A HIERARCHICAL AND MODULAR CIRCUIT IMPLEMENTING LEADING ZERO DETECTOR FOR A HIGH PERFORMANCE FLOATING-POINT PROCESSOR

Vojin G. Oklobdzija

Department of Electrical and Computer Engineering
University of California
Davis, CA 95616
vojin@ece.ucdavis.edu

Abstract

A novel circuit implementing a Leading Zero Detection (LZD) function is presented. The circuit is implemented by imposing a hierarchical and modular topology resulting in a very fast and efficient circuit in terms of the layout area. The circuit structure is derived from an algorithmic derivation of the LZD function. The basic blocks are designedto take advantage of the fastest cell for the particular technology; CMOS and ECL and are modular and scalable. The resulting circuit demonstrates a substantial advantage over the one obtained using logic synthesis.

1. Introduction

In any floating-point processor *normalization* is an operation that is performed either before the operands are brought to the arithmetic unit or after the computation (*post-normalization*) and it is not possible to perform it in parallel with the arithmetic operation. Therefore the time needed to detect the number of leading zero positions and execute appropriate shift is added to the critical path. The amount of the shift is determined by counting the number of zero digits from the left-most position until the first non-zero digit is reached. Afterwards the exponents are appropriately decremented by the shift amount. Often, the critical path includes leading zero detection, a pass through the shifter, and an adder. Those three time components all contribute to the critical path, hence it is essential to design those circuits to be as fast as possible.

Design of the Leading Zero Detector for a large number of input bits (sizes of 54 or 64 bits are quite common) is rather complicated. This is because each of the 6 result bits are dependent on all of the input bits, which in the case of a 64-bit word consists of 64 inputs. To design a circuit with such large and irregular fan-in dependencies is a problem, and it is to be expected that the resulting circuit will be complicated and slow. On the other hand the circuit can be designed with the use of *Logic Synthesis* tools because the VHDL description of this circuit is on the contrary, concise and clear. However, the circuit resulting from the use of logic synthesis is not expected to be the most efficient and fastest design possible simply because the logic synthesis tools have not achieved the required level of sophistication [2].

2. Design Approach

In an attempt to find an efficient circuit implementation, two principles were applied:

o *first, find structure in the circuit o make the design modular and hierarchical. This helps to keep the circuit fan-in and fan-out relatively small and the wiring of the circuit regular.*

o *second, implement the basic building block of the circuit using the fastest component of the given circuit technology.*

The first principle was achieved by the following design proceedure:

2.1 Design Procedure:

1. The LZD circuit for a relatively small number of input bits LZD^i (where i = 2 or 4), was designed first. This is a simple and straight forward process. Afterwards, a larger LZD was built from the smaller blocks in a recurrent way, using the second type of blocks LZD_m in a recurrent fashion, where m designates the number of LZD^i blocks acting as inputs to the LZD_m block. A LZD circuit is therefore built in a $\log_m(N)$ levels (where N is the number of input bits). The first row consists of the LZD^i type circuits followed by $\log_m(N)$ - 1 blocks of the LZD_m type. Therefore the speed of the resulting LZD circuit is expected to be proportional to log(N) as a function of the number of input bits.

2. The LZD_m block was built by favoring the use of multiplexers given that a multiplexer can be built as a very fast circuit (both in ECL as well as CMOS) technology. A multiplexer circuit can be built (in CMOS) to be faster than a regular two input gate (Fig.1). In ECL, the basic circuit structure accomodates building multiplexers, since current steering (mulitplexing) is the principle upon which a function is realized in ECL .

The modularity and the algorithm for design of the LZD can be derived from the Truth Table for an 8-bit LZD.

We observe that the valid bit V for the 8-bit LZD (indicating that there is a "valid" LZ string in this block) is simply formed by an OR function of V0 and V1.
The "position" bits P (indicating the position of the first non-zero bit) are formed (observing the table) by concatenating the complement of V0 (valid bit of the "left" nibble) with the postion bits (P0 or P1) of the "valid" nibble as:

```
if V0 = 1  than: P = -V0 / P0
           else:
           P=-V0 / P1
end;
```

This process defines the structure of the LZD_m which consists of an OR gate and an m-bit multiplexer. Both can be implemented as fast circuits. To form a larger size LZD, this process is repeated by cascading the LZD_m in a tree like structure.

