Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 4 (2024)

Design of Low Latency Multiply Accumulate Unit Using Counter Based Modular Wallace Tree Multiplier

Revathi S., Valli Shri P., Vidhyavarshini S., Kalieswari C.
Department of Electronics and Communication Engineering, National Engineering College, Kovilpatti, India

Abstract:- This project proposes a low- latency CBMWTM (Counter Based Modular Wallace Tree Multiplier) multiplier that enables the simple and effective implementation of the Wallace tree multiplier. The suggested multiplier makes use of a 7:3 counter with multiplexers and adders based on X-or gates, which are more effective than the other 7:3 counter designs already in use. The last stage consists of a single fast adder that receives carry from the pre-multiplier and previous stage output. This Project is implemented using Xilinx ISE Software Tool.

Keywords*:* Full Adder, 5:3 Counter, 7:3 Counter

1. Introduction

For embedding systems, the multiply- accumulate (MAC) unit is a key building block. A recent breakthrough in planning has been the creation of real-time edge applications. High-speed, low-power MAC devices are anticipated to be in high demand. The multiplier and the accumultialator make up the two separate blocks that make up the traditional MAC unit. Guard bits are used by the two N-bit accumulators and the N-bit multiplier (adder) that make up the N-bit MAC unit to prevent excess. A great deal of earlier work was rewarded by improvements to the multiplier and the adder. Three phases make up the multiplier. A partial product is produced by the first phase of the PPG process. A partial product matrix (PPM) will exist. The accumulation result is created by adding the second step to the first. Additional propagations generally result in significant route delay and greater power consumption. In order to solve these issues, a novel High- Performance Multiply- Accumulate Unit is suggested in this study. The suggested counter- based modular Wallace tree incorporates a portion of additions in the partial product reduction stage to lessen the propagation of critical path delays.

The intended MAC unit's final addition of the majority of the significant bits was not performed on the current multiplication. Instead, the final addition, including the collection of the majority of the significant bits, is handled by the PPR process of the subsequent multiplication. The length of carry propagations is shortened as a result. High speed binary carry selects adders (HSBCSA) are designed to count every carry in order to manage the overflow during PPR processing. The results of the experiment demonstrate that the suggested approach regularly works well in real-world situations. Below is a summary of the main features of the suggested design. To reduce the carry propagations, part of the additions are incorporated into the proposed CMWTM's PPR processing.

- The suggested High speed binary carry select adders (HSBCSA) are used to count the total carries in order to handle overflow on PPR procedure.
- The second step (the entire series of multiply- accumulate operations) is carried out in the last cycle (power saving) by employing the gating method.
- Verilog is used for coding, while the Xilinx ISE series is used for synthesis.

The effectiveness of the suggested CMWTM- HSBCSA-ES is assessed using measures like latency.

A tree-based multiplier operates in three primary stages: partial product production, tree reduction, and final addition. N2 AND gates are required in order to construct a partial product tree of the N row in an N-bit multiplier. The following procedure is the reduction of the tree, which reduces this partial product tree until only two rows remain. Half adders and full adders are used to add columns in a parallel manner. To create the finished product, two rows must be added in the final stage using any fast adder.

In order to decrease the partial product tree reduction delay even more, various topologies utilizing high-speed counters were suggested. A reverse pyramid is used to recalibrate the partial product generation tree in Counter Based Modular Wallace (CBMW) tree multipliers shape. After that, compressors or 4:3, 5:3, 6:3 and 7:3 counters are used, along with full and half adders, to complete the partial product tree reduction. Higher-order counters in a single column allow for the addition of more than three bits at once, reducing delay.

The CBMWT multiplier and the suggested counter-based modular Wallace (CBMW) tree multiplier are included in the design of a 7:3 counter that makes use of an ex-or gates-based adder module and a traditional full adder module. The results and simulations comparing the suggested multiplier architecture with other current architectures, as well as several 7:3 counters.

