EasyChair Preprint

№ 8653

Implementation of 24 Bit Multiplier Using Parallel Multiplication with Sorting Based Binary Counters for VLSI Applications

Pinniboyina Anand Venkat Seshu Babu and K. Rajasekhar

August 11, 2022

IMPLEMENTATION OF 24 BIT MULTIPLIER USING PARALLEL MULTIPLICATION WITH SORTING BASED BINARY COUNTERS FOR VLSI APPLICATIONS

Pinniboyina Anand Venkat Seshu Babu[1], Dr. K. Rajasekhar[2]
Student[1], M.Tech, VLSI & Embedded Systems, Dept. of ECE, UCEK(A), JNTUK, Kakinada, India.
Assistant Proffesor[2], Dept. of ECE, UCEK(A), JNTUK, Kakinada, India.
Email: anand.venkat56@gmail.com[1], rajakarumuri87@gmail.com[2]

ABSTRACT:

This project consists of an innovative way of rapid counters such as (7, 3), (15, 4) etc., binary counters and approximate (4:2) compressors which is based on sorting network. For improving the speed, a high compressor counters need to be employed. The counter inputs are divided asymmetrically into two pieces and then fed into sorting networks as inputs to construct sequences that are represented solely by one-hot code sequences. We can develop and further refine the (7, 3) counter using this method, which outperforms alternative designs in terms of latency, overall area, and power consumption in the vast majority of circumstances. A (15, 4) counter is developed, and it has a lower latency despite using less power and taking up less space. In addition, using a sorting network, we create approximation compressors (4:2). They built an 8 x 8, 16 x 16 bit multiplier to examine the performance of the circuits they constructed. A 24Bit multiplier is made and the effectiveness of the design is synthesized and simulated using Xilinx Vivado.

Keywords: - Binary counters, sorting network, approximate 4:2 compressor, 24 bit multiplier, one-hot code.

INTRODUCTION

The Wallace Tree structure, which is the basic multiplier's bottleneck, sums up all the partial products in a basic multiplier circuit. The summing of several operands is used in various areas of the circuit. The summation of several operands is used as the primary processing. A Wallace tree structure is well-known way of summarizing numerous operands, and its upgraded version, Wallace tree, is even more well-known. To speed up the summing, these approaches employ complete adders as (3,2) counters. The architecture used is referred to as a carry–save. Then onwards several research been published that look at ways to build a framework that speed up summing process. The fundamental concept is to use further bits at the same weight to build a counter or compressor having a greater compressing ratio comparing with (3,2) counter. By looking at the carry bits between neighboring columns, the compressors compress n rows into 2 rows. Compressors that compress four, five, or seven rows into two rows have been explored in certain articles (4,2), (5,2), and (7,2), respectively. They are, however, still part of the whole adder structure, which uses XOR gates as fundamental unit, and its logic are hard to reduce.

N rows are compressed into log2n rows by the counters. A symmetric stacking structure was

presented. It is really quick in comparison to other designs, however it is unsaturated. Then, on the crucial route, they employ a MUX to build a (7,3) saturated counter which influences the swiftness. Further recommended reducing the intended (6:3) counter to a (5:3) counter and combining three (5:3) counters to form a (15:4) counter. This strategy, however, is ineffective. We begin by reviewing the design as the principal comparison object. They suggested a rapid counter having symmetric stacking structure, and based on this (6,3) counter, they built a (7,3) saturated counter. Although it is the quickest of the seven counter designs (7,3) but lacks performance in delay reduction because it adds a multiplexer to route by not optimizing it. In contrast to the symmetric stacking structure, we begin by asymmetrically stacking two sorting networks. Approximate multipliers are commonly employed to speed up multiplication. Estimated booth encoding and partial product perforation are used to obtain an approximate multiplier.

A symmetric stacking structure is used in high-speed approximation (4:2) compressors. There are saturation counters with better efficiency in this design, namely (7,3) and (15,4). We begin by categorizing networks asymmetrically in this collection of designs. The new design is then optimized via logical simplification with the use of two extended bits. With the sorting network, we can also make exact/approximate (4:2) compressors. The sorting network is a high-performance parallel hardware network for sorting data. Any number can be sorted if a sorting network can sort a batch of data whose constituents are all 1-bit integers, according to the famous 0,1 principle. Only 1-bit data sorting is used in this article.

