A Novel Adder Configuration for High Efficiency

(IJCSNT) International Journal of Communication Systems and Network Technologies
ISSN Number: 2053-6283
Volume 10, Issue No. 1, 86–96

https://journals.mirlabs.in/index.php/ijcsnt
DOI-10.18486/ijcsnt/10.1.131
IJCSNT

Marcus Lloyde George[1]

Abstract

Binary adders are very important in digital signal processing. Adders are basic building blocks of more complex units such as multipliers, dividers, square units, etc. This paper first presents the proposal and implementation of a new adder model called the Prediction Adder (MLG Adder). The paper also presents performance of this adder in comparison to the performance of the various adders studied. This design is highly effective and has lower processing time and latency. The system was synthesized for a variety of FPGA targets using Xilinx ISE Design Suite 14.7 Commercial Edition and performance was simulated with ISIM and other tools available from Xilinx ISE Design Suite 14.7.

Keywords

Arithmetic Circuits, Binary Adders, Arithmetic Logic, FPGAs In Arithmetic, Summing Circuits

Received: 21 December 2020; **Revised:** 14 April 2021; **Accepted:** 22 April 2021;
Published: 30 April 2021;

[1]Department of Electrical and Computer Engineering, University of the West Indies, St. Augustine, Trinidad and Tobago.

Corresponding author:
Email: Marcus.George@sta.uwi.edu

Introduction

Arithmetic Logic Units (ALUs) are very important components of processors that perform various arithmetic operations such as multiplication, division, addition, subtraction, cubing, squaring, etc. Off all operations the operation of multiplication is most elementary and most frequently used in ALUs. It allows one number to be scaled by another number. The operation of multiplication also forms the basis of many other complex arithmetic operations such as cubing, squaring, convolution, etc. Multiplication consumes significant time compared to other arithmetic units used in basic mathematical computations[1]. As a result, it is beneficial in the area of mathematical computation to present faster and more efficient mechanisms for implementing mathematical operations. Many digital units for digital signal processing utilize multipliers in their operation. The more we can reduce the latency of the multiplier used the shorter the latency of the operation on the digital signal processor. One method of reducing the latency of the multiplier is by use of a adder with shorter latency. This paper first presents a review of several adder types such as the Ripple Carry Adder, Carry Lookahead Adder, Carry-Select Adder, Carry-Skip Adder, Carry-Increment Adder and Manchester-Carry Adder. In this paper we also propose the design of two alternative adder configurations and present performance of them in comparison to the performance of the various adders studied.

Review of Binary Addition Theory

A binary adder is basically a digital logic circuit which generates an arithmetic sum Sum (S) of two binary numbers A and B using only combinational logic theory. A 1-bit binary full adder takes as input three 1-bit inputs A, B and Carry-in (Cin) and computes a 1-bit Sum (S) and Carry-out (Co)[2].

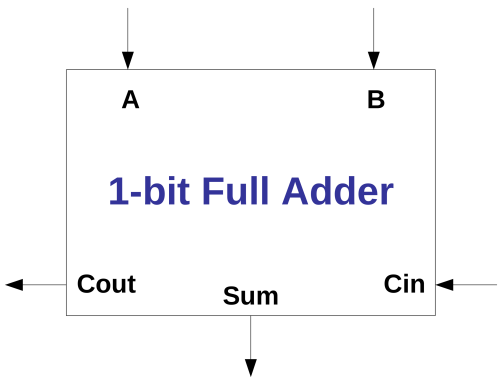

Figure 1. Datapath Interface Definition of 1-bit Full Adder

The operation of this 1-bit binary full adder operation is best described using the truth table below:

A ripple carry adder is a parallel binary adder which uses n full adders in parallel with all inputs asserted simultaneously to produce the sum which will be n-bits wide. A carry appears at the least significant adder of the system of adders and this carry is propagated through the n full adders to the most significantly, hence producing a wave of ripples outward. If the propagation delay of a 1-bit full adder is ta ns then the carry-in for the first stage arrives at the last stage in (n-1)ta ns. The ripple carry

Table 1. Truth Table for 1-bit Full Adder

A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

adder is simple but however has a long circuit delay due to the series of gates in the carry path from least significant to most significant bit. The longest circuit delay path through an n-bit ripple carry adder is 2n+2.

