CARRY SKIP ADDER (CSKA) HIGH-SPEED OPERATING UNDER WIDE RANGE OF SUPPLY LOGIC AT DIFFERENT LEVELS

PALEPU DIVYA[1], Mr.S.CHANDRA SEKHAR[2]

[1]PG Scholar in VLSI Design, [2]Associate Professor
Department of ECE,
Dhanekula Institute of Engineering & Technology,
Ganguru, Krishna Dist., Andhra Pradesh, India.

***Abstract* — This project presents a carry skip adder (CSKA) structure that has a higher speed yet lower energy consumption compared with the conventional one. The speed enhancement is achieved by applying concatenation and incrementation schemes to improve the efficiency of the conventional CSKA (Conv-CSKA) structure. In addition, instead of utilizing multiplexer logic, the proposed structure makes use of AND-OR-Invert (AOI) and OR-AND-Invert (OAI) compound gates for the skip logic. The structure may be realized with both fixed stage size and variable stage size styles, wherein the latter further improves the speed and energy parameters of the adder. Finally, a hybrid variable latency extension of the proposed structure, which lowers the power consumption without considerably impacting the speed, is presented. Complexity is decreased by using modified carry select adder with enhanced BEC unit.**

Keywords — CSKA, AOI, OAI, power consumption.

1. INTRODUCTION

The applications where the adders are used are multipliers, DSP to execute various algorithms like FFT, FIR and IIR. The adders come into the picture wherever the concept of multiplication comes. As the millions of instructions per second are performed in microprocessors,so the speed of operation is the most important constraint to be considered while designing multipliers. Due to device portability, miniaturization of device should be high and power consumption should be low. Devices like Mobile, Laptops etcrequire more battery backup.

So, a VLSI designer has to optimize these three parameters in a design. These constraints are very difficult to achieve.So depending on demand or application, some compromise between constraints has to be made. Ripple carry adders exhibits the most compact design but the slowest in speed whereas carry look aheadadder is the fastest one but consumes more area. Carry select adders act as a compromise between the two adders. In 2002, a new concept of hybrid adders is presented to speed up addition process by Wang et al. that gives hybrid carry look-ahead/carry select adders design. In 2008, low power multipliers, based on new hybrid full adders are presented in.

Much of the research efforts of the past years in the area of digital electronics have been directed towards increasing the speed of digital system. Recently the requirement of portability and the moderate improvement in battery performance indicates that the power dissipation is one of the most critical design parameter.

The three most widely accepted metrics to measure the quality of a circuit or to compare various circuit styles are area, delay and power dissipation. Portability imposes a strict limitation on power dissipation while still demands high computational speeds. Hence, in recent VLSI Systems, the power-delay product becomes the most essential metric of performance. The reduction of the power dissipation and the improvement of the speed require optimizations at all levels of the design procedure. Since, most digital circuitry is composed of simple and/or complex gates,we study the best way to implement adders in order to achieve low power dissipation and high speed.

Design of area and power efficient high-speed data path logicsystems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position.

The CSLA is used in many computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum. However, the CSLA is not area efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input Cin=0 and Cin=1, then the final sum and carry are selected by the multiplexers (mux).

Adder is about a digital circuit. In electronics, an adder or summer is a digital circuit that performs addition of numbers. In many computers and other kinds of processors, adders are used not only in the arithmetic logic units, but also in other parts of the processor, where they are used to calculate addresses, table indices and similar.

Although adders can be constructed for many numerical representations, such as binary-coded decimal or excess-3 but the most common adders operate on binary numbers. In cases where two's complement or ones' complement is being used to represent negative numbers, it is trivial to modify an adder into an adder–subtractor. Other signed number representations require a more complex adder.

