ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 2, March 2013

Implementation of Area Efficient 16bit Adder in SPARTAN-3 FPGA

K. Ramu, B.N. Srinivasa Rao
A I E T., Visakhapatnam, A I E T., Visakhapatnam

Abstract*— Fast addition plays an important role in advanced digital systems. In this paper we study efficient implementations of 16-bit full adders on FPGA devices, taking advantage of the specialized carry-logic. Most of fast adders are based on being able to calculate the carry propagation much faster without having to wait for it to ripple through each bit of the adders. The carry look-ahead technique is the most commonly used scheme for accelerating carry propagation. The carry look-ahead adder is to be designed with combination of 4-bit adders. This paper focus on the implementation of 16-bit full adder based on Very High Speed Integrated Circuit (VHSIC) Hardware Description Language. Finally the design is efficiently mapped to Hardware Resources in SPARTAN-3 FPGA. The design is implemented, simulated and synthesized by using VHDL.

***Index Terms*—Addition, Carry Look-Ahead Adder, VHDL.**

I. INTRODUCTION

Fast addition is an essential arithmetic function for most advanced digital systems. It heavily impacts the overall Performance of digital systems. Various adder structures can be used to execute addition [1, 2], such as serial and parallel structures. Most research works of adders are focused on the design of high-speed, low-area, or low-power Adders [3.5]. This paper is aimed to study and implement 16-bit adder based on VHDL. The purpose of using VHDL is that it offers many features for hardware design. When the actual addition is performed, there is no delay from waiting by using carry look-ahead adder. For any circuit larger than 4-bit, the carry look-ahead adder, the circuit becomes very complicated. So 16-bit carry look-ahead adder will be constructed by combing 4-bit carry look-ahead adders.

II. CARRY-LOOKAHEAD ADDER

The general strategy for designing fast adders is to reduce the time required to form carry signals. One approach is to compute the input carry needed by stage i directly from carry like signals obtained from all the preceding stages i-1,i-2,……0,rather than waiting for normal carries to ripple slowly from stages to stages. Adders that use this principle are called carry look-ahead adders. An n-bit carry look-ahead adder is formed from n stages , each of which is basically a full adder modified by replacing its carry output line ci by two auxiliary signals called gi and pi or generate and propagate, respectively ,which are defined by the following logic equation (1);

$g_i = x_i y_i \qquad p_i = x_i + y_i \qquad (1)$

The name generate comes from the fact that stage i generates a carry of 1(c_i=1) independent of the value of c_{i-1} if both x_i and y_i are 1; that is, if $x_i y_i$=1.Stage i propagates c_{i-1}; that is makes c_i=1 in response to c_{i-1}=1 if x_i or y_i is 1__in order words, if x_i+y_i=1. Now the usual equation $c_i=x_i y_i + x_i c_{i-1} + y_i c_{i-1}$, denoting the carry signal c_i to be sent to stage i+1, can be rewritten in terms of g_i and p_i.

$c_i = g_i + p_i c_{i-1} \qquad (2)$

Similarly, c_{i-1} can be expressed in terms of g_{i-1}, p_{i-1} and c_{i-2},

$c_{i-1} = g_{i-1} + p_{i-1} c_{i-2} \qquad (3)$

On substituting Equation (3) into Equation (2),

$c_i = g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-2} \qquad (4)$

Continuing in this way, c_i can be expressed as a sum-of-products function of the p and g outputs of all the preceding stages. For example, the carries in a four-stage carry look-ahead adder are defined as follows:

$c_0 = g_0 + p_0.c_{in}$
$c_1 = g_1 + p_1.g_0 + p_1.p_0.c_{in}$
$c_2 = g_2 + p_2.g_1 + p_2.p_1.g_0 + p_2.p_1.p_0.c_{in}$
$c_3 = g_3 + p_3.g_2 + p_3.p_2.g_1 + p_3p_2.p_1.g_0 + p_3p_2.p_1.p_0.c_{in} \qquad (5)$

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 2, March 2013

Fig. 1 shows the general form of a carry look-ahead adder circuit designed in this way [3].

