Research Article

Optimized Counter Design for Accelerated Summation in Digital Signal Processing Systems

G. Vishwanath

Vice Principal, Associate Professor and Head, Department of Electronics and Communication Engineering
Kakatiya Institute of Technology and Science for Women, Manik Bhandar, Nizamabad, Telangana, India.

Abstract: Efficient summation of multiple operands in parallel is crucial in various digital signal processing units. Accelerating this summation process requires high compression ratio counters and compressors. This study introduces a novel approach to fast saturated binary counters and exact/approximate (4:2) compressors utilizing sorting networks. The counter's inputs are asymmetrically divided into two groups and processed through sorting networks to produce reordered sequences, efficiently represented by one-hot code sequences. By establishing three special Boolean equations between the reordered sequence and the one-hot code sequence, the output boolean expressions of the counter are significantly simplified. Simulation results demonstrate the superiority of the proposed method over conventional approaches in terms of performance.

Keywords: Binary Counter, Exact/Approximate 4:2 Compressor, Multiplier, One-Hot Code, Sorting Network.

1. 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 adds all the partial products up with the Wallace Tree structure [1], whose performance is the bottleneck of the basic multiplier. Public-key cryptosystems, such as RSA and elliptic curve cryptography (ECC), use a big number multiplier based on the Toom-Cook [4] or Karatsuba algorithm [3] to construct modular multipliers. Many papers have studied these two algorithms and implemented them with hardware. In the papers, such as [5], many parts of the circuit utilize the summation of multiple operands. Fully homomorphic encryption (FHE) is a post-quantum cryptosystem that 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 method of multiple operands summation is the Wallace tree structure [1] and its improved method reduced Wallace tree [2]. 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. Since then, many papers have discussed how to construct a more time-efficient structure to accelerate the summation, such as [7]– [12]. The main idea is to construct a counter or a compressor with a higher compression ratio than the (3,2) counter by considering more bits at the same weight.

The compressors compress n rows into 2 rows by considering the carry bits between adjacent columns. Some papers have discussed (4,2) [8], (5,2) [9], and (7,2) [16] compressors, which compress four, five, or seven rows into two rows, respectively. However, they are still in the framework of full adders, which utilizes "XOR" gates as basic units, and their logical expressions are very difficult to simplify. The counters compress n rows into log2n rows. Some papers have discussed (4,3), (5,3), and (6,3) [10] and even (7,3) [11] and (15,4) [12] counters. They count the number of "1" s in the inputs. If a counter is a saturated counter, whose compressed results just right represent all the number of "1" s in the inputs, its compression efficiency achieves the limitation. The designs in [10] are all unsaturated and use too many "XOR" gates. Saturated counters are the main topics in this article. For example, (7,3) counter is a saturated counter because a 3-bit number can just right represent 0–7. Fritz and Fam [11] proposed a (6,3) counter that is constructed with a symmetric stacking structure. It is very fast compared with other designs but unsaturated. Then, Fritz and Fam [11] simply use a MUX on the critical path to construct a saturated (7,3) counter that affects the speed. Satish and Pande [17] reduce the (6:3) counter proposed in [11] to a (5:3) counter and combine three (5:3) counter into a (15:4) counter. However, this method is inefficient. Approximate multipliers are widely used in many error tolerant fields, such as digital image processing [18] and finite impulse response (FIR) filter [24], to accelerate the multiplication. Manikantta Reddy et al. [21] and Zervakis et al. [23] use approximate booth encoding and partial product perforation to get an approximate multiplier. Satish and Pande [17], Strollo et al. [18], Akbari et al. [20], and Venkatachalam and Ko [22] have discussed about approximate (4:2) compressors that can be used in approximate multipliers. The high-speed approximate (4:2) compressors in [18] are based on the symmetric stacking structure [11].

2. Proposed Method

Sorting Network Working Principle

The typical three- and four-way sorting networks are shown in Fig. 1. Each vertical line represents a sorter that 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. 1, This work considers 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).

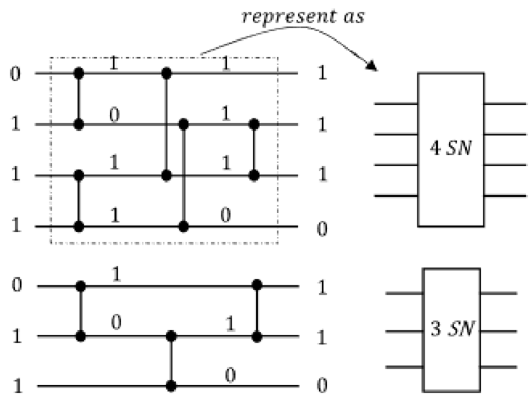


Fig 1. Three- and four-way sorting networks.

