Mr.N.B.Jilani, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

High Speed 32-Bit Vedic Multiplier Using Pipelined Architecture

[1]Mr.N.B.Jilani,[2]Dr.P.Prasanna Murali Krishna, [3]Perumalla Geeta Bharathi,[4]Venna Leelavathi,[5]Porumamilla Dharshini,[6]Sure Anusha

[1]professor,Department of ECE,KITS-Markapur
[2]Professor,Department of ECE,KITS-Markapur
[3,4,5,6] Students,Department of ECE,KITS-Markapur

Abstract— Based on the historical practice of Vedic multiplication, this proposed study details a large-number variant of the Vedic multiplier. Since it improves over the alternatives in terms of speed, area, hardware complexity, power consumption, and scalability, a Vedic multiplier is the way to go. While Xilinx is used for circuit design, the Modelsim tool is used for implementation using Verilog HDL. The Carry Save Adder, the Carry Lookahead Adder, and the Ripple Carry Adder are three 32-bit Vedic multiplier designs that are analyzed in this paper. They use different adder architectures. Around 0.082W was the power consumption of all three designs. With a latency of 26.466 ns and almost similar power usage, the 32-bit CLA-based Vedic multiplier was the quickest. The RCA-based multiplier outperformed all other Vedic multipliers in terms of area usage. Search Terms—Circuit Design, Power Consumption, Vedic Mathematics, Parallel Processing, and High-Speed Multiplication [4]. The procedure for practicing vedic multiplication is shown in Fig. 1. By dividing the multiplication operation into smaller, more manageable parts, this method offers a new solution to the problem. When compared to older multiplier designs, this may provide better results, particularly when taking into account the power and performance limitations of contemporary DSP systems.

Index Terms—High-speed multiplication, Power consumption, Vedic mathematics, Parallel processing, Circuit design.

I. INTRODUCTION

Multiplication is an essential operation in digital signal processing, a field that demands efficient and quick computation. In particular, DSP system performance is severely limited by the time required to complete these multiplication operations. There is a present push to create high-speed multipliers with low power consumption [1]. Recent years have seen a proliferation of multiplier algorithms and design processes, all with the same overarching goal: to remain under stringent power constraints while simultaneously increasing speed [2]. There is a potential solution to the problem of inefficient and slow multiplication in ancient Vedic mathematics. "More than 1700 percent faster than ordinary math" [3] is a common claim made about Vedic mathematics. There are sixteen'sutras,' or principles, that form the basis of this mathematical system from ancient India. 'Urdhva Tiryakb- hyam,' which translates to "Vertically and Crosswise," is one such sutra. There is much potential in incorporating algorithms from Vedic mathematics into current digital designs. Computer power and system efficiency might be significantly enhanced if engineers take use of these approaches' inherent efficiency. As an example, the 'Urd- hva Tiryakbhyam' sutra may be executed with much less time spent multiplying by employing parallel processing methods. Because of this, DSP algorithms may be executed more quickly, which is especially useful for real-time applications. Reduced power consumption is another benefit of using Vedic algorithms due to their simplified computational complexity. Because of the critical nature of power efficiency in embedded and mobile systems, this is of utmost importance [5]. Battery life and heat dissipation may both be improved using multipliers based on Vedic mathematics by lowering the number of

transistors switching during multiplication. In this paper, we go over the various Vedic multiplier methods and compare them with each other in terms of power, area (LUTs), and delay. The goal is to find the best way to combine modern technology with this old but useful algorithm by implementing the multipliers on a real FPGA.

II. LITERATURE REVIEW

A 32-bit multiplier may be designed in a variety of ways, each with its own set of advantages and disadvantages in terms of complexity, size, power consumption, and speed.

