Performance Analysis Of Fast Adders Using VHDL

R.P.P. Singh
ECE Department
Sri Sai College of Engg. & Tech.
Badhani (Pathankot), India
raminder_212003@rediffmail.com

Parveen Kumar
ECE Department
Beant College of Engg. & Tech.
Gurdaspur (Punjab), India
parveen.klair@gmail.com

Balwinder Singh
VLSI-ES Division, Centre for Development & Advanced Computing (CDAC), Mohali
balwinder_cdacmohali@yahoo.com

Abstract—**This paper presents performance analysis of different Fast Adders. The comparison is done on the basis of three performance parameters i.e. Area, Speed and Power consumption. Further, we present a design methodology of hybrid carry lookahead/carry skip adders (CLSKAs). This modified carry skip adder is modeled by using both fix and variable block size. In conventional carry skip adder, each block consists of ripple carry adder and skip logic is used after each block to generate carry for next block. The speed of operation depends on carry propagation from previous block to next block. In CLSKAs, we use carry lookahead logic in each block to generate carry for next block. The modified carry skip adders presented in this paper provides better speed and power consumption as compare to conventional carry skip adder and other adders like ripple carry adder, carry lookahead adder, Ling adder, carry select adder. The modified carry skip adders with fix block require few more CLB's because of Carry lookahead logic, whereas with variable block scheme, area optimization is achieved.**

***Keywords*— Adder, Ripple Carry Adder, Look Ahead Carry Adder, VHDL Simulation**

I. INTRODUCTION

Adders are most commonly used in various electronic applications e.g. Digital signal processing in which adders are used to perform various algorithms like FIR, IIR etc. In past, the major challenge for VLSI designer is to reduce area of chip by using efficient optimization techniques. Then the next phase is to increase the speed of operation to achieve fast calculations like, in today's microprocessors millions of instructions are performed per second. Speed of operation is one of the major constraints in designing DSP processors. Now, as most of today's commercial electronic products are portable like Mobile, Laptops etc. that require more battery back up. Therefore, lot of research is going on to reduce power consumption. Therefore, there are three performance parameters on which a VLSI designer has to optimize their design i.e. Area, Speed and Power. It is very difficult to achieve all constraints for particular design, therefore depending on demand or application some compromise between constraints has to be made.

II PRIOR WORK

In 1990, modified Carry-Skip Adders was presented by reducing first block delay with carry-lookahead adders using multidimensional dynamic programming [12]. In 1996, transistor-level simulation of the adders using HSPICE is done for area, time and power trade-off between different fast adders [6]. 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 [7]. In 2007, a new 54×54-bit multiplier is designed using high-speed carry-look-ahead adder and has been fabricated by CMOS technology [4]. In 2008, low power multipliers based on new hybrid full adders is presented [5]. In 2008, *Hasan Krad et al* worked on the performance analysis for a 32-Bit Multiplier with a Carry-Look-Ahead Adder and a 32-bit Multiplier with a Ripple Adder using VHDL [3].

III FAST PARALLEL ADDERS

A. *Ripple Carry Adder (RCA)*

Ripple carry adder can be designed by cascading full adder in series i.e. carry from previous full adder is connected as input carry for the next stage. Full adder is a basic building block of Ripple carry adder. Therefore, to design n-bit parallel adder, it requires n full adders. In our design we use 16 full adders to design a 16-bit parallel adder. The major limitation of Ripple carry adder is that as the bit length goes on increases, delay also increases. Therefore, Ripple carry adder is not suitable if large number bits are to be added.

The major element that causes delay is carry propagation, therefore it is important to calculate carry delay from input to output. For n-bit Ripple carry adder, Delay for carry can be calculated as: -

$$T_C = T_{FA}((x0, y0) \text{ to } c0) + (n-2) * T_{FA}(\text{cin to cout})) + T_{FA}(\text{cin to sout } (n-1)) \quad (1)$$

978-0-7695-3845-7/09 $26.00 © 2009 IEEE
DOI 10.1109/ARTCom.2009.132

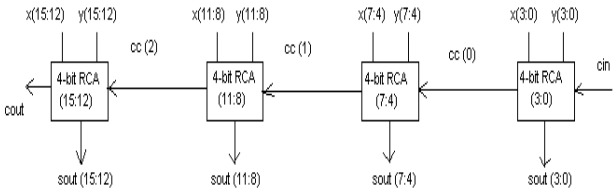

FIGURE 1: Schematic block diagram of 16-bit ripple carry adder .