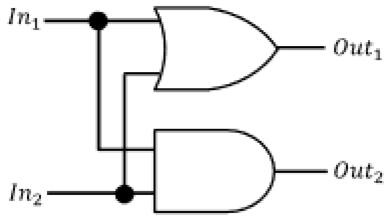


Fig.2. Two-input binary sorter

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.

Sorter for 1-Bit Data: 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 a sorter consumes one layer of two-input basic logic gates, and the three- and four-way sorting networks both consume three layers of two-input basic logic gates.

2.2. Proposed (7,3) counter.

This work constructs an efficient (7,3) counter. As the main comparison object, this work first briefly reviews the design in. Fritz and Fam 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 performance is worse because of simply introducing a MUX on the critical path without any optimization. To solve the problem, this work proposes this method of directly construct a (7,3) counter. Unlike the symmetric stacking structure, this work starts with two sorting networks asymmetrically, as illustrated in Fig. 2. By generating one-hot code sequences, this work establishes three special Boolean equations [see (2), (13), and (15), which significantly simplifies the Boolean expressions related to outputs.

Some Characteristics of Sorting Network: This work summarizes two characteristics of sorting networks. 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, 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, this work 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.

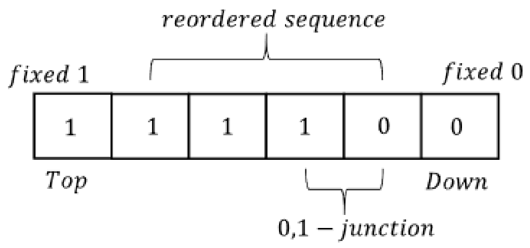


Fig.3. Definition of a sequence.

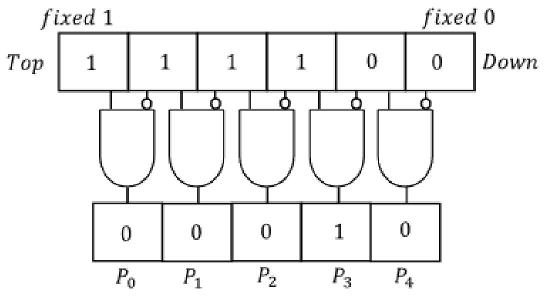


Fig.4. One-hot code generation circuit

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 This work ignore it while counting.

One-Hot Code Generation:

1) Asymmetric Prereorder: As illustrated in Fig. 2, 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, this work divides 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: As shown in Fig. 4, 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, this work still utilizes the four-way sorting network as an example, and then, This work have the structure in Fig. 5.

Table 1: 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

This structure uses a Boolean expression (ABbar) 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 "OR" and "&" represents "AND")

$$P_0|P_1|P_2|P_3|P_4 = 1.$$

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, this work have.

$$P_0|P_2|P_4 = \overline{P_3|P_4}.$$

All results of random separation satisfy this rule. This work also applies the same method on the three-way sorting network's output sequence and get the one-hot code sequence Q_0–Q_3. This sequence also satisfies the rule above.

Output Generation:

1) Basic Output Generation: Now, this work has 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 is 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 = $2^2C_2 + 2^1C_1 + 2^0S$. The sequence output from four-way sorting network is denoted as sequence H, including H_1–H_4 from left to right in Fig. 4. The sequence output from three-way sorting network is denoted as sequence I, including I_1–I_3 from left to right.

According to Table I, This work 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

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

Notice that the sequence Q, with the same method, satisfies.

$$Q_0|Q_1|Q_2|Q_3 = 1$$
$$Q_1|Q_2|Q_3 = \overline{Q_0}$$

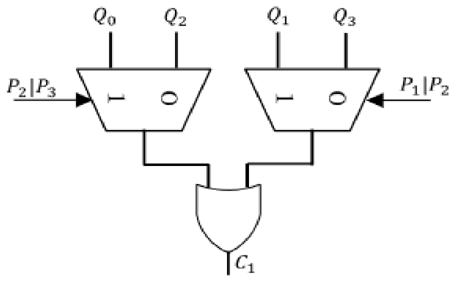


Fig.5. C_1 generating circuit.