The carry lookahead adder is a 2-stage adder which has reduced delay when compared to the ripple carry adder. This adder can be simply implemented by combination of the ripple carry adder and a carry logic unit. The ripple-carry adder is modified to provide propagate function Pi which is the exclusive-OR of inputs Ai and Bi. The generate function Gi is also provided and is basically the AND of Ai and Bi. The carry logic only consists of OR and AND gates. Therefore the delay for each of the carry signals produced could just be a couple of gate delays, hence minimizing the overall delay compared to the ripple carry adder. The parts of the full-adders associated with the carry propagation path are separated from those not associated[3].

The carry lookahead adder uses partial full-adders (PFA) in its operation. This PFA computes sum, propagation and generate signals instead of computing sum and carry as done in ripple carry adder. The propagation and generate signals are used in stage-2 of the carry look ahead adder in computation of carry signals. Consider a 4-bit carry lookahead adder. The carry signals will be computed as follows:

$$P_i = A_i \text{ XOR } B_i \tag{1}$$

$$G_i = A_i \text{ AND } B_i \tag{2}$$

$$C_1 = G_0 + C_0P_0 \tag{3}$$

$$\begin{aligned} C_2 &= G_1 + P_1(G_0 + C_0P_0) \\ &= G_1 + G_0P_1 + C_0P_0P_1 \end{aligned} \tag{4}$$

$$\begin{aligned} C_3 &= G_2 + P_2(G_1 + G_0P_1 + C_0P_0P_1) \\ &= G_2 + G_1P_2 + G_0P_1P_2 + C_0P_0P_1P_2 \end{aligned} \tag{5}$$

$$
\begin{aligned}
C_4 &= G_3 + P_3(G_2 + G_1P_2 + G_0P_1P_2 + C_0P_0P_1P_2) \\
&= G_3 + G_2P_3 + G_1P_2P_3 + G_0P_1P_2P_3 \\
&\quad + C_0P_0P_1P_2P_3
\end{aligned} \tag{6}
$$

In the case of the carry-select adder the k-bits are split into nonoverlapping groups of bits of lengths that can possibly be different. Each group will generate two sets of sum and carry bits. One set assumes that the carry-in is 0 while the other one assumes the carry-in is 1. Depending on the carry-out from the least significant stages the adder selects the sum of the subsequent stage matching that carry-out.

Design of Various High Speed Adders

The 4-bit Ripple-Carry Adder (RCA) was implemented by combining four 1-bit full adders. These adders were arranged in consecutive order and input bits A and B along with the output S were split among the 4 adders.

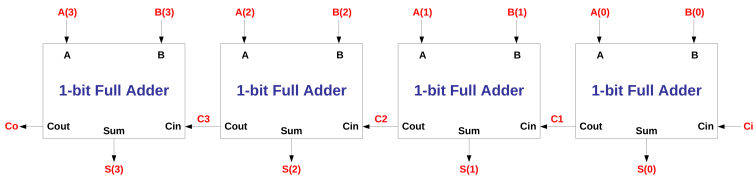

Figure 2. Datapath block diagram of 4-bit Ripple-Carry Adder (RCA)

The 4-bit Carry Lookahead Adder (CLA) was implemented by cascading four 1-bit inclusive full adders along with OR-AND gate pairs. The 1-bit inclusive full adder computed the outputs Generate (G) and Propagate (P).

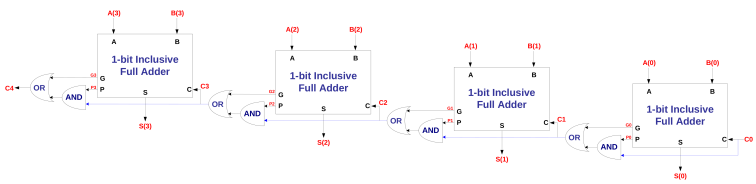

Figure 3. Datapath block diagram of 4-bit Carry Lookahead Adder (CLA)

The carry-select adders were constructed by simply the combination of full-adders and multiplexers. In the 2-bit version of the 4-bit Carry-Select Adder four 2-bit full adders were used along with two 2-bit 2 to 1 multiplexers and a 1-bit 2 to 1 multiplexer. Since addition in this adder occurs in groups of twos then the least significant 2 bits of A and B are assign to two adders, one adding them with Cin = 0, and the other Cin = 1. The sums of both adders are assigned to the inputs of a multiplexers and the value of the Cin to that stage determines which sum is selected. The same occurs for the most significant 2 bits of A and B.

