VLSI Design, 2002 Vol. 15 (3), pp. 637–645

Taylor & Francis
Taylor & Francis Group

Timing-Driven-Testable Convergent Tree Adders

JOHNNIE A. HUANG[a] and CHIEN-IN HENRY CHEN[b,*]

[a]*LSI Logic Corporation, Milpitas, CA 91436, USA;* [b]*Department of Electrical Engineering, Wright State University, Dayton, OH 45435, USA*

(Received 15 March 2001; Revised 30 January 2002)

Carry lookahead adders have been, over the years, implemented in complex arithmetic units due to their regular structure which leads to efficient VLSI implementation for fast adders. In this paper, timing-driven testability synthesis is first performed on a tree adder. It is shown that the structure of the tree adder provides for a high fanout with an imbalanced tree structure, which likely contributes to a racing effect and increases the delay of the circuit. The timing optimization is then realized by reducing the maximum fanout of the adder and by balancing the tree circuit. For a 56-b testable tree adder, the optimization produces a 6.37% increase in speed of the critical path while only contributing a 2.16% area overhead. The full testability of the circuit is achieved in the optimized adder design.

Keywords: Timing optimization; Balanced tree; Carry-lookahead adders; Tree adders; Testability; Iterative logic arrays

INTRODUCTION

A tree is a circuit constructed from identical modules interconnected in a regular fashion so that there is only one signal path between any two points. The modules or cells used to construct a tree circuit can have internal reconvergent fanout, but fanout is not allowed among the modules. A tree may have multiple outputs and modules interconnected by multiple-bit buses. The circuits studied in this paper are convergent tree circuits. In a convergent tree circuit, as one move from the input module to the output module, the number of (data) signal lines in the circuit decreases. The testing of these adders can be a very difficult task. A large and complicated circuit requires many test patterns to detect all functional faults. In order to decrease the test pattern to detect all possible faults for the entire circuit, the convergent tree circuit should be composed of identical modules interconnected in a one dimensional array so that the array interconnection allows the tests used for one module to be used on other modules [1].

Pseudo-exhaustive testing techniques based on partitioning are perfectly suited for circuits structured as iterative logic arrays (ILAs). The ILAs can be pseudo-exhaustively tested with a number of tests regardless of the number of cells in the ILAs such as the ripple-carry adder. In this paper, we are interested in the 56-b carry lookahead adder. To make a convergent tree C-testable [2], we must have one dimensional ILAs. In Fig. 1, we can see how a convergent tree module can be interpreted as the module of a one-dimensional array [3,4]. The testability of convergent tree circuits can be characterized in terms of the testing properties of their n one-dimensional array types.

The worst case propagation delays in carry-lookahead adders depend on how full adders are grouped structurally together into blocks as well as the number of levels and fanouts. A fully testable 56-b carry-lookahead adder is studied for timing as well as for testability. The 56-b carry-lookahead convergent tree adder has created one-dimensional arrays among GP modules. These arrays are used to make the circuit C-testable, thus, reducing the number of tests to detect all faults in the circuit. In the above design, the branch contribution to each of the carry modules causes racing and high fanout at the output of the carry module for the first 16 bits. The 5 fanouts at the carry-16 module cause a load imbalance that affects the delay of the circuit. This adder will then be compared to a timing driven testable adder optimized for better timing performance by reducing the number of fanouts by using a balanced convergent tree. Both designs will be tested and compared for timing, testability and area.

*Corresponding author.

ISSN 1065-514X print/ISSN 1563-5171 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/1065514021000012255

