E-ISSN 2581 –7957
P-ISSN 2277 – 3916
CVR Journal of Science and Technology, Volume 21, December 2021
DOI: 10.32377/cvrjst2109

Speculative Carry Addition Performance Improvement and Area Optimization using Modified Carry Generators

T. Subha Sri Lakshmi
[1]Asst. Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: rupashubha@gmail.com

Abstract: **Carry Speculative Adder (CSPA) is a technique for reducing the critical path delay of an arithmetic circuit that is designed using the carry speculation technique, with an assumption that the carry-out bit generated at a given position of input bits is only a dependent on the previous 'x' bits but not the LSB bit. Carry predictor, internal carry generator, sum generator blocks, and multiplier blocks are used in this architecture. The n-bit CSPA is subdivided into smaller block adders, each of which runs in parallel. A carry predictor circuit serves as a selection line for the multiplexer, with its inputs coupled to the bit pattern generated by internal carry generators, one of which is connected to binary one (vdd) and the other to binary zero (gnd). Each block adder is y bits in size, and the CSPA has 'm' independent block adders and '(m-1)' carry predictor circuits, where p = (n/y). With Carry Speculative Addition, modified carry generators are employed to minimize critical path delays and the consequent need for area and power. Data latching circuits are adjusted to provide continuous data to the circuit. In order to obtain accurate results, the CSPA with Modified Carry Generators (CSPA-M) is built using error detection and error recovery circuitry (VLCSPA-M). The proposed Carry Speculative Addition with Modified Carry Generator architecture is developed using Verilog HDL in both FPGA and ASIC platforms, and the design is simulated using Xilinx ISE and Cadence 45 nm Technology libraries.**

Index Terms: **Carry Speculative Adder (CSPA), Modified Carry Generator (CSPA-M), Verilog - HDL, FPGA – Xilinx ISE, ASIC – Cadence.**

I. Introduction

An adder is a digital circuit that adds two numbers together. Adders are utilized in the Arithmetic Logic Units (ALU) of many computers and other types of processors. They are also used in other areas of the processor to calculate things like addresses, table indexes, increment and decrement operators, and other things like that. Although adders can be built for a variety of number representations, such as binary-coded decimal or excess-3, binary numbers are the most frequent, it is simple to convert an adder into an adder–subtract or in circumstances where two's complement or ones' complement are used to represent negative values. Other signed number representations necessitate additional reasoning in addition to the simple adder. Multiple bits can be supported by the adders. Half adder and Full adder are two fundamental adders that can add two or three bits. Ripple Carry Adder (RCA), Carry Look Ahead Adder (CLAA) [1], Carry Save Adder (CSAA), Carry Skip Adder (CSKA), Carry Increment Adder (CINA), Carry Select Adder (CSIA), and Carry Bypass Adder are the adders that handle multiple bits (CBYA).

The speculative methodology is an optimization methodology for improving the delay of arithmetic circuits based on a prediction mechanism. Speculative adders are based on the observation that in classic adders, the critical path is rarely activated [2]. Each output of a speculative adder is dependent only on the previous k bits, rather than all previous bits. A new function speculation methodology for designing variable latency adders with low area overhead and excellent performance is Static Window Addition (SWA). It is a series of consecutive bits that is referred to as a window. Putting input bits into blocks can approximate the carry chain length to a high degree of certainty. In spite of varying adder widths, Variable Latency addition utilizing SWA [3] based speculative adders can be 10 times faster than the quickest Design Ware adder. The Correlation Aware Speculative Addition (CASA) [4] is a lightweight improvement to traditional speculative adders that takes advantage of the correlation between the most significant bit in the input operand and the carry-in value in order to improve accuracy. It demonstrates that the CASA produces a large reduction in mistake rate with minimal timing and area overhead.

II. Carry Speculative Addition

A. Carry Speculative Adder

Speculative Carry Select Addition (SCSA) [5] is presented in [3] as an efficient method for building low error rate speculative adders while maintaining low area overhead, high performance, and reliability. Signals in a chain can be divided into equal blocks by dividing each block into equal spaces. Addends' input bits are divided into blocks, and the carry bits between blocks are deliberately truncated to zero. Theorizing all the outputs of the block together reduces the overhead problem due to the separation of the outputs. The SCSA-based technique reduces space requirements by 43% while improving performance by 10% [6]. Offers a carry speculative adder in which a standard full adder is split into two distinct carry generators and sum generators, each with one additional logic gate, increasing speed while lowering power consumption. Comparing the carry speculative adder to a carry selection adder, the carry speculative adder reduces simulation latency by 25.69 percent, area by 14.62 percent and power consumption by 18.03 percent. In order to reduce critical route delays, carry

