Transactions Briefs

An Algorithmic and Novel Design of a Leading Zero Detector Circuit: Comparison with Logic Synthesis

Vojin G. Oklobdzija

Abstract— **A novel way of implementing the Leading Zero Detector (LZD) circuit is presented. The implementation is based on an algorithmic approach resulting in a modular and scalable circuit for any number of bits. We designed a 32 and 64 bit leading zero detector circuit in CMOS and ECL technology. The CMOS version was designed using both: logic synthesis and an algorithmic approach. The algorithmic implementation is compared with the results obtained using modern logic synthesis tools in the same 0.6 μm CMOS technology. The implementation based on an algorithmic approach showed an advantage compared to the results produced by the logic synthesis. ECL implementation of the 64 bit LZD circuit was simulated to perform in under 200 pS for nominal speed.**

I. INTRODUCTION

In any floating-point processor *normalization* is a required operation. It consists of an appropriate left shift until the first nonzero digit is in the left-most position. The amount of shift is determined by counting the number of zero digits from the left-most position until the first nonzero digit is reached. The exponents are appropriately decremented for the shift amount. The normalization is usually performed before storing the numbers in the register file (memory), commonly referred to as *post-normalization,* and referred to as *pre-normalization* before the operation is performed. In both cases, the special circuit implemented (in hardware) to detect the number of leading zeros is referred to as a *Leading Zero Detector* (LZD).

Applying a straight forward combinatorial approach in designing the LZD circuit is a rather complicated process because each bit of the result is dependent on all of the input bits, which in the case of 64 bit word, consists of 64 inputs. For example a 64 bit LZD circuit would consist of six outputs, each dependent on 64 inputs. It is obvious that such large fan-in dependencies are a problem and that the resulting circuit is likely to be complicated and slow. To design such a circuit using computer aided Boolean minimization techniques or the Karnaugh map method is cumbersome and slow and the resulting design does not exhibit any structure. An immediate solution would be to resort to the use of the logic synthesis tools and "let the tool do the job". This is perhaps the most commonly practiced approach, for any of the complex and complicated circuit of which LZD is a very good example.

Characteristic of the LZD circuit is its concise functional description. It is also very easy to describe the circuit using any of the common hardware description languages. Such a circuit naturally leads itself to the use of logic synthesis by describing the expected functionality in VHDL and simply letting the computer to do the rest.

On the other hand, we can use some intelligence in designing the circuit by attempting first to identify some common modules and impose hierarchy on the design. The LZD circuit in particular, is quite suitable for exploring possibilities of hierarchical and structural design. This yields to substantial improvement of the circuit regularity and speed, compared to straight-forward minimization. The resulting circuit performs well with low and regular fan-in and fan-out, leading to better wireability and better layout. However, LS tools have not yet reached a level of sophistication which can deal with hierarchical structures and create or impose hierarchy in the design. Their approach is to rather expand the logic in one level and optimize it via elaborate and laborious logic minimization, using hours of CPU time yet not being able to identify common modules and hidden structure. Therefore, the design presented in this paper results not only in an efficient and fast LZD, but also provides an efficiency measure of the logic synthesis tools and their limitations.

Manuscript received January 6, 1993; revised July 21, 1993. This work was supported by a minigrant from the Office of Research, University of California, Davis.

The author is with the Department of Electrical and Computer Engineering, University of California, Davis, CA 95616.

IEEE Log Number 9214288.

TABLE I
TWO BIT TRUTH TABLE FOR LZD

Pattern	Position	Valid
1X	0	yes
01	1	yes
00	X	no

II. DESIGN USING AN ALGORITHMIC APPROACH

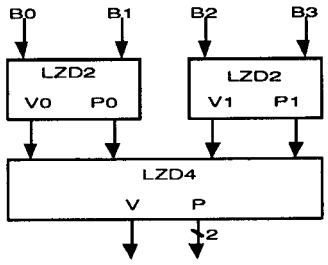
In our approach we use the inherent hierarchy associated with the leading zero detection process and map it into a hierarchical and modular design. In order to understand this design process, let us begin by first examining only the two bit case, as shown in the Table I. The pattern on the left designate the possible two bit combination. If the left-most bit is "1" we assign "0" to the *Position* and "1" to the *Valid* bit indicating that there is *zero* distance from the left-most bit to the first *nonzero* bit. If both bits are "0", we would set *Valid* bit to 0 indicating that this is not a valid position. Not only is this because we have only one bit to indicate position, but the next two-bit group might have more zeroes to follow and therefore position information is not complete. It is straightforward to construct the logic for the two bits representing the valid bit (V) and the position bit (P) as shown in Table I.