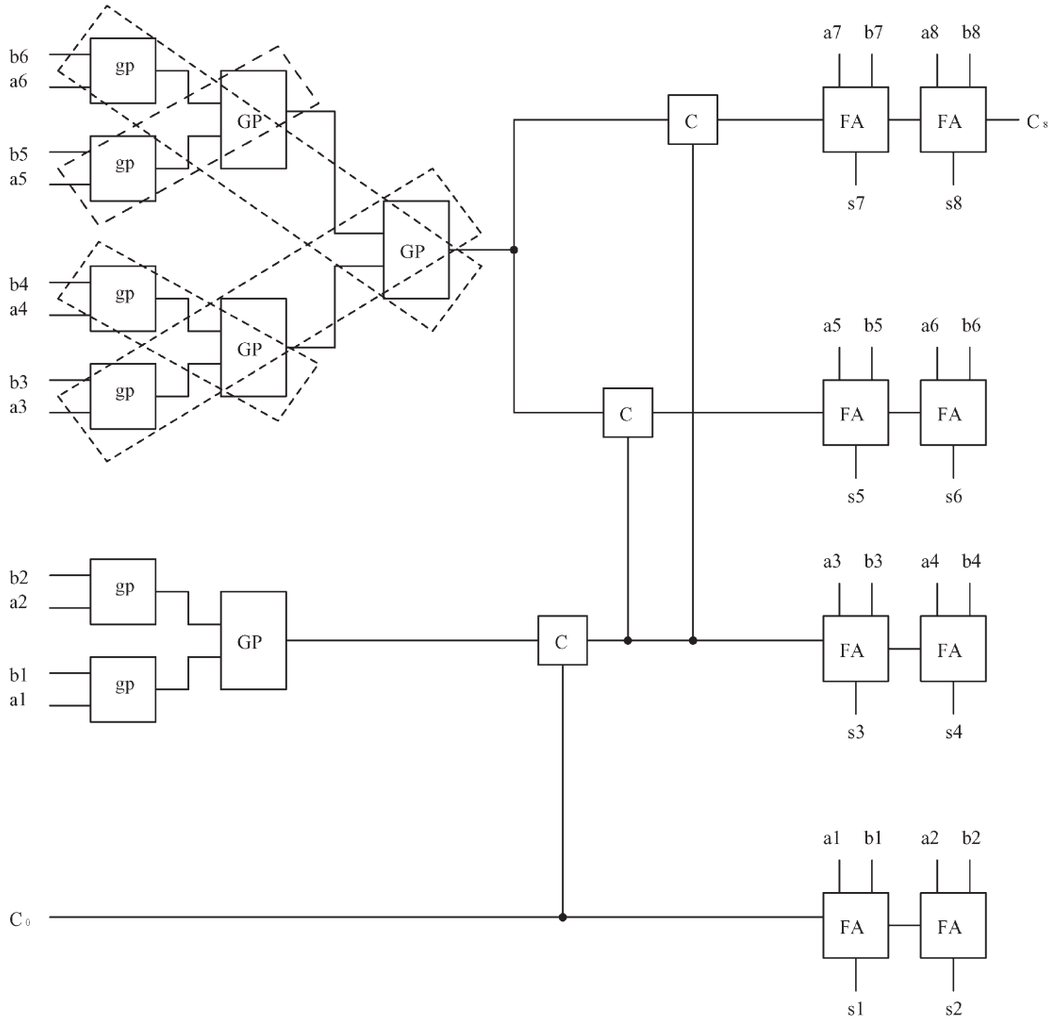

FIGURE 1 One dimensional arrays in convergent tree.

TESTABLE CONVERGENT TREE ADDER (TCTA)

The architecture of the carry-lookahead adder can be obtained from the fundamental carry operation *fco*. Let + denote the logic OR operation, and the juxtaposition of two variables denote the logic AND operation. To define the fundamental carry operation, we must first look at the concept of carry generation and carry propagation. For two operands $A = \{a_j, \ldots, a_k, \ldots, a_m, \ldots, a_0\}$ and $B = \{b_{n-1}, \ldots, b_k, \ldots, b_m, \ldots, b_0\}$ the carry generate and carry propagate are then defined as

$$p_i = a_i + b_i \quad \text{and} \quad g_i = a_i b_i,$$

then

$$[p_{i:j}, g_{i:j}] = (p_{i:k+1}, g_{i:k+1}) \quad fco \quad (p_{k:j}, g_{k:j}),$$

which is defined as

$$p_{i:j} = p_{i:k+1} p_{k:j}$$

and

$$g_{i:j} = g_{i:k+1} + p_{i:k+1} g_{k:j},$$

The *fco* operator has an associativity that can be represented by

$$[(p_{i:m+1}, g_{i:m+1}) \quad fco \quad (p_{m:k+1}, g_{m:k+1})] \quad fco$$
$$(p_{k:j}, g_{k:j}) = (p_{i:m+1}, g_{i:m+1}) \quad fco \quad [(p_{m:k+1}, g_{m:k+1})$$
$$fco \quad (p_{k:j}, g_{k:j})] \quad for \quad j < k < m < i.$$

Now we can express the carry output equation for a 4-b ripple carry adder as

$$(((pg_0 \ fco \ pg_1) \quad fco \quad pg_2) \ fco \ pg_3).$$

In this equation, we can see how the propagate and generate signals for the least significant bit (LSB) groups pg_0 and pg_1 are combined first; that result is then combined with the following group, and so on in a linear

