0% and 41.2%-72.7% better in ADP and PDP when the area or power is critical. The (31,5) counter, has more than 22.0% shorter delay than other existing designs. When they are embedded in a 16-bit multiplier, the multiplier achieves 31.8% and 32.2% better in ADP and PDP in maximum than those embedded in other counter designs. Exact/approximate (4:2) compressors are also proposed based on sorting network. And they performs approximately 10.2%-37.4% better in ADP and 22.3%-48.0% better in PDP when they are embedded in an 8-bit approximate multiplier.

REFERENCES

[1] C. S. Wallace, "A Suggestion for a Fast Multiplier," in IEEE Transactions on Electronic Computers, vol. EC-13, no. 1, pp.

14-17, Feb. 1964, doi:10.1109/PGEC.1964.263830.
[2] R. S. Waters and E. E. Swartzlander, "A Reduced Complexity Wallace Multiplier Reduction," in IEEE Transactions on Computers, vol. 59, no. 8, pp. 1134-1137, Aug. 2010, doi: 10.1109/TC.2010.103.
[3] P. L. Montgomery, "Five, six, and seven-term Karatsuba-like formulae," in IEEE Transactions on Computers, vol. 54, no. 3, pp. 362-369, March 2005, doi: 10.1109/TC.2005.49.
[4] J. Ding, S. Li and Z. Gu, "High-Speed ECC Processor Over NIST Prime Fields Applied With Toom–Cook Multiplication," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 3, pp. 1003-1016, March 2019, doi: 10.1109/TCSI.2018.2878598.
[5] R. Liu and S. Li, "A Design and Implementation of Montgomery Modular Multiplier," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-4, doi: 10.1109/ISCAS. 2019.8702684.
[6] W. Wang, X. Huang, N. Emmart and C. Weems, "VLSI Design of a Large-Number Multiplier for Fully Homomorphic Encryption," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 9, pp. 1879-1887, Sept. 2014, doi: 10.1109/TVLSI.2013.2281786.
[7] S. Asif and Y. Kong, "Analysis of different architectures of counter based Wallace multipliers," 2015 Tenth International Conference on Computer Engineering & Systems (ICCES), Cairo, 2015, pp. 139-144, doi: 10.1109/ICCES.2015.7393034.
[8] A. Najafi, B. Mazloom-nezhad and A. Najafi, "Low-power and high-speed 4-2 compressor," 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2013, pp. 66-69.
[9] A. Najafi, S. Timarchi and A. Najafi, "High-speed energy-efficient 5:2 compressor," 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2014, pp. 80-84, doi: 10.1109/MIPRO.2014.6859537.
[10] M.V. Sruthi "High-performance ternary designs using graphene nanoribbon transistors"in science direct
[11] C. Fritz and A. T. Fam, "Fast Binary Counters Based on Symmetric Stacking," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2971-2975, Oct. 2017, doi: 10.1109/TVLSI.2017.2723475.
[12] Q. Jiang and S. Li, "A design of manually optimized (15, 4) parallel counter", 2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC), Hsinchu, 2017, pp. 1-2, doi: 10.1109/EDSSC.2017.8126527.
[13] M. H. Najafi, D. J. Lilja, M. D. Riedel and K. Bazargan, "Low-Cost Sorting Network Circuits Using Unary Processing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 8, pp. 1471-1480, Aug. 2018, doi: 10.1109/TVLSI.2018.2822300.
[14] D.E. Knuth, The Art of Computer Programming, Volume 3: Sorting and Searching. Addison-Wesley, 1973.
[15] M. Mehta, V. Parmar and E. Swartzlander, "High-speed multiplier design using multi-input counter and compressor circuits," [1991] Proceedings 10th IEEE Symposium on Computer Arithmetic, Grenoble, France, 1991, pp. 43-50, doi: 10.1109/ARITH.1991.145532.
[16] A. Fathi, B. Mashoufi and S. Azizian, "Very Fast, High-Performance 5-2 and 7-2 Compressors in CMOS Process for Rapid Parallel Accumulations," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 6,

pp. 1403-1412, June 2020, doi: 10.1109/TVLSI.2020.2983458.
[17] T. Satish and K. S. Pande, "Multiplier Using NAND Based Compressors," 2019 3rd International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 2019, pp. 1-6, doi: 10.1109/IEMENTech48150.2019.8981067.
[18] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra and G. D. Meo, "Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 9, pp. 3021-3034, Sept. 2020, doi: 10.1109/TCSI.2020.2988353.