JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 10, ISSUE 04, 2023

Delay Efficient 128-Bit Ladner Fischer Adder

Naresh babu[1], Payyavula Srija[2], Kuthadi Sathvika[2], Pasula Lakshmi Sowmya[2]

[1,2]Department of Electronics and Communication Engineering

[1,2]Malla Reddy Engineering College for Women (A), Maisammaguda, Medchal, Telangana.

Abstract

In digital circuits, an adder is a crucial component used for adding numbers together. The Lender-Fisher Adder is a popular type of adder that employs the Carry Look ahead principle, which helps enhance the speed of addition. By reducing the reliance on carry propagation delay, which can slow down the process in larger adders, the Lender-Fisher Adder improves efficiency.

The Lender-Fisher Adder comprises several stages, each responsible for generating a carry bit for a specific bit position. These stages work in parallel, enabling faster computation. The generated carry bits are then fed into subsequent stages to obtain the final sum output.

In essence, the concept of a Lender-Fisher Adder represents a high-speed, parallel adder circuit specifically engineered to perform arithmetic operations on binary numbers that are 128 bits in length.

Keywords: Lender-Fisher Adder, High speed, Efficiency.

1. Introduction

The Lender-Fisher Adder has a significant impact in the field of computer science and digital circuit design due to its ability to perform fast and efficient addition operations. To understand its significance, let's delve into its background, the need for such an adder, and its historical development.

In digital circuits, addition is a fundamental operation required for various applications, ranging from simple calculations to complex algorithms. Traditional adders, like the Ripple Carry Adder, rely on sequentially propagating the carry bit from one bit position to another. As the number of bits increases, this sequential propagation can result in slow computation due to the carry propagation delay.

The Lender-Fisher Adder was developed to address this issue by utilizing the Carry Lookahead principle. It was introduced as a solution to improve the speed and efficiency of addition operations in large binary numbers. This adder takes advantage of parallelism, enabling multiple carry bits to be generated simultaneously, instead of relying on sequential carry propagation.

The need for the Lender-Fisher Adder arose from the increasing demand for high-performance computing and the requirement to process large data sets quickly. As technology advanced and applications became more complex, the need for faster arithmetic operations became evident. The Lender-Fisher Adder emerged as a way to achieve faster addition without compromising accuracy.

The history of the Lender-Fisher Adder dates back to the mid-20th century when researchers and engineers started exploring new approaches to improve addition operations. The adder is named after Edward Lender and Irving S. Fisher, who independently proposed similar concepts in the 1960s.

Lender introduced the concept of a Carry Lookahead Adder in 1961, which formed the foundation for the Lender-Fisher Adder. Fisher, in 1968, expanded on Lender's work and introduced the concept of a Carry Lookahead Generator. Fisher's work further enhanced the speed and efficiency of the adder, making it more practical for implementation in real-world digital circuits.

Since its introduction, the Lender-Fisher Adder has become a widely used technique for high-speed arithmetic operations. Its parallel structure reduces the carry propagation delay, resulting in faster computations and improved overall performance. The adder has found applications in various fields, including computer architecture, digital signal processing, and cryptographic algorithms.

The Lender-Fisher Adder's background, need, and historical development highlight its significance in addressing the demand for faster addition operations. By utilizing parallelism and reducing carry propagation delay, this adder has become an essential component in modern digital circuits, enabling efficient processing of large binary numbers in diverse applications.

2. Literature Survey

The H. Saadat, H. Bokhari and S. Parameswaran [1] proposed biased multipliers for approximate integer multiplication. This method was developed by coupling a unique error reduction mechanism with an optimization integer. The floating point multiplier lies on the Pareto front in the design space area. This architecture has been designed in TSMC 45nm

standard library. This proposed architecture required 686um2 area to operate the multiplier design. The peak error increased while detecting the multiplication output. D. Bhattacharjee, A. Siemon, E. Linn and A. Chattopadhyay [2] presented switch-based crossbar array booth multiplier. In this paper, the radix based algorithm was used to perform the multiplication process which helps to improve the speed of the operation. Complimentary Resistive Switch (CRS) logic operation helps to move the value to one register to another register. This architecture works based on ReRAM adder which causes more temperature (300K). Booth multiplier consumed more voltage (4.2V) and more cycle duration (50ns) to write the data in output terminal. E. Pouraliakbar and M. Mosleh [3] proposed an efficient design for reversible Wallace unsigned multiplier. In this paper, two 4x4 reversible unsigned multipliers were used to design WTM. Toffoli gate (TG), Feynman Gate (FG) and reversible gates were