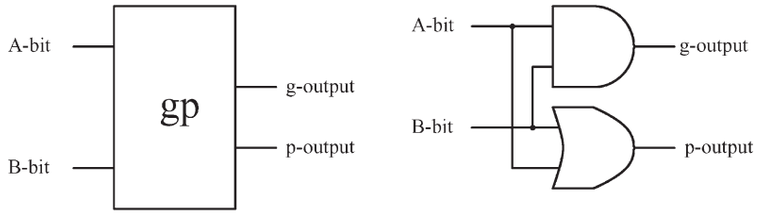

FIGURE 2 The gp module.

fashion. If instead we combined the two lower and upper groups simultaneously and then combined the results, we would get the following result for four bits

$$(pg_0 \ fco \ pg_1) \quad fco \quad (pg_2 \ fco \ pg_3).$$

As can be seen, by using the associative property of the *fco*, both results are equivalent.

TCTA Circuit

The convergent tree carry lookahead adder circuit was built by connecting several distinct modules. The first modules are the generate/propagate modules (gp and GP) and are used to find the carry out function in the carry lookahead adder. These modules are grouped for every 8 input bits to the adder. The results of these modules are combined with the carry from the lower 8 bits in the carry module and the sum is then calculated through groups of 4- and 8-b ripple carry adders. All of the modules will be studied in detail.

gp Module

In the previous section, we showed the equations to calculate the generate and propagate bits. Using those results, the carry out function for a 2-b sum can be expressed as

$$C_{\text{out}} = AB + C_{\text{in}}(A + B) = g_1 + C_{\text{in}}p_1.$$

The g and p values are calculated by using the *gp* module shown in Fig. 2.

Now that we have calculated the generate and propagate values for a single bit, we must extend the concept to include 2 bits. From the 2-b carry out equation we can find

$$C_2 = g_2 + C_1p_2.$$

When the value for C_1 is substituted, the equation becomes

$$C_2 = g_2 + g_1p_2 + C_0p_1p_2,$$

where we can define the 2-b carry generate and propagate, respectively as

$$G_2 = g_2 + g_1p_2$$

$$P_2 = p_1p_2.$$

C_2 now becomes

$$C_2 = G_2 + C_0P_2.$$

Figure 3 shows the logic implementation of the GP module.

For the 4-b carry out (C_4), the Boolean expression is

$$C_4 = g_4 + C_3p_4$$

where $C_3 = g_3 + C_2p_3$. When substituting the C_2, C_3 values,

$$C_4 = g_4 + g_3p_4 + g_2p_3p_4 + g_1p_2p_3p_4 + C_0p_1p_2p_3p_4$$

$$= g_4 + p_4(g_3 + (g_2 + g_1p_2)p_3) + C_0((p_1p_2)p_3)p_4.$$

The structure of the above expression is shown in Fig. 4.

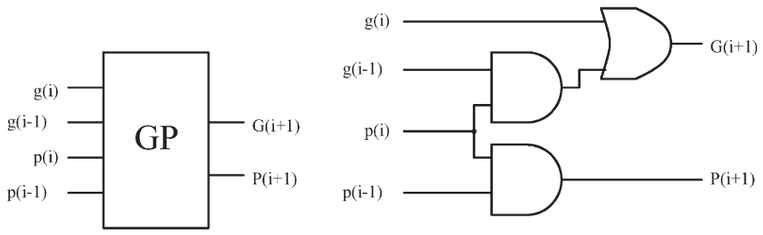

FIGURE 3 The GP module.

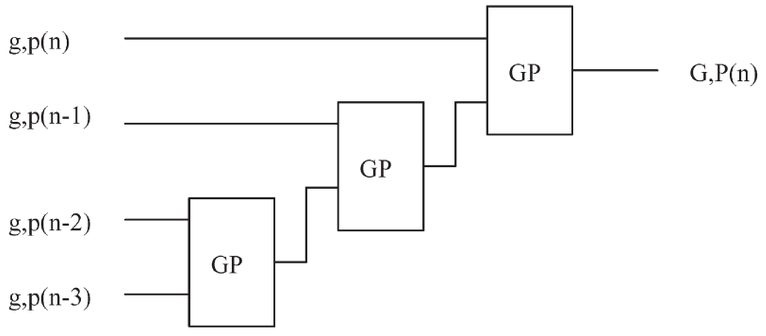

FIGURE 4 The 4-b GP module.