$$C_2 = P_4\big|\big(P_3\&\overline{Q_0}\big)\big|\big(P_2\&\big(Q_2\big|Q_3\big)\big)\big|(P_1\&Q_3)$$

As for C_1, the sum of subscripts of sequences P and Q equals 2, 3, 6, and 7; then, C_1 = 1 (see Table I). Thus, this work gets the following equation:

$$C_1 = \big(Q_0\&\big(P_2\big|P_3\big)\big)\big|\big(Q_1\&\big(P_1\big|P_2\big)\big)\big|$$
$$(Q_2\&(P_0|P_1|P_4))\big|(Q_3\&(P_0|P_3|P_4))$$

Note that.

$$P_0|P_1|P_4 = \overline{P_2|P_3}$$
$$P_0|P_3|P_4 = \overline{P_1|P_2}$$

Equation is reduced to the following equation:

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

They construct two multichannel selection constructions. Via the circuit shown in Fig. 6, C1 can be calculated time efficiently. As for S, it can be easily obtained by the following equation: S = (P1|P3) ⊕ (Q1|Q3)

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

where ⊕ denots "XOR."

2) Further Optimization: This work has got two sequences H1–H4 and I1–I3. Here, this work extends 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, this work has the following equation:

$$P_i = H_i \& \overline{H_{i+1}}, \quad i = 0, 1, 2, 3, 4$$
$$Q_i = I_i \& \overline{I_{i+1}}, \quad i = 0, 1, 2, 3.$$

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

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

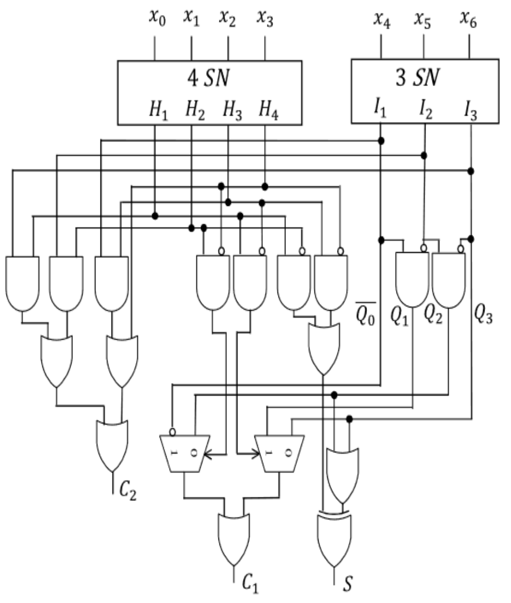


Fig.6. Overall (7,3) counter circuit.

Based on this, (10) is simplified to (14), which can also be calculated from the circuit in Fig. 6

$$\begin{aligned} C_1 &= (Q_0\&(P_2|P_3))|(Q_1\&(P_1|P_2))|(Q_2\&(\overline{P_2|P_3})) \\ &\quad |(Q_3\&(\overline{P_1|P_2})) \\ &= (Q_0\&(H_2|\overline{H_4}))|(Q_1\&(H_1|\overline{H_3}))|\left(Q_2\&\left(\overline{H_2|\overline{H_4}}\right)\right. \\ &\quad \left|\left(Q_3\&\left(\overline{H_1|\overline{H_3}}\right)\right)\right.. \end{aligned}$$

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 and so is the sequence Q. Then, this work gets the following equation:

$$I_i = I_i | I_{i+j}, \quad (i = 0, 1, 2, 3;\ j \ge 0;\ i + j \le 4)$$
$$H_i = H_i | H_{i+j}, \quad (i = 0, 1, 2, 3, 4;\ j \ge 0;\ i + j \le 5)$$

Overall Structure

The overall architecture is shown in Fig. 7. It is obvious that the paths from sequences H and I to C2, C1, and S is almost independent. This increases the parallelism of the circuit. However, as will be discussed later, the area of the proposed design will not increase with the parallelism. In fact, it decreases. The main reason for the area reduction is (2), (13), and (15), which significantly simplifies the output Boolean expressions.