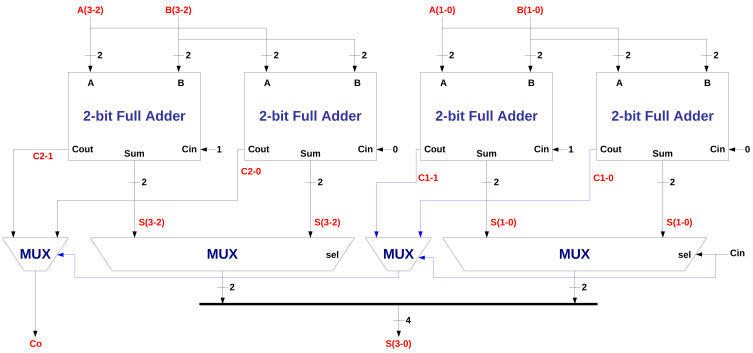

Figure 4. Datapath block diagram of 4-bit Carry-Select Adder (CSLA) - 2bit Version

Results and Testing for Various High Speed Adders

After implementation the adders were simulated using Isim to verify that they had been functionally correct. VHDL testbenches were created for each adder and timing simulations were performed. The test cases for these verification tests are shown in Table 2.

Table 2. Adder Test Cases

TEST#	INPUTS			OUTPUTS	
	A (HEX)	B (HEX)	Cin	Sum (HEX)	Cout
0	0	0	0	0	0
1	0	0	1	1	0
2	1	1	0	2	0
3	1	1	1	3	0
4	2	4	0	6	0
5	2	4	1	7	0
6	3	2	0	5	0
7	3	2	1	6	0
8	A	5	0	F	0
9	A	4	1	F	0
10	7	3	0	A	0
11	7	3	1	B	0
12	8	8	0	0	1
13	4	1	0	5	0
14	8	1	1	1	1

With these test cases in place the resulting Isim simulations for all adders (Figrure 18- 25) were obtained. All adders produced the correct sum and carry.

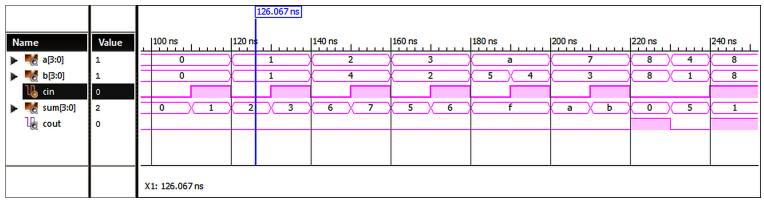

Figure 5. ISim simulation of 4-bit Ripple-Carry Adder (RCA)

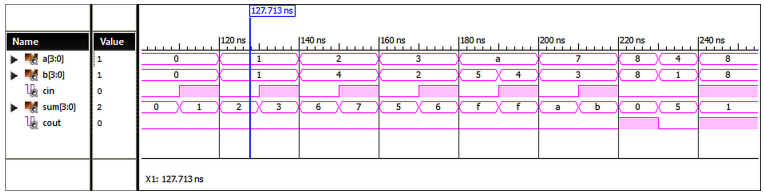

Figure 6. ISim simulation of 4-bit Carry Lookahead Adder (CLA)

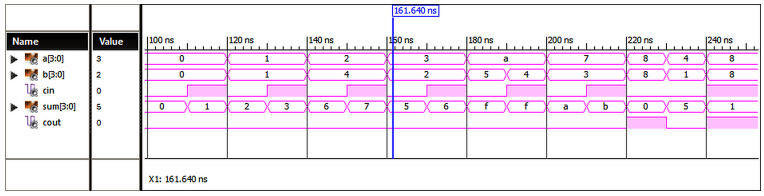

Figure 7. ISim simulation of 4-bit Carry-Select Adder (CSLA) - 2bit Version

Design of Prediction Adder (MLG Adder)

The prediction adder comprises of three fundamental components: 1-bit adder prediction block, adder correction block and multiplexer units for the purpose of selection of results from both adder prediction and adder correction blocks. The adder prediction block examines the inputs A, B and Cin and determines the possibility of the sum being '1' depending on these inputs. As long one of the inputs is '1' we assign LOGIC '1' to the output Sum. The carry Cout is determined by inputting inputs A and B to an AND gate. As long as both inputs A and B are LOGIC '1' then the carry output Cout will be assigned LOGIC '1'.