2. Literature Survey

A large number of previously proposed works that rely on MAC unit design in the literature. Typically, a multiplier has three phases. Partial product generation (PPG) is the initial phase. For an unsigned multiplication, for instance, AND gates can be used to produce a partial product matrix (PPM). Partial product reduction, or PPR, is the second phase. The PPM can be simplified to two rows by utilizing the Wallace tree or Dadda tree approaches. The final addition comes in the third stage. The final two rows are added together using an adder (sometimes known as the final adder). The final addition for an N-bit multiplier requires a (2N-1)-bit adder. Parallel prefix adders (PPAs) handle the last addition stage of partial products in the suggested architecture. This study proposes five different Wallace tree multiplier structures, employing the Kogge Stone, Sklansky, Brent Kung, Ladner Fischer, and Han Carlson adders. With the Xilinix 13.2 design suite, Verilog HDL is used to create each multiplier structure. The ISIM simulator is used to simulate the suggested structures, and the XST synthesizer is used to create them.

Numerous adder topologies have been suggested in order to balance latency trade- offs. A multiplier is a key component of every processing unit. Numerous multiplication algorithms are put forth, allowing for the creation of multiplier structures. Wallace tree multiplication is one of the many multiplication algorithm is advantageous in terms of operating speed. The need for circuits with high speed and small size is growing as technology advances. A novel Wallace tree multiplier structure is developed in order to increase the multiplier's speed without lowering its area parameter. Adders are used in circuits to build products. In the last stage, a single fast adder adds the carry from the previous column and the output from the previous stage.

3. Proposed Work

Our proposed block diagram consists of Partial Product Generation, 7:3 Counters, PISO and SISO Shift Registers and a comparator. Organizing the partial product tree into an inverse pyramid is the initial step in creating incomplete items. A 16x32 bit matrix with 7 groups of bits is produced when the partial product is empty and the inverse pyramid is padded with zero. Each column is applied serially to the 7:3 counter module after being shifted by one bit by the shifter that is parallel to the serial shift register (PISO) due to the partial product tree's parallel structure. The output S of the 7:3 counter is inserted into the same column, C1 is passed to the next column after a one-position shift, and C2 is passed to the next-to-next column.

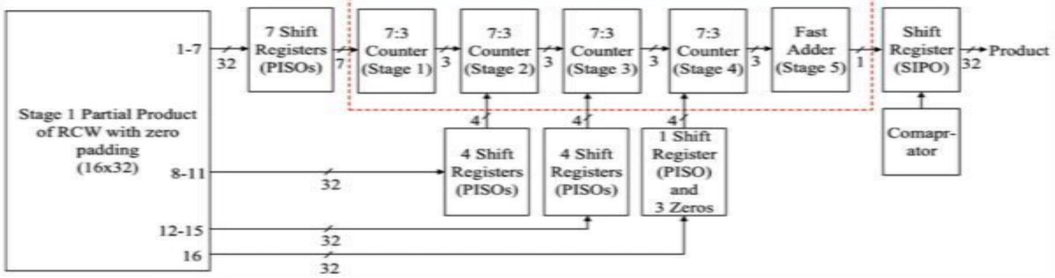


Fig 1 : Block schematic of a Modular Wallace Tree Multiplier with counter basis

Parallelism in design of multiplier typically leads to more changes between partial products. Incorporating an intermediate sum frequently results in enhanced performance; however, it concurrently introduces an irregular structure on the silicon surface and heightened power consumption due to the intricate routing of interconnects. Serial multipliers, aiming for improved space and power efficiency, sacrifice speed.

Thus, the choice between using a parallel or serial multiplier depends on the kind of application. The primary objective of the proposed the goal of the CBMWTM design is to provide a power-efficient multiplier with lower latency. The CBMWT multiplier employs a single 7:3 counter for each stage of partial product reduction when inputs are applied serially.

Delay is minimized as only one 7:3 counter is used per step, employing four standard full adders. The seven partial product terms that make up input bits 1 through 7 are represented by the output bits Sum, Carry1 and Carry2. The intermediate sum and carry bits are SFA1, SFA2, CFA1, CFA2, and Carry (or w14). Input1 receives the first three partial product bits - Input1, Input2, and Input3 while FA2 receives the next three partial product bits - Input4, Input5, and Input6 and FA3 receives Input7 along with the sum of FA1 and FA2, or SFA1 and SFA2, and carry bits of FA1, FA2, and FA3 are applied in the same manner.