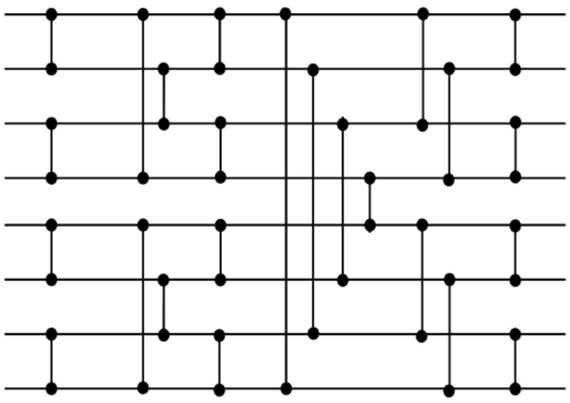


Fig.7. Eight-way sorting network

2.3 Compression Ratio Counters

This work has implemented an efficient (7,3) saturated counter, but, in some applications, such as high radix NTT [6] and large number multiplication [5], a (15,4) even (31,5) saturated counter will be useful. This work utilizes the above methods to construct (15,4) and (31,5) saturated counters and give them a brief but clear description.

Construct (15,4) Counter:

1) Seven- and Eight-Way Sorting Networks: Fig. 8 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.

$$\begin{aligned} Q_0|Q_2|Q_4|Q_6 &= \overline{Q_1|Q_3|Q_5|Q_7} \\ P_0|P_2|P_4|P_6|P_8 &= \overline{P_1|P_3|P_4|P_5|P_7} \\ I_i &= I_i|I_{i+j}, \end{aligned}$$

The outputs of the eight-way and seven-way sorting networks are denoted as sequence H (includes H_1–H_8 and extended to $H_0 - H_9$) and sequence I (includes I_1–I_7 and extended to I_0–I_8), respectively. By utilizing A&B logic, this work gets the one-hot code sequences P (P_0–P_8) and Q ($Q_0 - Q_7$). These sequences are similar to the sequences in (7,3) counter.

Table 2: 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

$$H_i = H_i|H_{i+j}, \quad (i = 0, 1, \ldots, 7, 8;\ j \geq 0;\ i + j \leq 9)$$

$$\sum_{n=i}^{n=j} Q_n = I_i \& \overline{I_{j+1}}, \quad (0 \leq i \leq j \leq 7)$$

$$\sum_{n=i}^{n=j} P_n = H_i \& \overline{H_{j+1}}, \quad (0 \leq i \leq j \leq 8)$$

Thus, This work directly give out the key equations, as shown in (17)–(19). Note that (17) is just an example; this regular satisfies all the random separation. The symbol “Σ” in (19) represents continuous “OR.”

Boolean Expressions of (15,4) Counter: The 4-bit output of the (15,4) counter is denoted as $C_3C_2C_1$ S. Table 3 shows the output corresponding to the number of the number of “1”s in input sequence. Similar to the method by which this work constructs the (7,3) counter, first, establish logic equations between $C_3C_2C_1$ S and sequences P and Q through the sum of the subscripts. Second, utilize (17)–(19) to optimize it. Then This work get (20)– (23). Because the original Boolean expressions are too long, here, this work express them with the Verilog syntax (especially ternary operator: “ab:c”).

$$\begin{aligned}
S &= (P_1|P_3|P_5|P_7) \oplus (Q_1|Q_3|Q_5|Q_7)\\
C_1 &= (((H_2\&\overline{H_4})\big|(H_6\&\overline{H_8}))?(Q_0|Q_4) : (Q_2|Q_6))\big|\\
&\quad (((H_1\&\overline{H_3})\big|(H_5\&\overline{H_7}))?(Q_1|Q_5) : (Q_3|Q_7))\\
C_2 &= ((H_4\&\overline{H_8})?Q_0 : Q_4)\big|((H_3\&\overline{H_7})?Q_1 : Q_5)\big|\\
&\quad ((H_2\&\overline{H_6})?Q_2 : Q_6)\big|((H_1\&\overline{H_5})?Q_3 : Q_7)\\
C_3 &= (H_1\&I_7)|(H_2\&I_6)|(H_3\&I_5)|(H_4\&I_4)|(H_5\&I_3)|\\
&\quad (H_6\&I_2)|(H_7\&I_1)|H_8.
\end{aligned}$$

Overall Structure:

The overall structure is shown in Fig. 9. “HI BUS” in Fig. 9 is a module, which mainly contains AB logic gates for calculating the related signals in (20)–(23). Signals R1 − R8 are related to (23). In (23), seven “AND” operations are needed between sequence H and sequence I, and R_1–R_8 denote the results of “AND”, i.e., $R_1 = H_1\&I_7$, $R_7 = H_7\&I_1$, and $R_8 = H_8$.

