European Journal of Molecular & Clinical Medicine
ISSN 2515-8260 Volume 07, Issue 05, 2020

# Modified Radix-16 Booth partial product generator for 64-bit binary multipliers

**R. NAGA PRAMEELA**[1], **GAMINI SRIDEVI**[2]

[1-]*PG Scholar Dept of ECE, Aditya Engineering College, Surampalem, E.G(Dt), AP, India*

[2-]*Professor, Dept of ECE, Aditya Engineering College, Surampalem, E.G(Dt), AP, India.*

[1-]*prameelaravuri860@gmail.com*

**Abstract***: **Redundant Binary Partial Product Generators are used to minimize, without any increase in the delayed partial product development block by one row, the maximum height of the partial product array generated by radix- 16 Modified Booth Encoded multiplier. The optimization for binary radix-16 (modified booth coded) multipliers is defined in this project to decrease the maximum volume of the partial product columns to [n/4], with n = 64-bit non-signed operand. This compares [(n + 1)/4] with the normal maximum height. Thus the overall height of a single device is limited. This multiplier improves ALU and processors efficiency. In contrast to the traditional booth multiplier, we test the suggested solution. In terms of field, delay and control, the logical synthesis showed its effectiveness. Using Verilog, the project will be built. The Simulation and Synthesis Xilinx ISE method is used.***

***Keywords: Binary multipliers, modified Booth recoding, radix-16.***

## 1. INTRODUCTION:

Binary multipliers are a commonly used part of the design of microprocessors and embedded systems and are therefore a significant aim for optimisation [1]–[6]. Present binary multiplication implementation is accompanied by the steps of [7]: 1) multiplier recoding in a number scheme in digit numbers; 2) multiplying each digit with a multiplicand result in some sections of the products; 3) reducing the list to two operands using multi-operand addition techniques; The normal recoding process covers -r digits by simply turning m groups into a signed-digit operand with digits in a minimally redundant digit [7], [8]. The output of each of these peculiarities multiplies two terms addition or subtraction which results in total carrying-out add-ons. High-energy latency was currently optimised, although the trees encountered increasing trouble with partial product reductions owing to unbalanced signal, due to the complicated wiring and flatulence.

In current and future units for multipliers (or multipliers-adding) the best pipelines are also key: 1) The pipelines are quite necessary, also for throughput-oriented uses, because they affect energy of all core [19]; and 2) the placing of pipelines at the same time, due to the appropriate amount to flip-flops and the signal propagation, can reduce total strength. Two radix-4 boot multipliers are introduced, thereby opening up analysis and extension to higher radii and unsigned multiplications to unsigned mantissa periods or integer arithmetic in a floating point unit). For a radix higher than 4, the odd multiple (usually adders) must be produced which results in time slackness required to cover" simplified assimilation of three bits.

Non-signed multiplication results in an optimistic method during the recoding process (this is centred on one row, which raises the overall height for the partial product list, not just in one row, but also in many columns). For all these purposes, strategies in [1] and 2] have to

be extended. The approach suggested makes a one row reduction when n is more than one m, since the maximal height of the regular unsigned multiplier is _(n+1)/m_. We use radix-16 since it is the most complicated example, row without raising the essential phase of part-product generation) relies on a particular timing of the different for all functional values of r and n and different implementing technologies (the reduction of a single unsigned multiplier is carried out with a method for synthesis, a typical cell library).Therefore we focus on a particular case: 64-bit radix-16 Booth, which for the nature of our scheme is codified among the functional meaning of radix. In comparison to the signed multiplier, the unsigned multiplier is more complicated for designing our framework. We use 64 bits since it is a broad word length representative. Instances such as authenticated, mixed unsigned/signed, radix-8 coding, n values can be generalised conveniently to other instances.

## 2. RELATED WORK:

D.Govekar. Al [1] build a hybrid adder-modified high speed booth multiplier. The Changed Booth Multiplier configuration with hybrid adder offers improved efficiency relative to traditional approaches with the Hold Look Ahead Adder. In contrast to the traditional approach the region is decreased by 4.8 percent and 3.710 percent respectively. The design is carried out using a simulation framework named Xilinx ISE10.1 and simulated by ModelSim15.7g.