From the above carry out expression, we can also have alternate GP expressions for four bits as

$$G_4 = g_4 + g_3p_4 + g_2p_3p_4 + g_1p_2p_3p_4$$
$$= (g_4 + g_3p_4) + p_3p_4(g_2 + g_1p_2)$$
$$P_4 = p_1p_2p_3p_4 = (p_1p_2)(p_3p_4).$$

From those equations, we found out (p_3p_4) and $(g_4 + g_3p_4)$ are the generate and propagate outputs of a 2-b GP module, which inputs are fed from bit-3 and bit-4 gp modules' outputs. Similarly, (p_1p_2) and $(g_2 + g_1p_2)$ are the outputs of a 2-b GP module, which inputs are fed from bit-1 and bit-2 gp modules' outputs. The alternate tree structure for a 4-b GP is shown in Fig. 5. It is obvious that the critical path of a 4-b GP in Fig. 5 is shorter than that in Fig. 4. Therefore, our timing driven convergent tree adder design adopts the timing optimized 4-b GP in Fig. 5.

8-b GP Functional Tree

The 8-b functional tree represented will be modified to reduce racing effects as well as fanouts. The design shown has a combination of gp and GP modules to produce the generate and propagate bits for an 8-b addition function. The 8-b carry equation is

$$C_8 = G_8 + C_0P_8.$$

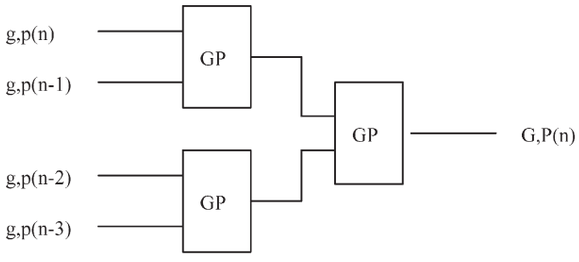

FIGURE 5 Timing optimized 4-b GP module.

By expanding the equation it takes on the following form

$$G_8 = g_8 + g_7p_8 + g_6p_8p_7 + g_5p_8p_7p_6 + g_4p_8p_7p_6p_5$$
$$+ g_3p_8p_7p_6p_5p_4 + g_2p_8p_7p_6p_5p_4p_3$$
$$+ g_1p_8p_7p_6p_5p_4p_3p_2$$
$$= (g_8 + g_7p_8 + g_6p_8p_7 + g_5p_8p_7p_6) + p_8p_7p_6p_5(g_4$$
$$+ g_3p_4 + g_2p_4p_3 + g_1p_4p_3p_2)$$

The propagate bit can be expressed as

$$P_8 = (p_8p_7p_6p_5)(p_4p_3p_2p_1)$$

The solutions for the generate and propagate bits produce a C-testable tree that can be seen in Fig. 6. These equations will be combined to optimize the design and change the tree model.

Carry Module

The carry module has the purpose of propagating the carry out through the carry-lookahead adder by combining the signals of 8-b functional trees with the carry out of the previous 8-b functional tree. The equation that characterizes the carry module is

$$C_n = G_n + C_{\text{in}}P_n.$$

The logic diagram can be seen in Fig. 7.

Convergent Tree Adder Design

Combining the above described modules, we reach the first design of 56-b TCTA that was synthesized, studied and tested. The 56-b TCTA is shown in Fig. 8. The design for this adder makes it fully testable. However, the adder has a significant problem, the output at the carry 16 module, C_{16}, has 5 fanouts. This will severely affect the delays of the circuit; furthermore, the high fanout produces very unbalanced circuits that will make delays vary significantly from one output bit to the next. Propagation differences can cause racing effects and

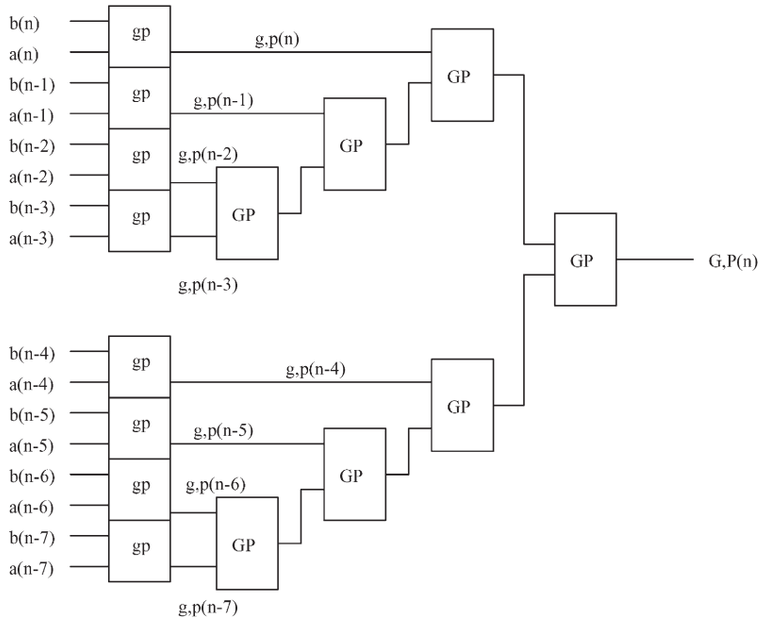

FIGURE 6 An 8-b GP functional tree.