After computing Sum and Cout, an examination of them would indicate that for cases where two (2) of the three inputs A, B, Cin are LOGIC '1' the Sum and Cout need to be corrected. As such the adder correction block examines cases where two (2) of the three inputs A, B, Cin are LOGIC '1' and corrects them. In this case the Sum and Cout signals are both inverted. The adder correction block also provides a selection signal which will be used further by the multiplexer units. The first multiplexer is used for selection of the sum Sum from the original and corrected sum signals, while the second multiplexer is used for selection of the carry Cout from original and corrected carry out signals.

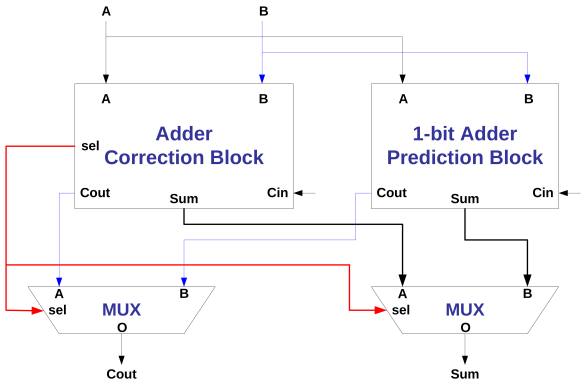

Figure 8. Datapath block diagram of 1-bit Prediction Adder

The 4-bit prediction adder was implemented by combining four 1-bit prediction adders. These adders were arranged in consecutive order and input bits A and B along with the output Sum were split among the 4 adders.

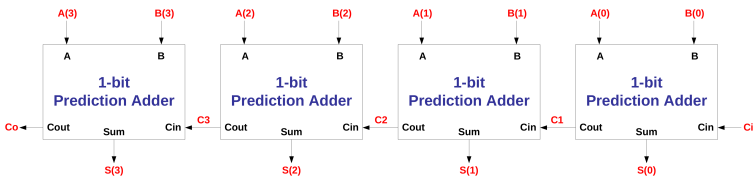

Figure 9. Datapath block diagram of 4-bit Prediction Adder

Results and Testing for Prediction Adder

After implementation the 1-bit prediction adder was simulated using Isim to verify that it had been functionally correct. A VHDL testbench was created for the adder and timing simulation was performed. From the simulation of Figure 3 the prediction adder correctly computes the sum and carry outputs.

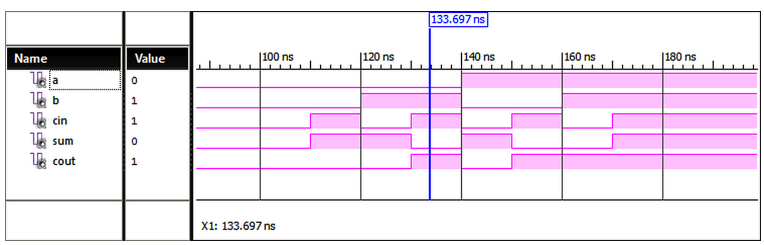

Figure 10. ISim simulation of 1-bit Prediction Adder

A VHDL testbench was created for the 4-bit adder and timing simulation was performed. The test cases for these verification tests are shown in Table 1.

Table 3. Adder Test Cases

TEST#	INPUTS			OUTPUTS	
	A (HEX)	B (HEX)	Cin	Sum (HEX)	Cout
0	0	0	0	0	0
1	0	0	1	1	0
2	1	1	0	2	0
3	1	1	1	3	0
4	2	4	0	6	0
5	2	4	1	7	0
6	3	2	0	5	0
7	3	2	1	6	0
8	A	5	0	F	0
9	A	4	1	F	0
10	7	3	0	A	0
11	7	3	1	B	0
12	8	8	0	0	1
13	4	1	0	5	0
14	8	1	1	1	1

With these test cases in place the resulting Isim simulation for the prediction adder was obtained. Based on the results of the Isim simulation shown in Figure 4 the 4-bit prediction adder produced the correct sum and carry.

