ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

VLSI Adder Implementation Using Generalized Ling Algorithm

Leya Anu George [1], Soumiya Rasheed[2], Vipin Thomas[3]

PG Holder [AE], Dept. of ECE, Ilahia College of Engineering, Mulavoor, Kerala, India[1]

Assistant Professor, Dept. of ECE, Ilahia College of Engineering, Mulavoor, Kerala, India[2]

Assistant Professor, Dept. of ECE, Ilahia College of Engineering, Mulavoor, Kerala, India[3]

ABSTRACT: Several addition algorithms are developed to improve speed of addition by manipulating the equations used for sum and carry. As with increase in speed of carry and sum computation, the time required for all mathematical calculation in a digital system can be reduced. Carry Look Ahead Adder calculates carry signals in advance based on input signal but carry calculation becomes complex beyond 4 bits. CLA is implemented in three stages; pre-processing, carry generation and parallel addition. Ling Adder is a special kind of CLA Adders. The improvement is in the carry generation stage which is the most intensive one. i.e. pseudo carry $h_i = c_i + c_{i-1}$ is propagated instead of c_i. Hardware implemented using Ling's algorithm is speedier than Carry Look Ahead adder. These expressions were modified by Ling, reducing complexity of carry generation but at the cost of sum generation complexity. However the total time for addition process is quite less than normal CLA. Ling's algorithm is hardware specific i.e. hardware derived for n bits is specific for it. The implementation differs for every n bit adder. Here hardware specific Ling Adder is transformed to general purpose Ling adder. Proposed adder is based on generalized Ling's algorithm where delay will be same for all addition process. In this thesis, 4,8,16,32 and 64 bit adders for CLA and Ling algorithm is implemented and the time delay, throughput and utilization was compared between two adders.

KEYWORDS: Carry look ahead adder, generalized Ling algorithm.

I. INTRODUCTION

A VLSI circuit has developed as a critical technological need in recent years with increase in demand for portable consumer electronics products. Adder is one of the most important functional components of energy efficient design units like CPU, ALU, floating-point unit and address generation unit like cache or memory access unit of these systems. For years, changing technology and operating constraints demanded adder implementations to obtain improvements in performance. Earlier, the prime constraint was area, leading to development of adders, such as carry-skip, which improves the speed of addition while maintaining low gate-count. As technology scaling continued, which let more logic gates per chip, complex parallel prefix schemes, yielding fast adder designs became viable. In this environment, implementations must be compared by optimizing the circuit sizing for speed under energy, output load, input size, and performance constraints.

With each new technology generation, the gap between addition algorithms and energy-efficient realizations has grown. Guidance for energy-efficient addition algorithm selection has-been nonexistent and only recently designs were compared from more than a single implementation point. Also, the reduced benefits were obtained from technology scaling. Adding to that supply and threshold voltages can no longer be reduced at the same rates as in previous technologies and static power dissipation from leakage continues to grow. Thus, improvements in energy efficiency must come from either restructuring the adder, optimally sizing transistors, or through the use of new devices.

The intent of this thesis is to propose a generalized Ling algorithm dependent energy efficient adder. Ling algorithm is a recursive algorithm, an improved version of carry look ahead adder. This algorithm is based on the propagation of a composite term in place of conventional look ahead carry giving a faster and less expensive adder. In this thesis Ling adder is introduced and described in a general manner in order to expose the essence of Ling's algorithm. The proposed

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

adder is modelled using Verilog HDL and simulated using ModelSim 6.5SE. Xilinx ISE 8.1i is used to synthesize the model.

Rest of the thesis is organized as follows. Section II frames the basic types of adders and introduces the recursive algorithm in existence, section III describes Carry look ahead and various parameters measured based on it. Section IV describes the model based on Ling algorithm, section V presents the synthesis and simulation results and the comparison between the two adders and section VI concludes the thesis with a foresight of future researches.

