Performance Analysis of 64-bit Carry Lookahead Adders Using Conventional and Hierarchical Structure Styles

Abdulmajeed Alghamdi and Fayez Gebali, IEEE Senior Member
Department of Electrical and Computer Engineering
University of Victoria
Canada

ABSTRACT

This paper introduces performance analysis of 64-bit Carry Lookahead Adders using conventional and hierarchical structure styles. We evaluate conventional carry lookahead adder (CLA) and hierarchical carry lookahead adder (HCLA) using different parameters. Our design is targeted into FPGA Virtex 7 family. Area, delay, and area-delay product of all design choices are reported. In the experimental results, we reduced CLA delay and area using radix-2 which performed better than traditionally used radix-4 CLA. In addition, we showed that CLA using conventional structure has better performance than the hierarchical structure.

Keywords:Conventional Carry Lookahead Adder (CLA) Hierarchical Carry lookahead Adder (HCLA),VLSI design and FPGA implementation

I. INTRODUCTION

Binary Adders are considered one of the most commonly used circuits in digital electronic applications. Various applications including DSP processors, embedded processors, and high-performance processor rely on the adder circuit in order to perform algorithm operations in their ALU unit. However, certain parameters involving power reduction, time delay and die area have to be taken into consideration in designing these processors [1], [2], [3].In the past, the major concern of the chip design was related to silicon area, and current nanometer technologies have brought power consumption to the main role. In addition, today's processors execute millions of instructions in which speed of operation becomes one of the major constraints of processor design.

II. MOTIVATION & CONTRIBUTION

Carry Lookahead Adder (CLA) using conventional structure is presented in many researches of a digital system [4], [5], [6], [7], [8], [9]. Most of the previous suggestions of enhancing CLA performance are presented using different approaches such as implementing CLA using different logic gate styles [10], [11], [12], [13], [14],various tree structures [15], [16], [17],or enhancing performance using optimizing techniques [18], [19].

In this work, a general methodology is developed for constructing 64-bit CLA and HCLA using number of modules (m) where there are no restrictions for radix-n in each module. The traditional approach requires that the module of CLA be 4.We provide a thorough study the effect of the radix on CLA and HCLA performance. Our contributions can be summarized into :

- We introduced and compared 64-bit CLA and HCLA using different radix to reach the best value of n in order to reduce delay and area.
- We showed that using radix-2 CLA has better performance than commonly used radix-4, as well as better performance than HCLA.

The remainder of the paper is organized as follows: In section III, we discuss N-bit CLA & HCLA basic modules design. Section IV and V show the different implementations of 64-bit CLA & HCLA using identical radix-n and HCLA using nonidentical radix-n respectively. Results and discussion are reported in Section VI. Finally, we offer the conclusion in section VII.

III. DESIGN OF CLA & HCLA MODULES

For N-bit CLA and HCLA, the two n-bit inputs $a[n-1:0]$ and $b[n-1:0]$ to be added are used to generate the carry propagate $p[n-1:0]$ and carry generate $g[n-1:0]$ signals according to the equations:

$$p_i = a_i \oplus b_i \tag{1}$$

$$g_i = a_i \cdot b_i \tag{2}$$

The logic in the bottom half of Figure 1 is called a Partial Full Adder (PFA) which is the responsible for generate and propagate the carry to the carry lookahead module.

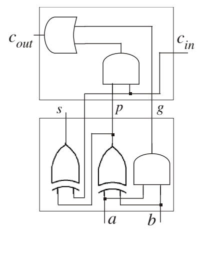

Fig. 1. Partial full adder circuit

Figure 2 shows the n-bit CLA basic module which accepts two n-bit input signals p and g, and the carry-in signal c_{in} and produces n + 1 bits carry signal $c_0 - c_n$ according to the equation:

$$\begin{aligned} c_0 &= c_{in} \\ c_1 &= g_0 + p_0.c_0 \\ c_2 &= g_1 + p_1.g_0 + p_1.p_0.c_0 \\ c_3 &= g_2 + p_2.g_1 + p_2.p_1.g_0 + p_2.p_1.p_0.c_0 \end{aligned}$$

In general, we can write the carry at bit i as:

$$c_i = c_{in} \prod_{j=0}^{i-1} p_j + \sum_{j=0}^{i-1} g_j \prod_{k=j+1}^{i-1} p_k, \qquad 0 \le i \le n \qquad (3)$$