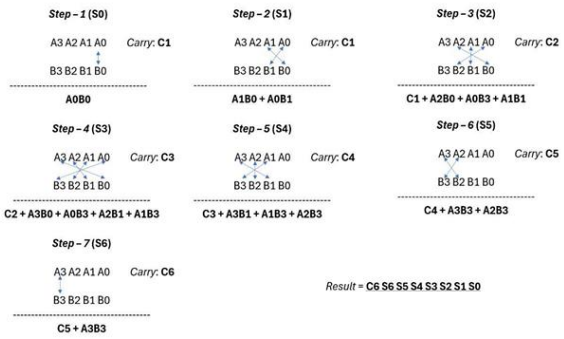

Fig. 1. Vedic Multiplication for 4-bit numbers,

Here we see the use of the "Vertical and Crosswise method," also known as Urdhva Tiryakbhyamsutra. One simple method that puts partial products in a grid and adds them up is an array multiplier, which forms the foundation of the Vedic Multiplier design [6]. An array of complete adders and multipliers results in a regular and predictable architecture, making it easy to comprehend and construct. The amount of adders and multipliers in an array may make it enormous, and the carry propagation in adders can restrict its performance. 2. The Radix-4 Booth Multiplier is a method that encodes the multiplier and thereby minimizes the number of partial products. Signed and unsigned integers are equally amenable to the Booth multiplier [7]. In comparison to simple array multipliers, it uses less partial products, therefore reducing the total number of partial products. Because there is less need to generate partial products, the speed might be quicker. But it's more complicated than an array multiplier, and there's extra design cost for partial product encoding and decoding. 3. Wallace Tree Multiplier: With no carriers propagated, a set of counters in the Wallace tree architecture adds all the components of all the partial products in each column simultaneously. This new matrix is lowered by an additional set of counters, and the process continues until a two-row matrix is generated. A common tool is the 3:2 counter.

In most cases, a carry propagate adder is used to attach the final results [8]. As opposed to array multipliers, this approach sums the partial products more efficiently by minimizing the number of addition steps. Having to account for tree structure and carry-save adders, the Wallace tree multiplier is more intricate in design and hence takes up more space [2]. Digital signal processing, encryption, picture and video processing, scientific computing, and other areas are finding high-speed and efficient multipliers to be more important because to the ever-increasing need for enhanced computing capabilities. Many modern apps rely on the underlying technology to handle massive volumes of data in real-time. Therefore, a solution is required that is highly scalable, uses less power, has low software and hardware complexity, and requires less processing time. Not all of the previously suggested approaches work. Section A. Approach and Benefits A number of design factors and performance measures, including complexity, area, power consumption, and speed, must be considered when comparing a 32-bit Vedic multiplier to other kinds of multipliers, such as the array multiplier, the Booth multiplier, and the Wal-lace tree multiplier. Several reasons why a Vedic multiplier might be preferable are as follows: 1. Quickness When it comes to multiplication time, Vedic multipliers—which are based on old Vedic mathematical techniques—may use algorithms that are more efficient. One technique that speeds up computing is Urdhva

Tiryakbhyam (Vertically and Crosswise). It does this by reducing the amount of partial products and addition steps.

Array multipliers are simple to implement, but they may have greater propagation delays than other methods because of the high number of adders required to add the partial products [2]. Reduce the amount of partial products, particularly for signed numbers, using Booth's approach. On the other hand, it may not always be faster than the Vedic technique for unsigned integers and may be more complicated [2].

- Wallace tree multipliers may reduce the amount of successive addition processes, which allows them to be extremely fast. Nevertheless, they are intricate and could need substantial hardware resources [2]. 2. The Complexity of Area and Hardware A more compact architecture is possible with Vedic multipliers as compared to array multipliers. Hardware complexity and space needs may be lowered due to the reduced number of adders and intermediary stages [9].
- Array multipliers may be complicated and resource-intensive due to the grid of AND gates and adders they need [9].
- Booth multipliers may get more sophisticated and take up more space due to the extra hardware needed for encoding and decoding the multiplier. One such enhanced version of the Booth multiplier that uses the Hsin-Lei encoder instead of the Booth encoder in Rizalafandi's design [10] reduces complexity but may not be as efficient as the Vedic multipliers.
- Despite their speed benefits, Wallace tree multipliers need sophisticated components that take a lot of space, such as carry-save adders [11]. 3. Energy Usage There may be fewer logic gates and stages in a Vedic multiplier, which might lead to lower power consumption.
- Array multipliers often consume a lot of power due to the high number of complete adders and AND gates they need [9]. Although booth multipliers may use less power than array multipliers, the extra hardware needed for encoding can cause them to still use more power than array multiplies. Though they're quick, the complicated structure of Wallace tree multipliers could cause them to use more power than necessary. [11]. 4. The capacity to expand
- The recursive and repeating character of Vedic multipliers makes them well-suited for scaling to greater bit widths.
- The practicality of array multipliers decreases with increasing bit-width, as the complexity and number of gates rise quadratically. Although it works OK at smaller bit widths, compared to Vedic multipliers, Booth's technique may not be the best choice at bigger ones.
- Wallace trees are scalable as well, but they may become complicated and drain resources as bit width grows.