Where T_{FA} (input to output) represent the delay of full adder on the path between it's specified input and output.

A. *Condition Carry Adder (CCA)*

Condition carry adder is based on the principle shown in figure.2. In this case instead of computing sum and carry directly by using full adder, it computes sum and carry depending upon status of previous carry i.e.

1. If ci = 0 then
   Si = ai xor bi & c_{i+1} = ai and bi (2)
2. If ci = 1 then
   Si = ai xnor bi & c_{i+1} = ai or bi (3)

A logarithmic time condition sum adder results if we proceed to the extreme of having single-bit adder at the very top. Thus taking the delay of 2-to-1 multiplexer as our units, the delay of a condition sum adder are characterized by the following recurrences:

$$T(k) = \text{Log}_2 k + T(1) \quad (4)$$

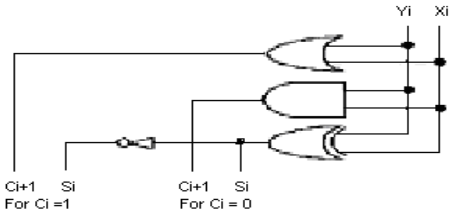

FIGURE 2: Single-bit position of Condition sum adder [1].

Where T (1) is the delay of circuit shown in figure.2, which is used at the top to derive the sum and carry bits with a carry-in of 0 and 1. An exact analysis leads to a comparable count for the number of single-bit multiplexers needed in a condition-sum adder. Assuming that K is a power of 2, the required number of multiplexers for a k-bit adder is:

$$(k - 1)(\text{Log}_2 k + 1) \quad (5)$$

B. *Lookahead Carry Adder (CLA)*

Lookahead carry algorithm speed up the operation to perform addition, because in this algorithm carry for the next stages is calculated in advance based on input signals. If X and Y are two inputs, "ci" is initial carry, "sout" and "cout" are output sum and carry respectively, then Boolean expression for calculating next carry and addition is:

Pi = xi xor yi --- Carry Propagation (6)

Gi = xi and yi --- Carry Generate (7)

C_{i+1} = Gi or (Pi and C_i)--Next Carry (8)

Figure.3 shows 16-bit look ahead carry adder that consists of four 4-bit adders each 4-bit look ahead carry generator. The latency through this 16-bit adder consists of the time required for:

1. Producing the G and P for individual bit positions (1 gate level).
2. Producing the G and P signals for 4-bit blocks (2 gate levels).

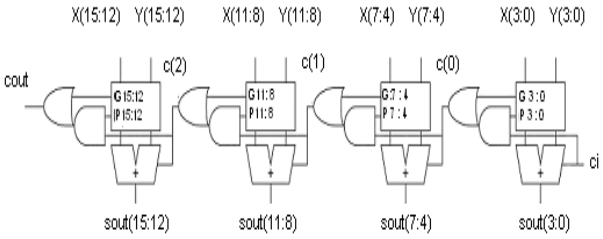

FIGURE 3: Schematic block diagram of 16-bit carry look ahead adder divided into 4 blocks [2].

3. Predicting the carry-in signals c4, c8 and c12 for the blocks (2 gate levels).
4. Predicting the internal carries within each 4-bit block (2 gate levels).
5. Computing the sum bits (2 gate levels).

Thus the total latency for 16-bit adder is a 9-gate level that is very less as compared to 16-bit ripple carry adder that is 32 gate levels. Thus, the delay of N-bit carry look-ahead adder based on 4-bit look-ahead blocks is:

$$T_{CLA} = 4 \log_4 N + 1 \text{ gate levels} \quad (9)$$

C. *Ling Adder Design*

The Ling Adder is a type of Look-Ahead Adder with a slight modification that results in significant hardware saving. Ling's modification consists of propagating $h_i = c_i + c_{i-1}$ instead of c_i. This results in reduction of number of gates required for implementation. Therefore, the Boolean expression for calculating next carry and sum are:

$$C_i = h_i (G_{i-1} + P_{i-1}) \quad (10)$$

$$S_i = P_i \text{ xor } h_i (G_{i-1} + P_{i-1}) \quad (11)$$

D. *Manchester Carry Chain Adder*

Similar to Ling's adder design, Manchester adder is also a type of Carry look-ahead adder. Manchester adder gives slight modification in calculating next carry to be propagated i.e. instead of using Boolean expression $C_{i+1} = G_i + C_i P_i$ to calculate next carry Manchester carry adder uses expression:

$$C_{i+1} = G_i + C_i t_i \quad (12)$$

$$t_i = X_i + Y_i \quad (13)$$

Thus, we can say that carry recurrence can be written in terms of t_i instead of P_i, which leads to slightly faster adder because in binary addition, t_i is easier to produce than P_i (OR instead of XOR). Table1 and figure.4 shows comparative study of 16-bit Manchester carry adder with

Carry look-ahead adder in terms of delay. CPLD XC9572 is used for implementation.

TABLE1: 16-BIT CARRY LOOK-AHEAD AND MANCHESTER ADDER DESIGN COMPARISON FOR AREA AND DELAY

Adders	Macrocells Used	Pad Delay	Delay (ns)	Delay (ns)
			Sum	Carry
CLA	31/72 (44%)	52	52	51
MCA	35/72 (49%)	53	51	33

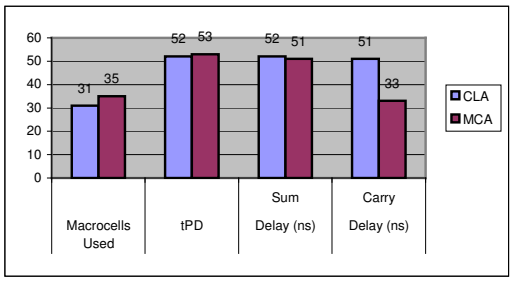

FIGURE 4: Graphical representation of Simulated results

E. Conventional Carry Skip Adder(CCSKA)

In case of N-bit Ripple carry adder, carry has to propagate through all N stages, which results in large delay in performing binary addition. In contrast, it is possible to skip carry over group of n-bits in case of Carry Skip Adder.

This results in less delay as compare 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:

$$T_{CSKA} = (b-1)+0.5+(N/b-2)+(b-1) \quad (14)$$

$$= 2b + N/b - 3.5 \text{ Stages} \quad (15)$$

Therefore, 8.5 Stages are required for 16-bit Carry Skip Adder that results in latency of 17 gate levels as compare to 32 gate levels required for 16-bit Ripple carry adder. Figure.5 shows 16-bit Carry-Skip Adder divided into 4 blocks and each block are a 4-bit Ripple Carry Adder.

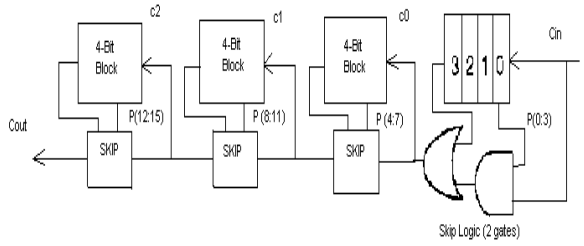

FIGURE 5: 16-Bit Carry-Skip Adder [1].

From equation (xi), it is observed that latency is directly proportional to block width i.e. if we decrease block width, then more number of blocks are required to make N bit adder which results in increase in latency because more number of skips are required between stages and vice-versa. Figure6 and table2 shows the simulated results in terms of delay achieved for 16-bit Carry skip adder-using width of 4 blocks, 2 blocks and 8 blocks.

TABLE2: 16-BIT CARRY SKIP ADDER USING 4 BLOCKS, 2 BLOCKS AND 8 BLOCKS.