In this paper, TaoLuo et al [3] present a multiplier in memory Booth centred on race track memory to mitigate this problem. As the building block of our multiplier, our proposed multiplier is a race track memory adder that saves 56.3% power compared to state-of-the-art magnet adder incorporated with the storage feature.

Elisardo antelo et.al [4] In this article, an algorithm for the recoded multiplier updated radix-16 binary boots decreases the range height of a partial product column in contrast with the full traditional scheme height to [n+1/4] for 64-bit unsigned translators. The system pipelined is used to decrease the total strength of 6.86% to 6.72% in two stages and to lower its total capacity to 7.15% to 6.84% in three stages. The planned power reduction is just 4%.

G. Haridas, and David. al [5] create a modern low-power, powerful area architecture utilising a multiplier with a direct-shape tree-based adjusted booth multiplier. By utilising twisted tree adder and adjusted treading tree adder in FIR philtres, the region in comparison to traditional FIR philtres is decreased by 23.29% and by 29.10% respectively. And the production is down 3.03%... The architecture is carried out using VHDL programming Xilinx 14.2 ISE tools, Model Sim. MATLAB Simulink function is used to calculate separate philtre coefficients for the implementation of a FIR philtre.

B.R, Prabhu and E. al [6] has indicated changed the hybrid hold looking-ahead adder booth algorithm. By integrating reversible logical functions with a hybrid look-ahead adder, the multiplier and the builder are proposed and a new complete adder architecture with reversible logic gate is provided. In contrast to the standard approach, the MAC indicated increased efficiency and reduction in area by 161.8% and 196.10% and latency by 34.35ns and 27.31ns. The concept is carried out using the Xilinx ISE simulator.

## 3. IMPLEMENTATION OF MULTIPLIER

We may conduct a quick transmission in parallel with the normal partial product generation to reduce the overall height for the partial product series. This short complement lowers the height by one row and is quicker than usual partial generation of a commodity.

Image.fig.1 displays the elements of the quick adder to add in the bit list. Following the brief addition, and displays a partial commodity bit list. We note a drop in maximal height from 17 to 16 for n = 64 in comparison with both estimates.