We can easily extend the two bit case to the four bit case. Let us designate the position bits (4 bits total) as $P0$ for the left-most two bits and $P1$ for the right-most two bits as shown in Fig. 1(b). Also, we will designate $V0$ and $V1$ as the valid bits for the first two and second two bits respectively starting from the left to right. The leading zero position can be represented as a function of those four bits as shown in the fifth column of Fig. 1(a) (minus sign represents complementation). Also, it should be noted that we are using "Big Endian" notation, i.e., we start indexing from left to right (this notation is used by IBM).

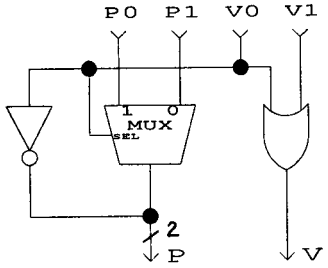
The 4 bit circuit has a depth of 2 logic levels; in the second level the valid bit is formed as a logical OR of the valid bits from the previous level. In other words, if there is a "valid" string of bits within the group in the previous level, then this group has a valid position bit. If all of the groups, however, do not show a "valid" output, this simply means that there are a string of zeros and that the first nonzero bit can be expected only within one of the groups to the "right". The left valid bit V_1 is inverted and concatenated with P_0 if $V_0 = 1$, or with P_1 if $V_0 = 0$ and $V_1 = 1$. This is achieved

pattern	position	position (binary)	valid	position
1011	0	00	yes	(-V0)P0
0100	1	01	yes	(-V0)P0
0011	2	10	yes	(-V0)P1
0001	3	11	yes	(-V0)P1
0000	X	XX	no	XX

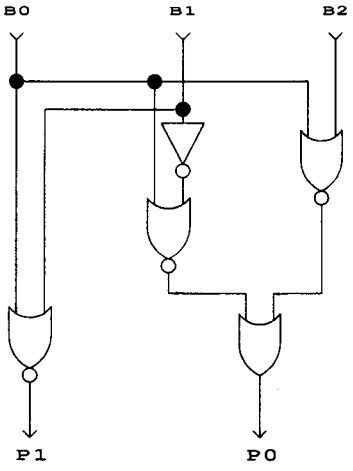
(a)



(b)



(c)



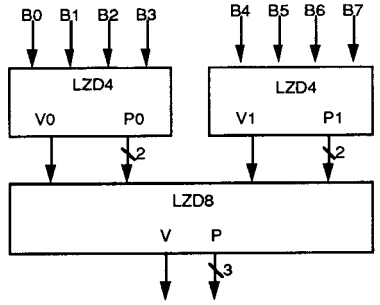
(d)

Fig. 1. Design of a 4 bit LZD. (a) Truth table. (b) Using two 2 bit LZD's. (c) The logic sturcture of the 4 bit LZD block. (d) One level implementation of the 4 bit LZD.

by simply multiplexing $P0$ and $P1$ to the output of the multiplexer. The logic structure of the 4 bit LZD group (LZD4) is shown in Fig.

8-bit LZD							
				"left" nibble		"right" nibble	
bit pattern		position	valid	P0	V0	P1	V1
1XXX	XXXX	000	yes	00	yes		
01XX	XXXX	001	yes	01	yes		
001X	XXXX	010	yes	10	yes		
0001	XXXX	011	yes	11	yes		
0000	1XXX	100	yes		no	00	yes
0000	01XX	101	yes		no	01	yes
0000	001X	110	yes		no	10	yes
0000	0001	111	yes		no	11	yes
0000	0000	XXX	no		no		no

(a)



(b)

Fig. 2. Design of an 8 bit LZD. (a) Truth table. (b) Using two 4 bit LZD's.

1(c). This implementation is particularly fast because the propagation delay of the multiplexer is smaller than the propagation delay of a gate. This is the case in several CMOS and ASIC libraries such as the ASIC library from LSI logic corporation [3].

A 4 bit LZD can also be implemented in one level directly from Fig. 1(a), avoiding the need to go into the above exercise. A one level implementation of a 4 bit LZD is shown in Fig. 1(d). It is not necessary that we limit our design to 2 bits per level. Naturally, the implementation depends on the technology used. The entire concept can be grouped in 4 bit groups. In a technology that tolerates high fan-in and fan-out we can compress even more bits into one level or one logic tree (such as in the case of ECL technology).