Carry Skip Adder	CLB'S	Delay (ns)	Delay (ns)	Power (mW)	Power (mW)
16-bit		Sum	Carry	Dynamic	Static
4 blocks	23	23.2	23.1	16.3	219.71
2 blocks	26	20.8	21.7	16.1	219.69
8 blocks	24	26.6	24.5	14.8	219.54

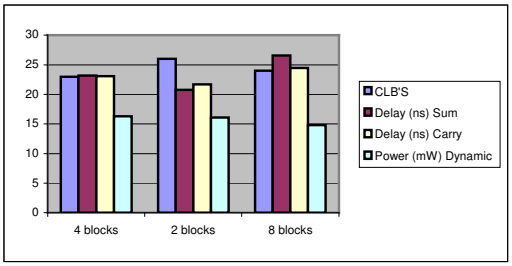

FIGURE 6: Simulated Results of Table1

F. Modified Carry Skip Adder (CLSKAs)

In conventional carry skip adder, each block consists of ripple carry adder and skip logic is used after each block to generate carry for next block. The speed of operation depends on carry propagation from previous block to next block. In CLSKAs, we use carry lookahead scheme in each block to generate carry for next block. This result in better performance in terms of speed as look ahead carry adder is faster than ripple carry adder. Figure7 shows modified CLSKA with fixed block size i.e. 4-bit each.

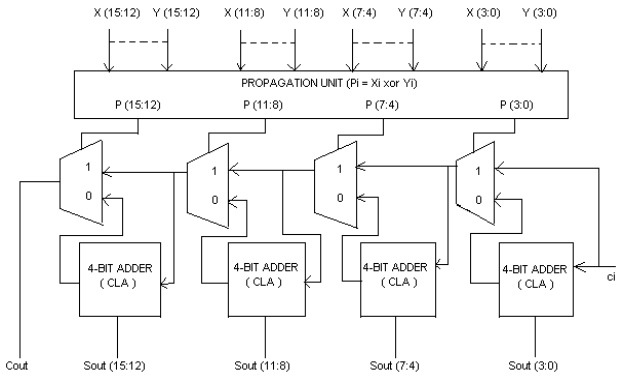

FIGURE 7: A 16-bit hybrid CLSKA with 4-bit block adders.

Now, next is another concept of designing adder by using variable block size [12]. Figure8 shows CLSKA model in which size of block is variable. Here, we use two full adders

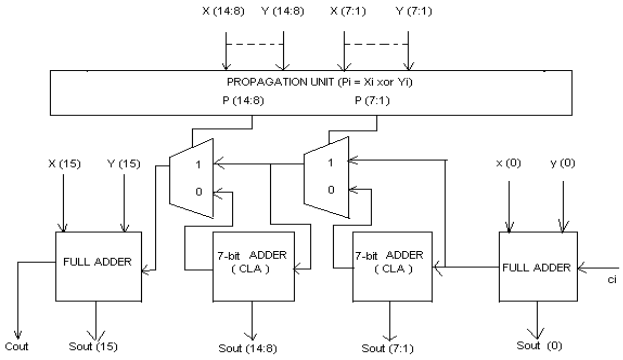

FIGURE 8: A 16-bit hybrid CLSKA with variable block size.

G. Carry Select Adder (CSA)

Carry select adder is based on the principle to calculate sum that is based on assuming input carry from previous stage. One adder calculates the sum assuming input carry of 0 while the other calculates the sum assuming input carry of 1. Then, the actual carry triggers a multiplexer that selects the appropriate sum [2]. Fig.9 shows the schematic block diagram of 16-bit Carry select adder consists of 4-blocks each of 4-bit Look ahead carry adder [11]. Carry output of each block is fed into next block as input carry.

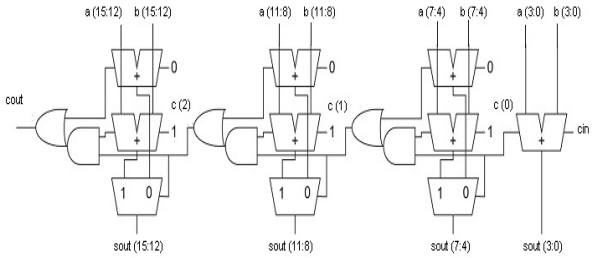

FIGURE 9: Schematic block diagram of 16-bit Carry select adder [2].

