An Advanced VLSI Architecture For A Three-Operand Binary Adder For Both High-Speed And Area-Efficient

SINGAM SAI KIRAN
M.Tech student ECE , VLSI system design, Avanthi institute of Engg & Tech.
Dr.S. Kishore Reddy
Associate professor, HOD ECE, VLSI design Avanthi institute of Engg & Tech.
E mail : kishorereddy416@gmail.com
P.V.RAJU
Assistant professor, ECE Department, Avanthi institute of Engg & Tech.
E Mali: vijnatharaju@gmail.com

ABSTRACT

The utilization of a three-operand binary adder is employed for executing modular arithmetic in various cryptography and pseudorandom bit generator (PRBG) methodologies. The carry save adder (CS3A) is widely employed as the primary approach for performing three-operandaddition. The ripple-carrying stage (n) of the CS3A introduces a propagation delay of O. The critical route time can be reduced by employing a parallel prefix two-operand adder, such as the Han-Carlson (HCA) method, for three-operand addition. The process of three-operand binary addition is executed by employing an innovative adder design that is both high-speed and area-efficient. This design incorporates pre-compute bitwise addition, followed by carry prefix calculation logic, resulting in a significant reduction in adder latency to O(log2 n). The proposed architecture has been subjected to functional validation on an FPGA device and has been generated utilising a 32nm CMOS technology library that is readily available. In the case of 32-, 64-, and 128-bit architectures, the recommended adder exhibited a performanceimprovement of 3.12, 5.31, and 9.28 times respectively compared to the CS3A after post- synthesis. The HC3B demonstrates superior performance in high-speed data processing due toits reduced spatial requirements, lower power consumption, and diminished delay when compared to the adder. Furthermore, it should be noted that the proposed adder exhibits loweraverage delay power (ADP) and peak delay power (PDP) compared to the existing three- operand adder techniques.

Keywords: ***carry save adder, FPGA, CMOS***

1. Introduction

Hardware multipliers rely heavily on multi-operand adders in order to calculate partial products. The use of multi-operand adders becomes important when performing multiplicationoperations that need the addition of numerous partial products. Due of the high powerconsumption in the multipliers and other arithmetic blocks, low-power electronics with low switching noise are required.

Due to the ability of inspecting operands for addition that might be either single bits or multiplebits, the adder's input and output can be in multiple bits [3]. To more directly depict a multi- operand adder, where partial sums are compressed and carry is less widely propagated, compressor trees may be used. In this work, many different kinds of multi-threaded arithmetic(MTA) are used. These include the Array tree adder, Wallace adder, Balanced delay tree adder,and Overturned-stairs adder. Despite its high power consumption and processing lag, adders are crucial in arithmetic-functioning machines. Adders are also used in multipliers, another component of resource-intensive arithmetic circuits.

Quick adders, sometimes known as other adders, are faster than the ripple carry adder (RCA) in terms of latency. However, they often have a bigger surface area and greater energy needs. The RCA, on the other hand, has a smaller footprint and lower power requirements, but it alsohas a longer latency. It is important to remember, however, that systems with a high number ofoperands often have resource constraints.

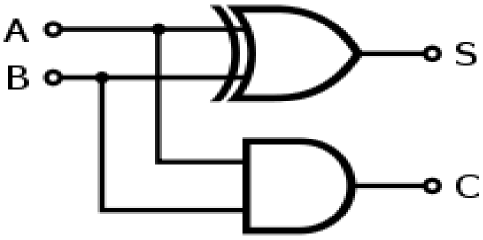
2. Existing Work

No significant effort towards delay refinement of RCA-BTA has been documented in the existing literature. Therefore, the critical route delay analysis is provided [5] to investigate thepotential for delay minimization of the RCA-BTA structure. Most of the suggested work in theliterature is geared at bettering the design metrics of fast adders, which in turn improves the performance of the WTA architecture.

2.1 Adder Types

2.1.1 Half Adder

The half adder is used to perform the addition of A and B. These results are S and C (the valuethat will theoretically be carried over to the next addition), both of which are produced by the method. The sum is 2C plus S at the end. The right half-adder arrangement uses an XOR gateand an AND gate to process S. By connecting the carry outputs of two half adders with an ORgate, a full adder may be created.



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 1: Half Adder

2.1.2 Full Adder

A full-adder may multiply a binary number by 8, 16, 32, or even more digits. The usual notation for the circuit's two-bit sum at the output is Cout or S. The following is the truth tablefor a complete adder with one bit.

