# Optimization and Speed Improvement Analysis of Carry-Lookahead Adder Structure

Brian D. Lee Vojin G. Oklobdzija*

Department of Electrical Engineering and Computer Science
University of California, Berkeley

## Abstract

The delay characteristics of carry-lookahead (CLA) adders are examined with respect to a delay model that accounts for fan-in and fan-out dependencies. Though CLA structures are considered among the fastest topologies for performing addition, they have also been characterized as providing marginal speed improvement for the amount of hardware invested. This analysis shows that this inefficiency can be explained by the suboptimal nature of common CLA implementations. Simulation results show that the CLA structures in wide use can be improved by varying the block sizes and the number of levels within each adder. Examples of optimal CLA structures are given and heuristic methods for finding these structures are presented.

## Introduction

Analysis of carry-lookahead adders is important in the design of high performance machines. In these designs, processor speed is a primary concern and carry-lookahead structures are often used because their delay times exhibit log dependence on the size of the adder and they are considered among the fastest circuit topologies for performing addition. However, adder comparisons [5, 4] have ranked CLA structures low on effective hardware utilization and this apparent inefficiency raises concerns over the optimality of current CLA implementations. Simulation results from this research show that the commonly used CLA structures can be improved by varying block sizes and levels within the adder.

Typical CLA implementations are made of lookahead units of relatively fixed sizes. This artificial constraint produces slack in the circuit and results in poor hardware utilization. The strategy of varying group sizes to reduce slack and improve performance is a promising idea and has been used successfully on carry-skip adders [3, 6]. A natural extension of this strategy is to also vary the number of lookahead levels. Since an accurate measure of the available slack is required to effectively implement these methods, this work uses a delay model that accounts for fan-in and fan-out dependencies. The parameter values used in the model are based on industrial data.

A simulation program has been written to compare different CLA structures. Preliminary data on varying block sizes was obtained through exhaustive search. Based on this data and an analysis of delay and slack in the CLA scheme, heuristics were chosen to find structures with completely variable block sizes and levels. The structures found by these heuristics are faster than more constrained topologies.

*Currently with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598

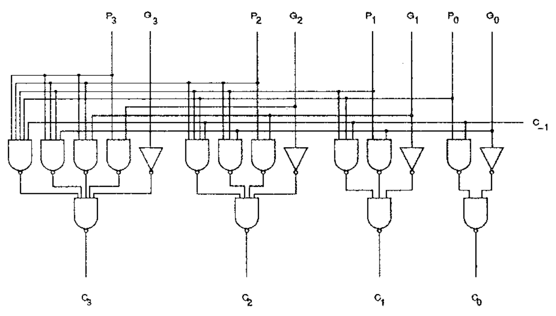

Figure 1: A 4-bit carry-lookahead unit with 2-level $C_i$ logic

## Carry Lookahead Structure

The simulation results in this paper are based on carry-lookahead adders that implement full lookahead. A description of the basic organization of carry-lookahead adders can be found in references such as [1]. Each adder consists of three main components – the propagate and generate generation circuitry, the carry-lookahead network, and the sum generation circuitry. This work concerns varying the sizes of circuit blocks and the number of levels in the carry-lookahead network to optimize adder delay.

Given a $n$-bit adder with inputs, $A$ and $B$, the logic equations for producing the initial propagate and generate signals and the final sum signals are

$$\begin{aligned} P_i &= A_i \oplus B_i \\ G_i &= A_i B_i \end{aligned}$$

and

$$S_i = P_i \oplus C_{i-1}$$

for $0 \leq i \leq n-1$. The simulations assume that $P_i$ and $S_i$ are produced by monolithic XOR gates instead of two levels of NAND gates and the sum XOR gate is assumed to have a fan-out of one.

The carry-lookahead network in a full carry-lookahead adder consists of a tree of block carry-lookahead (BCLA) units rooted at a single carry-lookahead (CLA) unit. Two different implementations of BCLA/CLA units are analyzed. Their performance differences are discussed later.