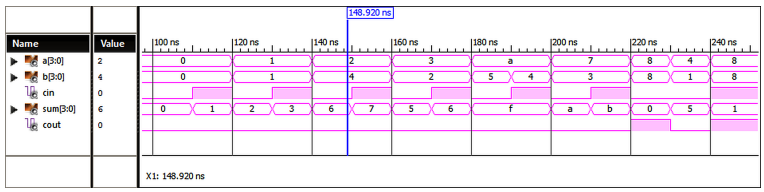

Figure 11. ISim simulation of 4-bit Prediction Adder

The Maximum Combinational Path Delay and Logic Utilization & Distribution for the prediction adder were obtained via the synthesis report after implementation on Xilinx ISE. Table 2 presents the information collected. Table 3 presents performance comparison of the 4-bit prediction adder with several other adders analyzed.

Table 4. Maximum Combinational Path Delay and Logic Utilization & Distribution for Prediction Adders

Adder Type	Maximum Combinational Path Delay / ns	Logic Utilization & Distribution
1bit Prediction Adder	7.824	2 input LUTs: 8 out of 15,360 1% Occupied Slices: 1 out of 7,680 1% Bonded IOBs: 5 out of 173 8%
4bit Prediction Adder	12.008	2 input LUTs: 8 out of 15,360 1% Occupied Slices: 6 out of 7,680 1% Bonded IOBs: 14 out of 173 8%

Table 5. Maximum Combinational Path Delay and Logic Utilization & Distribution for Adders

Adder Type	**Maximum Combinational Path Delay (ns)**	**Logic Utilization & Distribution**
Ripple Carry Adder	12.008	4 input LUTs: 8 out of 15,360 (1%) Occupied Slices: 6 out of 7,680 (1%) Bonded IOBs: 14 out of 173 (8%)
Carry Lookahead Adder	11.908	4 input LUTs: 8 out of 15,360 (1%) Occupied Slices: 6 out of 7,680 (1%) Bonded IOBs: 14 out of 173 (8%)
Carry-Select Adder (2B)	10.653	4 input LUTs: 8 out of 15,360 (1%) Occupied Slices: 5 out of 7,680 (1%) Bonded IOBs: 14 out of 173 (8%)
Prediction Adder	12.008	2 input LUTs: 8 out of 15,360 (1%) Occupied Slices: 6 out of 7,680 (1%) Bonded IOBs: 14 out of 173 (8%)

Conclusion

The results of validation indicate that the 4-bit prediction adder carry-select adder (2bit groups) produced the shortest latency among the adders analyzed. The 2-bit carry-skip adder provided the second best latency while the 2-bit carry-look ahead adder produced the third fastest latency.

To determine the effect of manipulating the basic building block on the adder, the faster adder – the 4-bit carry-select adder (2bit groups) – was selected as the platform for development while the 2-bit carry-skip adder and 2-bit carry-look ahead adder were used as the basic building block. It was observed from the systnthesis report that the use of carry-look ahead and carry-skip adders in place of full-adders did not improve the performance of the 4-bit carry-select adder (2bit groups).

References

[1] Mano, M. Morris and Charles R. Kime. 1997. Logic AND Computer Design Fundamentals. Prentice Hall, New Jersey. Volume 1.

[2] Koren, Israel. 2001. Computer Arithmetic Algorithms. 2nd Ed. A K Peters, Natick, Massachusetts. ISBM 1-56881-160-8.

[3] Ling, Huey. 1981. High-speed Binary Adder. IBM 1. RES DEVELOP. 0 VOL. 25 0 NO. 3 0 MAY 1981

[4] GS Tomar, Marcus L George, "Modified Binary Multiplier Architecture to Achieve Reduced Latency and Hardware Utilization", Wireless Personal Communication, Vol.98, No.4, pp.3549-3561, 2018.

[5] Chen, Chien-In Henry and Joel Yuen. 1992. An Efficient Approach to Pipeline Scheme for Concurrent Testing of VLSI Circuits. Proceedings of IEEE/ACM Design Automation. Pg 657-660. DI: 0-7803-0593-0/92.