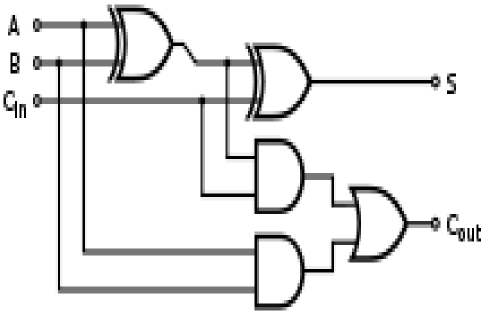


Figure 2 : Implimentation Of full adder

2.1.3 Ripple Carry Adders (Rca)

Ripple carry adder of N bits is created by concatenating N complete adders. The result of one full adder is sent into the input carry of the next full adder in this implementation. The followingequations are used in the computation of sum and carry. By bouncing from one full adder to the next, carry takes the longest critical route and experiences the most severe delay. Si = Ai *Bi * Ci

Ai + Bi + (Ai + Bi) Ci + 1. Where i is an integer between zero and n minus one

Although it takes the longest amount of time to add (O (n) time), RCA takes up the least amountof space (O (n) area). The delay of a ripple carry adder with N full adders chained together is 2N gate delays from Cin to Cout. The adder's latency grows proportionally as the number of bits rises. Figure shows an RCA block diagram.

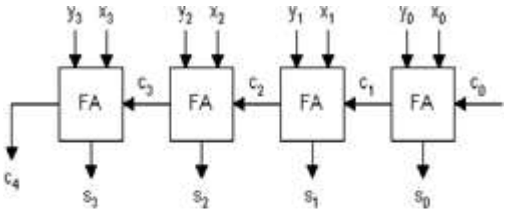


Figure 3: Block diagram of RCA

2.1.4 Carry Save Adder

In computer microarchitecture, a carry-save adder is used to compute the sum of three or morebinary data of n bits.

It's elementary school maths: "8+1=0, carry 1," "7+2+1=0, carry 1," "6+3+1=0, carry 1," and so on. By carrying the carry from one digit to the next during the calculation, we may have a better idea of the initial digit of the result. Adding two numbers of the same length will take time proportional to n if the computer can do several calculations simultaneously. In electrical

terms, using bits (binary digits), a carry must propagate from one end of a number to the other,taking time proportional to n if we have n one-bit adders. We won't rest until this is resolved, Fifth, we have no idea what the ultimate verdict will be.

6) We can't say if the sum is more than or less than a certain value (or even whether it's positiveor negative).A carry look-ahead adder has the potential to decrease the delay. However, the propagation delays increase as n increases because the distances signals must travel on the device expand in proportion to convey look-ahead. This is no longer the true, however, for really large numbers. Carry look ahead is mostly useless for public key cryptography, which needs integersizes between 512 and 2048 bits.

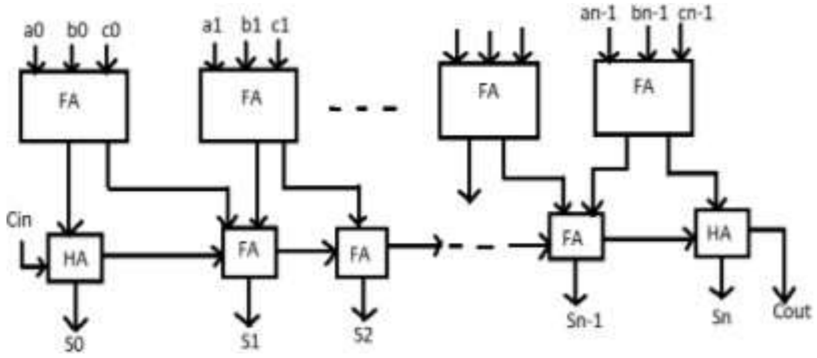


Figure 4: Carry Save Aadder

It is common custom to use either one three-operand adder or two two-operand adders when carrying out a binary addition with three operands. However, it is also possible to utilise a single two-operand adder. [2] The carry-save adder is a tool that is ideal for mathematical computations because of its small size and portability. The three-operand binary addition is a variation of the regular arithmetic that is used in cryptographic algorithms and PRBG approaches. On IoT-based hardware devices, the performance of MDCLCG and other alternative encryption algorithms is negatively impacted because CSA has a longer ripple-carrystage carry propagation delay. It is possible to use the Carry Save Adder to add together valuesof any size, ranging from 3 bits all the way up to n bits. The CSA and the Full Adder are equivalent to one another. Instead of the other adder, the partial product terms for each clusterwill be added using CSA. When compared to the CSA approach, using an alternate adder takesmuch more time. More than two numbers will need to be added together in a lot of different situations. The addition of the first two numbers, each of which has n bits, is the quickest and easiest technique to merge m numbers. And so on, until each of the three is in equilibrium. If