2. RELATED WORK

An efficient adder design basically improves the performance of a complex DSP system. ADDERS are a key building block in arithmetic and logic units (ALUs) and therefore increasing their speed and reducing their power/energy utilization powerfully have an impact on the speed and power utilization of processors. There are many works on the topic of optimizing the speed and an influence of these units that square measure reportable in. Obviously, it's extraordinarily desirable to realize higher speeds at low-power/energy consumptions that are a challenge for the designers of general purpose processors. One of the effective techniques to lower the facility consumption of digital circuits is to reduce the provision voltage because of quadratic dependence of the switch energy on the voltage. Moreover, the sub threshold current, that is the main leak part in OFF devices, has an exponential dependence on the supply voltage level through the drain-induced barrier lowering impact. Depending on the amount of the supply voltage reduction, the operation of ON devices might reside among the super threshold, near-threshold, or sub threshold regions. operational inside the super threshold region provides U.S. with lower delay and higher switching and leak powers compared with the near/sub threshold regions. Among the sub threshold region, the gate delay and leak power exhibit exponential dependences on the availability and threshold voltages. Moreover, these voltages are (potentially) subject to methodology and environmental variations among the nano-scale technologies. The variations increase uncertainties among the aforesaid performance parameters. Additionally, the small sub threshold current causes a large delay for the circuit's operative within the sub threshold region. In this paper, given the attractive features of the CSKA structure, we have focused on reducing its delay by modifying its implementation based on the static CMOS logic. The concentration on the static CMOS originates from the desire to have are liably operating circuit under a wide range of supply voltages in highly scaled technologies [10].The proposed modification increases the speed considerably while maintaining the low area and power consumption features of the CSKA. In addition, an adjustment of the structure, based on the variable latency technique, which in turn lowers the power consumption without considerably impacting the CSKA speed, is also presented. To the best of our knowledge, no work concentrating on design of CSKAs operating from the super threshold region down to near-threshold region and also, the design of(hybrid) variable latency CSKA structures have been reported in the literature. Hence, the contributions of this paper can be summarized as follows. 1) Proposing a modified CSKA structure by combining the concatenation and the incrementation schemes to the conventional CSKA (Conv-CSKA) structure for enhancing the speed and energy efficiency of the adder. The modification provides us with the ability to use simpler carry skip logics based on the AOI/OAI compound gates instead of the multiplexer. 2) Providing a design strategy for constructing an efficient CSKA structure based on analytically expressions presented for the critical path delay.3) Investigating the impact of voltage scaling on the efficiency of the proposed CSKA structure(from the nominal supply voltage to the near-threshold voltage). 4)Proposing a hybrid variable latency CSKA structure based on the extension of the suggested CSKA, by replacing some of the middle stages in its structure with a PPA, which is modified in this paper.

3. METHOD

3.1 HALF ADDER:

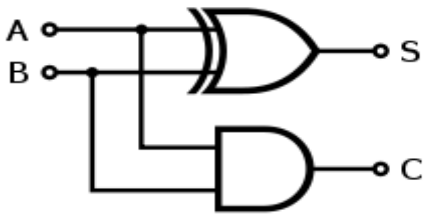


Fig :3.1 Half adder

The half adder adds two one-bit binary numbers A and B. It has two outputs, Sum and Carry. The simplest half adder design, shown in figure 3.1, incorporates an XOR gate for Sum and an AND gate for Carry.

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 3.1: Truth table of half adder

3.2 FULL ADDER:

With the addition of an OR gate to combine their carry outputs, two half adders can be combined to make a full adder.