[6] Hill, Eric L. and Mikko H. Lipasti. 2007. Transparent Mode Flip-Flops for Collapsible Pipelines. Pp 553-560. DI: 1-4244-1258-7/07.

[7] Choi, Jung Hwan, Byung Guk Kim, Aurobindo Dasgupta and Kaushik Roy. 2010. Improved Clock-Gating Control Scheme for Transparent Pipeline. Pg: 401 – 406. DI: 978-1-4244-5767-0/10.

[8] L. Benini and G. D. Micheli. 1996. "Automatic synthesis of low-power gated clock finite-state machines," IEEE Trans. on CAD, vol. 15, no. 6, pp. 630–643.

[9] Hennessey, John and David Patterson. 2003. Computer Architecture. A Quantitative Approach. 3rd ed. San Francisco: Morgan Kaufmann Publishers.

[10] P. Babighian et al. 2005. "A scalable algorithm for RTL insertion of gated clocks based on ODCs computation," IEEE Trans. on CAD, vol. 24, no. 1, pp. 29–42.

[11] H. Jacobson. 2004. Improved clock-gating through transparent pipelining," in Proc. of ISLPED. pp. 26–31.

[12] Panato, Alex, Sandro Silva, Flavio Wagner, Marcelo Johann, Ricardo Reis amd Sergio Bampi. 2004. Design of Very Deep Pipelined Multipliers for FPGAs. Proceedings of the Design, Automation and test in Europe Conference and Exibition Designer's Forum (DATE'04). IEEE Boston: Computer Society. DI: 1530-1591/04.

[13] Suryanarayana B. Tatapudi and Jose' G. Delgado-Frias. 2005. Designing Pipelined Systems with Clock Period Approaching Pipeline Register Delay. DI: 0-7803-9197-7/05.

[14] Jain, Anna, Baisakhy Dash, Ajit Kumar Panda and Muchharla Suresh. 2011. FPGA Design of a Fast 32-but Floating Point Multiplier Unit.

[15] Li, Zheng, Haimin Chen and Xianwen Yang. 2010. A Hardware Multiplier Design of Embedded Microprocessor. DI: 978-1-4244-6943-7/10.

[16] D. C. Michael, Y. Q. Zhang and Q. Li. 2005. Advanced Digital with the Verilog HDL etc translating, Beijing: Publishing House of Electronics Industry.

[17] Yilmaz, Mahmut, Derek R. Hower, Sule Ozev and Daniel J. Sorin. 2006. Self-Checking and Self-Diagnosing 32-bit Microprocessor Multiplier. International Test Conference. DI: 1-4244-0292-1/06.

[18] Qi, Haibing, Song Sun and Jianlan Feng. 2010. A FPGA Pipelining Design Method of Gradient Adaptive Lattice Joint Processor. 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics. pg 309-312. DI: 978-1-4244-5194-4/10.

[19] Mak, Terrence, Pete Sedcole, Peter Cheung and Wayne Luk. 2008. Wave-Pipelined Signaling for On-FPGA Communication. DI: 978-1-4244-2796-3/08.

[20] Zhang, Zhe and Xiaoming Hu. 2009. A Novel Pipelining Scheme for Network-on-Chip Router. 2009 Third International Symposium on Intelligent Information Technology Application. pg. 372-375. DI: 978-0-7695-3859-4/09.

[21] P. Bhattacharyya, "Performance Analysis of a Low-Power High-Speed Hybrid l-bit Full Adder Circuit," IEEE Trans. VLSI, vol. 23, 2015.

[22] H. Naseri and S. Timarchi, "Low-Power and Fast Full Adder by Exploring New XOR and XNOR Gates," IEEE Trans. VLSI, vol. 26, no. 8, 2018.

[23] H. E. Yantır, A. M. Eltawil and F. J. Kurdahi, "A Two-Dimensional Associative Processor," IEEE Trans. VLSI, vol. 26, no. 9, 2018.

[24] Ashish BAGWARI and I. KATNA, "Low Power Ripple Carry Adder Using Hybrid 1-Bit Full Adder Circuit," 2019 11th International Conference on Computational Intellisence and Communication Networks (CICN), Honolulu, HI, USA, 2019, pp. 124 - 127.