II.ADDERS

In the ripple carry adder, the addition speed is limited by the time required for the carries to propagate or ripple through all of the stages of the adder. The carry look ahead adder (CLA) offers a solution to this, based on Weinberger's recursive algorithm for carry computation, by calculating the carry signals in advance, based on the input signals. Predicting carry in advance is based on two functions of the full adder, called the carry generate and carry propagate functions. The carry generate function indicates when an output carry is generated. i.e. when both the input bits are one, given by $G = A.B$. A carry input may be propagated when either or both of the input bits are 1s,given by $P = A + B$. Here the carry out and sum are calculated as $Cout = G + P.C_{in}$ and sum is given by the equation, $S_i = (a_i \oplus b_i) \oplus c_i$. Though carry look ahead logic is a recursive solution, this problem can be dealt using many other recursive solutions.

A faster adder can be designed by using the Ling scheme, the group carry generate and propagate (G and P) are replaced by similar functions (called H or pseudo carry) which can be produced in fewer stages than the group G and P and is available one stage earlier than the corresponding G signal. G and P is found similar to Weinberger's method.

The general transformation of c_i is defined as: $c_i = \begin{cases} t_{i-1}h_{i-1}, & i > 0 \\ c_0, & i = 0 \end{cases}$

where the pseudo-carry, h_i is defined as $h_i = g_i + c_i$.

By factoring p_i out of the carry expression and propagating hi instead of C_{i+1}, all cases where carry is generated and/or propagated from the stage preceding stage i are included in h_i. The advantage of using pseudo-carry instead of carry is offset by the increased complexity of sum computation, which requires the real carry to form individual sum signals.

$$s_i = \begin{cases} a_i \oplus b_i, & h_{i-1} = 0 \\ a_i \oplus b_i \oplus t_{i-1}, & h_{i-1} = 0 \end{cases}$$

Ling adder reduces delay by using simplified versions of group generate. Conditional sum reduce delay but increase design time.

III.CARRY LOOK AHEAD ADDER

Carry look ahead adder is implemented in three stages; pre calculation stage where carry generate and propagate terms are calculated, carry logic stage where the carry inputs to all the stages are calculated in advance by using the formula, $C_{i+1} = g_i + p_i C_i$ and finally the adder stage where the results from the previous stages are combined to obtain the final sum as, $S_i = A_i \oplus B_i \oplus C_i$.

Unlike in ripple carry adder, in CLA the cumulative delay is eliminated as we do not have to wait for the carry to ripple though all of the stages before a final carry is available. Thus the CLA speeds up addition process. In CLA the carry logic is complicated.

In telecommunication, throughput shows the amount of bits successfully transferred between the source and destination. It is expressed in terms of bits per sec. Here it is found from the ratio between the number of bits and the time delay for respective number of bits. The elementary programmable logic block in Xilinx FPGAs is called slice that can perform logic functions. Logic resources are grouped in slices to create configurable logic blocks. A slice contains a set number of LUTs, flip-flops and multiplexers. Device Utilization indicates the FPGA element, such as slices, flip-flops, LUTs, and blocks of RAM. Used slices indicate how many of the FPGA element the compiled FPGA uses. Total slices indicate the total number of FPGA elements in the FPGA. Utilization percent indicates the percentage of the FPGA elements that the FPGA application uses.

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

IV.LING ADDER

Ling Adder is a special kind of CLA Adder. Here the first stage is same as that of the normal CLA. The improvement is in the carry generation stage which is the most intensive one. In this adder the carry generation is based on Ling's algorithm. Carry logic is less complex as compared to the normal CLA. Final adder stage is a little more complex than the CLA. In Ling adder carry logic is much faster than that of normal CLA. Sum calculation is a little more complicated. However the total time for addition process is quite less than normal CLA.
Ling algorithm is based on a different set of equations. The carry generate and carry propagate equations are same as that of normal CLA. Pseudo carry $h_i = C_i + C_{i-1}$ is propagated instead of C_i. Ling adder is implemented in three stages as CLA. First stage stays same but instead of carry it is pseudo carry which is generated as in Pseudo Carry logic stage as $h_{i+1} = g_i + t_{i-1}h_i$ and final carry can be found using $C_{i+1} = h_{i+1}t_i$. In the adder stage the results from the previous stages are combined to obtain the final sum, $S_i = p_i \oplus h_{i+1} + g_i p_{i-1} h_i$

Ling adder when compared with CLA has reduced carry computation complexity but increased sum computation complexity. Here the total number of terms involved is reduced at each pseudo carry generation unit thus leading to reduced time delay for computation. But it was observed that this occurred till a particular bit only.