produce incorrect outputs. In Fig. 8, since C_{24}, C_{32}, C_{40}, and C_{48} use the same carry value from C_{16}, the tree level will increase due to a higher number of partitioned inputs, resulting in a high fanout and unequal propagation delay path from inputs to outputs. Notice the critical path as the bold line starts in input bit 17.

TIMING DRIVEN TESTABLE CONVERGENT TREE ADDER (TDTCTA)

As shown earlier, the 56-b carry lookahead adder had several significant problems with the fanout of the 8-GP bit functional tree modules. We are going to start the optimization process by looking at the equations for the carry of the first four bits

$$C_4 = g_4 + C_3 p_4.$$

By expanding this equation we get

$$C_4 = (g_4 + p_4(g_3 + (g_2 + g_1 p_2)p_3)) + C_0(((p_1 p_2)p_3)p_4).$$

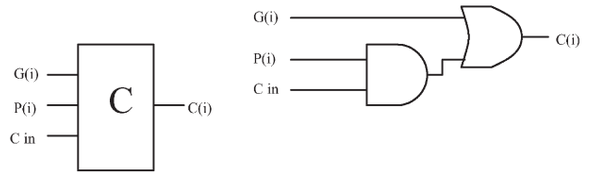

FIGURE 7 Carry module.

This equation can be regrouped and expressed as

$$C_4 = g_4 + g_3 p_4 + g_2 p_3 p_4 + g_1 p_2 p_3 p_4 + C_0 p_1 p_2 p_3 p_4.$$

This expression is more balanced and freer when we implement the logic diagram. We can now extend this notion to the 8-b GP functional tree module

$$C_8 = G_8 + C_0 P_8$$

where

$$\begin{aligned} G_8 &= g_8 + g_7 p_8 + g_6 p_8 p_7 + g_5 p_8 p_7 p_6 + g_4 p_8 p_7 p_6 p_5 \\ &\quad + g_3 p_8 p_7 p_6 p_5 p_4 + g_2 p_8 p_7 p_6 p_5 p_4 p_3 \\ &\quad + g_1 p_8 p_7 p_6 p_5 p_4 p_3 p_2 \\ &= g_8 + p_8(g_7 + p_7(g_6 + g_5 p_6)) + p_8 p_7 p_6 p_5(g_4 + p_4(g_3 \\ &\quad + p_3(g_2 + g_1 p_2))) \end{aligned}$$

and

$$P_8 = p_8 p_7 p_6 p_5 p_4 p_3 p_2 p_1 = (p_8(p_7(p_6 p_5)))\,(p_4(p_3(p_2 p_1))).$$

These two expressions can be optimized now by regrouping the equations as

$$\begin{aligned} G_8 &= (g_8 + g_7 p_8) + p_7 p_8(g_6 + g_5 p_6) + p_8 p_7 p_6 p_5((g_4 \\ &\quad + g_3 p_4) + p_3 p_4(g_2 + g_1 p_2)) \end{aligned}$$