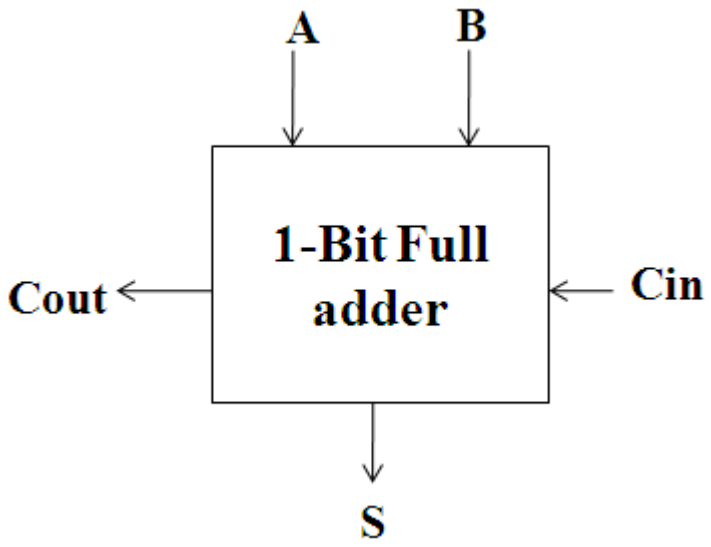


Fig 3.2: Full adder

The logic diagram of Full adder is shown in the figure 3.3. The full adder uses 2 XOR gates for the calculation of its sum and 1 XOR, 1 OR and 2 AND gates for the calculation of carry out.

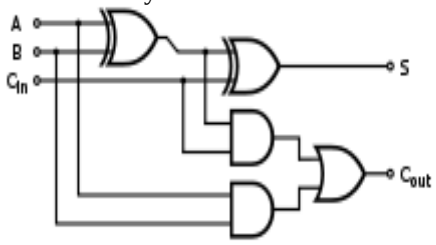


Fig 3.3: Logic diagram of Full adder

The fulladder is usually a component in a cascade of adders, which add 8, 16, 32etc binary numbers. The circuit produces a two-bit output sum represented by the signals C_{out} and S. The one-bit full adder's truth table is:

Inputs			Outputs	
A	B	Cin	S	Cout
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

Table 3.2: Truth table of 1 bit full adder

A full adder can be implemented in many different ways such as with a custom transistor-level circuit or composed of other gates. One example implementation is with

S=ABCin

Cout= (A.B)+ (Cin. (A B))

In this implementation, the final OR gate before the carry-out output may be replaced by an XOR gate without altering the resulting logic. Using only two types of gates is convenient if the circuit is being implemented using simple IC chips which contain only one gate type per chip.

In this, C_{out} can be implemented as Cout=(A.B) + (Cin . (A B)).

A full adder can be constructed from two half adders by connecting A and B to the input of one half adder, connecting the sum from that to an input to the second adder, connecting C_i to the other input and OR the two carry outputs. Equivalently, S could be made the three-bit XOR of A, B and C_i and C_{out} could be made the three-bit majority function of A, B and C.

3.3 FAST ADDERS:

3.3.1 Ripple Carry Adder:

Concatenating the N full adders forms N bit Ripple carry adder. In this carry out of previous full adder becomes the input carry for the next full adder. It calculates sum and carry according to the following equations. As carry ripples from one full adder to the other, it traverses longest critical path and exhibits worst-case delay.

Si = Ai xor Bi xorCi

Ci+1 = Ai Bi + (Ai + Bi) Ci; where i = 0, 1, …, n-1

RCA is the slowest in all adders (O (n) time) but it is very compact in size (O (n) area). If the ripple carry adder is implemented by concatenating N full adders, the delay of such an adder is 2N gate delays from Cin to Cout. The delay of adder increases linearly with increase in number of bits. The block diagram of RCA is shown in figure 3.4.

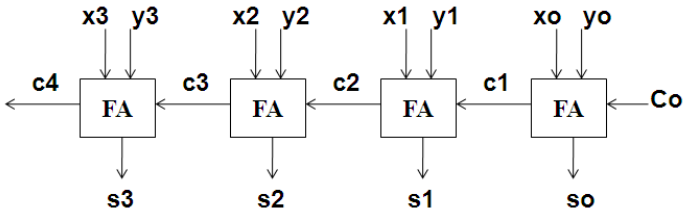


Fig 3.4: Block diagram of RCA

It is possible to create a logical circuit using multiple full adders to add N-bit numbers. Each full adder inputs a C_{in}, which is the C_{out} of the previous adder. This kind of adder is a ripple carry adder, since each carry bit "ripples" to the next full adder. It can be noted that the first (and only the first) full adder may be replaced by a half adder.