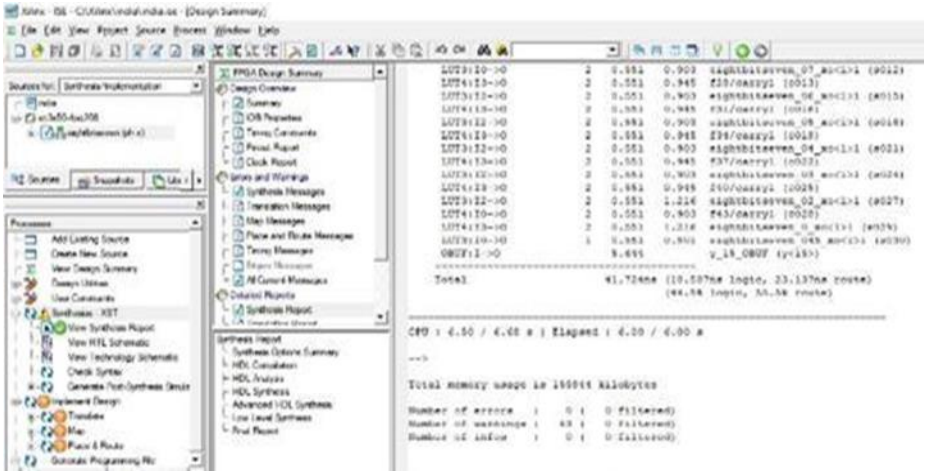
Following the construction of partial products, the procedure is repeated in the suggested multiplier to generate the multiplicand and multiplier's product. The first step is to produce objects that are unfinished. The partial product tree has an inverse pyramidal structure, as seen in Fig 1. A 16x32-bit matrix is formed by padding the inverse pyramid with zero when the partial product is empty. This matrix is then split into seven groups of bits. With weights of 20, 21, and 22, respectively, S, C1, and C2 are the output of the 7:3 counter. The output S is placed in column C1, which is moved by one, and C2 is then moved to the column that is moved by two. Before there are just two rows left, the leftover partial product and the output of the first stage 7:3 counter are fed into the second stage 7:3 counter. The identical procedures are then repeated for the next level.

The last stage employs a carry-save adder to merge its output with the carry from the preceding column, facilitating a serial output.This work proposes a low latency wallace tree multiplier architecture with reduced latency, utilizing a delay-efficient design of 7:3 counter comprising multiplexers and XOR gates. The proposed CBMW multiplier uses a sequential 7:3 counter to reduce partial products, and by enhancing locality, multi-bit addition in a single column lowers the multiplier's complexity. The lowest hardware utilization occurs when partial product tree reduction is performed each stage using a single 7:3 counter. Verilog in the Xilinx ISE is used to construct the proposed multiplier on the Spartan 3E FPGA.

Wallace tree multipliers offer a high-speed multiplication method that uses less power. The multiplier speed can be increased even more by using high speed 7:3 counters in the Wallace tree reduction. An algorithmic approach is presented in this study, to build the Wallace tree multipliers depending on counters. The efficient counter-based wallace modular tree multiplier of any size appropriate for Xilinx synthesis tools can be implemented with the suggested algorithm. A thorough comparison between the counter- based and traditional Wallace multipliers reveals that the counter-based multiplier can operate up to (10-20)% faster than the standard Wallace tree multiplier.