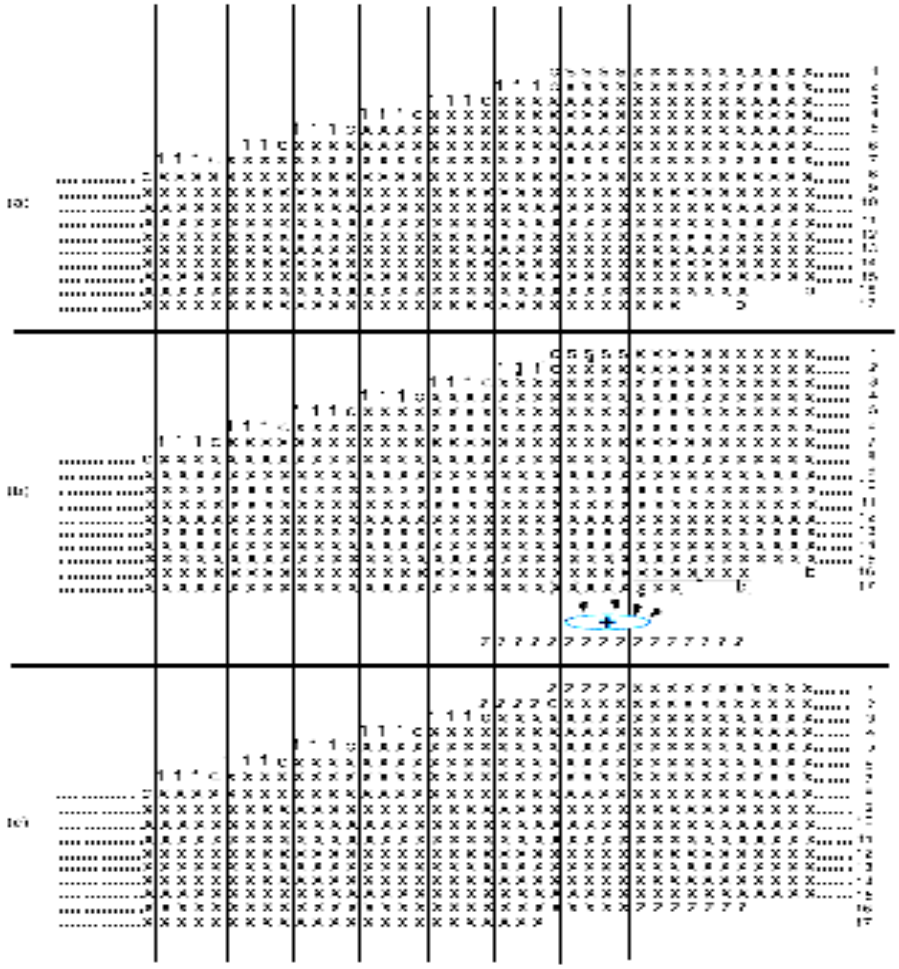

**Fig. 1: Radix-16 partial product reduction array**.

Since the partial goods are four bits moved to the left, that will require an expensive sign expansion. However, the sign extension shall be simplified with the combination of certain bits for each component product: the CSSS for the first parts product and the 111C for the other sections (with the exception of the subparts which do not have the negative effect, because the corresponding multiplier number is 0 or 1). The parts labelled with b in Fig.1 is compatible with the rationale 1 applied for the negative partial products for the complement of the two.

In two concurrent sections A and B as seen in Fig.2 we conduct the computation. Part A elements are created more rapidly than part B elements. In particular the part-A elements are extracted from the first part product: this is obtained directly from bit y3 as a consequence of no previous radix 16 digit pass digit; thus it was agreed to incorporate part A in a speculation by estimating two values, a result with carry-in = 0 and a result with carry-in = 1. The results are obtained from part- A. For a compound adder this can be measured easily.

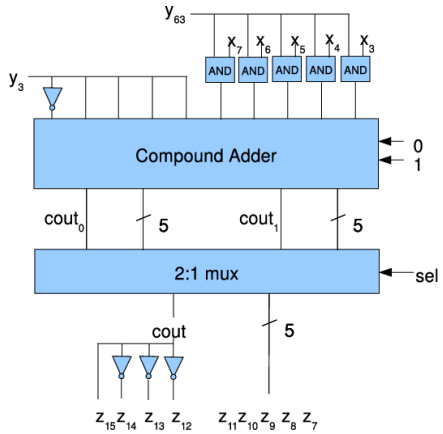


**Fig. 2: Speculative addition of part A.**

Fig. 2 Displays Component A implementation. Two potential outcomes are calculated by the compound adder. The right outcome is picked by a multiplexer until the transportation is collected (from Section B). Please keep in mind that the compound adder is just five bits so it stretches quickly over the three most important. Part B is difficult to determine. The key problem is that the lowest 7 bits of partial product 15 are required. Naturally waiting for partial product 15 to be produced is not the alternative since we want to escape the vital route by the short pause.