The layout of a ripple carry adder is simple, which allows for fast design time.However, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic. In a 32-bit ripple carry adder, there are 32 full adders, so the critical path (worst case) delay is 3 (from input to carry in first adder) + 31 * 2 (for carry propagation in later adders) = 65 gate delays. A design with alternating carry polarities and optimized AND-OR-Invert gates can be about twice as fast.

3.3.2 Carrylookahead adders:

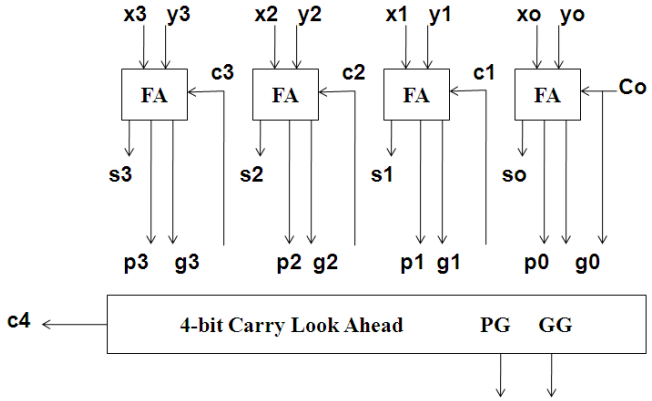


Fig 3.5: 4-bit adder with carry lookahead

To reduce the computation time, engineers devised faster ways to add two binary numbers by using carry-lookahead adders.

They work by creating two signals (P and G) for each bit position, based on if a carry is propagated through from a less significant bit position (at least one input is a '1'), a carry is generated in that bit position (both inputs are '1'), or if a carry is killed in that bit position (both inputs are '0').

In most cases, P is simply the sum output of a half-adder and G is the carry output of the same adder. After P and G are generated, the carries for every bit position are created. Some advanced carry-lookahead architectures are the Manchester carry chain, Brent–Kung adder and the Kogge–Stone adder.
Some other multi-bit adder architectures break the adder into blocks. It is possible to vary the length of these blocks based on the propagation delay of the circuits to optimize computation time. These block based adders include the carry bypass adder which willdetermine P and G values for each block rather than each bit and the carry select adderwhich pre-generates sum and carry values for either possible carry input to the block.

Other adder designs include thecarry save adder, carry-select adder, conditional-sum adder, carry-skip adder and carry-complete adder.

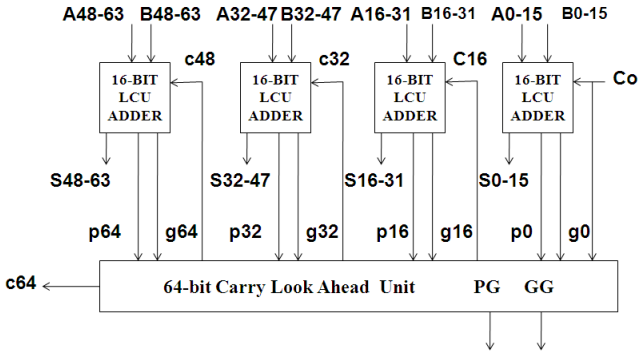


Fig 3.6: A 64-bit carry look ahead unit

By combining multiple carry lookahead adders even larger adders can be created. This can be used at multiple levels to make even larger adders. For example, the following adder is a 64-bit adder that uses four 16-bit CLAs with two levels of LCUs.

3.3.3 Carry Skip Adder (CSKA):

The carry-skip adder reduces the time needed to propagate the carry by skipping over groups of consecutive adder stages, is known to be comparable in speed to the carry look-ahead technique while it uses less logic area and less power.
Uniform sized adder:
A carry skip adder divides the words to be added into groups of equal size of k-bits. Carry Propagate pi signals may be used within a group of bits to accelerate the carry propagation. If all the pi signals within the group are pi=1, carry bypasses the entire group as shown in figure 3.7.