The first implementation generates carry signals in two levels of logic. Four-bit versions of these circuits are shown in Figures 1 and 2. A $k$-bit carry-lookahead unit of this type generates

$$\begin{aligned} C_j &= G_j + G_{j-1}P_j + G_{j-2}P_{j-1}P_j \\ &\quad + \cdots + G_0 P_1 \cdots P_j + C_{-1} P_0 \cdots P_j \end{aligned}$$

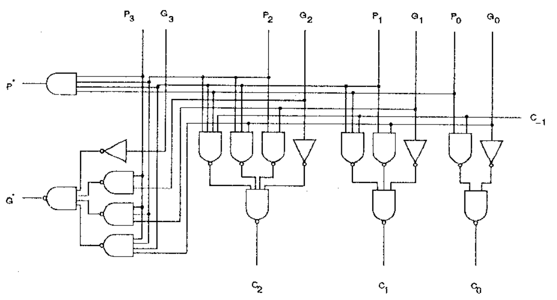

Figure 2: A 4-bit block carry-lookahead with 2-level $C_i$ logic

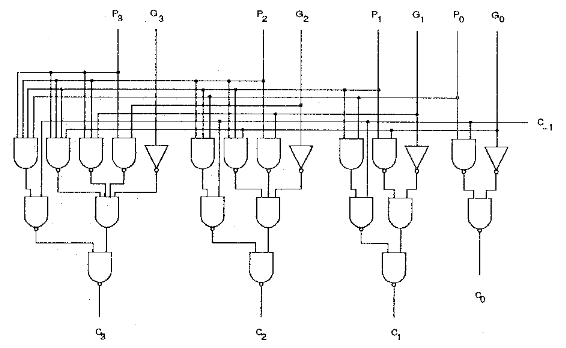

Figure 3: A 4-bit carry-lookahead unit with 3-level $C_i$ logic

where $0 \leq j \leq k-1$ and a $k$-bit block carry-lookahead unit of this type generates

$$\begin{aligned} P^* &= P_0P_1\cdots P_{k-1} \\ G^* &= G_{k-1}+G_{k-2}P_{k-1}+\cdots+G_0P_1\cdots P_{k-1} \\ C_j &= G_j+G_{j-1}P_j+G_{j-2}P_{j-1}P_j \\ &\quad +\cdots+G_0P_1\cdots P_j+C_{-1}P_0\cdots P_j \end{aligned}$$

where $0 \leq j \leq k-2$.

The second implementation generates carry signals in three levels of logic. Four-bit versions of these circuits are shown in Figures 3 and 4. A $k$-bit carry-lookahead unit of this type generates

$$C_j = G_j^* + C_{-1}P_j^*$$

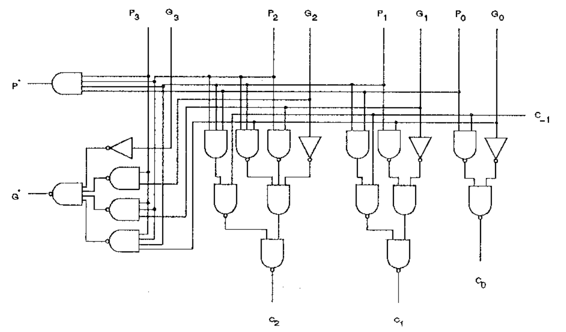

Figure 4: A 4-bit block carry-lookahead unit with 3-level $C_i$ logic

where $0 \leq j \leq k-1$ and a $k$-bit block carry-lookahead unit of this type generates

$$\begin{aligned} P^* &= P_{k-1}^* \\ G^* &= G_{k-1}^* \\ C_j &= G_j^* + C_{-1}P_j^* \end{aligned}$$

where $0 \leq j \leq k-2$. For both circuit blocks,

$$G_j^* = G_j+G_{j-1}P_j+G_{j-2}P_{j-1}P_j+\cdots+G_0P_1\cdots P_j$$

and

$$P_j^* = P_0P_1\cdots P_j.$$

The simulations also assume that $C_{-1}$ and all $A_i$, $B_i$ are latched and available at time $t=0$. Fan-out loading of $A_i$, $B_i$ and $C_{-1}$ is ignored and the adder delay is calculated as the time required to generate the slowest signal from among $S_i$ and $C_{n-1}$.

## Carry Lookahead Optimization

The basic goal of this research is to show how to optimize CLA structures by varying group sizes and lookahead levels. The purpose of these operations is to exploit the delay differences that represent underutilized time. Early signals can be delayed by modifying the network to make the signals the result of more lookahead computation. Since this allows the addition of larger operands in the same amount of time, slack reduction corresponds to adder structure optimization. The optimization requires a delay model that accurately measures slack in the circuit, an examination of the delay characteristics of the circuit blocks, and an analysis of critical delay paths to identify slack in the circuit. These requirements lead to a heuristic method for determining optimal CLA structures.