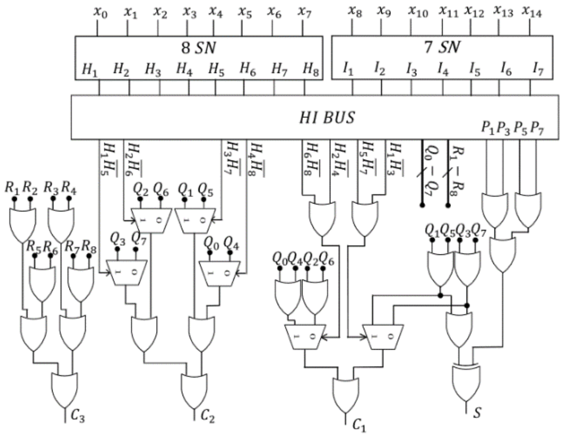

x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7
x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14}
8 SN
7 SN
H_1 H_2 H_3 H_4 H_5 H_6 H_7 H_8
I_1 I_2 I_3 I_4 I_5 I_6 I_7
HI BUS
$P_1P_3P_5P_7$
$Q_0 - Q_7$
$R_1 - R_8$
C_3
C_2
C_1
S

Fig.8. Overall (15,4) counter circuit

The critical path is from the input to C2, which consumes nine layers of 2IBLGs and a MUX. This is less than the structure in that needs six 2IBLGs and four XOR gates. Besides that, the proposed design needs 150 2IBLGs (the same calculation method as in Table II), while the design in [12] needs 135 2IBLGs. This area mainly comes from eight- and seven-way sorting networks, which consumes a total number of 84 2IBLGs. However, as will be discussed in Section VI.B, this design is flexible enough to be applied in high-performance or area-efficient scenarios.

Construct (31,5) Counter:

A high compression ratio (31,5) counter is also constructed with the proposed method. There are still three steps. First, divide the 31 bits into two parts. The two parts contain 16 and 15 bits, respectively. Then, put the two parts into two sorting networks of corresponding sizes. The outputs of sorting networks are extended by the fixed “0” and the fixed “1,” which are denoted as H_0–H_{17} and I_0–I_{16}, respectively. Second, one-hot code sequences P_0–P_{16} and Q_0–Q_{15} are generated by using the AB Boolean expression. Three trick Boolean expressions between reordered sequences and one hot code sequences are established, which has the same form as (17)–(19). Finally, generate and simplify the output expressions.

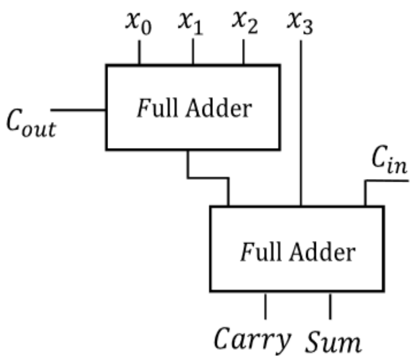


Fig. 9. 4:2 Compressor Combined by Full Adders.

This work directly gives out the outputs $C_4C_3C_2C_1$ S in (24)– (28), as shown at the bottom of the page. The independence of the equations increases the parallelism, so the proposed (31,5) counter is fast. The critical path is from input to C2, which consumes 13 layers of 2IBLGs and a MUX. This work observes the output equations of the proposed (7,3), (15,4), and (31,5) counters carefully, this work will find that, when This work doubled the input bit number from 7 to 15 (or from 15 to 31), the length of the equations increases, but the logic layers corresponding to circuits do not increase too much because of the abovementioned parallelism.

2.4 Exact/approximate (4:2) compressors: Exact (4:2) Compressor:

A (4:2) compressor has the same logical function, as shown in Fig. 10. To construct a high-speed (4:2) compressor, this work also introduces sorting networks. The four-way sorting network, as shown in Fig. 2, needs three stages to sort four inputs, and This work 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. This work redisplayed the first two stages of a 4-SN in Fig. 11 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 the following equation:

$$s_0 = (A\&\overline{B})|D$$
$$C_{\text{out}} = B.$$

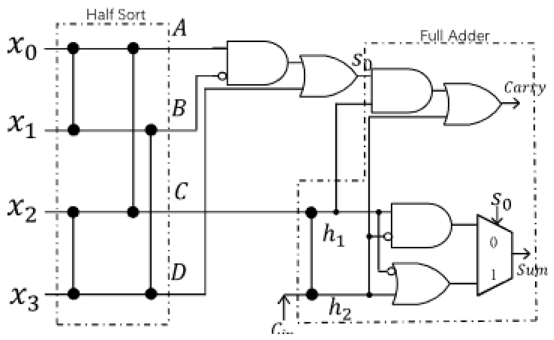


Fig.10. Proposed exact (4:2) compressor.