the gate delay is O (m log n), then an adder tree that has O (log m * log n) additions will needto be constructed. Using carry-saving augmentation might potentially lower the amount of timeneeded to complete the job in half.

It is possible to use a parallel prefixed two-operand adder like the Han-Carlson (HCA)[3] in order to cut down on the essential route time required for performing three-operand binary addition. We may be able to reduce the critical route latency by O(log2 n) if we increase the area by O(log2 n), and the same is true in the other direction. Therefore, a potent very large scale integration (VLSI) architecture is necessary in order to do the rapid binary addition whileemploying relatively simple hardware. This study introduces a novel high-speed area-efficientadder approach that employs pre-computed bitwise addition followed by carry-prefix calculation logic to accomplish the three operand addition. In contrast, the HCA-based three operand adder (HC3A) consumes less gate area and minimises propagation latency. This technique was developed to conduct the addition of three operands. The suggested Verilog HDL adder design is synthesised using a 32nm CMOS technology library that is available forpurchase commercially. The one-of-a-kind adder technique is put to the test, after which it is contrasted with the tried-and-true CS3A and HC3A approaches to adding three operands. In order to shorten the time required for the crucial route, it is possible to make use of a parallel prefix two-operand adder that has two stages. According to the findings of the study, the methods that are the most efficient for adding two operands together are logarithmic prefix adders and parallel prefix adders. There are a total of six unique topologies for these adder techniques, and they are referred to as the Brent-Kung, Sklansky, Knowles, Ladner-Fischer, Stone (KS), and HanCarlson (HC) topologies, respectively. When n is more than 16, Hanlon and Carlson emerge as the most efficient of the three methods.

Using the Han-Carlson adder, it is possible to carry out a three-operand addition in two separatestages, as seen in Figure. (HCA).

The blueprint for the three-operand version of the HC3A (HC3A) is quite detailed. The numberof black-grey cell stages that represent the length of the PG logic chain that the Han-Carlson adder makes use of.

3. Proposed Method

Integrated circuits are in danger of losing their field dependability as a result of increases in integration density and process shrinkage. It is essential for mission-critical systems such as server-class computers and embedded devices to have error detection that runsin parallel. Due to the fact that adding is the initial step in any computation, many adders finda home in extremely large size integrated circuits.

Any erroneous output may be recognisedby a comparison of two internal values included inside the carry select adders. It has been proposed in the past that concurrent error detecting adders may be used; these adders are able to recognise wrong outcomes from a single mistake. On the other hand, it is not always easy to identify a flawed product that is the consequence of a confluence of problems. If a second faultoccurs before the first fault is discovered as a result of the detection of an erroneous output, then the capacity to identify concurrent errors may be compromised or lost entirely. Finding the first flaw before the second one is necessary in order to have error-free adding machines. Because of this, the simplicity with which a failure in a concurrent error detecting adder may be recognised during maintenance, a reboot, or even during operation is of the utmostimportance.