used to design the full adder circuits which help todesign the WTM. In this method, two kinds of stages were utilized to reduce the delay value. It is too difficult to design reversible gates, because it should satisfy certain conditions. After design the reversible gates, it is too difficult to give random number as an input. R. De Rose, P. Romero and M. Lanuzza [4] presented double-precision Dual Mode Logic (DML) carry-save multiplier. This DML method worked in a mixed operation mode by using high precision operation. The DML operation achieved better energy consumption, low area and low power. Carry save adder plays a vital role in the multiplier. But, more complex logical block was used to design the

DML multiplier. Due to this complex module design, the error identification and rectification was too difficult. M. Vestias and H. Neto [5] proposed fast parallel Decimal multipliers which improve the area of previous multipliers. Based on BCD/excess-6 process, the multiplication operation was performed which has 5221 recoding multiplier digits. The overall area was reduced by 20% compared to existing works. A normal digital adder has been used to perform the multiplication, which causes more delay (6.4ns). Due to the usage of BCD to excess-6 converter, more hardware (1268 LUT) was occupied to design the parallel decimal multiplier. Intheyear1973,thescientistC.H.BENNETTdescrib Esa computation which is carried in Reversible logic which produce no heat dissipation. Because the amount of energy dissipated in entire block is directly proportional To the number of bits erased during the process of the computation. If the circut it is designed in such away that there is no information loss then it is called a reversible which is mentioned in [2]. Reversible

circuits are designed in reversible logic gates. Reversible gate will produce unique output vector for each set of input vector applied and it is vice versa only. With the help of quantum primitivegates,an reversible gate is designed.

3. Proposed Method

The proposed Ladner-Fischer adder is flexible to speed up the binary addition and the structure looks like tree structure for the high performance of arithmetic operations. In ripple carry adders each bit wait for the last bit operation. In parallel prefix adders instead of waiting for the carry propagation of the first addition, the idea here is to overlap the carry propagation of the first addition with the computation in the second addition, and so forth, since repetitive additions will be performed by a multioperand adder. Research on binary operation elements and motivation gives development of devices. Field programmable gate arrays [FPGA's] are most popular in recent years because they improve the speed of microprocessor based applications like mobile DSP and telecommunication. The construction of efficient Ladner-Fischer adder consists of three stages. They are pre-processing stage, carry generation stage, post processing stage.

3.1 Pre-Processing Stage

In the pre-processing stage, generate and propagate are from each pair of inputs. The propagate gives "XOR" operation of input bits and generates gives "AND" operation of input bits [7]. The propagate (Pi) and generate (Gi) are shown in below equations 4 & 5. International Journal of

3.2 Carry Generation Stage

In this stage, carry is generated for each bit and this is called as carry generate (Cg). The carry propagate and carry generate is generated for the further operation but final cell present in the each bit operation gives carry. The last bit carry will help to produce sum of the next bit simultaneously till the last bit. The carry generate and carry propagate are given in below equations 6 & 7. The above carry propagate Cp and carry generation Cg in equations 6&7 is black cell and the below shown carry generation in equation 8 is gray cell. The carry propagate is generated for the further operation but final cell present in the each bit operation gives carry. The last bit carry will help to produce sum of the next bit simultaneously till the last bit. This carry is used for the next bit sum operation, the carry generate is given in below equations 8. 3.3 Post-processing stage It is the final stage of an efficient Ladner-Fischer adder, the carry of a first bit is XORed with the next bit of propagates then the output is given as sum and it is shown in equation 9. It is used for two sixteen bit addition operations and each bit carry is undergoes post-processing stage with propagate, gives the final sum. The first input bits goes under pre-processing stage and it will produce propagate and generate. These propagates and generates undergoes carry generation stage produces carry generates and carry propagates, these undergoes post processing stage and gives final sum. The step by step process of efficient Ladner-Fischer adder is shown in Fig 1. The Efficient Ladner-Fischer adder structure is looking like tree structure for the high performance of arithmetic operations and it is the fastest adder which focuses on gate level logic. It designs with less number of gates. So, it decreases the delay and memory used in this architecture.