III. DESIGN AND PROPOSED METHODS

Ancient algorithms were used to create a 32-bit Vedic multiplier that was then adapted to the nary system for more efficient and speedier computations. To assess the outcomes, the Multiplier employs smaller chunks of code starting from the ground up. A 32-bit multiplication may be computed by combining smaller 16,8,4,2-bit multipliers in a sequential and organized fashion. Figure 3 depicts a 16-bit multiplier made up of four 8-bit multipliers, two 16-bit CSAs, one OR gate, and one 8-bit adder, whereas Figure 2 shows a 2-bit Vedic Multiplier architecture.

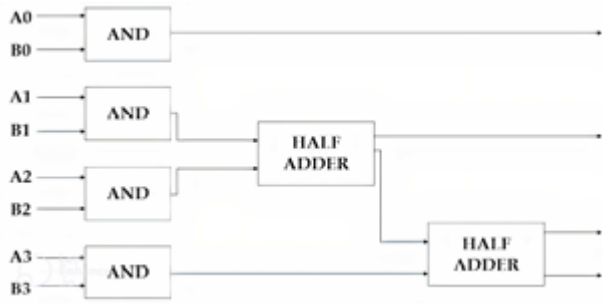

Fig. 2. 2-Bit Vedic Multiplier Schematic It is the building block for all other Vedic Multipliers

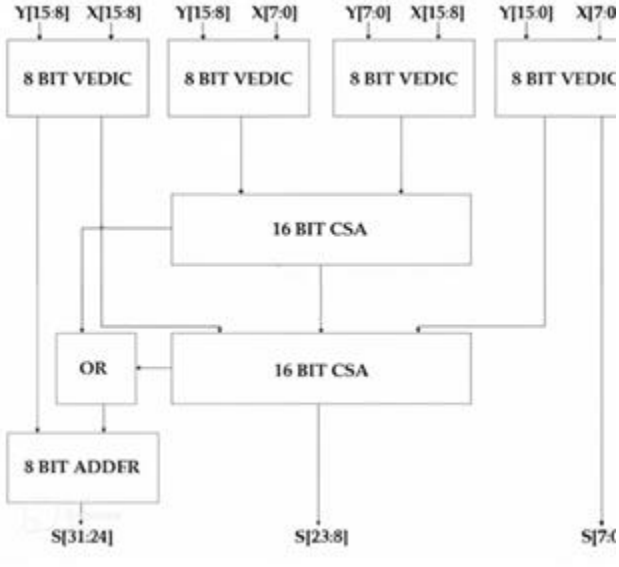

Fig. 3. 16-Bit Vedic Multiplier Schematic

The addition operation is the building block of multiplication and is implemented here using adders. Half-adders and full-adders are used to generate the sum of two-bit and three-bit inputs, with the carry involved, respectively. A Ripple Carry Adder (RCA) is a simple and easy-to-operate implementation of an adder, but it is very slow due to its step-wise process, which makes it reliant on the results of the last adder. As a result, there is a large delay caused by the cumulative effect. Using N-adders in an RCA configuration would result in a delay that is N-times the delay of a single adder, which is inefficient because the delay scales linearly with the input, N. An additional adder that may be used to reduce the delay of the preceding stage's carry generation is the Carry Lookahead Adder, which operates on the propagation and generation principle and uses basic mathematics to get the result. The idea behind it is to compute the carry bits before the total, which cuts down on calculation latency. The Carry Save Adder (CSA) is another common adder for binary multiplications; however, it differs from the others in that it produces a partial sum and a carry-out as its outputs. The outcome of adding the binary inputs is given by combining these outcomes in a systematic way in future phases.