3.1 Multiple Block Selection Adder, Size

In Figure 5, you can see a straightforward structure for an n-bit multi-block carry select adder.Its outputs are the carry output cn and the sum S: [sn1.. s0], while its inputs are the agehend X:[xn1.. x0] and the addend Y: [yn1.. y0]. It is composed of several little b bits. Block 0 is the name given to the first part of the series. Let's assume the kth block, which we'll refer to as blockk, has a bit width of nk. Every nk has the potential to be different from the others. Carryinformation is included in both the input cink and the output coutk of blockk. A row of full adders that executes a ripple carry addition on n bits makes up block0. An incrementer (INC) and a half adder (HA) are included in the construction of each full adder seen in the figure. A binary value consisting of just two bits plus a carry bit is input to an INC, and the resulting sumis output as a binary number consisting of only two bits. Each of the remaining blocks consistsof a row of 2:1 multiplexers (MUXs), a row of HAs, two rows of INCs (INC0 and INC1), anda row of INCs (INC0 and INC1). The gatelevel designs for the HA, INC, and MUX gates maybe seen in Fig. 5(b). INC0 and INC1 in each block, with the exception of block0, calculate two alternative sum results. One of these outcomes is based on the assumption that the carry inputcink is 0, and the other is based on the assumption that cink is 1. The row of MUXs selects the correct one. In this abbreviated form, a signal name that has two subscripts specifies, respectively, the block index and the position inside the block. The xk,j and yk,j representationsof the input bits at position j in blockk are written as (0 j nk). Where l = j + Pk1 m=0 nm, xk,j = xl. The column of INCs to which a signal belongs is indicated by the superscript that is placedafter the name of the signal. It might also be expressed as s 0 k,j (0 j nk), where j represents theresult of running INC0 at position k in blockk. The carry signals for bit j 1 are indicated by thenotation c 0 k,j and c 1 k,j, respectively. These signals are referred to as INC0 k and INC1 k. The ranges [xk,nk1.. xk,0] and [yk,nk1.. yk,0] serve as Blockk's inputs, while the range [sk,nk1.. sk,0] serves as Blockk's output range.

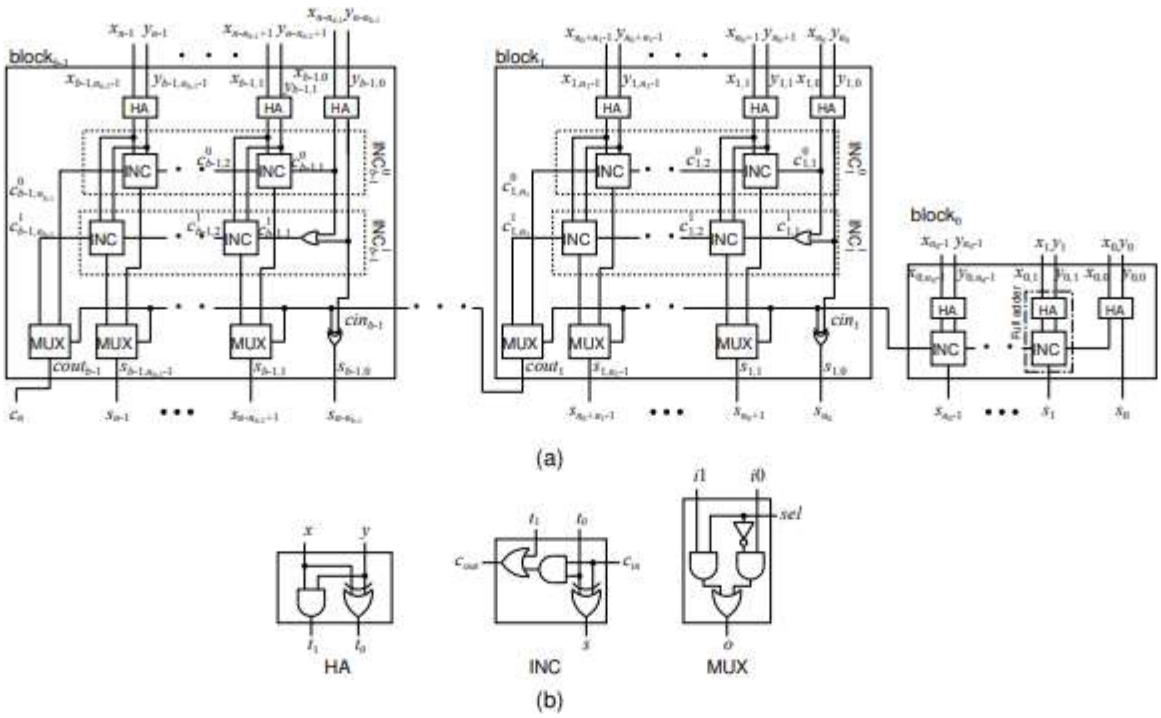


Figure 5: Multi-block carry select adder (a), and the gate-level designs of HA, INC and MUX (b).