The sum signals are obtaind as:

$$s = p_i \oplus c_i, \quad 0 \le i < N \qquad (4)$$

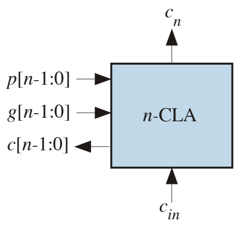

Fig. 2. n-bit CLA basic module

In contrast, Figure 3 shows an n-bit HCLA module that accepts n-bit $p[n-1:0]$ and $g[n-1:0]$ signals and produces two signals: group carry propagate p_{out} and group carry generate g_{out} as discussed in [20], [21]. The carry propagate and carry generate signals are given by:

$$p_{out} = \prod_{j=0}^{n-1} p_j \qquad (5)$$

$$g_{out} = \sum_{i=0}^{n-1} g_i \prod_{j=i+1}^{n-1} p_j \qquad (6)$$

The c_n of n-HCLA can be obtained through a small circuit which is given by:

$$c_n = c_{in} \cdot p_{out} + g_{out} \qquad (7)$$

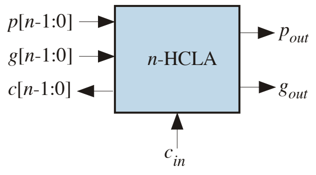

Fig. 3. n-bit HCLA basic module

IV. Implementation of Conventional CLA

In general, N-bit CLA using conventional structure can be obtained according to the equation $N = m \times n$ where m is the number of modules and n is the radix in each module.The lookahead modules compute the carry bits based on the generate and propagate values they receive.

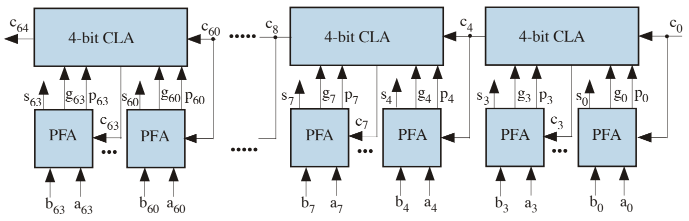
c_{64}
4-bit CLA
c_{60}
c_8
4-bit CLA
c_4
4-bit CLA
c_0
s_{63} g_{63} p_{63} s_{60} g_{60} p_{60}
c_{63}
PFA
s_7 g_7 p_7 s_4 g_4 p_4
c_7
s_3 g_3 p_3 s_0 g_0 p_0
c_3
b_{63} a_{63} b_{60} a_{60} b_7 a_7 b_4 a_4 b_3 a_3 b_0 a_0

Fig. 4. 64-bit CLA using radix-4

Previous studies have introduced the implementation of CLA using radix-4 as shown in Figure 4. This Figure shows a 64-bit CLA is designed using 16 modules with radix-4 in each module. However, Table I shows different values of n which are implemented in our work.

TABLE I. 64-bit CLA using different radix-n

No of modules (m)	Radix-n in each modules
2	32
4	16
8	8
16	4

V. Implementation of Identical & Nonidentical Radix-n HCLA

For both identical and nonidentical HCLA, the N-bit HCLA can be obtained by multiplying the radix-n in the first level (h_0) by the radix(n) in the the second levels (h_1) according to the equation: $N = n_{(h0)} \times n_{(h1)}$.

Figure 5 shows two levels implementation of 64-bit HCLA using identical radix-8.The two signals gg and gp indicate to group generate and group propagate respectively.These signals are still a single bit each they are not a vector. The carry out signal c_{64} is obtained at the top level of the hierarchy based on equation 7.

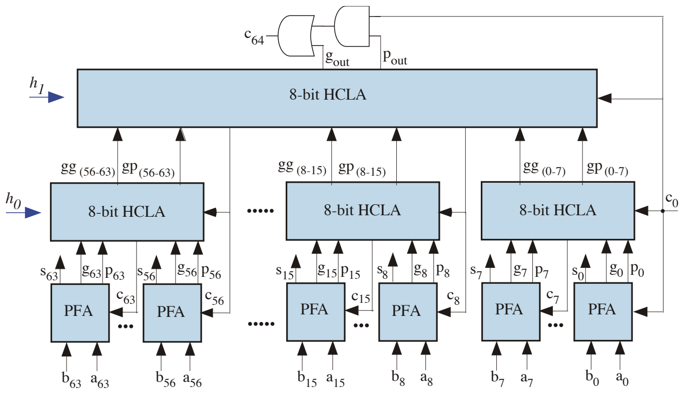
c_{64}
g_{out}
p_{out}
h_1
8-bit HCLA
$gg_{(56\text{-}63)}$ $gp_{(56\text{-}63)}$
$gg_{(8\text{-}15)}$ $gp_{(8\text{-}15)}$
$gg_{(0\text{-}7)}$ $gp_{(0\text{-}7)}$
h_0
c_0
PFA
s_{63} g_{63} p_{63} s_{56} g_{56} p_{56}
c_{63} c_{56}
s_{15} g_{15} p_{15} s_8 g_8 p_8
c_{15} c_8
s_7 g_7 p_7 s_0 g_0 p_0
c_7
b_{63} a_{63} b_{56} a_{56} b_{15} a_{15} b_8 a_8 b_7 a_7 b_0 a_0

Fig. 5. 64-bit HCLA using identical radix-n

On the other hand,HCLA using nonidentical radix-n deals with the case when the radix-n in level '0' and level '1' are not the same. Figure 6 shows 64-bit HCLA using nonidentical radix-n. The first level has four modules based on the radix-16 each. The second level has one module based on the radix-4.