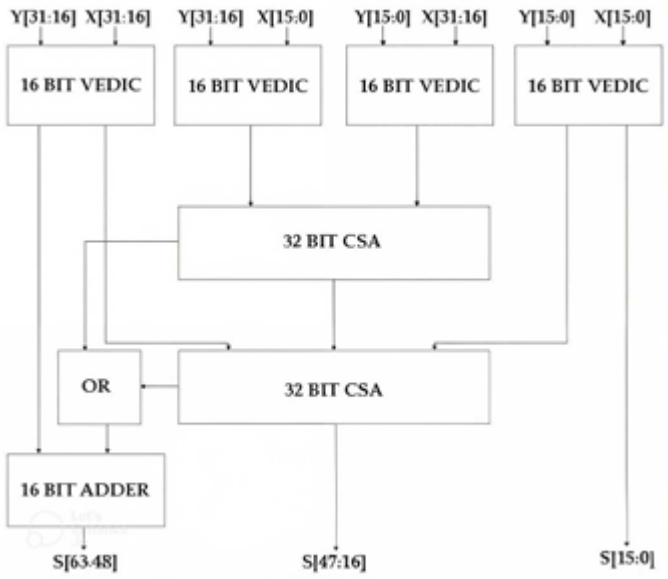

Fig. 4. 32-Bit Vedic Multiplier schematic

Although it is the most complicated adder and requires two stages of addition, the CSA adder's characteristics make it ideal for tasks that need quick computations, such as in a binary multiplier, and its latency is proportional to the number of bits [12]. The Vedic multiplier is built up of smaller modules that draw on past multiplication methods and provide a way to transfer them to current designs. Figure 2 shows the implementation of a 2-bit multiplier. To get the 4-bit output result, out, the bits of two 2-bit values are sent via AND gates and half adders. A1B0 plus A0B1 equals Out1 plus Carry-bit, A1B1 plus Carry-bit equals Out2 plus Out3, the result of adding the two variables. Then, as seen in Figure (x), they are

mixed to create a 4-bit Vedic multiplier. One may get a 4-bit Vedic multiplier by combining four 2-bit multipliers. To get the 8-bit output from the multiplication of the two 4-bit inputs, the two 4-bit variables a and b are sent into 2-bit multipliers, and the resulting values are passed into consecutive adders (CSA and RCA) [12]. The power consumption, delay latency, and eventual net delay have all been reduced by the implementation of various design improvements. After two 8-bit values are multiplied using four 4-bit Vedic multipliers, the resulting integers are added in a further adder stage, which then sends them on to the final adder for evaluation of the calculated product [13]. In order to determine the outcome of multiplying two 32-bit values, comparable techniques of combining 4 × 2n − 1 multipliers and adding their outputs in an adder stage are used. One example of a 32-bit Vedic Multiplier is seen in Fig.4.

$$A_0B_0 = Out_0$$

$$A_1B_0 + A_0B_1 = Out_1 + Carry - bit$$

$$A_1B_1 + Carry - bit = Out_2 + Out_3$$

IV. SIMULATION FRAMEWORK

Three separate implementations of CSA, CLA, and an RCA adder are used to test the efficacy of the whole project. In addition to comparing various architectures, the simulation framework is built to evaluate the uniqueness of the proposed 32-bit Vedic multiplier. A Linux computer was used to run the Xilinx ISE 14.7 toolkit, which was accessed via VirtualBox on a Windows 11 PC. A total of 63,400 Slice LUTs are used by the Artix-7 board (xc7a100t2csg324), an FPGA device, in the simulations. By adhering to a standardized approach, the simulation process guarantees that performance comparisons are accurate. The 32-bit Vedic Multiplier, the highest level, was developed and synthesized using the ISE Navigator tools. To verify the design against theoretical predictions, an RTL schematic was constructed. When it comes to efficiency, speed, and power consumption, as well as space usage, this suggested work is well-rounded and optimized. In contrast to traditional multipliers, our method provides an ideal trade-off without sacrificing any metrics. The code was compiled, the VHDL was created and executed, and the outcomes were retrieved from log files, evaluated by synthesis, and mapped to FPGAs.

The results show that all designs have about the same power consumption, however the suggested CLA-based Vedic multiplier has the lowest computational latency, making it ideal for fast applications. However, RCA-based designs are great for implementations with limited resources because, while being slower, they provide higher space efficiency. Our work emphasizes the benefits of combining Vedic mathematics with contemporary digital design by methodically assessing these trade-offs, proving its viability for HPC applications.