The standard 3&4 way sorting networks. A) Sorting Network Working Principle: Every standing bar is a sorter with 2 data inputs and outputs, with all data being one-bit values. The larger input is always sorted first, followed by the smaller.

B) 1-Bit Data Sorter: From previously stated, the sorter rearranges 2 inputs based on number. The logical circuit can easily sort two 1-bit data sets. A sorter uses 1 layer of 2-input basic logic gates, whereas 3 &4 way sorting networks use 3 layers of 2-input basic logic gates.

We doesn’t require exact or high-precision computing if small mistakes have no major impact on the results. Hence, approximation computing is encouraged as a novel method having high accuracy.

There is a system design in approximate computing which is having high-performance and is energy-efficient. Because addition mistakes are more sensitive than multiplication errors in such complicated computations, multipliers may tolerate a larger approximation than adders. Along with gain in factors such as performance and power consumption for these applications, the adoption of approximation arithmetic circuits will increase the image processing quality and deep learning. Rapid system with reduction in complex nature and consumed power emerge from arithmetic processes that are approximated. The compromise is to be a loss of accuracy, which would not necessarily impede machine learning and multimedia applications in their usual operation. The human eye's inability to discern subtle differences in photos and videos.

I. EARLIER WORK

An efficient (7, 3) counter was designed by making sequence of one-hot code. They have established 3 Boolean that reduces the Boolean equations having outputs.

By looking at the fig. 1 we can observe a (7, 3) counter. The H and I sequence are not depending on C2, C1 and S. Hence, an efficient (7, 3) saturated counter is made.

1) 7 & 8 Way Sorting Networks: To produce the outcome, this sorting network employs 6 layers of fundamental logic gates. By deleting 1 bit from an 8 way sorting network, a 7 way sorting network with 6 layers of basic logic gates may be obtained. Sequence H (includes H1–H8 and is extended to H0–H9) and sequence I (includes I1–I7 and is extended to I0–I8), respectively, are the 8 way and 7 way output of sorting networks. The one-hot code sequences P (P0–P8) and Q (Q0–Q7) are obtained by employing A&B logic. These sequences like those in the counter (7, 3).

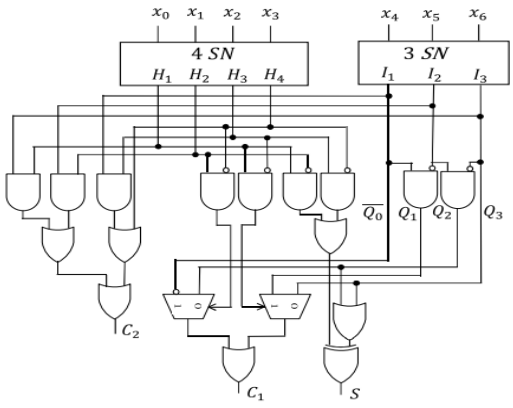

Fig. 1: (7,3) counter

2) (15, 4) Counter: C3C2C1 S is the 4-bit output of the (15, 4) counter. First, develop logic equations between C3C2C1 S and sequences P and Q using addition of the subscripts, similar to how we did with the (7, 3) counter. We adopt the Verilog syntax to express the original Boolean statements because they are excessively lengthy.

3) Overall Structure: HI BUS is a module that mostly consists of AB logic gates used to calculate the relevant signals. Between sequences H and I, seven AND operations are required, and the results are denoted by R1–R8, i.e., R1 = H1&I7,..., R7 = H7&I1, and R8 = H8.

The logical purpose of a (4:2) compressor is similar. This also offer sorting networks to help build a rapid (4:2) compressor. The output expressions have been changed to rectify the divergence caused by inadequate sorting.

II. PROPOSED WORK

Multiplication, division, addition, subtraction, cubing, squaring, and other arithmetic operations are performed by Arithmetic Logic Units (ALUs). Multiplication is the most basic and commonly utilized operation in ALUs of all the operations. It enables the scaling of one integer by another.

Two separate 24 bits numbers are multiplied by a 24 bit multiplier.

Because 1 is greater than 0, when there are ones, all 1s are at the beginning of the series, and when there are 0s, all 0s are at the end, as seen in Fig. A sequence's definition. The reordered sequence must have a place where the two 1s and 0s meet if both ones and zeros exist. If the sequence only contains ones and zeros, manipulate it by inserting one at top and zero at down to make sure that the 0, 1-junction exits indefinitely.