3.2 Capability to Identify Problems in Parallel Operations

One stuck-at fault in an INC' or its ascendant HA' never affects both the total output and one of the two carry signals at the same time, as stated in Section 3.1. Since only a single stuck-at fault may exist inside an INC' or its parent HA', this is the situation. This means that (1) the sum signal of the INC', (2) one of the carries of the INC', or (3) all three signals (the two carriesand the total) will be affected by a fault in the INC' or its ascendant HA'. By comparing the expected parity pS with the actual parity S and the values of cn and c ′ n, we may assess whether

or not an error was caused by a fault. The first case involves using one of INC's two carry signals for the addition operation and the other for parity prediction. Errors in the parity prediction carry signal only affect one bit of the parity prediction carry bits, hence the final tally is unaffected. Incorrect results caused by the flaw may then be identified and corrected. Let ck,j represent the carry that is affected by the fault that occurred at an INC' when the carrysignal for addition is erroneous. A defect at an INC' has an effect on the carry, denoted by ck,j.If the problem affecting ck,j does not propagate to other nodes, it will be uncovered when it causes a second error in the total signal, sk,j. The reason for this is because the faulty carry signal is not being used for parity prediction. However, errors will be introduced into the q sumsignals sk,j+1, sk,j+q if the incorrect value of ck,j is passed on at q consecutive carry signals. This happens because ck,j is being incorrectly propagated. So, the wrong answer is the sum signals sk,j, sk,j+1, , sk,j+q. Here, the q carry signals c ′ k,j+1, , c ′ k,j+q are also incorrect because the logic used to generate them is the same as that used to generate the ck,j+1, , ck,j+q, and both use the same carry inputs (ck,j, ck,j+1, , ck,j+q1). Signals for q+1 sum sk,j, sk,j+1, and sk,j+q, as well as q carry c′ k,j+1, and c′ k,j+q, will be off as a result. Therefore, we will use q incorrect signals to determine the expected parity (pX pY) (pCb1 pC0), and q+1 incorrectsignals to determine the overall signal parity. Each of these two parities will be different fromthe other due to the one variation in the total amount of erroneous signals used in each computation, allowing the error to be pinpointed. In the second case, the total result is not the same as the right one since a bit of the sum result was flipped. However, the calculated parity is correct since all of the carry bits used to make the prediction were correct. The impact of a mistake may be determined by comparing the predicted and actual parities of the final outcome.In the third instance, the INC'-derived total bit and all of the carry bits would be incorrect. Theacquired sum bits and carry bits used for parity prediction in the upper locations do not conflictwith one another, with the exception of the position of the flawed INC'. This is because it has been shown that both carry bits are incorrect. Even if the parity prediction carry bits are correct, the total bit from the INC' will be inverted because of the lower bits. Consequently, the expected and actual levels of equality are not the same. The adder has two carry inputs, cink and cin' k, one for each adder block. If the cink is wrong, then the sk,0 will be wrong as well. The error on cink may also lead to problems with subsequent sum outputs (say, q additional sum outputs). It will also lead to errors at the q carry signals c ′ k,i, much as the explanation provided in scenario 1. As a result, there will be q + 1 incorrect sum outputs and q incorrect carry signals c ′ k,i, both of which will be identified for the same reasons as in instance (1). When performing an XOR or MUX' operation, only one of the sum bits, the carry bit, or the projected parity is altered. Therefore, the effect of an error in them may be determined by comparing the projected parity with the parity and by comparing the two carry outputs of the adder. Keep in mind that if one of the input operands does not match its parity input, the projected parity may be off. This occurs because the parity input is used to make parity predictions. As a result, it is possible to find contradiction between X and pX or between Y andpY even if the adder is error-free. This is so because it is possible to verify these discrepanciesseparately. As can be seen in Figure 3, the proposed adder is well-suited for usage in systems that conduct parity-based error detection. Real-world designs like [21] and [22] make use of arithmetic circuits with parity-based error detection. The adder's expected parity is utilised forthe result's parity bit, while the parities of the operands (pX and pY) are read from memory ora register file and used to perform the addition. By monitoring the system's parity checker and comparing the adder's cn and c ′ n carry outputs, one may ascertain whether or not the adder has generated a faulty result.