Now we can take two groups of 4 bits and form a LZD for an 8 bit word by simply following the same concept that we did in the example of 4 bits. The truth table is given in Fig. 2(a) and the design in Fig. 2(b).

From the above discussion we can deduce the hierarchical structure for the LZD and arrive at the following algorithm for generating the number of leading zeros:

Algorithm for generating LZ count:

(1) Form the pair of bits B_i, B_{i+1} for $i = 0$ to N-2 with bit 0 being the leftmost one

(2) Determine P and V bits for each pair

(3) for the next level determine the P_g and V_g bits as function of two pairs of inputs P and V in this level in the following way:

$V_g = V_l + V_r$ where "+" is logical OR operation of the left and right inputs

if $V_l = 1$ then $P_g = 0, P_l$ where "," designates concatenation

else if $V_r = 1$ then $P_g = 1, P_r$

otherwise $V_g = 0$

Repeat step (3) $\log(N) - 2$ times

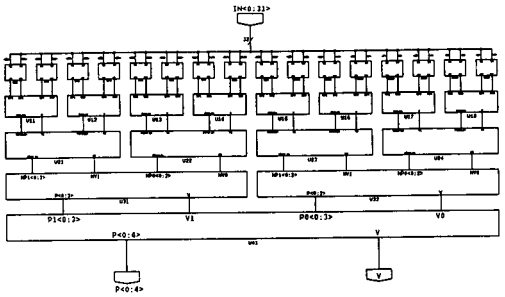


Fig. 3. 32 bit LZD circuit composed of 2 bit groups.

TABLE II
SIMULATION CONDITIONS

T_{ox}=150A, V_T=0.6V, L_{eff}=0.6u $R_{metal\ 1}$=120 mohm/sq		
NC	4.0 V, 125 C	Nominal
WC	2.8 V, 125 C	Worse Case

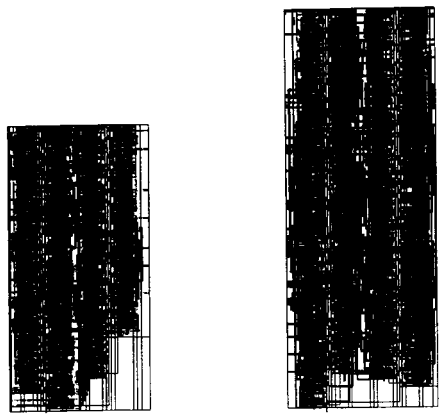
It can easily be concluded that the logical depth of this circuit is $\log_2(N)$ stages where the path through each stage is of the complexity of the multiplexer or one level of equivalent logic. The multiplexer is actually implemented using a pass-transistor structure and therefore is even faster than a regular CMOS gate. This is the reason for the extraordinary speed of this scheme for the LZD implemented in CMOS. In addition, we might want to save one level and implement this scheme in $\log(N) - 1$ levels instead of $\log(N)$ levels. This is achieved by starting with the groups of 4 bits and proceeding in the way described by the algorithm. Using CMOS technology, this is usually the best that can be achieved in terms of logic levels, because any further compression of the number of levels would pass the point of diminishing returns by increasing the fan-in and fan-out of the circuits, thus negatively affecting the speed. However, the same concept can be applied to groups of 4 bits instead of 2 bits, which is more appropriate for ECL and BiCMOS technologies. In that case there is an additional speed advantage, because the speed of this LZD implementation is proportional to $\log_4(N)$ stages and therefore faster. The structure of the 32 bit LZD composed of the 2 bit groups is shown in Fig. 3.

III. IMPLEMENTATION AND LOGIC SYNTHESIS EXPERIMENT

We implemented six LZD prototypes of various sizes. They were laid out and simulated for worst case conditions. In addition, we repeated those designs using logic synthesis tools, produced layouts and simulated the results. Our second goal was to gain performance by resizing the transistors as we went down the path. Larger transistors result in better driving capability and their size will be increased where space is available. Given that our structure is tree like, as the signal moves from the first stage to the second and third, the available space increases. By filling this space with transistors of larger size, the resulting LZD layout takes more of a rectangular shape. Such an approach is used successfully in an adder based on recurrence solving [1]. Therefore our objective was to have some performance gain by applying this findings. This provided enough data for performance analysis and evaluation of the LS tool. The technology and simulation parameters are shown in Table II.