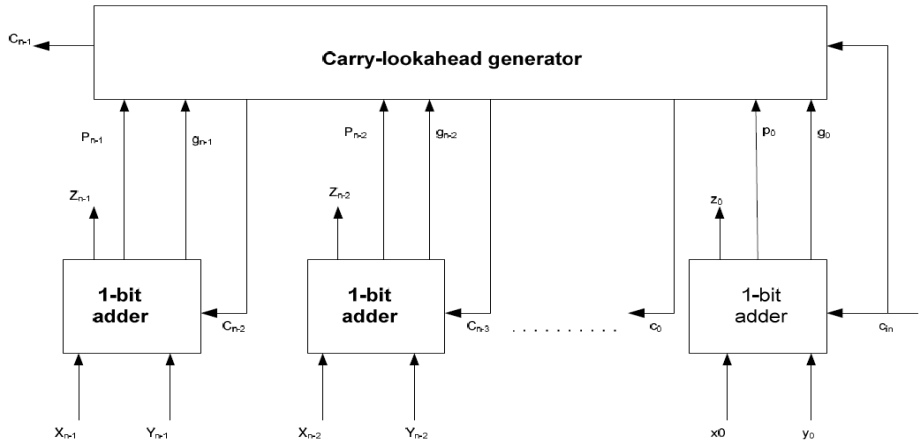

Fig. 1 Overall Structure of Carry Look-ahead Adder

A. Adder Expansion

The methods of handing carry signals in the two main combinational adder designs considered so far, namely, ripple carry propagation and carry look-ahead Fig. 1 can be extended to larger adders of the kind needed to execute add instructions in a 32-bit computer. If the n 1-bit (full) adder's stages are replaced in the n-bit ripple-carry design with n k-bit adders, an nk-bit adder can be obtained. Four 4-bit adders such as the 4-bit carry look-ahead circuit of Fig. 2 can be connected in this way to form the 16-bit adder appearing in Fig. 3. This design represents a compromise between a 16-bit stage ripple-carry adder, which is cheap but slow, and a single-stage 16-bit carry look-ahead adder, which is fast, expensive, and impractical because of the complexity of its carry-generation logic. The circuit of Fig.3 effectively combines sets of four x_iy_i inputs into groups that are adder via carry look-ahead; the results computed by the various groups are then linked via ripple carries [3].The components designed for 1-bit addition have been replaced with similar but larger components intended for 4-bit addition. The expanded design of Fig. 1 can be got. Again 1-bit adders with 4-bit adder are being replaced, but now each adder stage produces a propagate-generate signal pair pg instead of cout and a carry look-ahead generator converts the four sets of pg signals to the carry inputs required by the four stages. The "group" g and p signals produced by each 4-bit stage are defined by [3].

$$g = x_iy_i + x_{i-1}\,y_{i-1}(x_i+y_i) + x_{i-2}y_{i-2}(x_i + y_i)\,(x_{i-1} + y_{i-1}) + x_{i-3}y_{i-3}(x_i+y_i)\,(x_{i-1} + y_{i-1})\,(x_{i-2} + y_{i-2}) \quad (6)$$

$$p = (x_i + y_i)\,(x_{i-1} + y_{i-1})\,(x_{i-2} + y_{i-2})\,(x_{i-3} + y_{i-3}) \quad (7)$$

It is not hard to show that the logic to generate the group carry signals cout, c11, c7 and c3 in Fig. 4 is exactly the same as that of the carry look-ahead generator of Fig. 2.