Compared to CLA, in case of Ling adder, throughput is lesser in case of 4, 8, 16 bits. But the throughput shows an increase from 32 bit onwards. As there is increased complexity of sum calculation in case of Ling adder, the utilization obtained for Ling adder will always be higher compared to CLA.

V. RESULT AND DISCUSSION

The time delay, throughput and utilization obtained for various bits of CLA and LING adder were plotted against the number of bits. The proposed adder is based on generalized Ling's algorithm. Since it is not hardware specific current consumption can be made almost linear. Delay will be same for all addition process. Since the delays are same this algorithm can be used for IP address calculation. Addition process will be faster than normal CLA. So by using these subroutines in Math coprocessor effectively we can increase the overall system performance.

4, 8, 16, 32, 64 bit Ling adder and CLA was synthesized and compared. The value for time delay, throughput and utilization were observed for both CLA and Ling adder and is given in Table 1.

Table 1 CLA vs Ling Adder

Bits	Time delay (ns)		Throughput (Mbps)		Utilization (%)	
	CLA	LING	CLA	LING	CLA	LING
4	11.747	11.832	340	338	0.65	0.78
8	17.102	18.583	467	430	1.3	1.9
16	27.814	19.087	575	550	2	4
32	49.237	48.801	649	655	4	8
64	92.084	81.001	695	790	9	20

From the above obtained results the time delay, throughput and utilization for both the adders were compared graphically.

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

From Fig 1, it was observed that the time delay till 16 bit is greater for Ling. But as the no: of bits increase there is a considerable reduction in the time delay. i.e. beyond 16 bits the time delay is more for CLA.

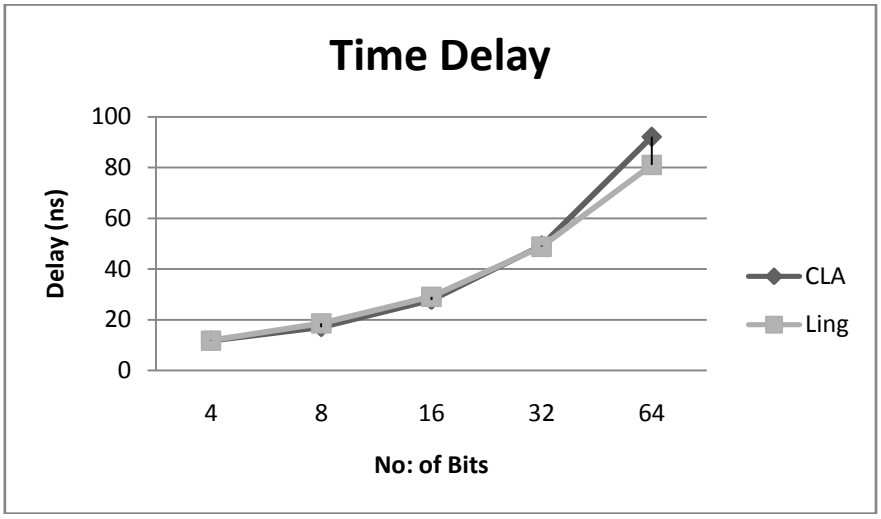

Fig 1 Bit vs. Time Delay for Ling and CLA

Throughput is calculated from the ratio between the number of bits by the time delay obtained in respective cases. From Fig 2, in case of throughput CLA have a clear gain over Ling adder till 16 bits, and the trend reverse from 32 bit onwards.