TABLE III
PERFORMANCE OF THE NEW LZD CIRCUIT UNDER NOMINAL AND WORSE-CASE CONDITION

Bits	WC [nS]	NC [nS]
25	7.69	4.49
32	7.7	4.52
53	9.08	5.35
64	9.09	5.37
112	10.7	6.41
128	10.7	6.43



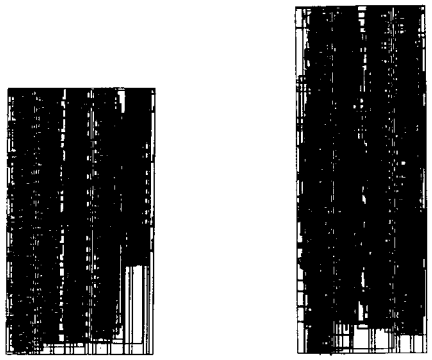
(a) (b)

Fig. 4. Layout of the 32 bit LZD. (a) for the algorithmic design and (b) LS result.

A. The Layout

The regularity of the novel LZD design can also be used to produce a more efficient layout. By creating each cell as a basic building block for each stage, the entire circuit can be routed primarily in metal lines flowing in the direction of the data-path. This results in better performance and facilitates the inclusion of the LZD in the regular data-path of the floating-point or any other unit that needs a LZD circuit.

The layout of the 32 bit LZD is shown in Fig. 4, (a) for the algorithmic design and, (b) one obtained as a result of LS. Observe that the algorithmic layout has 4 rows of cells which were placed using the Timberwolf package resulting in an area of 163 × 340 μ. The results obtained using LS have a 35% larger area resulting in 186 × 403 μm. They are plotted to the same scale and placed next to each other for easy comparison. In terms of speed, the algorithmic LZD introduces a delay of T_a = 4.5 nS, while the LZD resulting from LS has a delay of T_{ls} = 5.8 ns (both for the typical case) which is 29% slower. Therefore we can say that the algorithmically designed LZD is roughly one–third smaller and faster than the equivalent LZD resulting from logic synthesis.



(a) (b)

Fig. 5. "Rectangular" layout of the LZD circuit produced using (a) the algorithmic approach and (b) LS

We have also tried to explore the regular and hierarchical structure of this design by applying the approach described by Vuillemin and Guibas [1] to the layout of this circuit. The idea in [1] is to lay the tree-like structures like this one on a rectangular pattern in such a way that as the signal progresses down the levels, the size of the cells is made larger, increasing their driving capability so that the signal can drive more inputs in a shorter time. We implemented this idea and laid out the 32 bit LZD circuit such that the width of the circuit was kept constant. The layout of the "rectangular" LZD structures obtained using the algorithmic approach and LS are shown in Fig. 5. They are plotted on the same scale (as in Fig. 4) to point to the difference in size between the algorithmic LZD (a) and LS (b). The "rectangular" layout of the algorithmic LZD is 206×363 μm [Fig. 5(a)] while the LS resulted in 181×473 μm. The algorithmic approach has resulted in a 14.5% smaller layout compared to the LS result in this case.

B. Performance

The performance of the novel LZD was simulated under nominal and worst case conditions, denoted NC and WC, respectively. The NC and WC conditions are characterized in Table II together with the parameters typical for this CMOS process. The table shows the speed of the LZD for different sizes starting from $N = 25$ to $N = 128$ bits. This is also shown in Table I(a) for the unbuffered LZD (without the output buffers) for nominal and worst case conditions.

In terms of performance, the "rectangular" layout introduced a delay of 6.9 nS for the algorithmic LZD and 7.7 nS for the one resulting from LS (for the nominal case). The performance advantage of the algorithmic LZD was 12% over LS for the NC. For worst-case conditions, this advantage was 19%. However, when we compared the "rectangular" layout versus the regular layout in terms of performance in both cases, the results showed that overall, the "rectangular" approach did not improve the performance in every case.