Received on 24-06-2021, Revised on 03-07-2021, Accepted on 26-08-2021.

speculation is used in the Carry Speculative Adder (CSPA). Figure 1 shows the block diagram of the carry speculative adder (CSPA). It consists of three n-bit Carry Predictor Circuits that work independently, each with a tiny block adder. Each block adder with the exception of the leftmost one, has a size of y bits. In a CSPA, where p = (n/y), there are “p” independent block adders and “(p1)” carry predictor circuits.

This technique is based on predicting circuit delay in order to measure the effectiveness of the arithmetic circuits. According to speculative adders, a few critical paths are engaged in traditional adders.

Rather than including all previous bits in its outputs, a speculative adder only counts the last l bits as inputs. The Static Window Addition (SWA) method is proposed in [8] as an innovative function speculation technique for creating highly effective variable latency adders with minimal area overhead. Input bits contained within a window are arranged in a specific order. Putting input bits into blocks can approximate the carry chain length to a high degree of certainty. Using SWA-based speculative adders, Variable Latency addition employing adders ranging anywhere from 5 to 40% smaller is 10% faster than the quickest Design Ware addition. A correlation aware speculative addition [4] is made in [9], which intelligently uses the correlation between the significant bits of the input operands and the carry-in values to improve the accuracy of [9] speculative adders. Specifically, it shows that CSPA significantly reduces error rates and reduces overhead while reducing the time and area spent on errors.

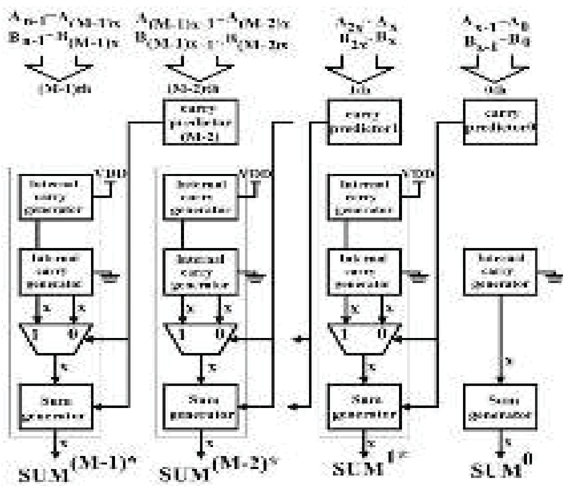


Figure 1. Block Diagram of Carry Speculative Adder

B. Carry Generator

As illustrated in Figure 2, the carry bit is created after three gate delays in a traditional full adder. Figure 2 illustrates how the carry and sum generators are separated in the CSPA's block adder, using a Modified Full Adder (MFA). A further logic gate is added to more efficient Transmission Full Adder (TFA), which produced an extra gate delay and increased power consumption. With two carry generators and a sum generator, CSPA [1] is implemented in block adders to save electricity.

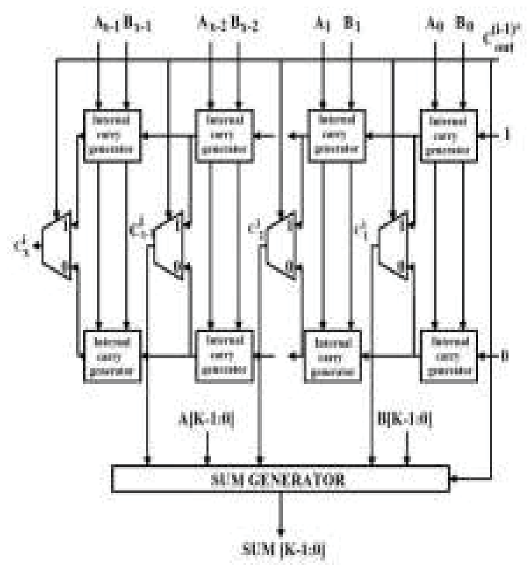


Figure 2. Structure of Block Adder

C. Modified Carry Generators

As for CSPA [6], it utilizes a modified full adder to split the Sum and Carry generators, which increases power consumption and area. To save space, the carry generator has been replaced with two distinct carry generators for Carryin=1 and Carryin=0, respectively. The structure of carry generator is shown in Figures 3(a) and 3(b). Ripple carry circuits have carry-outs of full adders that are as the incoming carry-in of the next most significant full adder. In spite of the fact that ripple carry occurs in each stage, it is called a ripple carry circuit. It is only during the subsequent carry-in of a ripple carry adder that bits for the sum and carry out of any half adder stage are valid. The logic circuitry should have propagated this delay through its logic circuits.