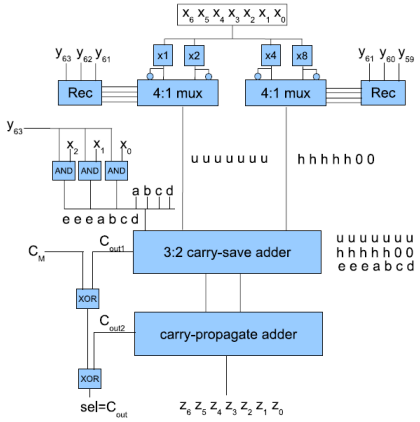


**Fig. 3: Computation of part B**.

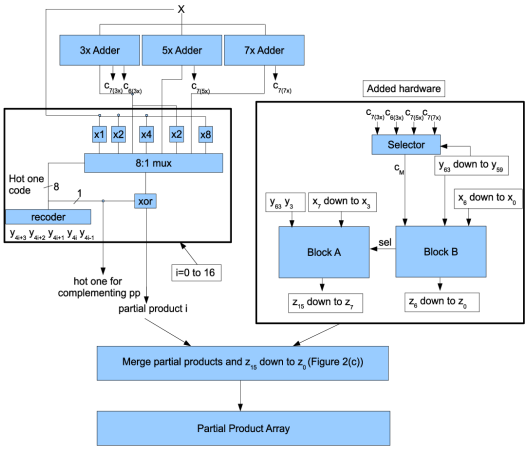


**Fig.4: High level view of the recoding and partial product generation stage in our existing method.**

In calculating component B, the approach of calculating the highest proportion of the part product 15 will also contribute to an incoherence. In fact, since partial product 15 results from an unexpected multiple, the potential carriage of the 7 small bits in the most important component of the partial product is already integrated. We should not generate this carrier again during the measurement of component B. This dilemma has been fixed as follows. Let's first analyse the scenario with positive chances. Fig.3 shows that component B computation will produce two outputs: one from the carry-saving adder (Cout1), while one from the transported adder (Cout2). This shows a two output computation. In order to prevent contradictions, we detect and remove from the two carriages created in part B the transport that has propagated to the most important part of the partial product 15 (this is what we call CM).

## 4.PROPOSED METHOD

### Modified full adder- 1 (FA1):

In these works, the descriptions of the multiplier and truncated multiplier of the Wallace tree are analysed with a standard complete adder as well as with a tweaked whole adder-1.

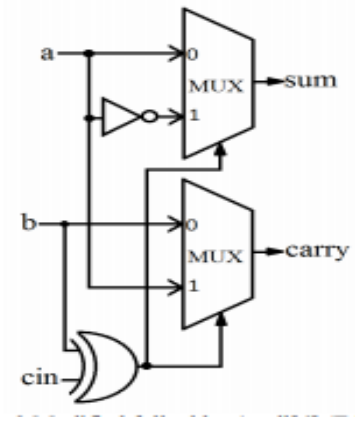


**Fig.5: Modified Full adder -1 Cell**

### Modified full adder- 2 (FA2):

This analysis demonstrates that the updated full adder 3 implementations are less common and have a less effective effect [7], and the comprehensive research on the Wallace tree multiplier with standard complete adder, modified full adder1 and modified full adder-2.

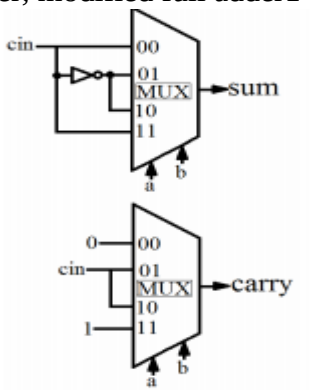


**Fig.6: Modified full Adder-2 cell (7) (FA2)**

The usage of adjusted complete adder-3 is also seen to be better as regards delay and location.

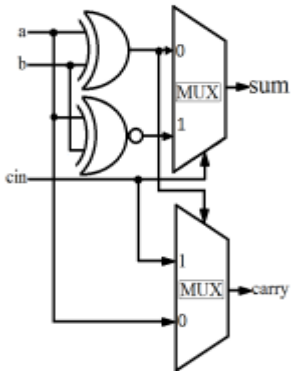


**Fig.7: Modified full Adder-3 cell (8) (FA3)**

**A. Ripple Carry Adder (RCA):**

Ripple Carry Adder is a simple adder circuit that contains individual full adder cells and propagates the carry produced by the addition between them [4]. Only after the carrying from the previous stages is used as an input in the current stage is the results calculated [4]. Owing to these gaps in spreading, there is a substantial downside to the delay. The 32-bit RCA is shown in Fig.5.