and

$$P_8 = (p_8 p_7 p_6 p_5)\,(p_4 p_3 p_2 p_1).$$

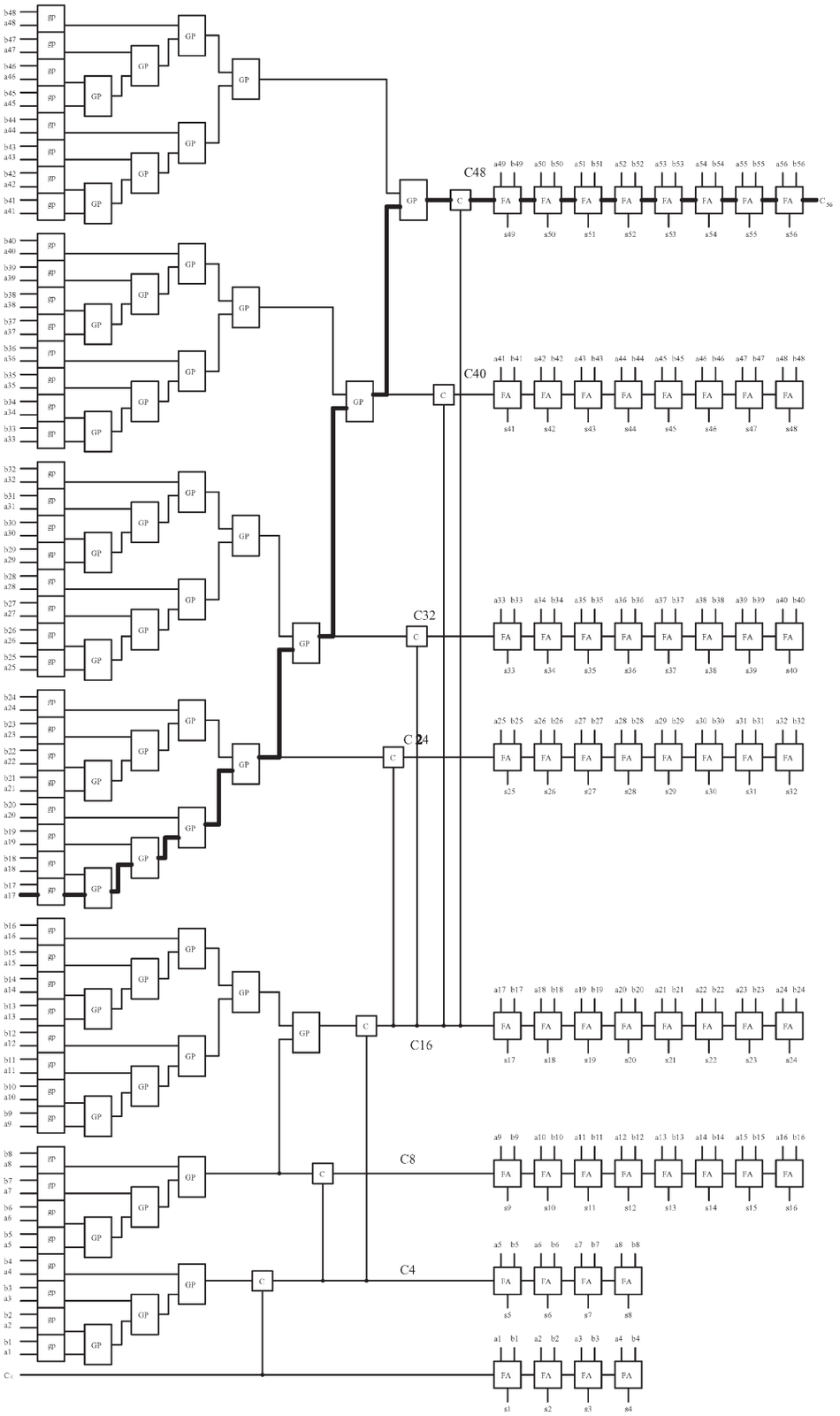

FIGURE 8 56-b TCTA.

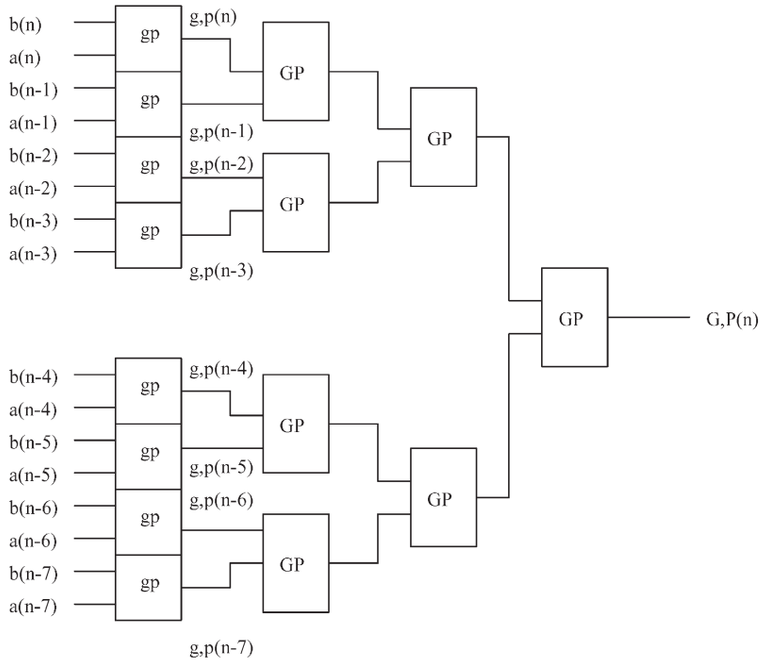

FIGURE 9 Optimized 8-b GP functional tree.