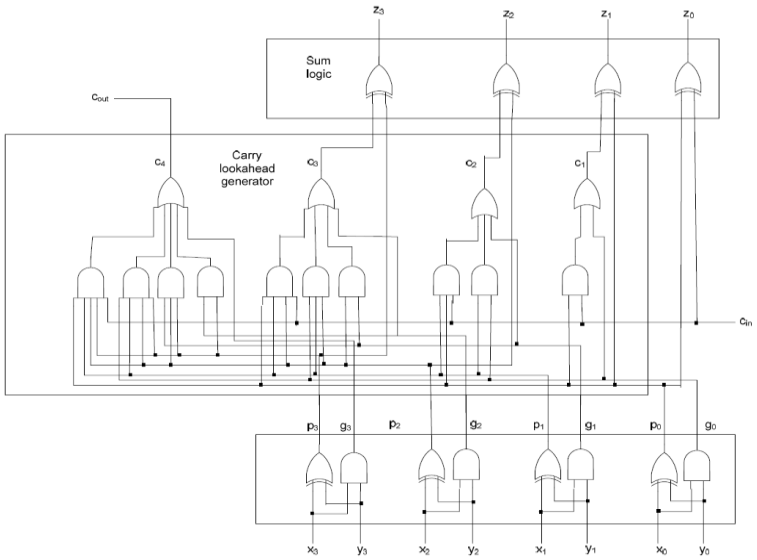

Fig. 2 A 4-Bit Carry Look-ahead Adder

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 2, March 2013

A. 4-bit CLA Structural model Design

To achieve this, individual gates are designed in behavioral model and all the components are interconnected in structural model in VHDL as shown in the fig.2. By using post layout simulation we can find out the actual delays of this model. The technology schematic and simulation results are shown in fig 3 and 4 respectively.

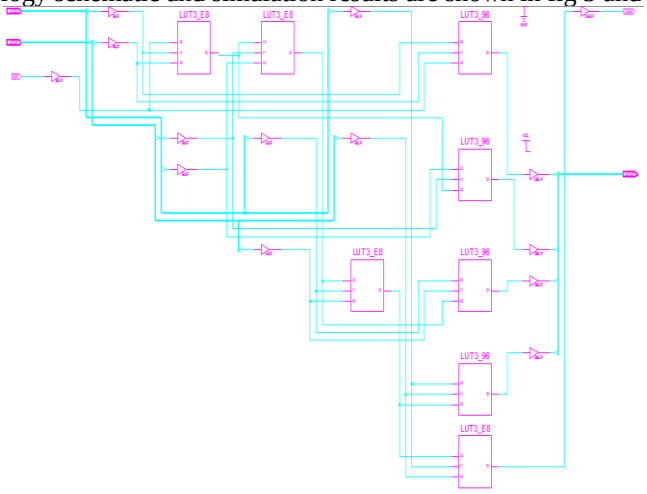

Fig. 4 4-bit CLA Technology Schematic

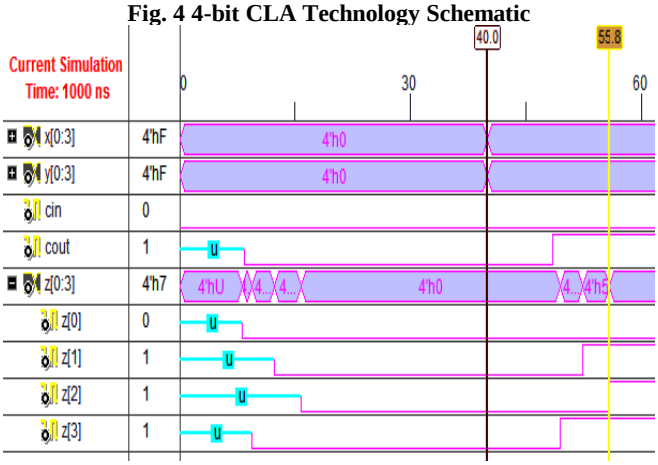

Fig. 5 4-bit CLA Post layout simulation

The following figures shows the technology schematic and post layout simulation results of VHDL optimized 4-bit CLA.