I. CARRY SAVE ADDER

Basically, carry save adder is used to compute sum of three or more n-bit binary numbers. Carry save adder is same as a full adder. But as shown in figure.10, here we are computing sum of two 16-bit binary numbers, so we take 16 half adders at first stage instead of using 16 full adders. Therefore, carry save unit consists of 16 half adders, each of which computes single sum and carry bit based only on the corresponding bits of the two input numbers. Let x and y are two 16 bit numbers and produces partial sum and carry as s and c as shown in table3:

$$Si = xi \text{ xor } yi \quad (16)$$

$$Ci = xi \text{ and } yi \quad (17)$$

The final addition is then computed as:

1. Shifting the carry sequence C left by one place.
2. Placing a 0 to the front (MSB) of the partial sum sequence S.
3. Finally, a ripple carry adder is used to add these two together and computing the resulting sum.

TABLE 3: CSA COMPUTATION

X:	1 0 0 1 1
Y: +	1 1 0 0 1
S:	0 1 0 1 0
C:	1 0 0 0 1
Sum:	1 0 1 1 0 0

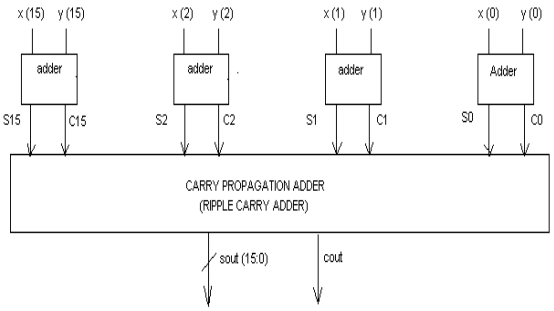

FIGURE 10: Computation flow of CSA

When adding together two numbers, using a half adder followed by a ripple carry adder is faster than using two ripple carry adders. This is because a ripple carry adder cannot compute a sum bit without waiting for the previous carry bit to be produced, and thus has a delay equal to that of *n* full adders. A carry-save adder, however, produces all of its output values in parallel, thus the total computation time for a carry-save adder is less than ripple carry adders.

IV. RESULTS AND DISCUSSION

To demonstrate the performance of modified carry skip adder we compare it with other adders like ripple carry adder, lookahead carry adder Ling adder, carry select adder, carry save adder. We design all adders using VHDL (Very High Speed Integration Hardware Description Language) for 16-bit unsigned data. To get power, delay and area report, we use XILINX 9.1 i as synthesis tool and Modelsim XE III 6.2g for simulation. FPGA-Spartan III is used for implementation. The modified Carry Skip adder architecture (hybrid carry lookahead/carry skip adders) with fix block size (four blocks of 4-bit each) gives better result than other adders in terms of Speed (Delay=15.0 ns) but require more Area (29 CLB's) and power consumption (Dynamic Power=13.9mW) whereas with variable block

size architecture, power consumption and area (26 CLB's) also improve.

TABLE 4: AREA, POWER AND DELAY REPORT FOR ADDERS

ADDERS (With Fix Block Size)	CLB'S	Delay (ns)	Delay (ns)	Power (mW)	Power (mW)
Block Size = 4-Bit		*Sum*	*Carry*	*Dynamic*	*Static*
Ripple Carry	24	24.1	23.5	7.6	218.7
Conditional carry	24	23.9	23.3	7.6	218.7
Look Ahead	26	20.9	20.6	13.3	219.4
Lings	20	23.3	23.1	13.3	219.4
Carry Select	27	17.1	17.7	10.1	219
Carry Save	29	23.1	22.8	9	218.9
Conventional Carry Skip Adder	23	23.2	23.1	16.3	219.7
Modified carry Skip Adder	29	15	14.6	13.9	219.4

TABLE 5: AREA, POWER AND DELAY REPORT FOR ADDERS

ADDERS (With Variable Block Size)	CLB'S	Delay (ns)	Delay (ns)	Power (mW)	Power (mW)
2 X 7-bit + 2 X 1-bit		*Sum*	*Carry*	*Dynamic*	*Static*
Ripple Carry	24	22.9	22.5	7.9	218.7
Look Ahead	24	25.3	24.3	7.6	218.7
Conventional Carry Skip Adder	26	22.3	20.5	14.9	219.5
Modified carry Skip Adder	26	16.8	12.2	13.8	219.5