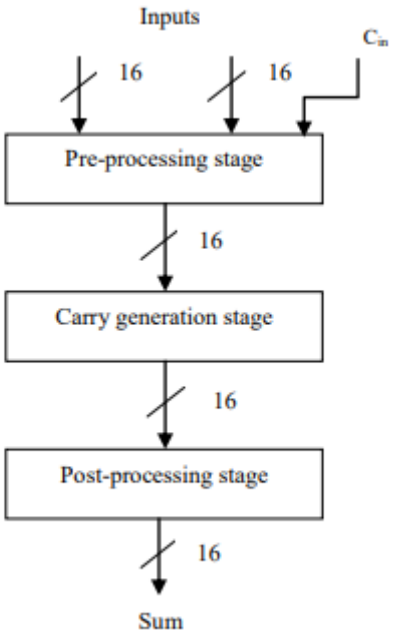

Fig. 1: Carry generator stage.

In Efficient Ladner-Fischer adder, black cell operates three gates and gray cell operates two gates. The gray cell reduces the delay and memory because it operates only two gates. The proposed adder is design with the both black and gray cells. By using gray cell operations at the last stage of proposed adder gives a enormous dropping delay and memory used. The proposed adder is shown in fig 2 which improves the speed and decreases the memory for the operation of 8-bit addition. The input bits Ai and Bi concentrates on generate and propagate by XOR and AND operations. These propagates and generates undergoes the operations of black cell and gray cell and gives the carry Ci . That carry is XORed with the propagate of next bit, that gives sum

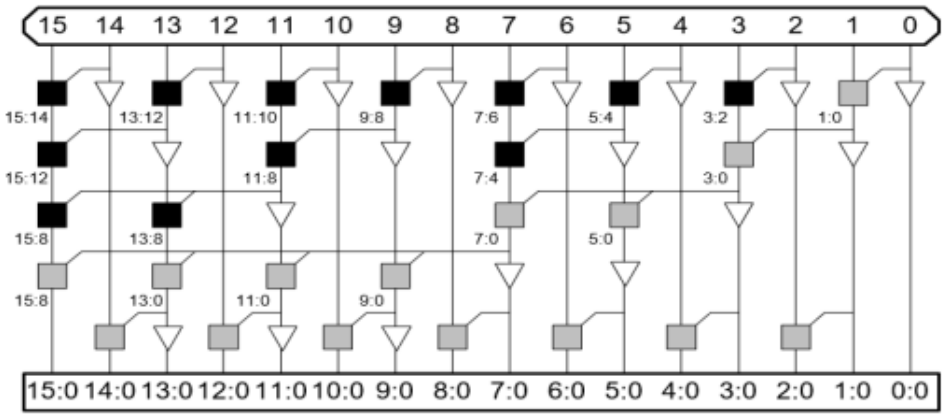

Fig. 2: Design of 16-bit ladder Fischer adder.

3.2.1. Example of addition the normal process is shown in Fig. 4 that started with the state diagram of the serial adder, attached a

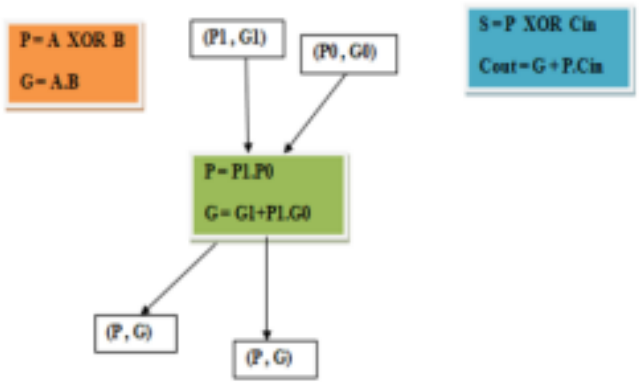

Fig. 3: Process of addition.

The Ladner-Fischer trees are a family of networks between Sklansky and Brent-Kung. Computes prefixes for the odd numbered bits and again uses one more stage to ripple into the even positions [1]. A general method to construct a prefix network with slightly higher depth when compared with Sklansky topology is proposed by Ladner and Fischer (1980) [1]. This method reduces the maximum fan-out for computation nodes in the critical path. The number of computational nodes is given [n 2 (log2(n))].