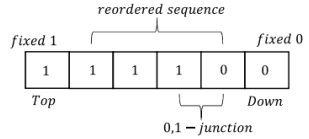

Fig. 2: Definition of a sequence.

The reordered sequence consists of same number of 1s and 0s as the original. Also the added ones will affect the overall number of 1s in the new sequence, it is established therefore it is ignored while counting.

As a result, the 3&4 way sorting networks take similar amount of time to complete. We divide a (7, 3) counter's seven inputs into two sections based on this. There are four bits in one portion and three bits in the other.

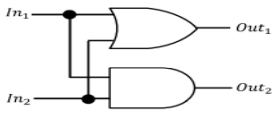

Fig. 3: Two-input binary sorter.

Find the code sequences 0, 1-Junction and One-Hot encoding. The junction represents a rearranged sequence under the extended fixed one and zero. As a result, we'll keep using the 4 SN as an example, resulting in the structure depicted in Figure 1. The design uses an equation to generate a modified sequence. There is only one 1 in the sequence because the reorganized sequence consists of one junction only.

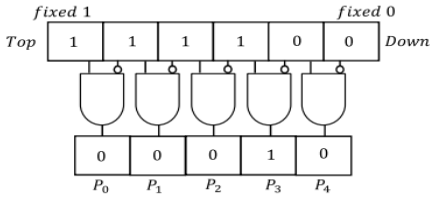

Fig. 4: One-hot code generation circuit.

Thus it states that the sequence P0–P4 is nothing but a one hot encoding which satisfy the needs.

$$P0|P1|P2|P3|P4 = 1. \quad (1)$$

The sequence is arbitrarily split into two parts which is given in equation (2).

$$P0|P2|P4 = P3|P4. \quad (2)$$

The output sequence coming from 3SN, exact procedure is followed to get one hot encoding sequence Q0–Q3.

1) To generate output we got two sequence P and Q. P0 = 1 indicates that the 4N sorting network's input sequence has no ones, P1 = 1 shows that there is one 1, Pi = 1 indicates that the input sequence contains I 1 s, and Q indicates that the input sequence contains I 1 s. C2, C1, and S are the outputs (7,3) counter where C2 have more weight and S have the least weight. 22C2 + 21C1 + 20 S = 22C2 + 21C1 + 20 S = 22C2 + 21C1 + 20 S = 22C

When C2 = 1 at least four 1s are present in the input sequence of the (7, 3) counter. P4 = 1 denotes that there are four 1s in sequence H, while Q0 = 1 denotes that there are no 1s in sequence I. P4&Q0 = 1 shows that there are 4 + 0 = 4 1s in the 7 bits input. Thus, C2 = 1 when total subscripts of P and Q are equal or greater than 4. As a result, C2 may be written as

$$C2 = (P4\&(Q0|Q1|Q2|Q3))|(P3\&(Q1|Q2|Q3))|(P2\&(Q2|Q3))|(P1\&Q3). \quad (3)$$

Equation t the sequence Q, by same procedure in (2), satisfies

$$Q0|Q1|Q2|Q3 = 1 \quad (4)$$

$$Q1|Q2|Q3 = Q0. \quad (5)$$

Put (4) and (5) into (3), we get

$$C_2 = P_4|(P_3\&\overline{Q_0})|(P_2\&(Q_2|Q_3))|(P_1\&Q_3). \quad (6)$$

C1 equals 1 when the addition of the subscripts of the sequences P and Q equals 2, 3, 6, and 7. As a result, we have:

$$C_1 = (Q_0\&(P_2|P_3))|(Q_1\&(P_1|P_2))| (Q_2\&(P_0|P_1|P_4))|(Q_3\&(P_0|P_3|P_4)). \quad (7)$$

$$P_0|P_1|P_4 = \overline{P_2|P_3} \quad (8)$$
$$P_0|P_3|P_4 = \overline{P_1|P_2}. \quad (9)$$

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

$$S = (P1|P3) \oplus (Q1|Q3). \quad (11)$$

where $\oplus$ denotes XOR.