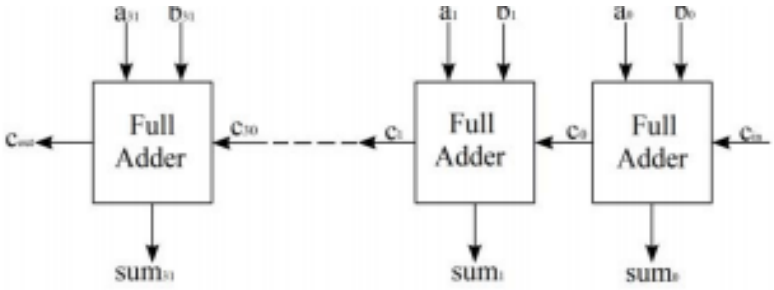


**Fig 8: 32 – bit ripple carry adder**

**Compute Add Increment (CAI) adder:**

Various RCA blocks for measuring the effects are included in the CAI adder. In addition to add-ons to carry(c1) and value the first RCA block is given as the input. For the other RCA block the transmission is given as a logical '0' to receive a temporary sum (sum1). Circuit increment consists of half adders, inserting a provisional number and bringing the final total and carriage (cy). Circuit increase is accomplished by OR operation between carrying (cy) and execution of the previous level. The increase circuit is carried out. Owing to the assumption that the transport period for the RCA phases is '0' and hence the transmission time declines. The CAI 32bit architecture is seen in the diagram.