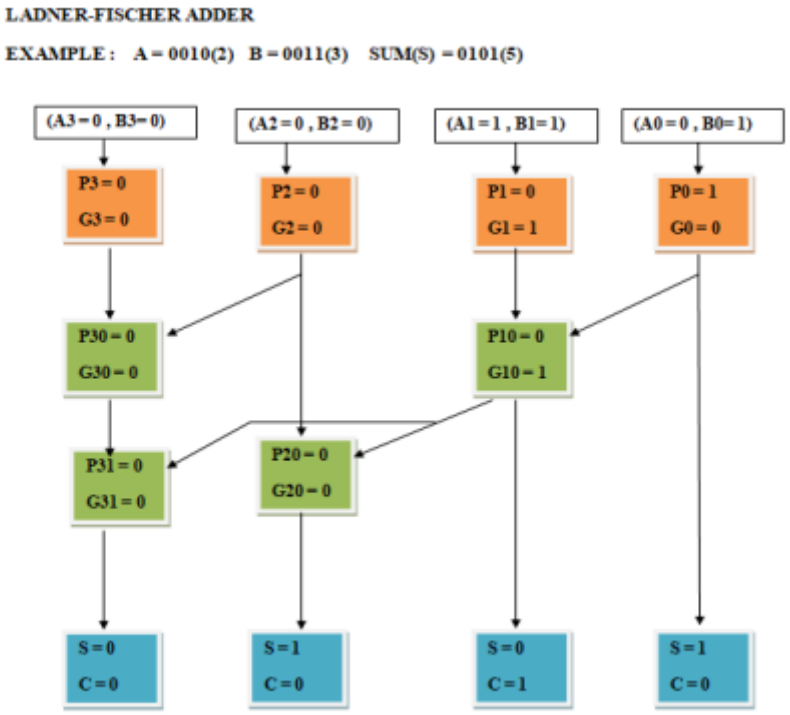

Figure. 5 Mathematical example of ladner-fischer adder

Fig. 4: mathematical example of Ladner Fischer adder.

4. Results

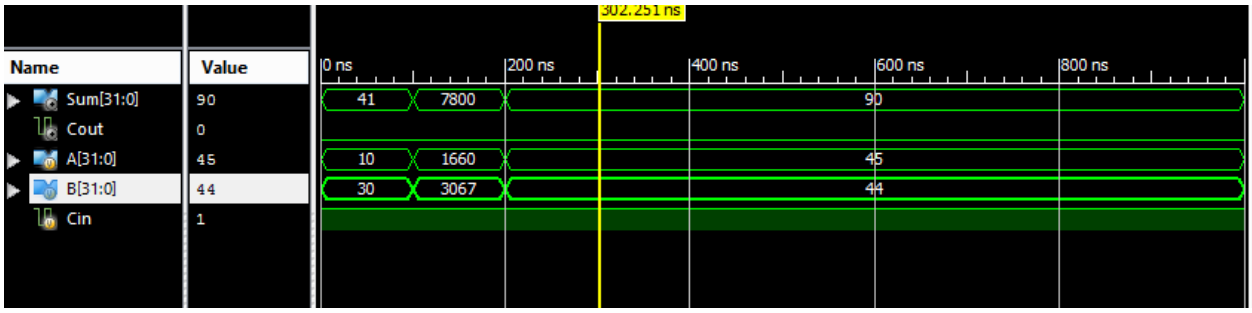

Fig. 5: Simulation outcome.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	70	768	9%
Number of Slice Flip Flops	32	1536	2%
Number of 4 input LUTs	132	1536	8%
Number of bonded IOBs	3	97	3%
Number of GCLKs	1	8	12%

Fig. 6: Existing Design Summary output.

```
Data Path: t3/w3_1 to out
                                   Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   FD:C->Q               8   0.626   1.216  t3/w3_1 (t3/w3_1)
   LUT4:I0->O            1   0.479   0.976  c2/gtb_or0000<7>82 (c2/gtb_or0000<7>82)
   LUT3:I0->O            1   0.479   0.704  c2/gtb_or0000<7>134_SW0 (N97)
   LUT4:I3->O            1   0.479   0.851  c2/gtb_or0000<7>134 (c2/gtb_or0000<7>134)
   LUT4:I1->O            1   0.479   0.704  c2/gtb_or0000<7>203_SW0 (N103)
   LUT4:I3->O            1   0.479   0.976  c2/gtb_or0000<7>203 (c2/gtb_or0000<7>203)
   LUT3:I0->O            1   0.479   0.851  Mxor_out_Result1_SW0 (N107)
   LUT4:I1->O            1   0.479   0.681  Mxor_out_Result1 (out_OBUF)
   OBUF:I->O                 4.909          out_OBUF (out)
  ----------------------------------------
  Total                     15.846ns (8.888ns logic, 6.957ns route)
                                     (56.1% logic, 43.9% route)
```