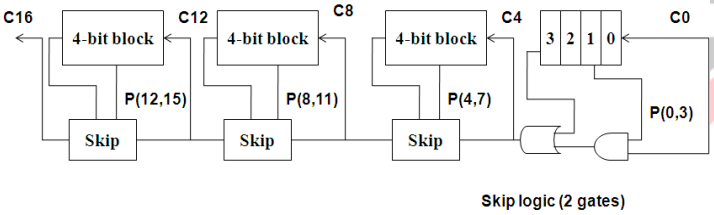


Fig 3.7: Carry skip adder

P = pi * pi+1 * pi+2 *… pi+k
In this way, the delay is reduced as compared to ripple carry adder. The worst-case carry propagation delay in a N-bit carry skip adder with fixed block width b, assuming that one stage of ripple has the same delay as one skip, can be derived:
TCSKA = (b -1)+0.5+(N/b-2)+(b -1) = 2b + N/b – 3.5 Stages
Block width tremendously affects the latency of adder. Latency is directly proportional to block width. More number of blocks means block width is less, hence more delay.

3.3.4 Variable Block Adder:

The idea behind Variable Block Adder (VBA) is to minimize the critical path delay in the carry chain of a carry skip adder, while allowing the groups to take different sizes. In case of carry skip adder, such condition will result in more number of skips between stages.
Such an adder design is called variable block design, which is tremendously used to fasten the speed of adder. In the variable block carry skip adder design, we divided a 32-bit adder into 4 blocks or groups. The bit widths of groups are taken as: First block is of 4 bits, second is of 6 bits, third is 18 bit wide and the last group consist of most significant 4 bits.

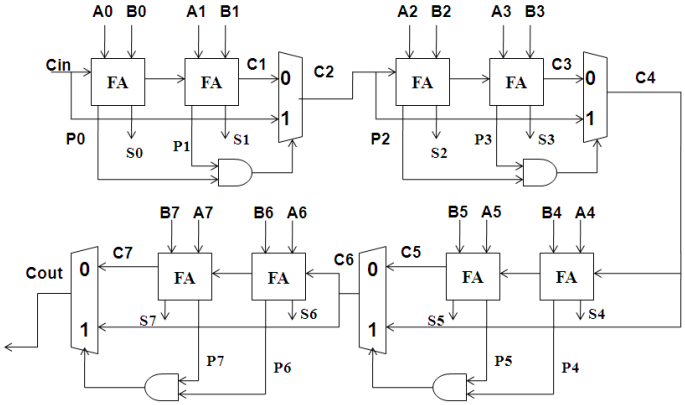


Fig 3.8: Architectural block of 8-bit Carry skip adder

The carry skip adder provides a compromise between a ripple carry adder and a CSLA adder. The carry skip adder divides the words to be added into blocks. Within each block, ripple carry is used to produce the sum bit and the carry. The Carry Skip Adder reduces the delay due to the carry computation i.e. by skipping over groups of consecutive adder stages.

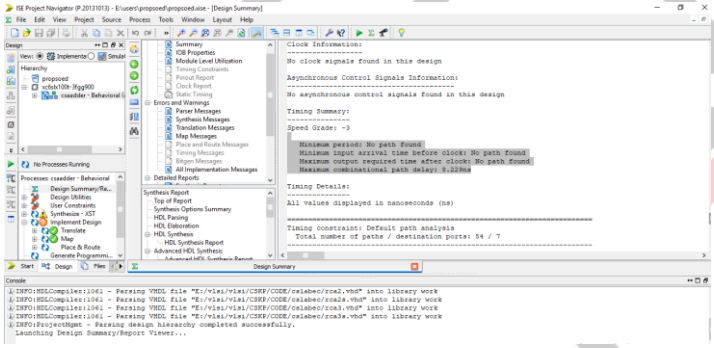
- If each Ai # Bi in a group, then we do not need to compute the new value of Ci+1 for that block; the carry-in of the block can be propagated directly to the next block.
- If Ai = Bi = 1 for some i in the group, a carry is generated which may be propagated up to the output of that group.
- If Ai = Bi = 0, a carry will not be propagated by that bit location.

The basic idea of a carry-skip adder is to detect if in each group all Ai # Bi and enable the block‘s carryin to skip the block when this happens as shown in figure 3.8.