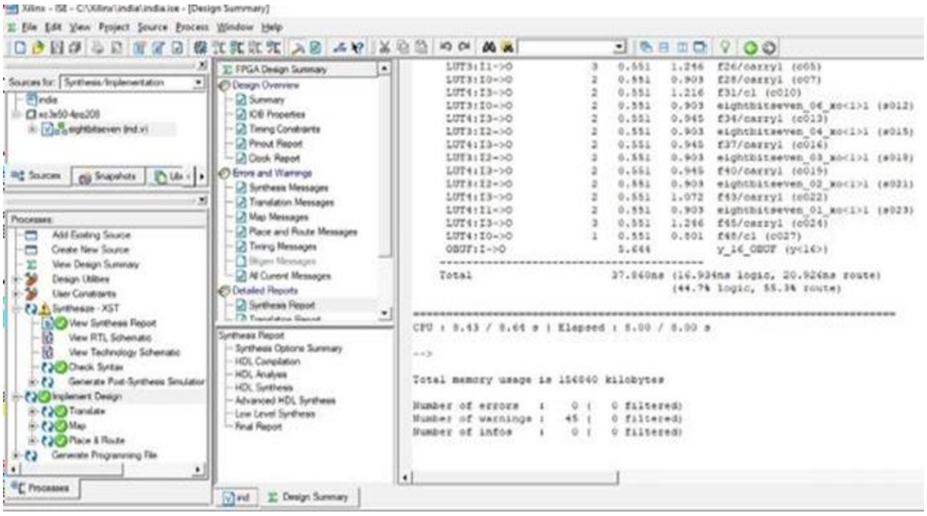
4. Result & Discussion

The Performance of the multiplier is analysed by using the Xilinx. The Xilinx helps us to find the time taken to execute the 32 bit multiplier using full adder, 32 bit multiplier using 5:3 counter, 32 bit multiplier using 7:3 counter. The following screenshots show the delay for the various multiplier. From these multiplier the delay is reduced for the 32 bit multiplier using 7:3 counter (using X-or gate & Multiplexer).



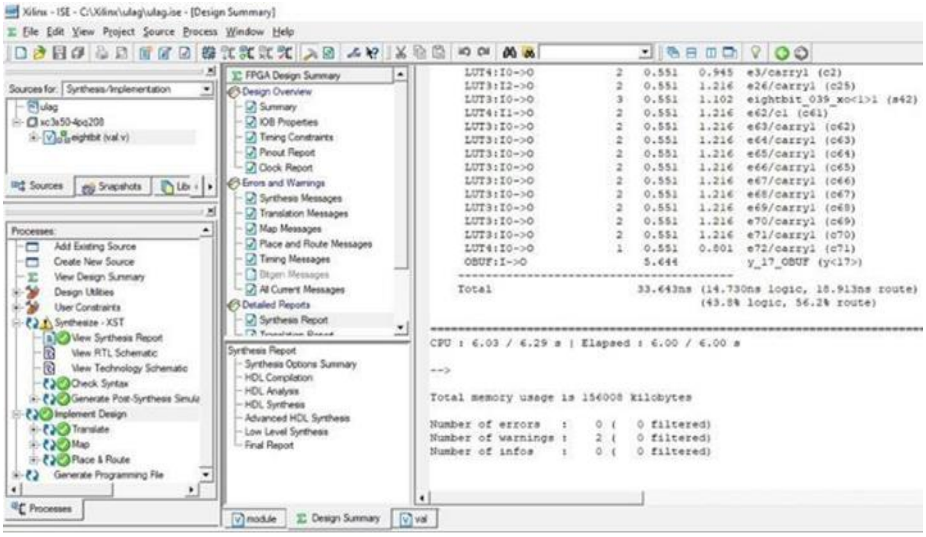
```
     LUT3:I0->O            2   0.551   0.903  eightbitseven_07_xo<1>1 (s012)
     LUT4:I3->O            2   0.551   0.945  f28/carry1 (c013)
     LUT3:I2->O            2   0.551   0.903  eightbitseven_06_xo<1>1 (s015)
     LUT4:I3->O            2   0.551   0.945  f31/carry1 (c016)
     LUT3:I2->O            2   0.551   0.903  eightbitseven_05_xo<1>1 (s018)
     LUT4:I3->O            2   0.551   0.945  f34/carry1 (c019)
     LUT3:I2->O            2   0.551   0.903  eightbitseven_04_xo<1>1 (s021)
     LUT4:I3->O            2   0.551   0.945  f37/carry1 (c022)
     LUT3:I2->O            2   0.551   0.903  eightbitseven_03_xo<1>1 (s024)
     LUT4:I3->O            2   0.551   0.945  f40/carry1 (c025)
     LUT3:I2->O            2   0.551   1.216  eightbitseven_02_xo<1>1 (s027)
     LUT4:I0->O            2   0.551   0.903  f43/carry1 (c028)
     LUT4:I3->O            2   0.551   1.216  eightbitseven_0_xo<1>1 (s034)
     LUT3:I0->O            1   0.551   0.801  eightbitseven_045_xo<1>1 (s030)
     OBUF:I->O                 5.644          y_15_OBUF (y<15>)
    ----------------------------------------
    Total                     41.724ns (18.587ns logic, 23.137ns route)
                                       (44.5% logic, 55.5% route)

=========================================================================
CPU : 6.50 / 6.68 s | Elapsed : 6.00 / 6.00 s

-->

Total memory usage is 155044 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :   43 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

Fig 2 : Screenshot for the delay of 32 bit multiplier using full adder