V. RESULTS AND DISCUSSION

The data shown in Figure 5 is the outcome of the simulation and its subsequent compilation. Based on the final findings, we can say that the 32-bit CLA-based Vedic multiplier is the quickest with a latency of 26.466 ns, that all of them used almost the same amount of power, and that the RCA-based multipliers were the most efficient in terms of area usage.

TABLE I. DESIGN COMPARISON BETWEEN CSA, CLA AND RCA 32-BIT VEDIC MULTIPLIER

DESIGN	POWER	AREA (LUTs)	DELAY
CSA	0.082W	3063	32.317ns
CLA	0.082W	2989	26.466ns
RCA	0.082W	2016	32.438ns

There are three parts to the total amount of power that is used: dynamic, switching, and static. While input changes are the only source of dynamic and switching power consumption, leakage is the culprit behind steady-state power

usage [14]. The continual charging and discharging of the transistor-level capacitors is the source of the power usage. What controls how often a node switches for a certain circuit is a factor of α called the activity factor, which in turn affects the dynamic and switching power. Because it determines the likelihood, the value of α, which ranges from 0 to 1, is obtained when simulations are conducted. This probability is used to identify node switching. There is a quadratic link between the dynamic power consumption and the transistors' VDD, which is another major element affecting the power [15].

If you think of transistors as pull-up and pull-down resistors in an RC model with capacitors, you can readily understand the delay of transistors; this model has its own rise and fall period. As per the Elmore-delay model, the values of the resistors and capacitors determine the rise and fall delay. As the width of a transistor rises, the resistance lowers, leading to an increased delay time. This is because the delay is directly proportional to the resistance, which is inversely proportional to the width of the transistor [16]. An FPGA uses what are called Lookup Tables, or LUTs. For programming FPGAs, these are the bare minimum requirements. For various input combinations, it keeps distinct values of outputs in a truth table, which is how it functions. The size of a LUT is determined by the amount of inputs that are available for it. The typical architecture of a LUT with n inputs consists of 2n single-bit memory cells, a 2n:1 multiplexer, or something similar (for example, two 2n- 1:1 muxes followed by one 2:1 mux).

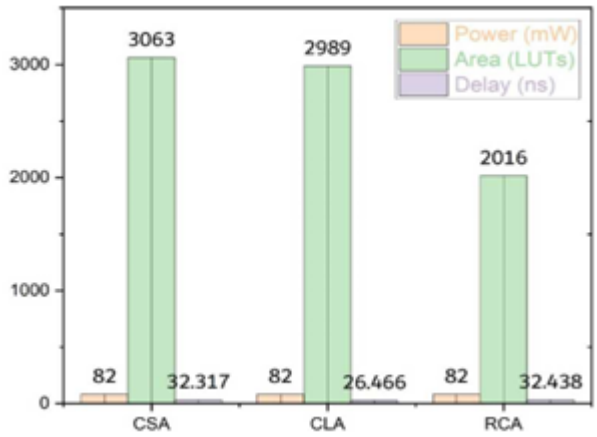

Fig. 5. Design Comparison Graph

Based on these numbers, we can say that power efficiency isn't a deciding factor when choosing the design that maximizes efficiency among the three options as they all used about the same amount of power (0.082W). Among the three designs depicted, CSA employs the most LUTs, which indicates that it consumes a substantial amount of room. CLA, which uses a considerable amount of acreage, follows CSA. The rapid performance it produces is sufficient justification for this. When compared to the other methods for the 32-bit Vedic multiplier, the RCA design uses the fewest LUTs and takes up the least amount of space because it takes a relatively simple approach to adding two numbers using full adders cascaded together. By reducing the total time of the delay—which often causes a delay in simple cascaded designs—the CLA-adder—which operates on the concept of producing and propagating—displays the least delay. Saving a lot of time while simulating the Vedic multipliers is possible by computing the carry. In this case, RCA works very slowly and causes a significant increase in delay timing, mostly because of the time it takes to generate a carry and then use it as an input to the following stage. Based on our findings, a CLA design is optimal for reducing power delay; nevertheless, an RCA design is required for optimizing area, which requires sacrificing delay.