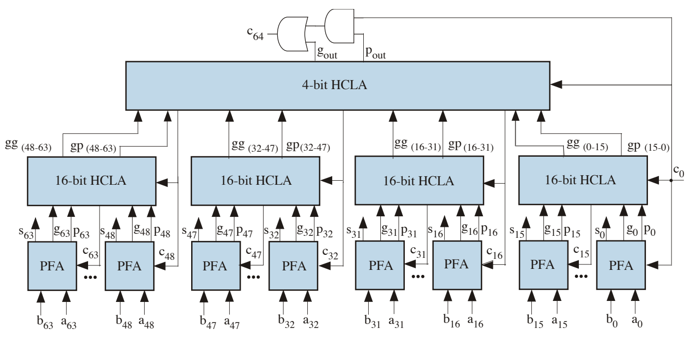

Fig. 6. 64-bit HCLA using nonidentical radix-n

Tables II illustrates two HCLAs implementation using nonidentical radix-n. The first column indicates the number of levels while the second column shows the radix-n in each level.It can be seen that we need three modules to design 64-bit HCLA using radix-32 in the first level and radix-2 in the second level. In contrast, we need five modules to design 64-bit HCLA using radix-16 in the first level and radix-4 in the second level as shown in Figure 6.

TABLE II. 2-LEVELS 64-BIT HCLA USING NONIDENTICAL RADIX-n

Levels (h)	Radix-n in each level	
h_0	32	16
h_1	2	4
No of modules	3	5

VI. Implementation Results and Discussion

In this paper, the hardware implementation of 64-bit CLA and HCLA using various radix-n has been carried out.We used VHDL for writing RTL code and Xilinx ISE tool to simulate and synthesize the design. The simulation assists to verify the design and the synthesis report shows the results of speed and area.The results show a comparison between conventional and hierarchical structures of CLA. Tables III illustrates 64-bit CLA implementation results using different radix-n. The design targeted into FPGA Virtex 7 family to estimate the worst-case delay (T),area (A),area-delay product ($A \times T$). Area estimation is based on the number of slices utilized in each design.

TABLE III. 64-BIT CLA IMPLEMENTATION RESULTS OF DELAY, AREA AND AREA-DELAY PRODUCT

m	$radix - n$	$T(ns)$	A	$T \times A$
2	32	3.55	210	746
4	16	3.66	117	428
8	8	2.57	92	236
16	4	1.51	96	145
32	2	1.35	86	116

It is noticeable that increasing the value of n leads to the worst-cases delay, area and area-delay product. CLA using radix-16 produces 2.15 ns delay compared to radix-4. In addition, designing CLA using radix-32 occupied almost double the area of radix-4. Although radix-4 is commonly used in designing CLA, radix-2 produces smaller delay and area as shown in Table III which positively affect the overall performance.Increasing the value of n in CLA design produces large delay and area due to utilized resources and routing complexity.

Table IV shows the simulation results of 64-bit HCLA using two levels implementation.It can be shown that 64-bit HCLA designed with identical radix-8 produces the least delay and area-delay product at the same time. The HCLA using nonidentical radix such as radix-16 in level '0' and radix-4 in level '1' has the most delay and area-delay product simultaneously.

TABLE IV. 64-BIT HCLA IMPLEMENTATION RESULTS OF DELAY, AREA AND AREA-DELAY PRODUCT

h_0	h_1	$T(ns)$	A	$T \times A$
32	2	5.62	240	1349
16	4	9.98	201	2006
8	8	5.57	207	1153

The implementation results in Table III and Table IV can be summarized as following:

- Increasing the value of n doesn't contribute to improve the performance of CLA design
- CLA using radix-2 has better performance than commonly used radix-4, as well as better performance than HCLA
- CLA using different radix-n has always less delay and smaller area compared to HCLA
- HCLA using identical radix-n has less delay and smaller area compared to HCLA using nonidentical radix-n
- Increasing the value of n leads to a huge area for both CLA and HCLA design
- CLA using conventional structure is more efficient than hierarchical structure

VII. Conclusion

In this work, comparative performance analysis of 64-bit CLA and HCLA has been carried out. Our design targeted FPGA Virtex 7 family to estimate the delay, area, and area-delay product. The results showed that by varying the CLA and HCLA radix-n how performance is changed. As a result, CLA using radix-2 performed faster than traditionally used radix-4 CLA, hence improving the overall performance. In addition, CLA using conventional structure produced 75.7 % less delay and 57.2% smaller area compared to the hierarchical structure.