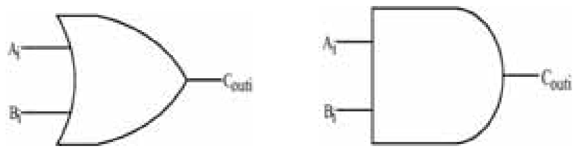


Figure 3(a) and 3(b). Structure of Modified carry Generator for Cin = 1 and 0.

Carryin = 0 and the logic gate AND is used as a replacement for the four-gated 1bit carry generator. A logic gate "OR" is used when Carryin is 1, as well as the four-gated 1-bit carry generator. There are two types of modified carry generators instead of two carry generators, so the block adders take up less space with one gate delay. To segregate the carry and sum generators in the CSPA's block adder, a Modified Full Adder (MFA) is used. Due to two gate delays and increased power consumption, MFA has an additional logic gate than TFA to lessen the carry bit delay. A CSPA chip consists of a multi-function adder with a sum generator and two carry generators to save power. As soon as an input pattern is received, the carry generators and carry estimator circuits are simultaneously active. Similarly, the internal carry generator generates equivalent internal carry signals when signals "1" and "0" are input. Carry predictors enable block adders to generate their expected carryout bits as shown in Figure 4.

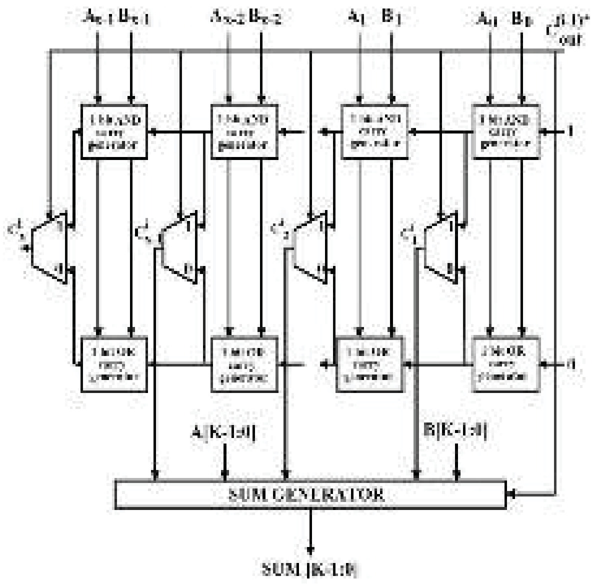


Figure 4. Block Diagram of Carry Speculative Adder

III. Variable Latency Circuit

A. Carry Speculative Adder

When the critical path delay is employed as the execution period, a variable latency design can reduce waste timing of the circuit. Two clock cycles are employed in variable latency design. The CSPA with error detection and recovery circuits is shown in Figure 5.