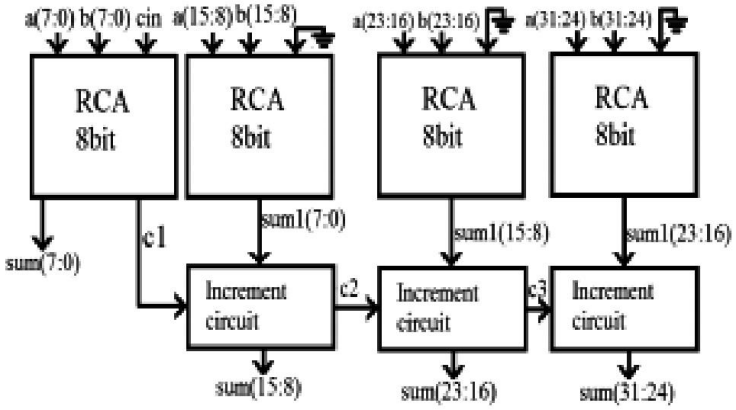


**Fig 9: The architecture for the CAI 32bit**

## 5.Simulation Method

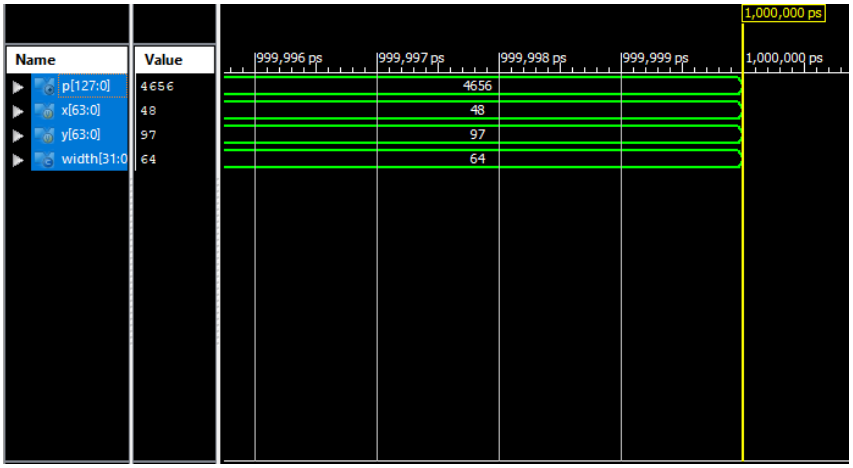


**Figure 10A. output wave form of proposed multiplier method**

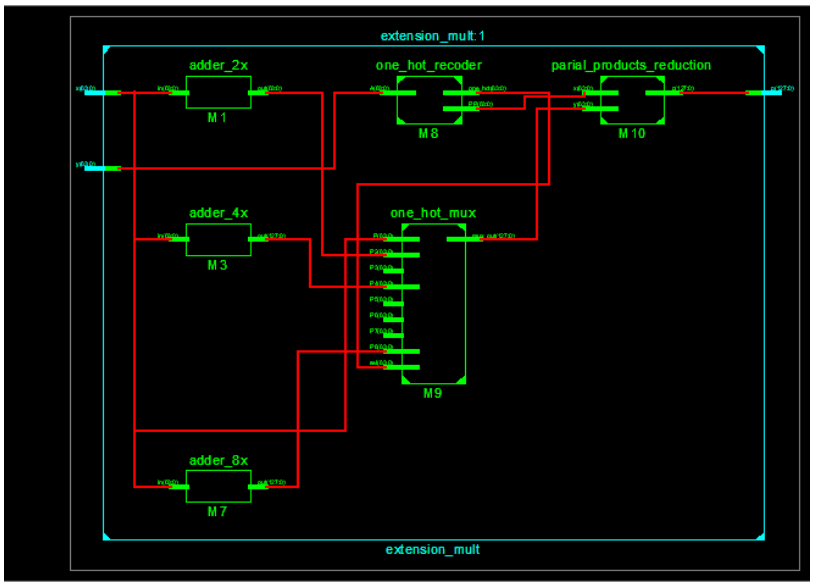


**Figure 10B. The RTL schematic of proposed multiplier**

```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: No path
   Maximum output required time after clock: No path
   Maximum combinational path delay: 27.685ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 8331536
---------------------------------------------------------
Delay:               27.685ns (Levels of Logic = 164
  Source:            y<0> (PAD)
  Destination:       p<127> (PAD)

  Data Path: y<0> to p<127>
```

**Figure 10C.The timing summary of proposed multiplier**

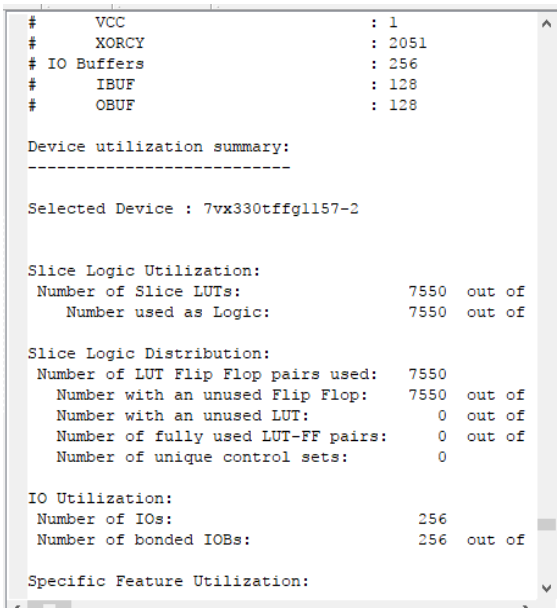


**Figure 10D. the area report of proposed multiplier**

The functionality of the circuit was tested using the Xilinx ISE software. The signals a and b of size 63-bit are given as Inputs. Here **m** is the mode selection bit, when m=0 it will act as approximate multiplier for some test cases and when m(mask)=1 it will act as an accurate multiplier for all test cases. The two input signals a, b are of values {97,48}, when m=0 then the multiplier output is 4656 can be observed in output variable P. The simulated result is shown in Fig.10A.The RTL schematic for proposed multiplier is shown in Fig.10B.The timing summary of proposed multiplier is shown in Fig.10C.The area report of proposed multiplier is shown in Fig.10D.

## 6. RESULTS AND DISCUSSION

The study has been carried out by considering Look Up Table(LUT), Time Delay and Power Consumption. The Look Up Tables used in the proposed system are 7550 as compared to the previous work of 12887. It indicates very less area is used for the proposed design. The consumed path delay in the proposed system is 27.685ns as compared to the previous work of 36.802ns.

**Comparison**

| Parameter | existing | proposed |
|---|---|---|
| **LUTS** | **12887** | **7550** |
| **Time** | **36.802ns** | **27.685ns** |
| **Area used** | **27%** | **16%** |

The performance of the proposed system is compared with that of the existing method. We can observe that the designed multiplier is effective and efficient in terms of area-delay trade off, delay (speed) and power utilization when compared to the previous one.

## 7. CONCLUSIONS

Using the proposed algorithm, the multiplier performs stronger power delay tests than traditional booth multipliers. Here we have provided a method to minimise for 64-bit 128-bit radix-16 Booth the overall height of the part feature ranges recoded magnitude multipliers by one. This decrease will allow greater versatility in the design and without any additional pause for n-to-32 for cellular design of the reduction tree of the pipelined socket. In general computers, as well as optical signal processors, smart phone device processors and various arithmetical systems, using Booth encoding the proposed Booth algorithm may be commonly used.