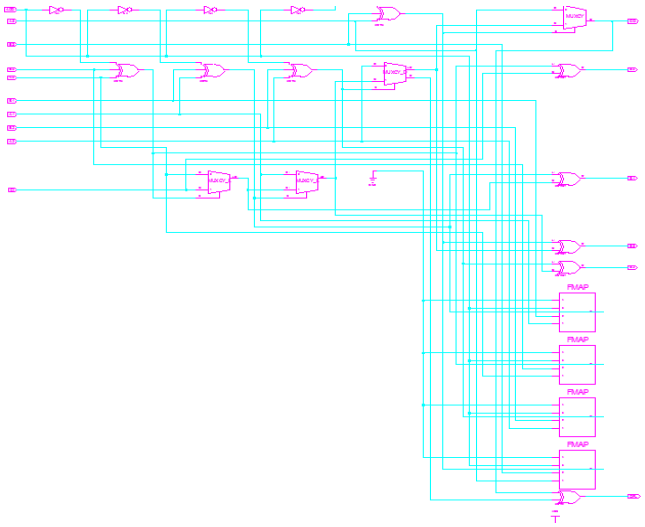

Fig. 6 4-bit CLA Technology Schematic (optimized)

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 2, March 2013

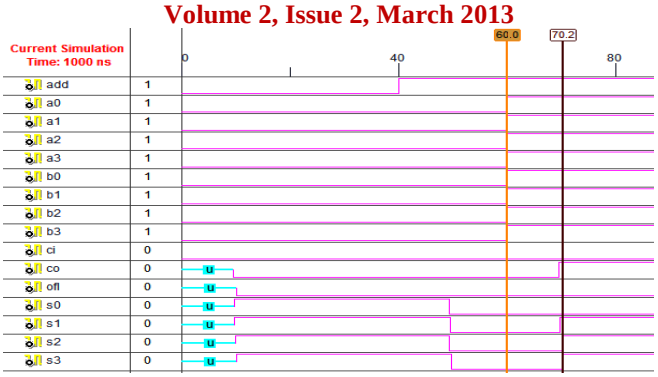

Fig. 7 4-bit CLA (VHDL Primitive) Post layout simulation

III. 16-BIT ADDER DESIGN

The design goal is to minimize the number of gates used; operation speed is not of concern. The circuit is required in several versions that handle different data word sizes, including 4, 8 and 16. Assume that we have standard gate-level and 4-bit register-level components available as building blocks. The lowest cost adders employ ripple-carry propagation. Four 4-bit CLAs are interconnected as ripple adder form as shown in the fig.8. here the carry is propagated to the next stage 4-bit CLA. The internal carry propagation delay of 4-bit adder is reduced by using the technique CLA. So that the overall speed of the 16-bit Adder is increased. Moreover the VHDL Primitive is designed by using faster design elements. So that the sum delays as well as the carry delay is very much improved. For faster Full Adders designs VHDL Primitive can be used to reduce design time and to achieve the speed. The comparison is shown in the table 1.

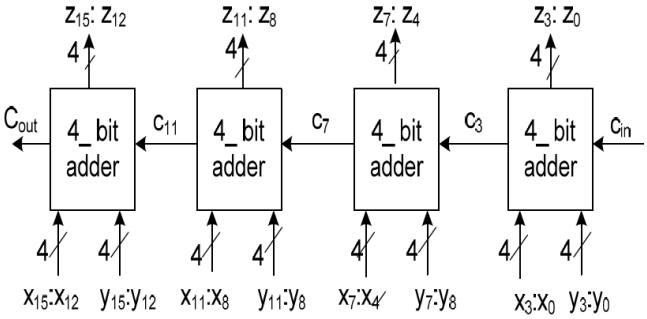

Fig. 8 16-Bit Composed of 4-Bit Adders Linked by Ripple Carry Propagation

The hierarchical design is shown in fig 9. The same concept is used in VHDL Primitive also but the number of 4-bit CLA internal stages are reduced.