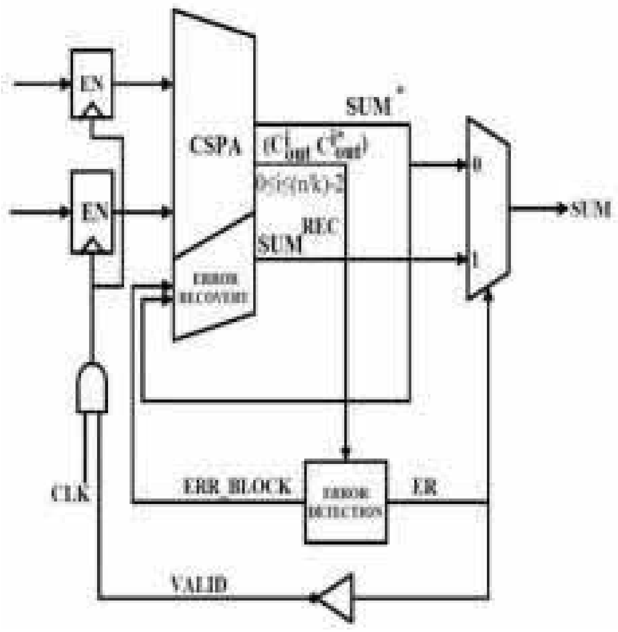


Figure 5. Variable Latency Carry Speculative Adder Circuit

The internal carry generators and carry estimator circuits function in parallel when an input pattern is received. With regards to carry in signals "1" and "0," the internal carry generator generates equivalent internal carry signals. The block adders expected carryout bits are generated by the carry predictor. Error_ block signals and error signals are generated by the error detection circuit. The error detection circuit generates Error block [4, 6] impulses, which specify which block circuit gave erroneous and precise measurements. The error signal indicates whether or not an error has occurred. When the error signal ER is set to “0” and the VALID signal is set to “1,” the findings are correct. The results of the CSPAs are then verified as correct and used as the output. When the ER signal is “1” and the VALID signal is “0,” the results computed by the CSPA are incorrect, and the correct results are recovered from the Error Recovery Circuit in the second cycle, and the result is presented as SUMREC (i.e. Recovered sum). The input registers are blocked if an error occurs, and no fresh input is stored into the device. Because the CSPA's error rate is low, the VLCSPA's average latency is more similar than that of the CSPA.

B. Carry Speculative Adder with Modified Carry Generator

Carry Speculative Adder, error - detecting circuit, error recovery circuit, and data latched circuit, and cascaded multiplexer is used in configurable latency carry speculative adder, as shown in Figure 6.

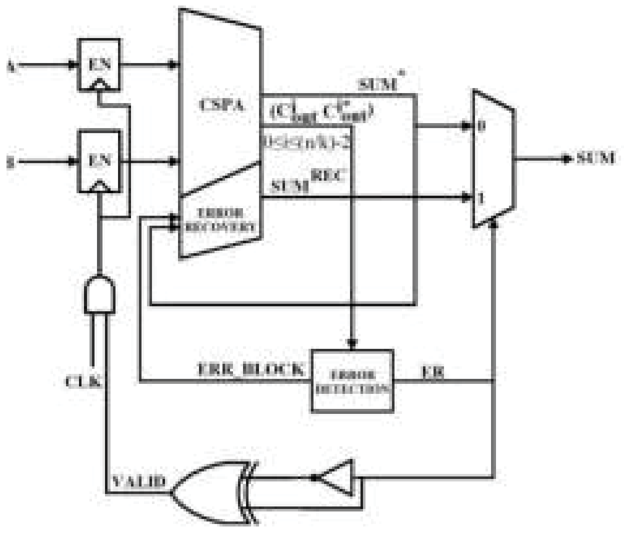


Figure 6. Variable Latency Modified Carry Speculative Adder Circuit

When an input pattern is received, VLCSPA returns the CSPA result in a cycle. Error_ block signals and error signals are generated by the error detection circuit. When an error appears, the error recovery circuit recovers the results based on the Error block. The results from the error detection circuit and CSPA are fed into a multi-bit multiplexer, and the error is utilized as the multiplexer choose signal. Whenever the ER signal is "1," the signal is recovered from the error recovery circuit. When the ER signal is “0,” the CSPAM [5] (i.e. Carry Speculative Adder with Modified Carry Generators) results are selected. The input registers are deactivated when an error occurs, and no new input is latched in the circuit. The not gate in figure 6 is replaced by the XOR gate in this data latching. An exclusive or operation is conducted between the error signal and the counterpart of the error signal in the latching circuit. After the previous data has been recovered, the valid signal enables latching fresh data into input registers.

C. Error Detection Circuit

The addition operation in CSPA is based on guesswork, which might result in accurate or inaccurate results. An error detection circuit is used to determine if the output is accurate or not. The Block diagram of an error detection circuit is shown in Figure 7. The Error Detection Circuit also detects whether or not an error has occurred. The following is how error detection works. When the Ci out and Ci *out signals of each block adder are fed into the error detection circuit, it is checked to see if they are equivalent. This is accomplished by performing an exclusive or (XOR) operation on Ci out and Ci* out, where Ci out is the carryout bit of the ith block adder that is generated by the multiplexor and Ci *out is the projected carryout bit generated by the ith carry prediction circuit. If Ci out and Ci *out are the same, the carry out bit of the ith block adder is accurate. Otherwise, the Ci *out signal projected is not correct. The output of the OR operation on the XOR output of the gates yields the error signal. If the error detection circuit detects an Error, the error signal is "1," else it is "0" [6].