### Delay Model

Logic gate delays are modeled as

$$\begin{aligned} \text{delay} &= f(\text{fi},\text{fo}) \\ &= A + B\cdot\text{fi} + (D + E\cdot\text{fi})\cdot\text{fo}\,. \end{aligned}$$

Simpler delay models that use unit gate delays are inadequate because they do not reveal all the slack in the circuit.

The simulations use the following delay functions:

$$\begin{aligned} t_{NAND} &= 0.1058+0.1175\text{fi}+(0.0825+0.0148\text{fi})\text{fo} \\ t_{AND} &= 0.2825+0.1675\text{fi}+(0.0911+0.0037\text{fi})\text{fo} \\ t_{INV} &= 0.265+0.1016\text{fo} \\ t_{XOR} &= 0.945+0.05645\text{fo}. \end{aligned}$$

The constants in these functions are based on LSI Logic Corporation's 1.5 Micron Compacted Array ™ Technology [2]. To limit complexity, the models assume that all logic gates are single, possibly large, gate structures rather than multiple levels of smaller gates.

### BCLA/CLA Delay Characteristics

An understanding of BCLA/CLA delay characteristics is important for finding opportunities to reduce slack in CLA structures In particular, the fan-in and fan-out properties of the circuits must be analyzed.

The input and output loading on the propagate and generate signals of a BCLA can be derived by induction on the circuits of Figures 1, 2, 3, and 4. By inspection, the following results are obtained. Given a $k$-bit BCLA unit connected to the $jth$ input of a $m$-bit BCLA unit

- $G^*$ of the $k$-bit BCLA unit has fan-out $(m - j)$
- $G^*$ of the $k$-bit BCLA unit has worst case fan-in $k$
  Specifically, in the two-level implementation of $G^*$, the first level NAND gate $i$ associated with input $G_i$ has fan-in $k - i$.
- $P^*$ of the $k$-bit BCLA unit has fan-in $k$
- $P^*$ of the $k$-bit BCLA unit has fan-out $(m - j)(j + 1)$

Examples of these relationships are shown graphically in Figures 5 and 6. The analysis for a BCLA unit feeding into a CLA unit is similar

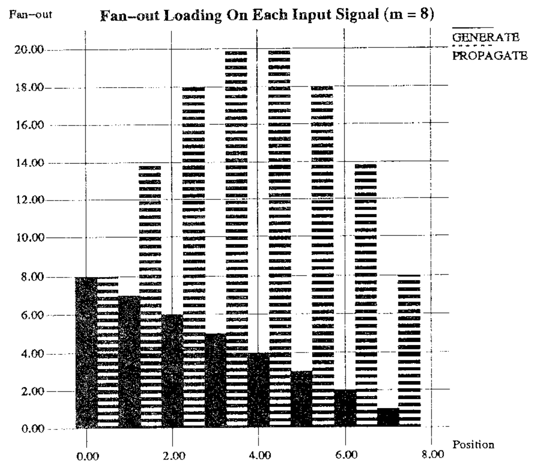

Figure 5: Fan-out loading on input signals of an 8-bit BCLA/CLA unit

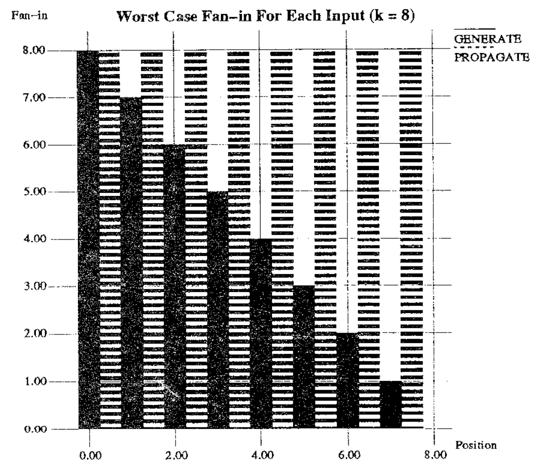

Figure 6: Worst case fan-in gates for the inputs of an 8-bit BCLA unit

and gives the same results.