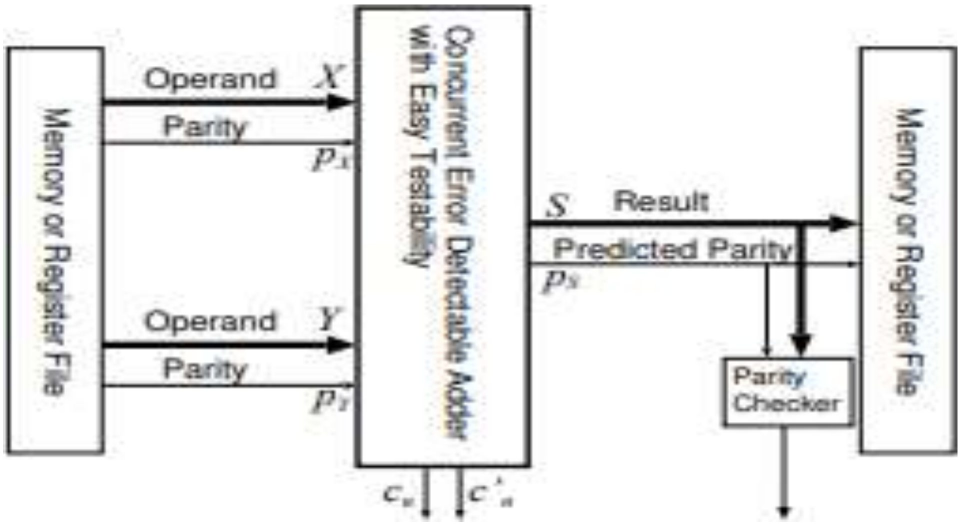
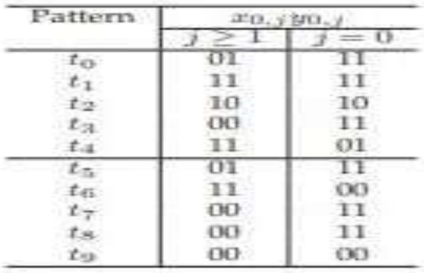


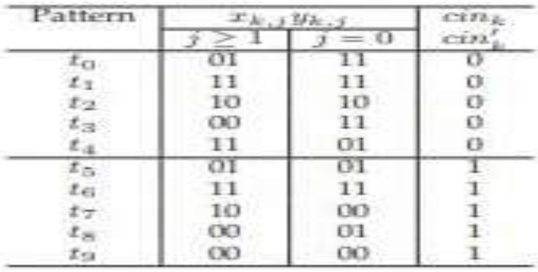
Figure 6: Example of a datapath circuit with the proposed adder in a system using parity- based error detection

TABLE 1
Input test patterns for block_0.



Pattern	$x_{0,j}y_{0,j}$	
	$j \geq 1$	$j = 0$
t_0	01	11
t_1	11	11
t_2	10	10
t_3	00	11
t_4	11	01
t_5	01	11
t_6	11	00
t_7	00	11
t_8	00	11
t_9	00	00

TABLE 2
Input test patterns for block_k $(k \geq 1)$ shown in Fig. 2(b).



Pattern	$x_{k,j}y_{k,j}$		cin_k
	$j \geq 1$	$j = 0$	cin'_k
t_0	01	11	0
t_1	11	11	0
t_2	10	10	0
t_3	00	11	0
t_4	11	01	0
t_5	01	01	1
t_6	11	11	1
t_7	10	00	1
t_8	00	01	1
t_9	00	00	1

4. Hardware Requirements

"Very Large Scale Integration" is the abbreviation for VLSI. This is the area of study concerned with cramming ever more logic devices into ever smaller spaces. With very large scale integration (VLSI), circuits that formerly required many boards are now able to fit into an area only a few millimetres wide. There is now a great chance to accomplish goals that werepreviously impossible. Everyday items like computers, cars, smartphones, and even cutting- edge digital cameras all use VLSI circuits. We'll examine how extensive knowledge in severalsubfields of the same topic is required for all this in the next chapters. While very large scale integration (VLSI) has been around for some time, developments in computing have led to a huge increase in the number of tools available for designing VLSI circuits. Additionally, IC capabilities has risen dramatically over time in terms of computing power, utilisation of available space, and yield in accordance with Moore's law. Thanks to the confluence of these two developments, novel ICs with a wide range of capabilities are now within reach. The proliferation of tiny computer devices, such as those found in embedded systems and ubiquitous computing, means that even the shoes you wear may one day do valuable tasks, such as monitoring your heartbeats.

There are three types:

1. Analog:

2. Amplifiers, Data converters, Filters, Hase Locked, Sensors, etc., that need a high degree ofaccuracy yet have a low transistor count.

2. Application-oriented integrated circuits (ASICS)

Improvements in IC manufacturing have allowed us to pack more and more powerful circuits into ever-tinier packages. Because of this, we can fit a greater amount of functionality into thesame footprint. The primary use of this skill is in the creation of ASICs. These integrated circuits (ICs) are purpose-built, meaning they are designed to do a single task very well. Digitalsignal processing (signal filters, picture compression, etc.) is where you'll most often see this used. To put it simply, a digital wristwatch typically consists of a single IC doing all the time-keeping tasks, in addition to other functions like games, calendars, etc.