VIII. Acknowledgment

Alghamdi would like to thank the staff at King Abdullah Scholarship Program (KASP), Saudi Cultural Bureau in Canada, and Ministry of Education for funding and sponsoring my studies & researches at the University of Victoria.

References

[1] H. Elmiligi, M. W. El-Kharashi, and F. Gebali, “Power consumption of 3d networks-on-chips: Modeling and optimization,” *Microprocessors and Microsystems*, vol. 37, no. 6, pp. 530–543, 2013.

[2] B. H. Meyer, J. J. Pieper, J. M. Paul, J. E. Nelson, S. M. Pieper, and A. G. Rowe, “Power-performance simulation and design strategies for single-chip heterogeneous multiprocessors,” *Computers, IEEE Transactions on*, vol. 54, no. 6, pp. 684–697, 2005.

[3] R. Uma, V. Vijayan, M. Mohanapriya, and S. Paul, “Area, delay and power comparison of adder topologies,” *International Journal of VLSI design & Communication Systems (VLSICSj Vo1. 3, No. 1*, 2012.

[4] M. H. Hjakazemi and A. Baniasadi, “An alternative hybrid power-aware adder for high-performance processors,” *Journal of Low Power Electronics*, vol. 10, no. 1, pp. 38–44, 2014.

[5] R. Singh, P. Kumar, and B. Singh, “Performance analysis of fast adders using vhdl,” in *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom’09. International Conference on*. IEEE, 2009, pp. 189–193.

[6] M. Hajkazemi, A. Haghdoost, and A. Baniasdi, “Reconfiguring the carry look-ahead adder using application behavior in embedded processors,” in *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on*. IEEE, 2010, pp. 183–187.

[7] P. Gurjar, R. Solanki, P. Kansliwal, and M. Vucha, “Vlsi implementation of adders for high speed alu,” in *India Conference (INDICON), 2011 Annual IEEE*. IEEE, 2011, pp. 1–6.

[8] L. M. Kalyani Garimella, S. R. Sudha Garimella, K. Duda, and E. Fetzer, “New generation carry look twice-ahead adder cl2a and carry look thrice-ahead adder cl3a,” in *Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on*. IEEE, 2013, pp. 1387–1390.

[9] R. Kumar and S. Dahiya, “Performance analysis of different bit carry look ahead adder using vhdl environment,” vol. 2, 2013.

[10] J. Samanta, M. Halder, and B. P. De, “Performance analysis of high speed low power carry look-ahead adder using different logic styles,” *International Journal of Soft Computing and Engineering (IJSCE) ISSN*, pp. 2231–2307, 2013.

[11] P. Celinski, J. F. López, S. Al-Sarawi, and D. Abbott, “Low depth, low power carry lookahead adders using threshold logic,” *Microelectronics journal*, vol. 33, no. 12, pp. 1071–1077, 2002.

[12] K. Ueda, H. Suzuki, K. Suda, H. Shinohara, and K. Mashiko, “A 64-bit carry look ahead adder using pass transistor bicmos gates,” *Solid-State Circuits, IEEE Journal of*, vol. 31, no. 6, pp. 810–818, 1996.

[13] K.-H. Cheng, S.-W. Cheng, and W.-S. Lee, “64-bit pipeline carry lookahead adder using all-n-transistor tspc logics,” *Journal of Circuits, Systems, and Computers*, vol. 15, no. 01, pp. 13–27, 2006.

[14] M. C. Osorio, C. Sampaio, A. Reis, R. P. Ribas *et al.*, “Enhanced 32-bit carry look-ahead adder using multiple output enable-disable cmos differential logic,” pp. 181–185, 2004.

[15] C. Dacheng, “Vhdl implementation of a fast adder tree,” 2005.

[16] C.-C. Wang, P.-M. Lee, R.-C. Lee, and C.-J. Huang, “A 1.25 ghz 32-bit tree-structured carry lookahead adder,” vol. 4, pp. 80–83, 2001.

[17] S. H. Kim and S.-K. Chin, “Formal verification of tree-structured carry-lookahead adders,” pp. 232–232, 1999.

[18] 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,” *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 2, pp. 569–583, 2009.

[19] H. Dao and V. G. Oklobdzija, “Application of logical effort techniques for speed optimization and analysis of representative adders,” in *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, vol. 2. IEEE, 2001, pp. 1666–1669.

[20] F. Gebali and A. Ibrahim, “Optimized structures of hybrid ripple carry and hierarchical carry lookahead adders,” Microelectronics, 2015, in print.

[21] “http://www.cs.sfu.ca/coursecentral/150/eyal/lectures/cla.pdf.”