The loading on carry signals may be derived by a similar analysis of the circuit diagrams. Each carry signal, $C_i$, of a BCLA/CLA unit is the $C_{-1}$ of BCLA units on previous levels. An example of this is shown in Figure 7. In particular, it connects to its $(i + 1)th$ fan-in unit, the

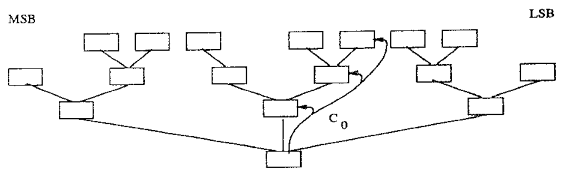

Figure 7: Fan-out path within the CLA network of $C_0$ of the final lookahead unit

zeroth fan-in unit of its $(i + 1)th$ fan-in unit, the zeroth fan-in unit of the the zeroth fan-in unit of its $(i + 1)th$ fan-in unit, etc. Each carry signal also connects to an XOR gate in the sum generation circuitry. A BCLA unit of size $k$ contributes $k - 1$ to the fan-out loading of its input $C_{-1}$ signal. In the two-level implementation of $C_i$, the worst case first-level NAND gate has fan-in $i + 2$. In the three-level implementation of $C_i$, $C_{-1}$ always connects to an input NAND gate with a fan-in of 2.

These fan-in and fan-out properties have an important effect on the critical delay analysis in BCLA/CLA units. Unlike typical analyses, the $P^*$ delay can not be neglected with respect to the $G^*$ delay. Even though computing $P^*$ requires one less level of logic than calculating $G^*$, the potentially high group propagate fan-out loading may place generation of $P^*$ on the critical path. The $P^*$ delay path should be compared to the $G_0$-to-$G^*$ delay path which contains the worst case fan-in gate.

Another important delay path is the carry propagation (assimilation) path. This is the path from $C_{-1}$ to some $C_i$ and is critical when $C_{-1}$ arrives much later than all $P_i$ and $G_i$. Assuming a $k$-bit BCLA unit, the worst case path for the two-level implementation of $C_i$ is from $C_{-1}$ to $C_{k-2}$. For the corresponding three-level implementation of $C_i$, the paths are equal for all $i$ because $C_{-1}$ feeds gates of constant fan-in. When $C_{-1}$-to-$C_i$ delays are a significant fraction of the total adder delay, the three-level implementation should produce faster adders than the two-level implementation because of its superior fan-in properties. This condition should hold for larger adders.

An issue related to BCLA delay is the relationship between group size and number of lookahead levels. Clearly, larger BCLA units have longer delays than smaller units. Also, adding more levels of BCLA units tends to increase delay because of the extra logic levels. However, at some size, implementing a single large BCLA unit as multiple levels of BCLA units is advantageous. Unfortunately, determining this breakpoint is difficult because the delay of a BCLA unit depends on the block sizes on the next level of lookahead and typically, those sizes are determined by the number and sizes of blocks on all previous levels.

## Critical Path Analysis

Identifying critical paths in CLA structures is important because the remaining non-critical delay paths represent opportunities for slack reduction. Standard CLA analyses assume that the critical path in the adder is always as shown in Figure 8. Unfortunately, the validity of this assumption is not guaranteed when variable group sizes and lookahead levels are allowed. However, the actual critical path will have an analogous form. The first part of the critical path is the delay to the generation of a carry signal in the CLA unit and the last part of the critical path is the propagation of this carry signal back through some subtree of the BCLA network. Furthermore, each subtree of the network has an analogous critical path. Opportunities to reduce slack can be found in each portion.

The delay to the CLA unit depends only on $G^*$ and $P^*$ delays. Most

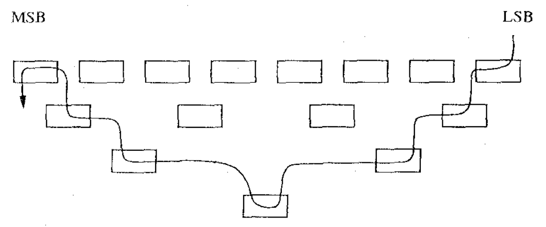

Figure 8: Critical path assumed by typical analyses

if this delay is expected to be from $G_0$-to-$G^*$ delays because this path has both high fan-out loading and worst case fan-in gates. On a given level, the critical path might depend on the generation of $P^*$, but on the next level, the critical path will most likely be the group generate computation because $G^*$ is a function of all the input propagate signals except $P_0$. The adder delay after the CLA unit depends only on carry propagation delays. All $P_i$ and $G_i$ signals have already settled and the critical path follows the worst $C_{i-1}$-to-$C_i$ path in a BCLA unit on each level.

Assuming that the generation of $G^*$ is the critical path through a BCLA unit, then the group feeding $G_i$ can be larger than the group feeding $G_j$ for $i > j$, since $G_j$ feeds a higher fan-in gate. This will reduce the slack between the different $G_i$-to-$G^*$ delays. This argues for fewer levels of smaller lookahead units in the least significant bits than in the most significant bits of every subtree of lookahead units.