This expression can also be described using the fundamental carry operation as

$$(pg_4 fco\,(pg_3 fco\,(pg_1 fcopg_2)))$$

$$fco\,(pg_8 fco\,(pg_7 fco\,(pg_5 fcopg_6)))$$

and then it can be optimized to

$$((pg_1 fcopg_2) fco\,(pg_3 fcopg_4)) fco\,((pg_5 fcopg_6)$$

$$fco\,(pg_7 fcopg_8)).$$

The equations above give shape to the new optimized 8-b GP functional tree that can be seen in Fig. 9. Using these expressions we avoid the large fanout problem as well as the unequal propagation delay path from various inputs to respective outputs. In this new adder design, the inputs are partitioned every 8 bits to obtain a more balanced tree circuit. Now using the new 8-b functional tree module we get the new 56-b testable timing driven convergent tree adder circuit from Fig. 10. The comparison of module count between 56-b TCTA and the above mentioned timing optimized tree design, 56-b TDTCTA, is shown in Table I.

SYNTHESIS AND SIMULATION RESULTS

Both 56-b TCTA and TDTCTA were synthesized and tested in CMOS technology. Table II summarizes the results of critical path delay and silicon area.

HITEC [5] was used for fault grading of both 56-b TCTA and TDTCTA. The total equivalent single stuck-at faults of 56-b TCTA and TDTCTA are 2,672 and 2,600, and all faults are detected by 218 and 190 test vectors, respectively. Both 56-b TCTA and TDTCTA are testable designs with 100% fault coverage. The fault grading results are summarized in Table III.

CONCLUSION

Implementation of timing driven TCTAs has been presented. The architecture of this design has been studied and analyzed. This circuit was then optimized through the use of the associativity property of the fundamental carry operation. By losing a small amount of area, the performance of the circuit was significantly increased. For 56-b adder, the timing optimized design uses two more GP modules than the original design, and therefore, there is an increase in the area of the design. The addition of the two modules balances the circuit and increases the speed. This addition produces a significant improvement of timing by 6.37% over the original design while maintaining its full testability.

For future work in this area, other circuits will have to be studied and analyzed to take advantage of the timing optimization process. The reduction of the high fanout and the balancing of the circuit would produce faster circuits. The circuits will still maintain the testability properties using the associativity property of the fundamental carry operation.