## 8. REFERENCES

1. S. Kuang, J. Wang, and C. Guo, "Modified booth multipliers with a regular partial product array," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 56, no. 5, pp. 404–408, May 2009.
2. G.sridevi and T.narmada, "Implementation of FIR Digital Filters for Odd Length using Power Reduction Technique "international journal of VLSI system design and communication systems,Vol.5,No.7,july 2017.
3. G.sridevi and O K kennedy , "performance analysis of a low power and high speed carry select adder", international conference on current trends in computer ,electronics and communication ,September 2017.
4. F. Lamberti et al., "Reducing the computation time in (short bit-width) twos complement multipliers," IEEE Trans. Comput., vol. 60, no. 2, pp. 148–156, Feb. 2011.
5. N. Petra et al., "Design of fixed-width multipliers with linear compensation function," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 5, pp. 947– 960, May 2011.
6. K. Tsoumanis et al., "An optimized modified booth recorder for efficient design of the add-multiply operator," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 4, pp. 1133–1143, Apr. 2014.
7. A. Cilardo et al., "High speed speculative multipliers based on speculative carry-save tree," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 12, pp. 3426– 3435, Dec. 2014.
8. S. Vassiliadis, E. Schwarz, and D. Hanrahan, "A general proof for overlapped multiple-bit scanning multiplications," IEEE Trans. Comput., vol. 38, no. 2, pp. 172–183, Feb. 1989.
9. D. Dobberpuhl et al., "A 200-MHz 64-b dual-issue CMOS microprocessor," IEEE J. Solid-State Circuits, vol. 27, no. 11, pp. 1555–1567, Nov. 1992.
10. E. M. Schwarz, R. M. A. III, and L. J. Sigal, "A radix-8 CMOS S/390 multiplier," in Proc. 13th IEEE Symp. Comput. Arithmetic (ARITH), Jul. 1997, pp. 2–9.
11. J. Clouser et al., "A 600-MHz superscalar floating-point processor," IEEE J. Solid-State Circuits, vol. 34, no. 7, pp. 1026–1029, Jul. 1999.
12. S. Oberman, "Floating point division and square root algorithms and implementation in the AMD-K7 microprocessor," in Proc. 14th IEEE Symp. Comput. Arithmetic (ARITH), Apr. 1999, pp. 106–115.
13. R. Senthinathan et al., "A 650-MHz, IA-32 microprocessor with enhanced data streaming for graphics and video," IEEE J. Solid-State Circuits, vol. 34, no. 11, pp. 1454–1465, Nov. 1999.
14. K. Muhammad et al., "Speed, power, area, latency tradeoffs in adaptive FIR filtering for PRML read channels," IEEE Trans. Very Large Scale Intgr. Syst., vol. 9, no. 1, pp. 42–51, Feb. 2001.
15. G. Colon-Bonet and P. Winterrowd, "Multiplier evolution: A family of multiplier VLSI implementations," Comput. J., vol. 51, no. 5, pp. 585–594, 2008.

16. R. Riedlinger et al., “A 32 nm, 3.1 billion transistors, 12 wide issue itanium processor for mission-critical servers,” IEEE J. Solid-State Circuits, vol. 47, no. 1, pp. 177–193, Jan. 2012.
17. B. Cherkauer and E. Friedman, “A hybrid radix-4/radix-8 low power signed multiplier architecture,” IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 44, no. 8, pp. 656–659, Aug. 1997.
18. D. Lutz, “ARM FPUs: Low latency is low energy,” presented at the 22nd IEEE Symposium in Computer Arithmetic, Jun. 2015, [last visited Jul. 1, 2016]. [Online]. Available: http://arith22.gforge.inria.fr/slides/s1-lutz.pdf
19. V. G. Oklobdzija, D. Villeger, and S. S. Liu, “A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach,” IEEE Trans. Comput., vol. 45, no. 3, pp. 294–306, Mar. 1996.