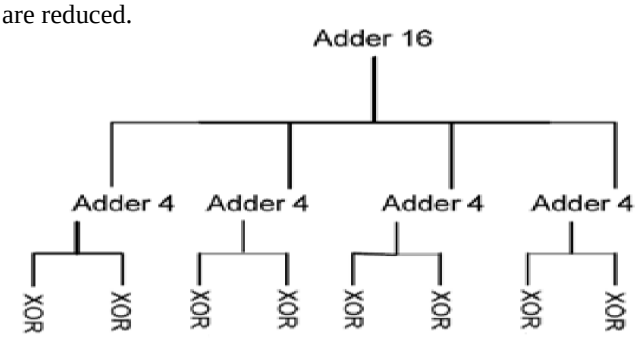

Fig. 9 Hierarchical Level for 16-Bit Carry Look-head Adder

The following table describes the comparison between general CLA design and VHDL Primitive. From this easily we can understand that VHDL Primitive Adder is faster than any other adder, because the VHDL Primitive is optimized design.

Table 1. Comparison between design elements

Design name/Delay	Sum Delay(ns)	Carry Delay(ns)	Final Delay(ns)
General CLA(4-bit)	15.8	8.4	15.8
optimized CLA	13.2	9.5	13.2

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 2, March 2013

16-bit Adder(Using General CLA)	41	33.6	41
16-bit Adder(Using optimized CLA)	38.7	38	38.7

IV. CONCLUSION

In this paper 16-bit Adder is designed and compared with another 16-bit Adder designed by using VHDL Primitive and found 2.3ns difference in between to designs. So that is concluded that 16-bit Adder design using VHDL Primitive is faster than any other design in Semi-custom, because the VDHL Primitive is optimized 4-bit CLA.

ACKNOWLEDGMENT

This material is based upon work supported by the students of Avanthi Institute of Engineering and Technology. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily react the views of AIET

REFERENCES

[1] I. Koren, Computer Arithmetic Algorithms, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 07632, 1993.

[2] J.-C. Lo, .A fast binary adder with conditional carry generation., IEEE Trans. Computers, vol. 46, no. 2, pp. 248.253, Feb. 1997.

[3] A. Tyagi, .A reduced-area scheme for carry-select adders. IEEE Trans. Computers, vol. 42, no. 10, pp. 1163.1170, Oct.1993.

[4] C. Nagendra, M. J. Irwin, and R. M. Owens, .Area-time power tradeoffs in parallel adders. IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing, vol. 43, no. 10, pp. 689.702, Oct. 1996.

[5] T.-Y. Chang and M.-J. Hsiao, .Carry-select adder using single ripple-carry adder., Electronics Letters, vol. 34, no. 22 pp. 2101.2103, Oct. 1998.

[6] Andrew S. Tanenbaum, Structure Computer Organization, Fourth edition, Prentice Hall, Inc., 1999.

[7] Samir Palnitkar, Verilog HDL: A Guide to Digital Design and Synpaper, Sun Microsystems, Inc., 1996.

[8] William Stallings, Computer Organization & Architecture, Sixth edition.

[9] David A. Patterson, John L. Hennessy, Computer Organization and Design, (The Hardware /Software Interface), Third edition, Elsevier Inc., 2005.

[10] May Phyo Thwal, Khin Htay Kyi, and Kyaw S war Soe "Implementation of Adder-Subtractor Design with Verilog HDL, World Academy of Science, Engineering and Technology 39 2008.

AUTHOR BIOGRAPHY

K. Ramu was born in Visakhapatnam, Andhra Pradesh, India. He has received his B.Tech degree in Electronics and Communication Engineering from Jawaharlal Nehru Technological University, Kakinada, India and pursuing M. Tech in VLSI System Design from Jawaharlal Nehru Technological University, Kakinada, India. Currently he is a student in Avanthi Institute of Engineering and Technology Makavarapalem, Visakhapatnam, Andhra Pradesh, India. His area of interests VLSI Semi-Custom design.

B.N.Srinivasa rao received his B.Tech degree in Electronics and Communication Engineering from JNT University, Hyderabad, India and M.Tech in VLSI System Design from JNT University, Hyderabad, India. He is currently working as an Assistant Professor in Avanthi Institute of Engineering and Technology, Visakhapatnam, Andhra Pradesh, India. He has 5 years teaching and 9 years industrial experience. He has 7 publications in various International conferences. His area of interest VLSI Semi and full custom design. He guided many projects for B.Tech and M.Tech students.