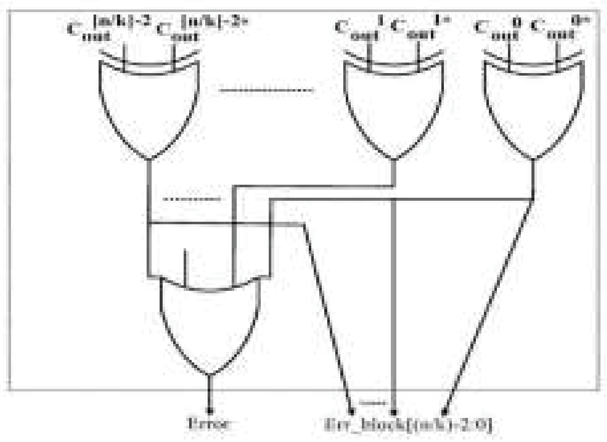


Figure 7. Schematic of Error Detection Circuit

D. Error Recovery Circuit

The Error Recovery Circuit corrects inaccurate partial sum bits [5] of the block adders based on the Error block signal. The Error Recovery Circuit block schematic is illustrated in Figure 8.

IV. Simulation Results

From conception to device programming with the Integrated Software Environment (ISE®) from Xilinx, you can approach your idea from conception to completion. The ISE Project Navigator supervises and processes your design as it proceeds through the ISE design cycle. The design entry phase of the IISEs is its first phase. You build your source files during design entry based on your design goals. Creating a top-level design file can be done using a schematic or a Hardware Description Language (HDL).

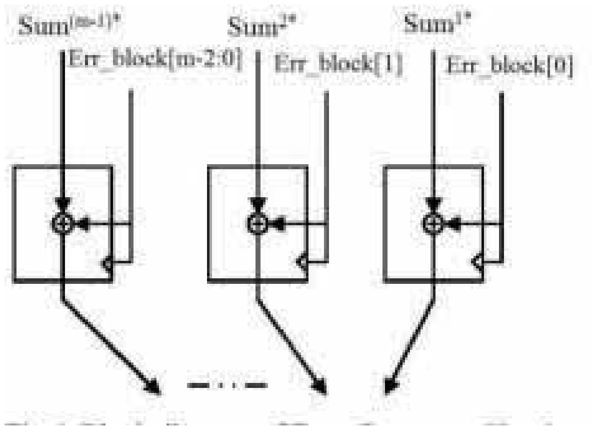


Figure 8. Structure of Error Recovery Circuit

There are a variety of file formats that are used to create the lower-level source files in your design. Design entry is followed by simulation, and then synthesis is conducted. The resulting netlist files are sent to be implemented, after which VHDL, Verilog, or mixed language designs are converted. The implementation of design is the process of converting the logical design into a physical format that a target device can use. In Design, each implementation procedure can be run independently or in one step. Implementing a Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD) has different results (FPGA). Throughout the process of designing, the functioning is continually tested. Simulator The software can be used to test the timeliness of your design and functionality. Simulators translate VHDL and Verilog codes to circuits and display logical results in the simulation to evaluate the function of the circuits. A programming file can be generated by Design and used to configure the device. Configuration of Xilinx devices involves downloads of programming files from a host computer and generation of configuration files from XST Synthesis.

The critical route delay of SCSA is 19 to 40% less than that of the Kogge-Stone adder for an error rate of 0.01 percent as shown in Table I and Table II. The critical route delay of SCSA is equivalent to that of speculative addition in VLSPA for an error rate of 0.01 percent. This suggests that SCSA has a lot of potential. The area of SCSA is 10 to 43 percent lower than Kogge-Stone adder at an error rate of 0.01 percent. The area requirement of the speculative adder in VLSPA is -20 to 8% for an error rate of 0.01 percent. For various bit widths, it can be seen that the area of SCSA is always smaller than that of VLSPA. This is due to the fact that SCSA speculates on the level of the window, whereas VLSPA speculates on the level of the individual bit position. In VLSPA, the critical path delay of a speculative adder is 12 to 27% less than that of a Kogge-Stone adder. However, the error detection block's critical path delay is 4 to 8% longer than the speculative adder's, negating the advantage of speculation. The error recovery block has a delay that is less than half that of the speculative addition and error detection blocks. VLCSPA, on the other hand, has nearly identical critical path delays for error detection and speculative addition, both of which are 14 to 36 percent faster than the Kogge-Stone adder. When speculation is true, the critical path latency of VLCSPA is 6 to 19 percent lower than that of VLSPA. The error recovery block's critical path delay is less than half that of speculative addition and error detection. VLSPA has a 14 to 32 percent bigger surface area than Kogge-Stone adder. This is related to the error detection and recovery blocks' area overhead. VLCSPA, on the other hand, has a -6 to 17% less area requirement than the Kogge Stone Adder. When the bit width is 512, the VLCSPA adder has a 6 percent smaller area than the Kogge-Stone adder. To put it another way, VLCSPA has the potential to be smaller and faster than the Kogge-Stone adder.