3. Systems-on-a-chip (SOC):

These mixed signal circuits (digital and analogue on the same chip) are quite sophisticated. Anexample of a system on a chip is a network processor or a wireless radio chip.

4.1 VLSI's Benefits

While our focus in this book will be on ICs, it is important to note that the capabilities and limitations of ICs heavily influence the design of the larger system. There are several essentialways in which integrated circuits enhance system features. When compared to digital circuits made of discrete components, ICs provide three main benefits:

- Size. In contrast to discrete components, which may be as large as millimetres or centimetresin size, integrated circuits include transistors and wires that are just micrometres in size. Since smaller components have less parasitic resistances, capacitances, and inductances, they are faster and use less power than their larger counterparts.

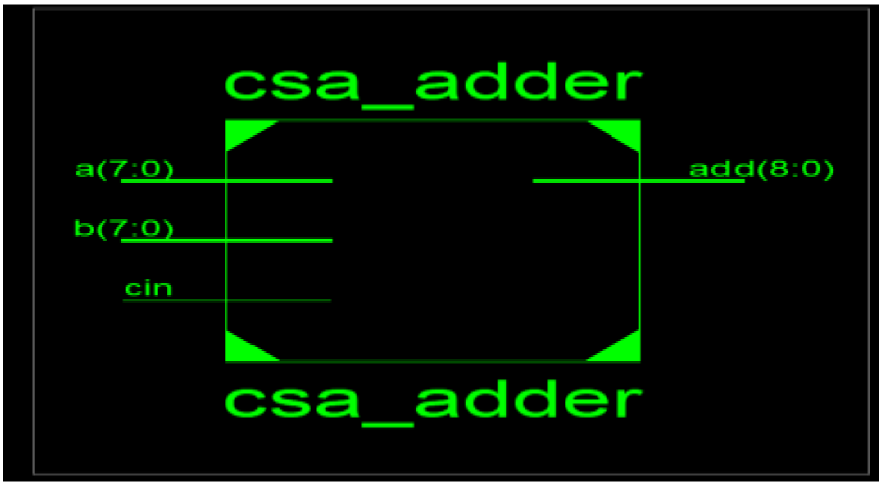
- Speed. Within a chip, it is significantly faster to flip a signal from logic 0 to logic 1 than it isto do so across chips. Internal chip communication may be several orders of magnitude quicker

than inter-chip communication on a PCB. Smaller components and connections mean less parasitic capacitance to slow down the signal, which is why circuits on a chip can operate at such fast speeds.

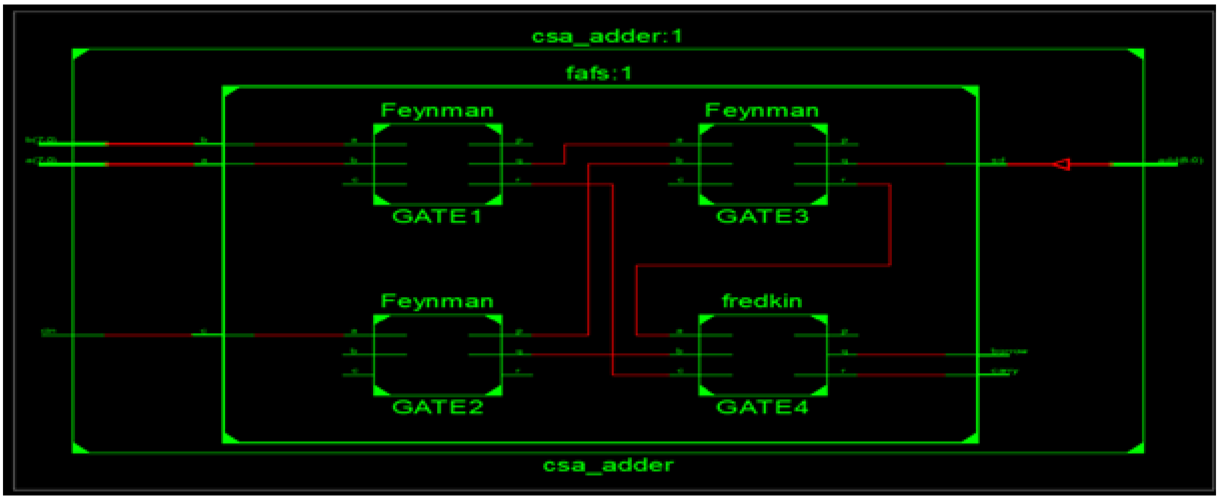
The use of energy. Within a semiconductor, logic activities use far less energy. Again, smallercircuit sizes on the chip are mostly responsible for the reduced power consumption; circuits with smaller sizes have fewer parasitic capacitances and resistances, respectively.