```
     LUT3:I1->O            3   0.551   1.246  f26/carry1 (c05)
     LUT3:I0->O            2   0.551   0.903  f28/carry1 (c07)
     LUT4:I3->O            2   0.551   1.216  f31/c1 (c010)
     LUT3:I0->O            2   0.551   0.903  eightbitseven_06_xo<1>1 (s012)
     LUT4:I3->O            2   0.551   0.945  f34/carry1 (c013)
     LUT3:I2->O            2   0.551   0.903  eightbitseven_04_xo<1>1 (s015)
     LUT4:I3->O            2   0.551   0.945  f37/carry1 (c016)
     LUT3:I2->O            2   0.551   0.903  eightbitseven_03_xo<1>1 (s018)
     LUT4:I3->O            2   0.551   0.945  f40/carry1 (c019)
     LUT3:I2->O            2   0.551   0.903  eightbitseven_02_xo<1>1 (s021)
     LUT4:I3->O            2   0.551   1.072  f43/carry1 (c022)
     LUT4:I1->O            2   0.551   0.903  eightbitseven_01_xo<1>1 (s023)
     LUT4:I3->O            3   0.551   1.246  f45/carry1 (c024)
     LUT4:I0->O            1   0.551   0.801  f48/c1 (c027)
     OBUF:I->O                 5.644          y_16_OBUF (y<16>)
    ----------------------------------------
    Total                     37.860ns (16.934ns logic, 20.926ns route)
                                       (44.7% logic, 55.3% route)

=========================================================================
CPU : 8.43 / 8.64 s | Elapsed : 8.00 / 8.00 s

-->

Total memory usage is 156040 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :   45 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