VI. CONCLUSION

We have examined three 32-bit Vedic multiplier designs—the Carry Save Adder, the Carry Lookahead Adder, and the Ripple Carry Adder—that use various adder architectures in this study. Although there were notable variations in speed and area usage, the data demonstrate that power consumption remained constant across all configurations. Thanks to its effective carry propagation mechanism, the CLA-based multiplier was the quickest, achieving a delay of 26.466 ns. The RCA-based device, on the other hand, had the greatest delay while having the smallest area footprint because to its simplicity; this shows the trade-off between area and speed. Finally, certain application needs determine the best Vedic multiplier architecture to use. The CLA-based architecture is superior for processes that need a high rate of speed. Despite its slowness, the RCA-based architecture is an alternative to consider if area optimization is critical. By outlining the benefits and drawbacks of each possible Vedic multiplier architecture, this study helps readers choose the best one.

REFERENCES

1. D. Basha et al., "Rca-csa adder based vedic multiplier," International Journal of Applied Engineering Research, vol. 12, pp. 7603–7613, 01 2017.
2. V. Sumit and D. Deepak, "Delay-power performance comparison of multipliers in vlsi circuit design," International journal of Computer Networks Communications, vol. 2, 07 2010.
3. S. Ms.Dharani et al., "Design and analysis of high-speed low-power vedic multiplier with 3-1-1-2 compressor using reversible logic gates," IOP Conference Series: Materials Science and Engineering, vol. 1059, p. 012024, 02 2021.
4. K. Liew and H. H. Lo, "Performance comparison review of 32-bit multiplier designs," in Intelligent and Advanced Systems (ICIAS), vol. 2, 06 2012, pp. 836–841.
5. P. Saha et al., "Asic design of a high speed low power circuit for factorial calculation using ancient vedic mathematics," Microelectronics Journal, vol. 42, pp. 1343–1352, 12 2011.
6. M. Zangeneh and A. Joshi, "Designing tunable subthreshold logic circuits using adaptive feedback equalization," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, pp. 1–1, 04 2015.
7. K. Kumar, A. Tyagi, Y. R. Lata, V. Kumar, A. Kumar, and A. Jain, "Performance Enhancement of 6T And 9T SRAM Using 90 nm Technology," 2024 2nd World Conf. Commun. Comput. WCONF 2024, 2024, doi: 10.1109/WCONF61366.2024.10692130.
8. K. N., "High speed area efficient 32 bit wallace tree multiplier," International Journal of Computer Applications, vol. 124, pp. 25–28, 08 2015.
9. S. S. Anu Thomas, Ashly Jacob and S. Sudhakaran, "Comparison of vedic multiplier with conventional array and wallace tree multiplier," International Journal of VLSI System Design and Communication Sys- tems, vol. 04, pp. 244–248, 04 2016.
10. R. Hussin et al., "Improved booth encoding for reduced area multiplier," in 2006 IEEE International Conference on Semiconductor Electronics, 2006, pp. 773–775.
11. A. M. Ghorpade and A. M. Muchandi, "Multiplier design using carry save adder," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 05, 04 2016.
12. J. Kuppili, M. Abhiram, and N. A. Manga, "Design of vedic math- ematics based 16 bit mac unit for power and delay optimization," in 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), 2021, pp. 1–4.
13. M. B. Murugesh et al., "Modified high speed 32-bit vedic multiplier design and implementation," in 2020 International Conference on Elec- tronics and Sustainable Communication Systems (ICESC), 2020, pp. 929–932. 12T
14. P. Gupta, N. Singh, V. Kumar, and K. Kumar, "Implementation of Dual Node SRAM Unit

10.1109/IC2E362166.2024.10827406. Cell," 2024. doi:

15. J. Zidar et al., "Dynamic voltage and frequency scaling as a method for reducing energy consumption in ultra-low-power embedded systems," Electronics, vol. 13, p. 826, 02 2024.
16. H. Bhardwaj, S. Jain, and H. Sohal, "An innovative interconnect structure with improved elmore delay estimation model for deep submicron technology," Analog Integrated Circuits and Signal Processing, vol. 111, pp. 1–21, 06 2022.