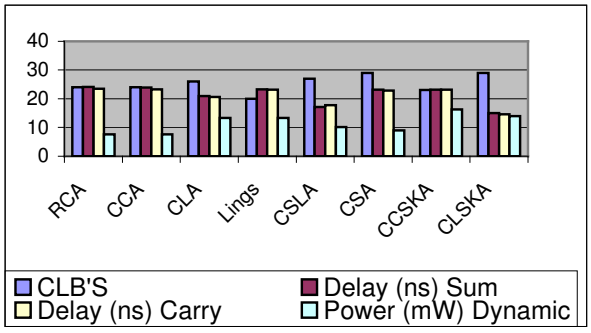

FIGURE 11: Performance comparison of adders

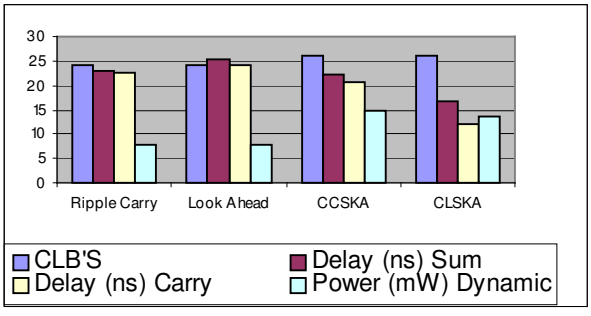

FIGURE 12: Performance comparison of adders

V. CONCLUSION

From the results and discussion, it is observed that there are trade-offs between performance parameters i.e. Area, Power and Delay. For designing delay efficient adder, we have proposed a hybrid carry lookahead/carry skip adders in which carry lookahead logic is used instead of ripple carry adder in each block to generate output sum and carry bit for next block. This result in fast operation but at the cost of few more CLB's due to carry lookahead logic.

VI. REFERENCES

[1] B. Parhami, Computer Arithmetic, Algorithm and Hardware Design, Oxford University Press, New York, pp. 91-119, 2000.

[2] Pong P. Chu "RTL Hardware Design Using VHDL: coding for Efficiency, Portability and Scalability" Wiley-IEEE Press, New Jercy, 2006

[3] Hasan Krad and Aws Yousif Al-Taie, "Performance Analysis of a 32-Bit Multiplier with a Carry-Look-Ahead Adder and a 32-bit Multiplier with a Ripple Adder using VHDL", Journal of Computer Science 4 (4): 305-308, 2008

[4] Asadi, P. and K. Navi "A novel high-speed 54-54-bit multiplier", Am. J. Applied Sci., 4 (9): 666-672, 2007

[5] Z. Abid, H. El-Razouk and D.A. El-Dib, "Low power multipliers based on new hybrid full adders", Microelectronics Journal, Volume 39, Issue 12, Pages 1509-1515, 2008

[6] Nagendra, C.; Irwin, M.J.; Owens, R.M.,"Area-time-power tradeoffs in parallel adders", Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on Volume 43, Issue 10, Page(s): 689 – 702, 1996

[7] Wang, Y.; Pai, C.; Song, X., "The design of hybrid carry look-ahead/carry-select adders, Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on Volume 49, Page(s): 16-24, 2002.

[8] May Phyo Thwal, Khin Htay Kyi, and Kyaw Swar Soe, "Implementation of Adder-Subtracter design with VerilogHDL", International Journal of Electronics, Circuits and Systems Volume 2 number 3, 2008

[9] Min Cha and Earl E. Swartzlander, Jr, "Modified Carry Skip Adder for reducing first block delay", Proc. 43rd IEEE Midwest Symp. on Circuits and Systems, Lansing MI, Page(s): 346-348, 2000

[10] Behnam Amelifard, Farzan Fallah, Massoud Pedram, "Closing the gap between Carry Select Adder and Ripple Carry Adder: A new class of Low-power and High-performance Adders", Proceedings of the Sixth International Symposium on Quality Electronic Design (ISQED'05) , 2005

[11] Jin-Fu Li, Jiunn-Der Yu, Yu-Jen Huang, "A Design Methodology for Hybrid Carry-Lookahead/Carry-Select Adders with Reconfigurability", IEEE, 2005

[12] Pak K. Chan, et al, Delay Optimization of Carry-Skip Adders and Block Carry-Lookahead Adders Using Multidimensional Dynamic Programming, IEEE Transactions on Computers, vol. 41, No. 8, pp. 920-93, 1992