As a result of SCSA's lower critical path delays, errors of 0.010 percent and 0.250 percent result in ten percent less delays than those caused by Design Ware adders. SCSA, as width increases, has a 43.0 percent smaller area than Design Ware adder, with an error rate of 0.01 percent. The area of the SCSA is 21.0 to 56.0 percent smaller than the Design Ware adder for an error rate of 0.25 percent. SCSAs with a lower mistake rate have a larger surface area than those with higher error rates. The error rate and the type of error must be balanced as a result. Occasionally, the error rate can increase slightly when an application uses error-tolerant algorithms to minimize the failure area. A tradeoff is created between the error rate and the time needed for the task to be completed.

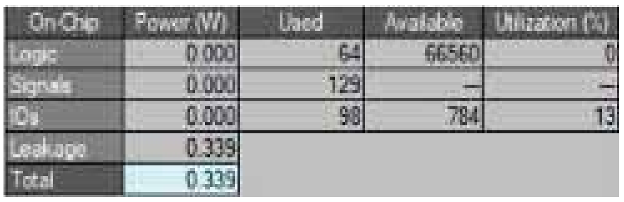
TABLE I.
COMPARISON OF VARIOUS ADDER WIDTHS W.R.T WINDOW SIZE

Adder Width	Window Size ($P_{err} = 0.01\%$)	Window Size ($P_{err} = 0.25\%$)
64	14	10
128	15	11
256	16	12
512	17	13

TABLE II.
COMPARISON OF VARIOUS ADDER INTERMS OF DELAY & POWER

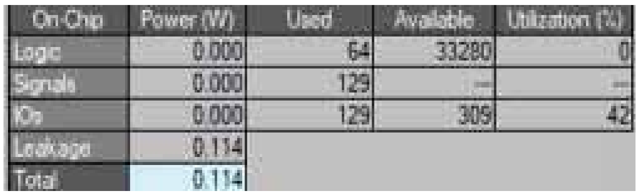
Adder Type	Methodology	Delay	Time Complexity	Power
SCSA (16 bit)	Without Variable Latency Design	19. 279 ns	7.42sec	0.6 5 W
CSPA (16 bit)	With Variable Latency Design	16. 224 ns	6.52 sec	0.3 39 W
Modifie d CSPA (16 bit)	With Variable Latency Design	13. 08 ns	5.66 ns	0.1 14 W

Power consumption of carry speculative adder (CSPA) and modified carry speculative adder using carry generators (CSPAM) is shown in Figure 9 and 10 using Xilinx tool. The maximum combinational path delay of CSPA is 55.680 ns and for CSPAM are 11.367 ns



On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	64	66560	0
Signals	0.000	129	---	---
IOs	0.000	98	784	13
Leakage	0.339			
Total	0.339			

Figure 9. Power Consumption of CSPA



On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	64	33280	0
Signals	0.000	129	---	---
IOs	0.000	129	309	42
Leakage	0.114			
Total	0.114			

Figure 10. Power Consumption of CSPAM

In this paper both the designs i.e., CSPA and CSPAM are simulated and verified using both FPGA and ASC tools. Figure 9 and 10 results are given with the help of Xilinx x-Power Analyzer tool. By observing the power results the CSPAM has less power compared to CSPA due to modified carry generators used in it. Figure 11 shows the RTL schematic of CSPA which is generated with the help of Cadence RTL Compiler tool.