The slack in the second portion of a critical path arises from different $C_i$ delay times in a lookahead unit. The subtrees fed by $C_j$ should be faster than those fed by $C_i$ for $j > i$. This indicates that fewer levels of smaller lookahead units should handle the most significant bits of each subtree than should handle the least significant bits.

The criteria for each portion of the critical path contradict each other. This indicates that the best opportunities to increase group sizes or add levels of lookahead may occur in the middle bits of each subtree rather than at the ends. Unfortunately, this criteria can not be used to make specific decisions except for trivial cases. The problem is that the fan-in and fan-out dependencies of the delay model interfere with local optimizations. The whole adder must be analyzed because decisions on each level depend on the sizes of the next level which depend on the structure of the previous levels. Heuristic methods which guess a size for the CLA unit and then work backwards to create a network of BCLA units of the desired size are used to deal with this problem.

## Heuristics

The basic algorithm for finding an optimal CLA structure is a greedy method. The starting point is a single CLA unit. At each step, adder size is increased by adding levels or increasing group sizes. The location and amount of size increase is determined heuristically. The process stops when the adder reaches the desired size. The decision to increase the number of lookahead levels is reduced to the decision to increase group sizes by viewing all the inputs into the carry-lookahead network as groups of size one. When a group is made larger, the extra bits are added to the most significant end of the group and the inputs to these new positions are connected directly to the initial propagate and generate generation circuitry.

The first heuristic increases adder size by one bit in each step. The block that receives this extra bit is chosen so that the new structure has the smallest possible increase in delay. Increasing the size of the CLA unit is not allowed and the whole process is repeated for different starting CLA unit sizes. The current range of starting sizes is from 2 to 16.

The second heuristic uses a more complex metric. The starting point is a 2-bit CLA unit and at each step, bits are added to achieve the maximum increase in adder size per unit delay increase. This metric is determined for each group (including the CLA unit) as follows. Compute a new adder delay based on increasing the size of the group under consideration by one. Determine the total number of bits that could be added to the network without exceeding the new delay value. This number includes the bit that would be added to the current group and assumes the process of adding the other bits starts at the CLA unit and works back by levels to the initial propagate and generate circuitry. The metric is calculated as the total number of bits that could be added divided by the increase in delay. Note that the number of bits that can be added is the minimum of the number allowed by the delay limit and the number required to reach the desired adder size.

## Results

Figures 9 and 10 show results for the two different implementations

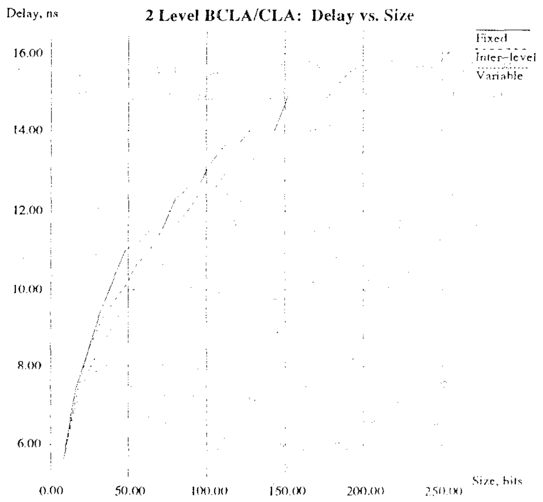

Figure 9: Delay versus adder size in adders using a 2-level BCLA/CLA carry implementation

of BCLA/CLA units. Each graph has three curves corresponding to allowing only fixed-sized groups (**Fixed**), allowing different sized groups only on different levels (**Inter-level**), and allowing variable sizes and levels anywhere (**Variable**). The delay values are for the best structures of each category. The **Fixed** and **Inter-level** curves were obtained by simulation of all possible combinations. If an integral number of groups of the chosen size for a level handled more signals on that level than was necessary, then the size of the group corresponding to the most significant bits of the adder was decreased to eliminate the excess capability. The **Variable** curve for the two-level BCLA/CLA carry implementation (Figure 9) was obtained using the first heuristic. This heuristic worked well for this lookahead implementation but not as well for the three-level implementation. The **Variable** curve for the three-level BCLA/CLA carry implementation (Figure 10) was obtained using the second heuristic since it produced better results than the first heuristic.