The summation of s_0, C_{in}, and C is calculated with a “Full Adder” (as shown in Fig. 11), which has been modified. Equation (30) describes the “Full Adder”.

$$\begin{aligned} h_1 &= C|C_{\text{in}} \\ h_2 &= C\&C_{\text{in}} \\ \text{Carry} &= (s_0\&h_1)|h_2 \\ \text{Sum} &= s_0?(\overline{h_1}|h_2) : (h_1\&\overline{h_2}) \end{aligned}$$

Approximate (4:2) Compressors:

This work also uses the name “Yang1” and “Yang2” in [18] to represent the approximate (4:2) compressors, which has 1 and 2 errors, respectively, as proposed in [19]. This work names the approximate (4:2) compressors proposed in [18] with one and two errors as “Strollo1” and “Strollo2,” respectively.

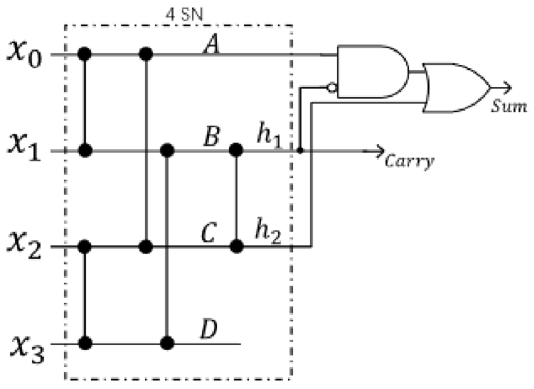


Fig.11. Proposed approximate (4:2) compressor with one error.

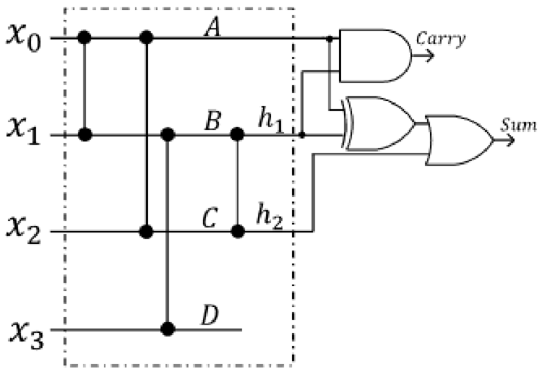


Fig. 12. Proposed approximate (4:2) compressor with two errors.

In Fig. 12, This work constructs 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.

$$\text{Carry} = A \& h_1$$
$$\text{Sum} = h_2 | (A \oplus h_2)$$

To construct a faster approximate (4:2) compressor, a sorter is discarded in 4 SN, as shown in Fig. 13. Although it is uncertain that the sequence [A, h1, h2] is sorted completely, this work assume that the sequence is sorted completely. In order to correct the deviation introduced by incomplete sorting, the output expressions are modified.