2) Further Optimization: We are having 2 sequences H1–H4 and I1–I3. Here, we extend sequence H1–H4 by H0 and H5. Repeat for sequence I. I0 is fixed 1, and I4 is fixed zero. The obtained equations are:

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

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

Furthermore, the subsequences from the sequence Q or P are supplied, and their subscripts are consecutive, the result of adding them up may be simply described by sequence I or H.

In the earlier works a counter and compressors have been implemented that reduces the count of multipliers. In the proposed design a similar optimized way as earlier work we can extend the design to higher bit multipliers without modifying the architecture of earlier proposed multipliers.

By using 8*8 multiplier which was implemented in the previous work, a 24*24 multiplier is designed without changing the previous architecture. A (15,4) or (31,5) saturated counter will be advantageous in various applications.

IV. EXPERIMENTAL RESULTS

On synthesizing the Verilog code a RTL schematic can be generated in which we get to see the modules, sub modules and gate level design.

The RTL schematic is shown in fig. 5

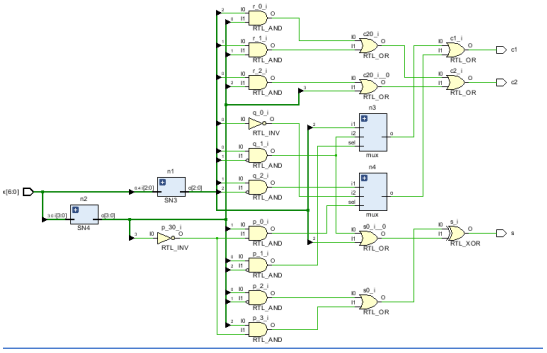

Fig. 5: RTL Schematic

Similar to RTL schematic, technology schematic can also be generated after synthesizing the code. It consists of more number of block which is not possible to show that's why top block is only shown in the below fig.

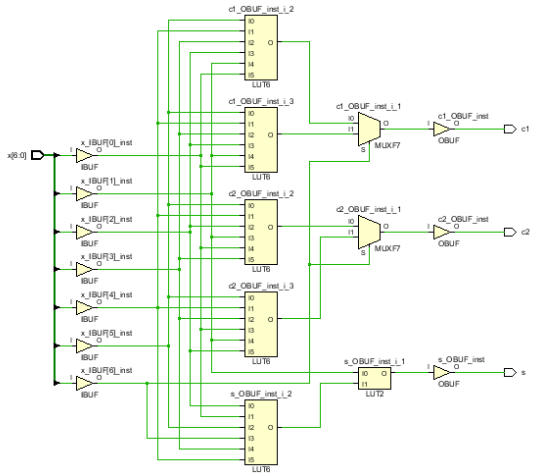

Fig. 6: Technology Schematic

To check the correctness of the output and to observe the output waveform we need to do simulation process after completing the Verilog code. The output waveform is shown in Fig. 7

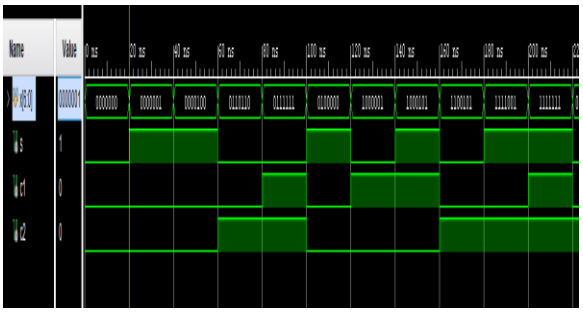

Fig. 7: Simulation Results

Evaluation table for Area, Delay:

	AREA(LUT's)	DELAY(ns)	POWER(Watts)
Bc_73	6	5.804	2.022
Bc_154	44	8.775	3.275
Bc_315	139	11.776	5.632
Mul 8	122	12.041	14.187
Mul 16	739	17.002	41.444
Mul 24	1218	12.862	63.288

V. CONCLUSION

We can obtain optimized parameter values for higher bit multipliers by using the proposed multipliers which are 8 bit multiplier by extending it to 24 bit multiplier without changing its architecture. Hence, it can be used for higher bit multiplications.

Thus area for the higher order multipliers would be same as that of 8 bit multiplier which results in area efficient circuit.

VI. 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] 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.

[5] 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.

[6] 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.2017.2723475.10.1109/EDSSC.2017.8126527.

[7] 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.

[8] 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.

[9] 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.