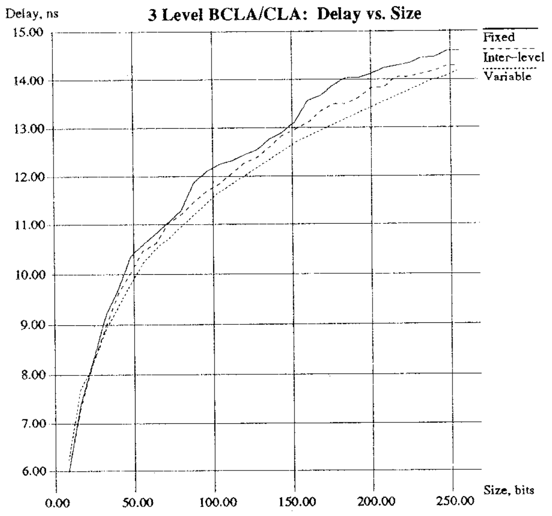

Figure 10: Delay versus adder size in adders using a 3-level BCLA/CLA carry implementation

Figure 11 gives the optimal 32-bit adder structure found by each heuristic. The 1's represent input from the initial bit positions.

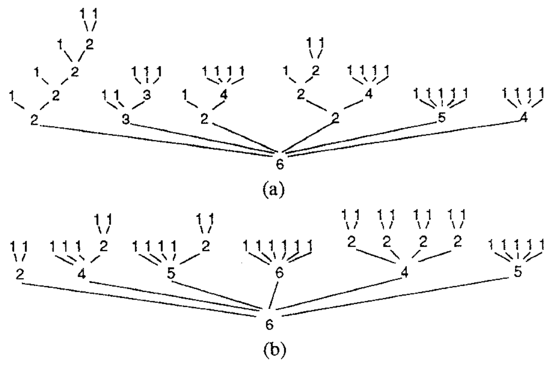

Figure 11: 32-bit adders found by (a) the first heuristic using 2-level BCLA/CLA units (delay = 8.5249 ns) and (b) the second heuristic using 3-level BCLA/CLA units (delay = 8.9188 ns)

The results correspond well to theory. Adders of fixed-size groups are slower than adders allowing variable inter-level groups. Adders with variable groups and levels are faster than both other types. The optimal **Variable** structures tend to have more levels and larger group sizes in the middle of the adder than on the ends. Results of comparing the different lookahead implementations are mixed. The **Fixed** and **Inter-level** three-level implementations performed much better than the two-level implementations. However, the heuristics improved the performance of the two-level implementations more than they improved the three-level implementations. The delay differences between the **Variable** adders of the two implementations is smaller, though the three-level implementation is still faster for larger adders.

## Conclusion and Future Work

Varying group sizes and lookahead levels improves the performance of commonly used CLA implementations. Unfortunately, finding these improved structures is difficult because of delay fan-in and fan-out dependencies. In general, the whole adder structure must be known before a decision to increase group sizes or the number of lookahead levels can be made. Fortunately, simple heuristics can deal effectively with this problem. Simulation results show that heuristic methods can find CLA adder structures with variable group sizes and levels that are faster than more constrained carry-lookahead adders.

Work is in progress to re-run the simulations in this paper for ECL delay data. This is particularly important in the domain of high performance machines where bipolar is the dominant technology. Also, more heuristics and implementations will be examined. For example, a hybrid of the two heuristics presented here will be tried and an adder structure that uses the three-level BCLA carry implementation with the two-level CLA carry implementation will be tested.

## Acknowledgements

The authors would like to thank Professor Pak Chan for sharing his ideas and insight on carry-lookahead adders. His many helpful discussions and suggestions are greatly appreciated.

## References

[1] Kai Hwang. *Computer Arithmetic: Principles, Architecture, and Design.* John Wiley & Sons, Inc., 1979.

[2] LSI Logic Corporation. *Databook: 1.5 Micron Compacted Array ™ Technology*, July 1987.

[3] Vojin G. Oklobdzija and Earl R. Barnes. On implementing addition in VLSI technology. *Journal of Parallel and Distributed Computing*, 5:716–728, 1988.

[4] Robert Sherburne, Jr. Processor design tradeoffs in VLSI. Technical Report UCB/CSD 84/173, University of California, Berkeley, 1984.

[5] J. Sklansky. An evaluation of several two-summand binary adders. *IRE Trans.*, EC-9(2):213–226, June 1960.

[6] Silvio Turrini. Optimal group distribution in carry-skip adders. In *Proceedings of the 9th Symposium on Computer Arithmetic*, pages 1–18, September 1989.