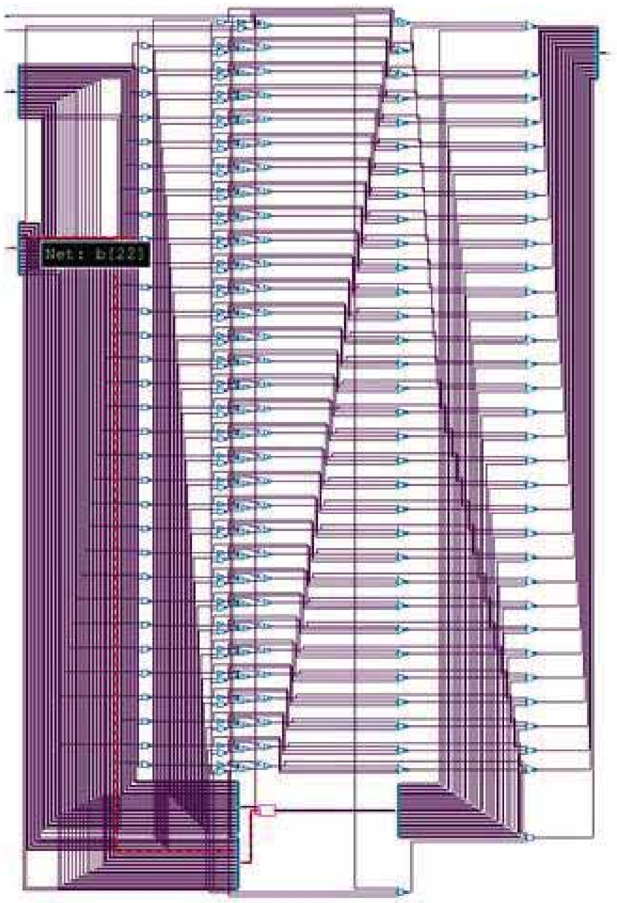
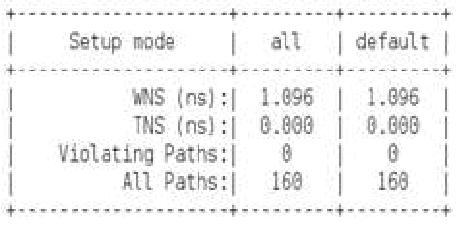


Figure 11. The RTL Schematic of CSPA

Figure 12 give the Pre-synthesis timing report of CSPA in which it consists of 160 paths and the timing is 1.096 ns and there are no violating paths. Pre synthesis and Post Synthesis are performed with the help of Cadence SoC Encounter.



```
+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  1.096  |  1.096  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|   160   |   160   |
+--------------------+---------+---------+
```

Figure 12. The Pre-Synthesis timing report of CSPA

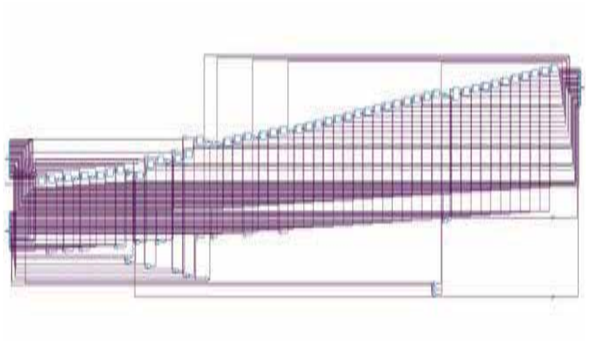


Figure 13. The RTL Schematic of Carry Predictor

Figure 13 shows the RTL schematic of carry predictor circuit which plays the major role in CSPA and CSPAM. Figure 14 shows the final chip layout structure of CSPAM.