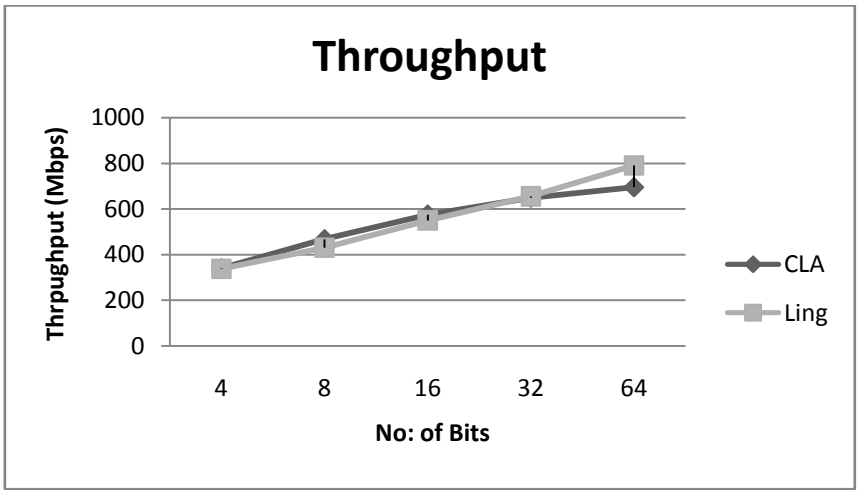

Fig 2 Bit vs. Throughput for Ling and CLA

ISSN (Print) : 2320 – 3765
ISSN (Online) : 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

Percentage of slices utilization found from the ratio of the actual number of slices used to total number of slices available. From Fig 3.it is observed that utilization is always greater for Ling adder as the sum complexity is higher for Ling adder.

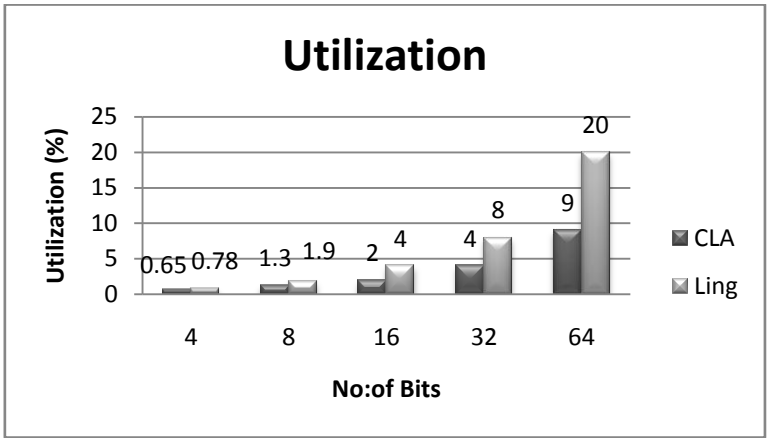

Fig 3 Bit vs. Utilization For Ling And CLA

VI.CONCLUSION

By taking advantage of possible energy-delay tradeoffs on, algorithm and recurrence a 64 bit adder is developed-which operates faster than normal CLA. Adder complexity is reducing as the bit size increases. The word size of existing PC is 32 bit and is now upgrading to 64 bit. By effectively using this addition operation in Math coprocessor we can improve the overall system performance. Since the time of addition decreases, obviously the subtraction time reduces and so does the multiplication and division time. So I strongly believe that adder optimization of 64 bit and higher is a hot topic for future investigation and research.

REFERENCES

[1] B.R. Zeydel, Dursun Baran, Vojin G. Oklobdzija, "Energy-efficient Design Methodologies": High-performance VLSI Adders IEEE journal of solid-state circuits, vol. 45, no. 6, June 2010
[2] R. W. Doran, "Variants of an improved carry look-ahead adder," IEEE Trans. Comput., vol. 37, no. 9, pp. 1110–1113, Sep. 1988.
[3] A.Weinberger and J. L. Smith, "A logic for high-speed addition," Nat. Bur. Stand. Circ., vol. 591, pp. 3–12, 1958.
[4] S. Knowles, "A family of adders," in 14th IEEE Symp. Computer Arithmetic,Adelaide, Australia, Apr. 14–16, 1999.
[5] H. Ling, "High-speed binary adder," IBM J. Res. Devel., vol. 25, no. 3, pp. 156–166, May 1981.
[6] V.G.Oklobdzija, B.R.Zeydel, H.Q.Dao, S.Mathew,and R.Krishnamurthy, "Comparison of high performance VLSI adders in energy delay space"IEEE Trans. VLSI Syst., vol.13,no.6,pp.754–758,Jun.2005.
[7] Giorgos Dimitrakopoulos and Dimitris Nikolos , "High Speed Parallel Prefix VLSI Ling Adders", IEEE transactions on computers, Vol 54, No.2 February 2005.
[8] Jackson, R.; Talwar, S, "High speed binary addition", published in IEEE conference on Signals, systems and Computers on 2004.
[9] Youngmoon Choi and Earl E. Swartzlander, Jr.," Speculative Carry Generation With Prefix Adder", IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 16, No. 3, March 2008