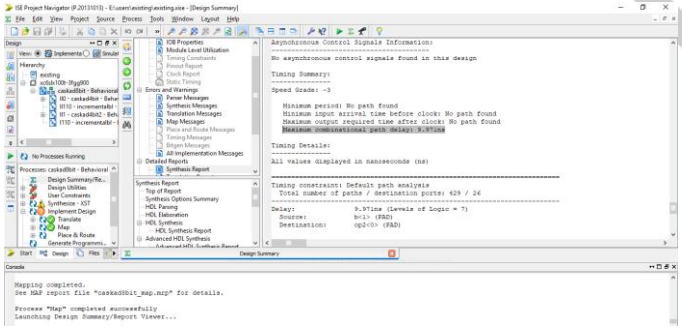
In general, a blockskip delay can be different from the delay due to the propagation of a carry to the next bit position. With carry skip adders, the linear growth of carry chain delay with the size of the input operands is improved by allowing carries to skip across blocks of bits, rather than rippling through them.

4. SIMULATION RESULTS

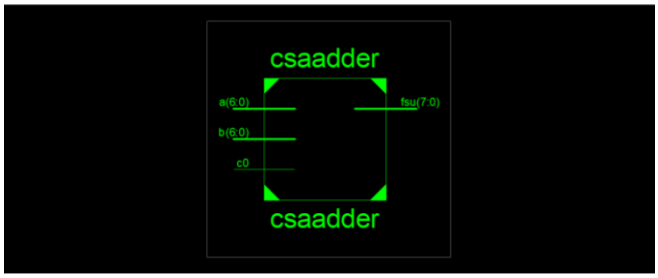
1) EXSISTING SYSTEM AREA:



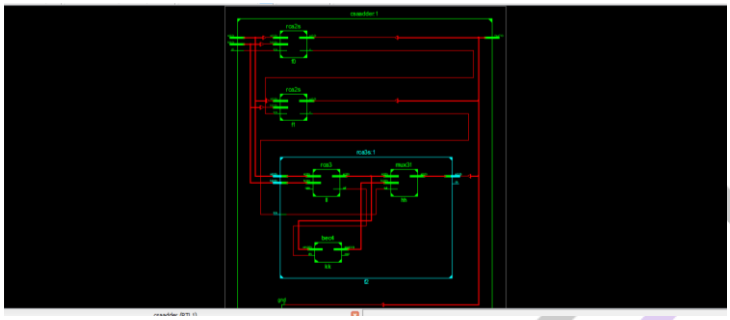
PROPOSED SYSTEM AREA:



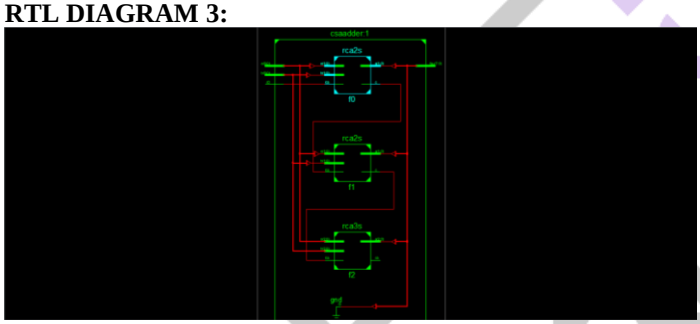
RTL DIAGRAM:



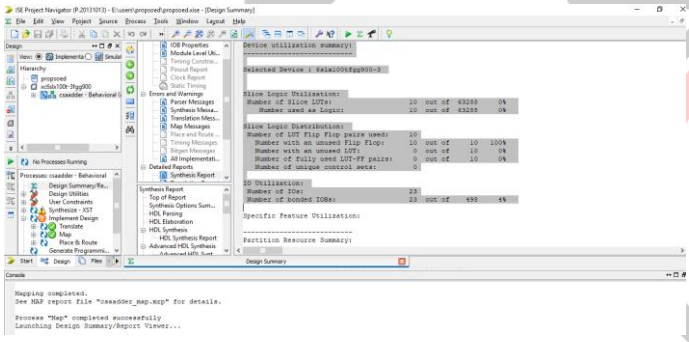
RTL DIAGRAM 2:



RTL DIAGRAM 3:



EXSISTING SYSTEM DELAY:



PROPOSED SYSTEM DELAY:

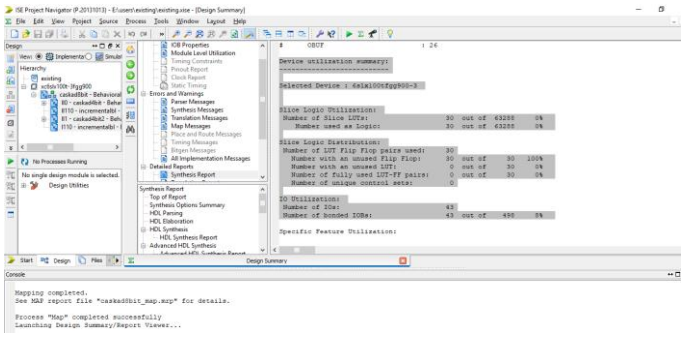


TABLE OF SIMULATION RESULTS DESCRIPTION:

PARAMETERS	EXISTING SYSTEM	PROPOSED SYSTEM
Area	206 gates	86 gates
Time	9.971 n.s	8.229 n.s

CONCLUSION

In this paper, a static CMOS CSKA structure called CI-CSKA was proposed, which exhibits a higher speed and lower energy consumption compared with those of the conventional one. The speed enhancement was achieved by modifying the structure through the concatenation and incrementation techniques. In addition, AOI and OAI compound gates were exploited for the carry skip logics. Further, enhanced by decreasing the complexity of adder using modified carry select adder architecture.

REFERENCES

Good Teachers are worth more than thousand books, we have them in Our Department.

[1] I. Koren, Computer Arithmetic Algorithms, 2nd ed. Natick, MA, USA: A K Peters, Ltd., 2002.
[2] R. Zlatanovici, S. Kao, and B. Nikolic, "Energy–delay optimization of 64-bit carry-lookahead adders with a 240 ps 90 nm CMOS design example," IEEE J. Solid-State Circuits, vol. 44, no. 2, pp. 569–583, Feb. 2009.
[3] S. K. Mathew, M. A. Anders, B. Bloechel, T. Nguyen, R. K. Krishnamurthy, and S. Borkar, "A 4-GHz 300-mW 64-bit integer execution ALU with dual supply voltages in 90-nm CMOS," IEEE J. Solid-State Circuits, vol. 40, no. 1, pp. 44–51, Jan. 2005.
[4] V. G. Oklobdzija, B. R. Zeydel, H. Q. Dao, S. Mathew, and R. Krishnamurthy, "Comparison of high-performance VLSI adders in the energy-delay space," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 6, pp. 754–758, Jun. 2005.
[5] B. Ramkumar and H. M. Kittur, "Low-power and area-efficient carry select adder," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 2, pp. 371–375, Feb. 2012.
[6] M. Vratonjic, B. R. Zeydel, and V. G. Oklobdzija, "Low- and ultra low-power arithmetic units: Design and comparison," in Proc. IEEE Int. Conf. Comput. Design, VLSI Comput. Process. (ICCD), Oct. 2005, pp. 249–252.
[7] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 43, no. 10, pp. 689–702, Oct. 1996.
[8] Y. He and C.-H. Chang, "A power-delay efficient hybrid carrylookahead/ carry-select based redundant binary to two's complement converter," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 1, pp. 336–346, Feb. 2008.
[9] C.-H. Chang, J. Gu, and M. Zhang, "A review of 0.18 μm full adder performances for tree structured arithmetic circuits," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 6, pp. 686–695, Jun. 2005.
[10] D. Markovic, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow-power design in near-threshold region," Proc. IEEE, vol. 98, no. 2, pp. 237–252, Feb. 2010.