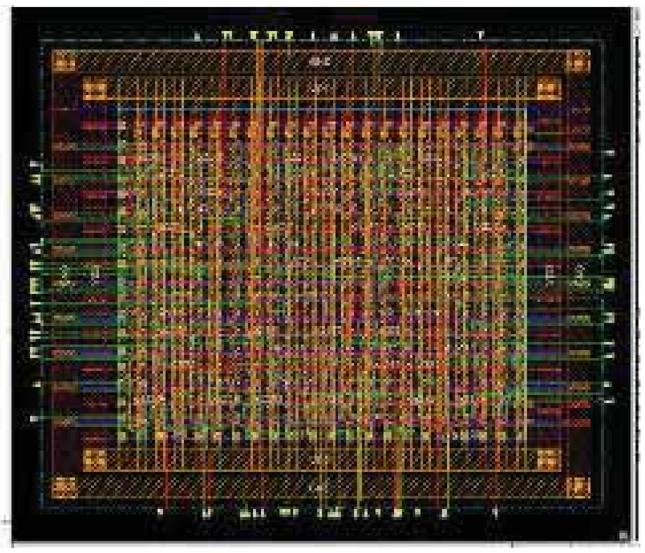


Figure 14. Chip Layout of CSPAM

V. Conclusions

First, the CSPA splits the carry and sum generators, allowing for faster calculation of the carry signal and partial sum bit. Secondly, the block adders suggested to carry predictor circuit only predicts the carryout bit using input bits near the MSB. The prediction circuit's hardware cost is lowered, while the CSPA's error rate increases just a little. Thirdly, an error model was presented to investigate the correlation between the quality of the block adder, the length of the carry predictor circuit, and the CSPA error rate. Fourthly, the EEDP model was presented to calculate the optimal block adder size for segmenting the "n" bit adder for balanced latency, power consumption, and error rate. Fifthly, when errors occur, the proposed error detection circuit identifies which block adder produced an inaccurate carry-out bit, and the error recovery circuit focuses solely on restoring block adders with incorrect partial sum bits. A variable latency adder that adds mistake detection and recovery to the speculative adder. The suggested variable latency adder outperforms the quickest Design Ware adder in both delay and area, according to simulation findings. To fully investigate the design space and its applications, such as non-uniform input distributions, non-uniform input arrival timing profiles, and other arithmetic operations like multiplication and multi-operand addition, more study is required. The ETA (Error Tolerant Adder) series of low-power, high-speed adders are more economical than traditional adders on the market, especially in low-accuracy applications. The possible uses of the ETA (Error Tolerant Adder) are mostly in sectors where accuracy is not a stringent necessity or where super low power consumption and high-speed performance are more important than accuracy. The DSP application for portable devices such as cell phones and laptops is an example of such applications. When compared to the trustworthy adder, the adder provided here is considerably faster. This approach is very beneficial in computationally intensive applications that are resistant to slight computation errors. A mechanism has also been provided for detecting and correcting mistakes. This produces a fast variable latency adder that is 1.5 times faster than a standard fast adder.

References

[1] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, “IMPACT: IMPrecise adders for low-power approximate computing,” in Proc. Int. Symp, Low Power Electron. Design (ISLPED), Aug. 2011, pp. 409–414.

[2] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo, “Enhanced low-power high speed adder for error tolerant application,” in Proc. Int. SoC Design Conf. (ISOCC), 2010, pp. 323–327.

[3] K. Du, P. Varman, and K. Mohanram, “High performance reliable variable latency carry select addition,” in Proc. Design Autom. Test Eur. Conf. Exhibit. (DATE), Mar. 2012, pp. 1257–1262.

[4] Gai Liu, Ye Tao, Mingxing Tan, and Zhiru Zhang, Computer Systems Laboratory, Electrical and Computer Engineering, “CASA: Correlation-Aware Speculative Adders”, 2014.

[5] Junjun Hu and WeikangQian, Shanghai “A New Approximate Adder with Low Relative Error and Correct Sign Calculation”, 2014.

[6] Ing-Chao Lin, Senior Member, IEEE, Yi-Ming Yang, and Cheng-Chian Lin, “High-Performance Low-Power Carry Speculative Addition With Variable Latency”, IEEE transactions on very large scale integration (vlsi) systems, 2014.

[7] Du, P. Varman, and K. Mohanram, “Static window addition: A new paradigm for the design of variable latency adders,” in Proc. Int. Conf. Comput. Design (ICCD), Oct. 2011, pp. 455–456.

[8] N.V.Mujadiya, “Instruction scheduling on variable latency functional units of VLIW processors,” in Proc. Int. Symp. Electron. Syst. Design (ISED), 2011, pp. 307–312.
[9] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low-power digital signal processing using approximate adders,” IEEE Trans. Computer.- Aided Design Integration. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.
[10] Y. Chen et al., “Variable-latency adder (VL-adder) designs for low power and NBTI tolerance,” IEEE Trans. Very Large Scale Integration. (VLSI) Syst., vol. 18, no. 11, pp. 1621–1624, Nov. 2010.
[11] Y. Liu, Y. Sun, Y. Zhu, and H. Yang, “Design methodology of variable latency adders with multistage function speculation,” in Proc. 11th Int. Symposium. Qual. Electron. Design (ISQED), Mar. 2010, pp. 824–830.
[12] Zhu, W. L. Goh, K. S. Yeo, and Z. H. Kong, “Design of low-power high-speed truncation-error tolerant adder and its application in digital signal processing,” IEEE Trans. Very Large Scale Integration. (VLSI) Syst., vol. 18, no. 8, pp. 1225–1229, Aug. 2010.