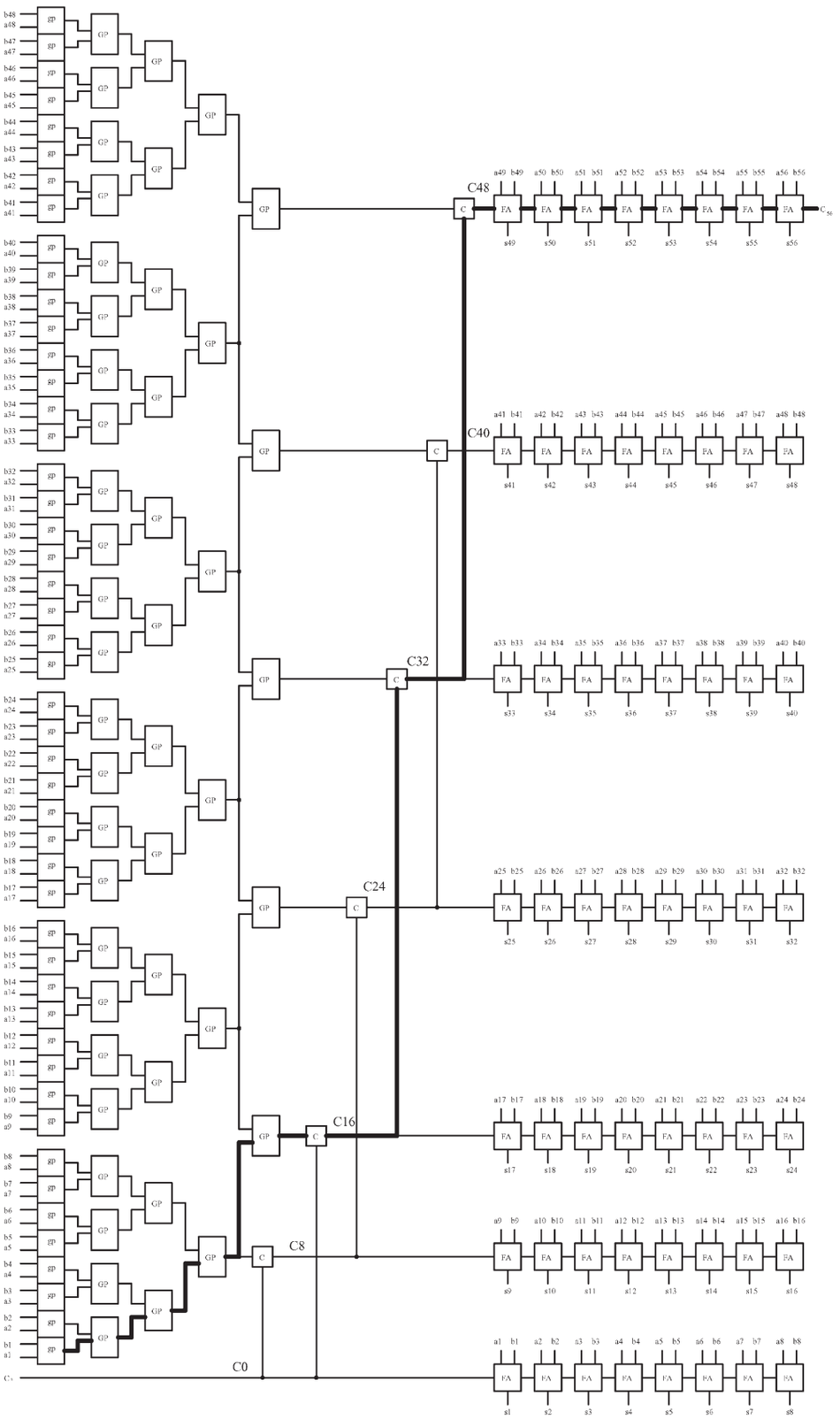

FIGURE 10 56-b TDTCTA.

TABLE I Comparison of module count

Module count	56-b TCTA	56-b TDTCTA
gp Modules	48	48
GP modules	45	47
Carry modules	7	6
Max. fan out	5	2

TABLE II Critical path delay of 56-b TDTCTA

Properties	56-b TCTA	56-b TDTCTA	% change
Area	$1924 \times 2654\,\lambda^2$	$1964 \times 2656\,\lambda^2$	-2.156
t_{phl}	$7.846\,ns$	$7.631\,ns$	2.74
t_{plh}	$11.61\,ns$	$10.585\,ns$	8.83
t_p	$9.728\,ns$	$9.108\,ns$	6.37

TABLE III Fault grading of 56-b TCTA and TDTCTA

Design	Faults	Coverage	Vectors
56-bit TCTA	2672	1.0000	218
56-bit TDTCTA	2600	1.0000	190

References

[1] Becker, B. (1988) "Efficient testing of optimal time adders", *IEEE Transactions on Computers* **37**, 1113–1120.

[2] Friedman, A.D. (1973) "Easily testable iterative systems", *IEEE Transactions on Computers* **22**, 1061–1064.

[3] Blanton, R.D. and Hayes, J.P. (1996) "Testability of convergent tree circuits", *IEEE Transactions on Computers* **45**(8), 950–963.

[4] Lombardi, F. and Sciuto, D. (1992) "Constant testability of combinational cellular tree structures", *Journal of Electronic Testing: Theory and Applications* **3**, 139–148.

[5] Niermann, T.M. and Patel, J.H. (1991) "Method for automatically generating test vectors for digital integrated circuits", *Proceedings on European Design Automation Conference (EDAC)*, 214–218.

Johnnie A. Huang received his M.S. degree from Electrical Engineering of Wright State University, Dayton, Ohio, summer 1997. Since then he has been with LSI Logic Corp., Milpitas, CA, as an engineer working in the area of logic and memory design and test, and fault tolerant memory design.

Chien-In Henry Chen received the B.S. degree from the National Taiwan University, the M.S. degree from the University of Iowa, and the Ph.D. degree from the University of Minnesota, all in electrical engineering. He is currently Professor of Electrical Engineering at Wright State University at Dayton, Ohio. From June 1999 to August 2000 he was on leave with Baynacre, Inc., developing silicon-accurate timing verification technologies. His research is primarily in the areas of digital and mixed-signal design synthesis and testing, timing analysis and optimization for VDSM IC, and IC chip design for signal processing, communication and networking. He has written over 70 publications in professional journal and conference proceedings. He is a technical committee member of 1995, 1996, 2000–2002 IEEE International ASIC SOC Conference. He was a plenary speaker of 1995 the 6th VLSI Design CAD Symposium.