The performance of the "rectangular" layout was better than the regular layout in case of 1 pF output load. The difference was 15% for the nominal case and 12% for the worst case which favors the "rectangular" layout. The performance difference is shown in Table IV(a). In the case of no load (or light load) on the output, the "rectangular" layout was worse: 10% (nominal case) and 23% (worse case). This is explained by the fact that this circuit maintains regular fan-in and fan-out, and neither one of them increases when reaching the levels closer toward the output. The performance was not affected because the input capacitance grew proportionally, as did the driving capacity of the gate. The increase in delay caused by the capacitance increase, more than offset the improved driving capability, resulting in a slight loss. Therefore, the idea [1] which worked well for a CLA-like adder structure did not work in our case. This is shown in Table IV(b). We observe that under the no-load conditions, the LZD circuit obtained via regular layout performs better. However with 1.0 pF load, the rectangular LZD outperforms the regular one in both NC and WC. This is attributed to the stronger driving capabilities of the final stages. However, we feel that just adding stronger buffers at the output nodes would still make the regular case perform better.

In Table V we compare the results for a 32 bit LZD using our approach with the one obtained using logic synthesis without load and with 1.0 pF loading capacitance on the outputs. The algorithmic LZD outperforms the one obtained via LS under all conditions. The only exception has been the marginal difference of 0.25 nS in favor of the implementation obtained by synthesis which is attributed to the rectangular layout rather than the way LZD has been implemented. With a 1.0 pF load and use of rectangular layout, algorithmic LZD is 12% faster under nominal conditions (6.9 versus 7.7 nS) and 15% faster under worst conditions (11.4 versus 13.6 nS). In any one case, we have demonstrated that our algorithmic LZD circuit outperforms LS, and that the improvement in performance ranges from 12% (NC rectangular layout, 1 pF load) to 56% (WC regular layout, no load). The improvement in the area is from 14.5% (rectangular layout) up to 35% (regular layout). This clearly demonstrates the superiority of the algorithmic approach.

TABLE IV
PERFORMANCE COMPARISON OF REGULAR VERSUS RECTANGULAR 32 BIT LZD LAYOUT WITH (a) 1 pF LOAD. (b) NO LOAD CONDITION.

Speed NC (WC) for 1.0pF load [nS]	
Regular Layout (163X340μ)	Rectangular Layout (206X363μ)
7.95 (12.8)	6.9 (11.4)

(a)

Speed NC (WC) for 0pF load [nS]	
Regular Layout (163X340μ)	Rectangular Layout (206X363μ)
4.5 (7.7)	4.93 (9.5)

(b)

TABLE V
COMPARISON OF THE ALGORITHMIC LZD AND LOGIC SYNTHESIS RESULTS

Comparison of the Algorithmic LZD circuit with the LZD obtained via Logic Synthesis										
Algorithmic LZD (regular layout)					LZD obtained with Logic Synthesis					
	no load		1.0 pF load		regular	no load		rectang.	1.0 pF load	
Area[u]	WC [nS]	NC [nS]	WC [nS]	NC [nS]	Area[u]	WC [nS]	NC [nS]	Area[u]	WC [nS]	NC [nS]
163×340 86mils	7.7	4.5	12.8	7.95	186×403 116mils	12	5.8	181×473 133mils	13.6	7.7

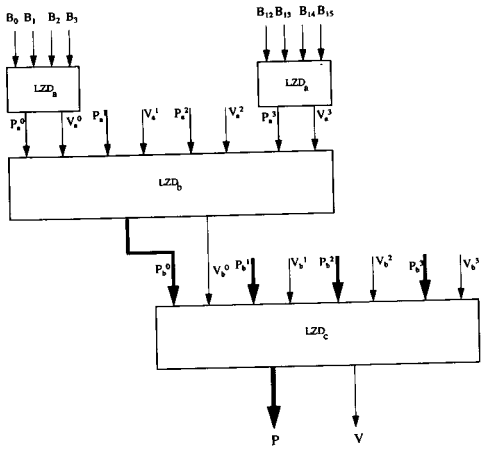


Fig. 6. Structure of 64 bit ECL LZD circuit.

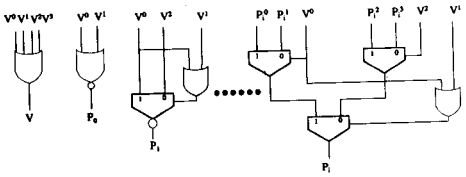


Fig. 7. ECL structure of the second level LZD circuit.