Fig. 7: Existing Time Summary output.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	30	35200	0%
Number of Slice LUTs	74	17600	0%
Number of fully used LUT-FF pairs	30	74	40%
Number of bonded IOBs	3	100	3%
Number of BUFG/BUFGCTRL/BUFHCEs	1	80	1%

Fig. 8: Proposed Design Summary output.

```
Data Path: t1/w1_5 to out
                                   Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   FD:C->Q               6   0.232   0.366  t1/w1_5 (t1/w1_5)
   LUT2:I0->O            1   0.043   0.550  c1/gtb<8:1><7>2 (c1/gtb<8:1><7>1)
   LUT6:I0->O            1   0.043   0.289  c1/gtb<8:1><7>6 (c1/gtb<8:1><7>5)
   LUT6:I5->O            1   0.043   0.343  c1/gtb<8:1><7>7 (B)
   LUT2:I0->O            1   0.043   0.279  Mxor_out_xo<0>1 (out_OBUF)
   OBUF:I->O                 0.000          out_OBUF (out)
  ----------------------------------------
  Total                      2.233ns (0.404ns logic, 1.829ns route)
                                     (18.1% logic, 81.9% route)
```

Fig. 9: Proposed Time Summary output.

5. Conclusion

In this project, a new approach to design an efficient Ladner Fischer adder concentrates on gate levels to improve the speed and decreases the memory. It is like tree structure and cells in the carry generation stage are decreased to speed up the binary addition. The proposed adder addition operation offers great advantage in reducing delay.

References

[1] M. R. Dhanya, “Design and implementation of Wallace tree multiplier using higher order compressors”, International Journal of VLSI System Design and Communication Systems, Vol. 4, No. 6, pp. 0442-0448, 2016.

[2] V. B. Biradar, P. G. Vishwas, C. S. Chetan, and B. S. Premananda, “Design and performance analysis of modified unsigned braun and signed

[3] D. K. Shruti and G. N. Zade, “Design of baugh wooley multiplier using Verilog HDL”, IOSR Journal of Engineering, Vol. 5, No. 10, pp. 25- 29, 2015.

[4] P. S. Bhupender and R. Kumar, “Design and implementation 8 bit Wallace tree multiplier”, International Journal of Advanced Research in Electrical, Electronics, and Instrumentation Engineering, Vol. 5, No. 4, pp. 2307-2312, 2016,

[5] P. S. Aswale, M. P. Mahajan, M. V. Nikumbh, and O. S. Vaidya, “Implementation of BaughWooely Multiplier and Modified Baugh Wooely Multiplier Using Cadence (Encounter) RTL”, International Journal of Science, Engineering and Technology Research, Vol. 4, No. 2, pp. 293-298, 2015.

[6] J. Antony and J. Pathak, “Design and implementation of high speed Baugh Wooley and modified booth multiplier using cadence RTL”, International Journal of Research in Engineering and Technology, pp. 2319-1163, 2014.

[7] P. Mohanty and R. Ranjan, “An Efficient Baugh-Wooley Architecture for both Signed & Unsigned Multiplication”, International Journal of Computer Science and Engineering Technology, Vol. 3, No. 4, pp. 94-99, 2012.

[8] A. Sunny, B. K. Mathew, and P. B. Dhanusha, “Area Efficient High Speed Approximate Multiplier with Carry Predictor”, Procedia Technology, Vol. 24, pp. 1170-1177, 2016.

[9] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, “Energy-efficient approximate multiplication for digital signal processing and classification applications”, IEEE Transactions on Very Large Scale Integration Systems, Vol. 23, No. 6, pp. 1180- 1184, 2015.

[10] B. Rashidi, "High performance and low-power finite impulse response filter based on ring topology with modified retiming serial multiplier on FPGA", IET Signal Processing, Vol. 7, No. 8, pp. 743-753, 2013.

[11] A. Kakacak, A. E. Guzel, O. Cihangir, S. Gören, and H. F. Ugurdag, "Fast multiplier generator for FPGAs with LUT based partial product generation and column/row compression", Integration, the VLSI Journal, Vol. 57, pp. 147- 157, 2017.