3. Simulation and synthesis analysis

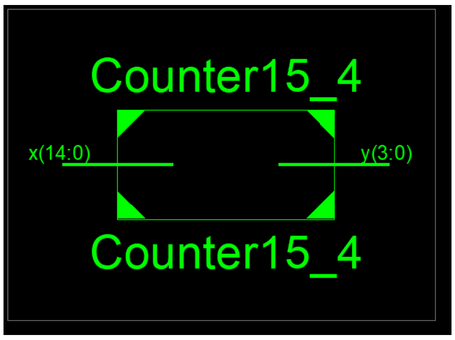


Fig. 13: Top module for 15:4 counter

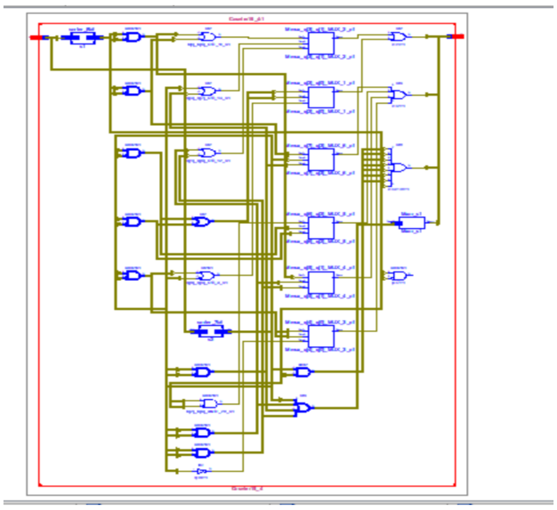


Fig 14: RTL schematic for 15:4 counter

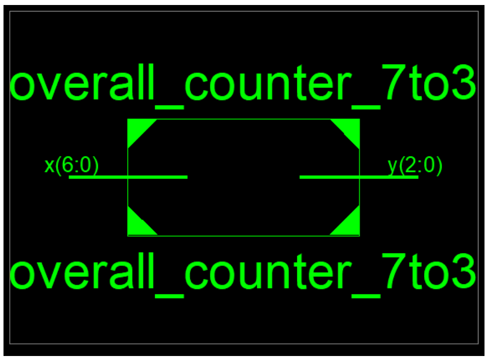


Fig 15: Top module for 7:3 counter

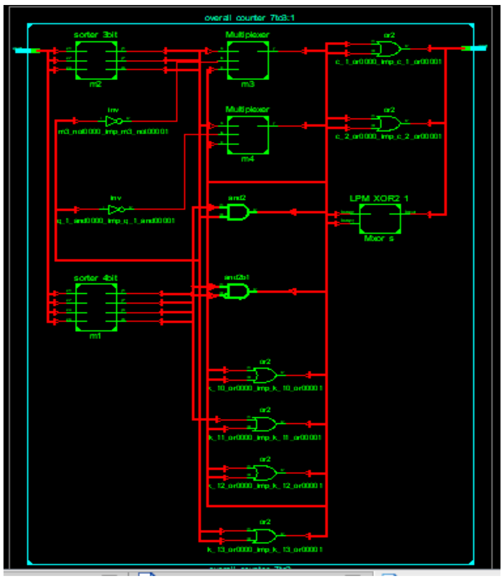


Fig 16: RTL schematic for 7:3 counters.

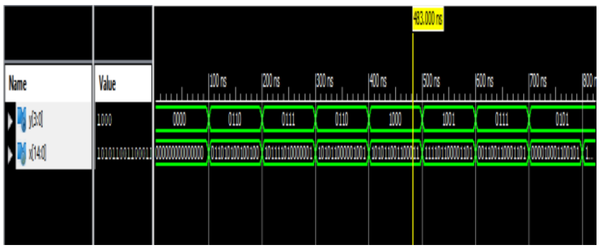


Fig 17: simulation result for 15:4 counter

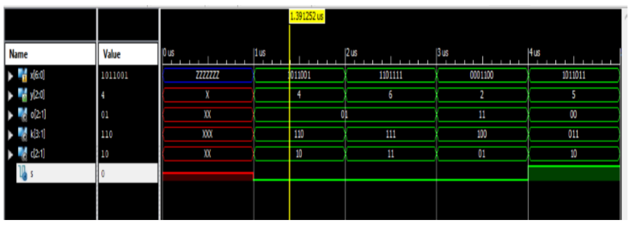
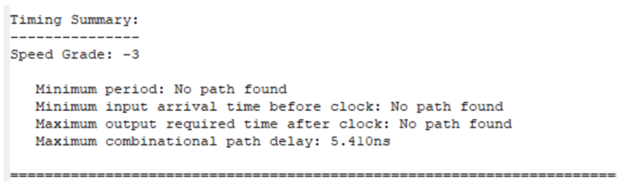


Fig 18: simulation result for 7:3 counter

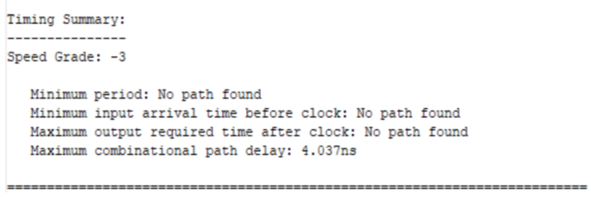


```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 5.410ns

========================================================================
```

Fig 19: synthesis report for Delay gates in existing 15:3 counter



```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 4.037ns

========================================================================
```

Fig 20: Synthesis Report for Delay in proposed 15:4 counter.