Fig 3 : Screenshot for the delay of 32 bit multiplier using 5:3 counter.



```
     LUT4:I0->O            2   0.551   0.945  e3/carry1 (c2)
     LUT3:I2->O            2   0.551   1.216  e26/carry1 (c25)
     LUT3:I0->O            3   0.551   1.102  eightbit_039_xo<1>1 (s42)
     LUT4:I1->O            2   0.551   1.216  e62/c1 (c61)
     LUT3:I0->O            2   0.551   1.216  e63/carry1 (c62)
     LUT3:I0->O            2   0.551   1.216  e64/carry1 (c63)
     LUT3:I0->O            2   0.551   1.216  e65/carry1 (c64)
     LUT3:I0->O            2   0.551   1.216  e66/carry1 (c65)
     LUT3:I0->O            2   0.551   1.216  e67/carry1 (c66)
     LUT3:I0->O            2   0.551   1.216  e68/carry1 (c67)
     LUT3:I0->O            2   0.551   1.216  e69/carry1 (c68)
     LUT3:I0->O            2   0.551   1.216  e70/carry1 (c69)
     LUT3:I0->O            2   0.551   1.216  e71/carry1 (c70)
     LUT4:I0->O            1   0.551   0.801  e72/carry1 (c71)
     OBUF:I->O                 5.644          y_17_OBUF (y<17>)
    ----------------------------------------
    Total                     33.643ns (14.730ns logic, 18.913ns route)
                                       (43.8% logic, 56.2% route)

=========================================================================
CPU : 6.03 / 6.29 s | Elapsed : 6.00 / 6.00 s

-->

Total memory usage is 156008 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    2 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

Fig 4 : Screenshot for the delay of 32 bit multiplier using 7:3 counter

The delay analysis was performed in Xilinx ISE software where the design was implemented for various counter. From that 7:3 counter (using X-or gate & Multiplexer) has less delay and is more efficient.

5. Comparison Table

The below table is an comparison table for 32 bit multiplier using various methods like full adder, 5:3 counter, 7:3 counter (using X-or gate & Multiplexer). The delay of 7:3 counter is 33.643 ns, the delay of 4.217 ns are reduced from previous counters.

Table 1 : Comparison Table for 32 bit Multiplier

Title	Wallace Tree Multiplier		
	Full Adder	5:3 Counter	7:3 Counter
Number of 4 i/p LUT's	154	149	154
Number of slices Occupied	83	84	87
Number of slices containing related logic only	83	84	87
Number of bonded Input Output Buffers	33	32	33
Total number of equivalent gates in the design	924	894	924
JTAG gate count increase for IOBs	1584	1536	1584
Delay(ns)	41.724	37.860	33.643

The below table is an comparison table for 16 bit multiplier using various methods like full adder, 5:3 counter,7:3 counter (using X-or gate & Multiplexer). The delay of 7:3 counter is 15.273ns, the delay of 3.449ns are reduced from previous counters.

Table 2 : Comparison Table for 16 bit Multiplier

Title	Wallace Tree Multiplier		
	Full Adder	5:3 Counter	7:3 Counter
Number of 4 i/p LUT's	33	30	24
Number of slices Occupied	16	16	16
Number of slices containing related logic only	18	18	13
Number of bonded Input Output Buffers	33	30	24
Total number of equivalent gates in the design	198	180	147

JTAG gate count increase for IOBs	78	768	768
Delay(ns)	17.986	18.722	15.273

A 7:3 counter that uses an X-or gate, multiplexer-based adder module, and a conventional full adder module is designed with the Counter Based Modular Wallace Tree Multiplier (CBMWTM). The outcomes and simulations that contrast the proposed multiplier architecture with various existing architectures, together with a number of counters for both 16- and 32-bit multipliers.

6. Conclusion

This paper proposes a scalable and low- latency CBMW multiplier. It makes the Wallace tree multiplier simple and effective to use. The suggested multiplier employs a 7:3 counter made up of an X-or gate and multiplexer based Adders, that are more effective than the other 7:3 counter designs now in use in Circuits, Systems, and Signal Processing. Due to greater locality and area preservation, there is a notable reduction in delay as just one 7:3 counter is required every stage. Xilinx ISE is used to synthesize the modular multiplier architecture, and the output is contrasted with multipliers of a comparable design that have been published in the literature. The MAC unit's performance is assessed using the suggested CBMWTM in its design.

References

[1] Yamini devi Ykuntam , Katta Pavani and Krishna Saladi, "Design and analysis of High speed wallace tree multipliers using parallel prefix adders for VLSI circuit designs", in IEEE 11th ICCCNT, 2020.

[2] Kokila Bharti Jaiswal, Nithish Kumar V, Pavithra Seshadri and Lakshminarayanan G, "Low Power Wallace Tree Multiplier Using a Modified Full Adder", 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), 2015.

[3] Y. Jiang, A. Al-Sheraidah, Y. Wang, E. Sha, J. Chung, "A novel multiplexer-based low-power full adder", IEEE Trans. Circ. Syst. II Exp. Briefs, 2004.

[4] C.W. Tung, S.H. Huang, "A high- performance multiply-accumulate unit by integrating additions and accumulations into partial product reduction process", IEEE Access 8, 2020.

[5] W. J. Townsend, E. E. Swartzlander, and J. A. Abraham, "A Comparison of Dadda and Wallace Multiplier Delays", in Proc. SPIE Annu. Meeting Opt. Sci. Technol. San Diego, CA, USA, 2003.