IV. HIGH-PERFORMANCE ECL IMPLEMENTATION

In a Floating-Point processor a *critical path* consists of: *leading zero detector, shift* and *rounding* operation, where each of the operations contributes approximately equally to the delay. The technology of implementation may generally differ from low power CMOS to ECL technology which is very relevant even today due to the renewed attention given to it. Implementation of LZD by the algorithm described is challenging because the rules that are applied for ECL technology are very different from those applied to CMOS. Also ECL has a tendency to combine as much logic as possible into one *level* or *logic tree,* generally allowing for larger fan-in. Finding a suitable ECL structure for any algorithmic and technology independent design is not easy. Therefore we had to solve two problems:

1) compress the design in as few levels as possible
2) identify a common and characteristic structure that also implements itself well in ECL

The first problem has been to define a 4-way LZD structure which leads to 3 levels (or 3 ECL trees) for a 64 bit LZD circuit. The first level is trivial to design and it is very much similar to the 4 bit group design shown in Fig. 1(d) except by using wired-OR (in ECL), we were able to implement everything in on the level of gates. The structure of 64 LZD implemented in ECL technology is shown in Fig. 6.

The second level will indicate the LZD position based on the input from the 4 LZD groups from the previous level. The structure of this group is not as regular as in the CMOS case. However, it was possible to identify a common multiplexer based structure that can be applied in general for any position bit ($i = 2, 3 \ldots k$). This structure is shown in Fig. 7.

Fortunately our algorithmic approach favors multiplexer structure which suits an ECL circuit very well. The bits P_0 and P_1 (as well as V) are implemented separately. The ECL tree used for this computation is shown in Fig. 8. It is three transistor levels high, which is about as much as we can implement in one ECL tree. Therefore the decision to calculate P and V based on the input of the previous four groups, rather than two or eight, seems to be optimal. The depth of the n-bit LZD circuit implemented in this way is $\log_4(n)$ levels. In our case for 64 bit LZD, we have a depth of 3 ECL *trees*. Using the advanced Motorola BiCMOS process, this circuit produces the result in $T_{cr} = 200$ pS (nominal time) for a 64 bit LZD circuit. It should be noted that the first level, which is one gate deep, could be integrated into the second level yielding an implementation in only 2 ECL trees.

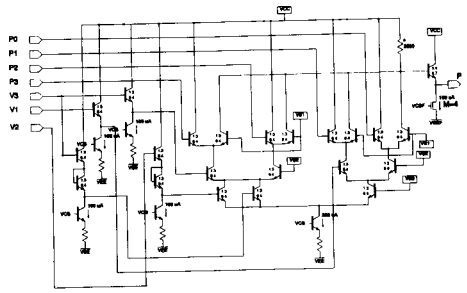


Fig. 8. ECL circuit implementation of the second level (64 bit LZD).

V. CONCLUSION

In this paper we have described an algorithmic approach to designing a leading zero detector. This circuit has been implemented in 0.6 μ CMOS technology and is compared to the results obtained using logic synthesis under various conditions and for different layout approaches. The algorithmic approach outperformed LS consistently, with improvements in speed ranging from 12%–56% and improvements in layout area ranging from 14.5%–35%. We have clearly demonstrated the superiority of the algorithmic approach on this circuit. The results generally indicate that careful analysis of the problem and clever management of the hierarchy pays big dividends in the performance of critical circuits, especially data-paths. Although very useful, LS tools are still not capable of managing hierarchies and making intelligent choices when it comes to design, and therefore they should be treated accordingly. The resulting LZD circuit has remarkable performance, which is important since it is often a part of the critical path in the floating-point unit.

ACKNOWLEDGMENT

I gratefully acknowledge V. Chang for running the simulation and Timberwolf program. I thank the referees and R. Maeder for careful reading and useful remarks. The idea was conceived of in June 1987 while the author was on the TF-1 project at IBM T. J. Watson Research Center in New York. I am grateful to M. Denneau of IBM for his ideas and discussion during the project. The author is thankful for the generous support from Sun Microsystems Laboratories, G. Taylor, D. Ditzel, and P. Hansen in particular.

REFERENCES

[1] J. Vuillemin and L. Guibas, "On fast binary addition in MOS technology," in *Proc.*, ICCC'82, New York, Sept. 28, 1982.
[2] V. G. Oklobdzija, "An implementation algorithm and design of a novel leading zero detector circuit," presented at 26th Asilomar Conf. on Signals, Systems and Computers, Oct. 26–28, 1992.
[3] LSI Logic, *1.0 micron cell-based product databook.* 1991.