4. Conclusion

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. This work presents a novel method of fast saturated binary counters and exact/approximate (4:2) compressors based on the 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. Both simulation and synthesis were done using Xilinx 14.5. Using the above method, this work construct and further optimize the (7,3) counter that can perform 27.0%, 26.2%, and 52.0% better in maximum than other designs in delay, area-delay product, and power-delay product, respectively. Similarly, the (15,4) counter is constructed, and it achieves approximately 35.3% shorter delay, while it significantly consumes less power and area.

References:

[1]. C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron. Comput., 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," IEEE Trans. Comput., 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,"IEEE Trans. Comput., vol. 54, no. 3, pp. 362–369, Mar. 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 Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 3, pp. 1003–1016, Mar. 2019, doi:10.1109/TCSI.2018.2878598.

[5]. R. Liu and S. Li, "A design and implementation of montgomery modular multiplier," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Sapporo,Japan, May 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 Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 9, pp. 1879–1887, Sep. 2014, doi: 10.1109/TVLSI.2013.2281786.

[7]. S. Asif and Y. Kong, "Analysis of different architectures of counter based wallace multipliers," in Proc. 10th Int. Conf. Comput.Eng. Syst. (ICCES), Cairo, Egypt, Dec. 2015, pp. 139–144, doi:10.1109/ICCES.2015.7393034.

[8]. A. Najafi, B. Mazloom-nezhad, and A. Najafi, "Low-power and highspeed 4-2 compressor," in Proc. 36th Int. Conv. Inf. Commun. Technol.,Electron. Microelectron. (MIPRO), Opatija, Croatia, May 2013,pp. 66–69.

[9]. A. Najafi, S. Timarchi, and A. Najafi, "High-speed energy-efficient 5:2 compressor," in Proc. 37th Int. Conv. Inf. Commun. Technol., Electron.Microelectron. (MIPRO), Opatija, Croatia, May 2014, pp. 80–84, doi:10.1109/MIPRO.2014.6859537.

[10]. 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, doi: 10.1109/ICCES.2015.7393033.

[11]. C. Fritz and A. T. Fam, "Fast binary counters based on symmetric stacking," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 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," in Proc. Int. Conf. Electron Devices Solid-State Circuits (EDSSC), Hsinchu, Taiwan, Oct. 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," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 8, pp. 1471–1480, Aug. 2018, doi:10.1109/TVLSI.2018.2822300.

[14]. D. E. Knuth, The Art of Computer Programming: Sorting and Searching,vol. 3. Reading, MA, USA: Addison-Wesley, 1973

[15]. M. Mehta, V. Parmar, and E. Swartzlander, "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, 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,"IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 6, pp. 1403–1412, Jun. 2020, doi: 10.1109/TVLSI.2020. 2983458.

[17]. T. Satish and K. S. Pande, "Multiplier using NAND based compressors,"in Proc. 3rd Int. Conf. Electron., Mater. Eng. Nano-Technol. (IEMENTech), Kolkata, India, Aug. 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," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 67, no. 9, pp. 3021–3034, Sep. 2020, doi:10.1109/TCSI.2020.2988353.

[19]. Z. Yang, J. Han, and F. Lombardi, "Approximate compressors for error resilient multiplier design," in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFTS), Amherst, MA, USA, Oct. 2015, pp. 183–186, doi: 10.1109/DFT.2015.7315159.

[20]. O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers,"IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4,pp. 1352–1361, Apr. 2017, doi: 10.1109/TVLSI.2016.2643003.

[21]. K. Manikantta Reddy, M. H. Vasantha, Y. B. Nithin Kumar, and D. Dwivedi, "Design of approximate booth squarer for error-tolerant computing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 5, pp. 1230–1241, May 2020, doi: 10.1109/TVLSI.2020.2976131.

[22]. S. Venkatachalam and S.-B. Ko, "Design of power and area efficient approximate multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 5, pp. 1782–1786, May 2017,doi:10.1109/TVLSI.2016.2643639.

[23]. G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi,"Design-efficient approximate multiplication circuits through partial product perforation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 10, pp. 3105–3117, Oct. 2016, doi:10.1109/TVLSI.2016.2535398.

[24]. W. Liu et al., "Design and analysis of approximate redundant binary multipliers," IEEE Trans. Comput., vol. 68, no. 6, pp. 804–819, Jun. 2019, doi: 10.1109/TC.2018.2890222.