5. Simulation Result

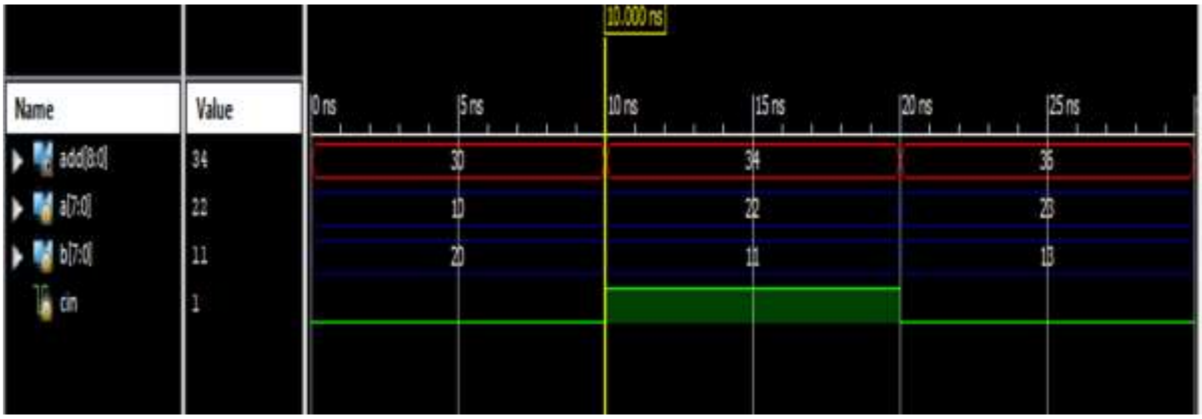
RTL:



Internal Block Diagram



Simulation Results:



Conclusion

This research presents a high-speed, area-efficient adder strategy and related VLSI architecture for computing modular arithmetic for use in cryptography and PRBG applications. In this method, the addition of three input operands is computed concurrently using a four-stage prefix adder. The suggested architecture is smaller overall due to the elimination of unnecessary prefix calculation steps in both PG logic and bit-addition logic. The idea of the hybrid Han-Carlson two-operand adder (HHC2A) is expanded upon in the hybrid Han-Carlson three-operand adder (HHC3A) design. Similar to the suggested adder design, the Kogge Stone hybrid Han-Carlson three-operand adder is written in Verilog HDL. All of these devices' core area and delay timing are synthesised using the Spartan 3 technology library. For better area and latency performance, we implemented our adder in the FIR filter's adder section. By considering both area and delay, we determined that our adder was the best option.

References

[1] M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal, and Y. M. Jang, " FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime fifield," *IEEE Access*, vol. 7, pp. 178811–178826, 2019.

[2]Z. Liu, J. GroBschadl, Z. Hu, K. Jarvinen, H. Wang, and I. Verbauwhede, "El liptic curve cryptography with effificiently computable endomorphisms and its hardw are implementations for the Internet of Things," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 773–785, May 2017.

[3] Z. Liu, D. Liu, and X. Zou, "An effificient and flflexible hardware implementation of the dual-fifield elliptic curve cryptographic processor," *IEEE Trans. Ind. Electron.*, vol. 64,no. 3, pp. 2353–2362, Mar. 2017.

[4] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Design*. New York,NY, USA: Oxford Univ. Press, 2000.

[5] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, no. 170, pp. 519–521, Apr. 1985.

[6] S.-R. Kuang, K.-Y. Wu, and R.-Y. Lu, "Low-cost high-performance VLSI architecture for montgomery modular multiplication," *IEEE Trans.Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 2, pp. 434–443, Feb. 2016.

[7] S.-R. Kuang, J.-P. Wang, K.-C. Chang, and H.-W. Hsu, "Energy-effificient high-throughput montgomery modular multipliers for RSA cryptosystems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 11, pp. 1999–2009, Nov. 2013.