3. Implementation

The structure of LZD_2 (used in our CMOS implementation) is shown in Fig. 2. A larger size multiplexer can be implemented in ECL technology, therefore m=4 is the right choice in the case of ECL. However, implementation of LZD_4 is somewhat more elaborate, nevertheless it follows from the design procedure described in 2.1 and preservs the principles outlined at the begining of the Section 2. Position bit m-1 is a logical NOR function of Vm-1+Vm-2, Pm-2 is obtained by multiplexing Vm-1 and Vm-3 in a multiplexer controlled by Vm-1 and Vm-2, and bits Pm-3 to P0 are obtained by using a complex multiplexer tree, the implementation of which is shown in Fig. 3.

The way in which LZD is constructed from LZD^i and LZD_m blocks is shown in Fig.4.

4. Results

The performance of the CMOS implementation of this LZD was simulated under nominal (NC) and worse case conditions (WC) for an implementation in 0.7u Leff, tripple metal CMOS technology. Fig. 5. shows the speed of this novel LZD for different sizes starting from N=25 to N=128 bits. The layout of this novel circuit (Fig.6) is also regular and compact which is another atribute contributing to its extraordinary speed. The Algorithmic approach outperformed logic synthesis consistently ranging from 10% - 36% in terms of the performance and 13% - 26% in terms of the layout area. The ECL implementation of a 64-bit LZD using the outlined design procedure resulted in under 200pS simulated delay under nominal conditions.

5. Conclusion

An Algorithmic approach to designing a Leading Zero Detector is described. This circuit has been implemented in 0.7μ CMOS technology and is compared to the results obtained using Logic Synthesis. We have clearly demonstrated with this circuit the superiority of the algorithmic approach. The lessons learned apply not only to this particular design (of a LZD), but could be taken quite generally as an indication that in the performance of critical, especially data-path, circuits careful analysis of the problem and clever management of the hierarchy pays big dividends. Although very useful, synthesis tools are still not capable of managing hierarchy and making intelligent choices when it comes to design and therefore they should be treated accordingly. The novel LZD has also shown to be very useful since it is often a part of the critical path in the floating-point unit and the results obtained are quite remarkable.

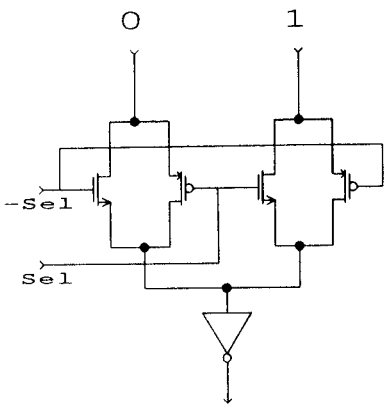
Acknowledgment: *Contribution of Vincent Chang and careful reading of Ron Maeder are gratefuly acknowledged.*

6. References

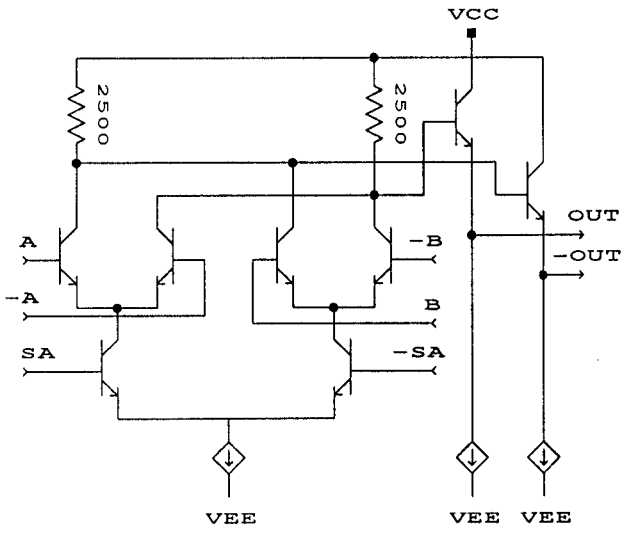
[1] J. Vuillemin, L. Guibas, *"On Fast Binary Addition in MOS Technology"*, Proceedings of ICCC'82, New York, September 28, 1982.

[2] V.G.Oklobdzija,*"An Algorithmic and Novel Design of a Leading Zero Detector Circuit"*, Submitted to IEEE Transactions on VLSI Systems, 1993.

Figures and Tables



CMOS



ECL

Fig. 1. Multiplexer Circuits

Table 1. Truth Table for an 8-bit LZD

8-bit LZD				"left" nibble		"right"nibble	
bit pattern		P	V	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

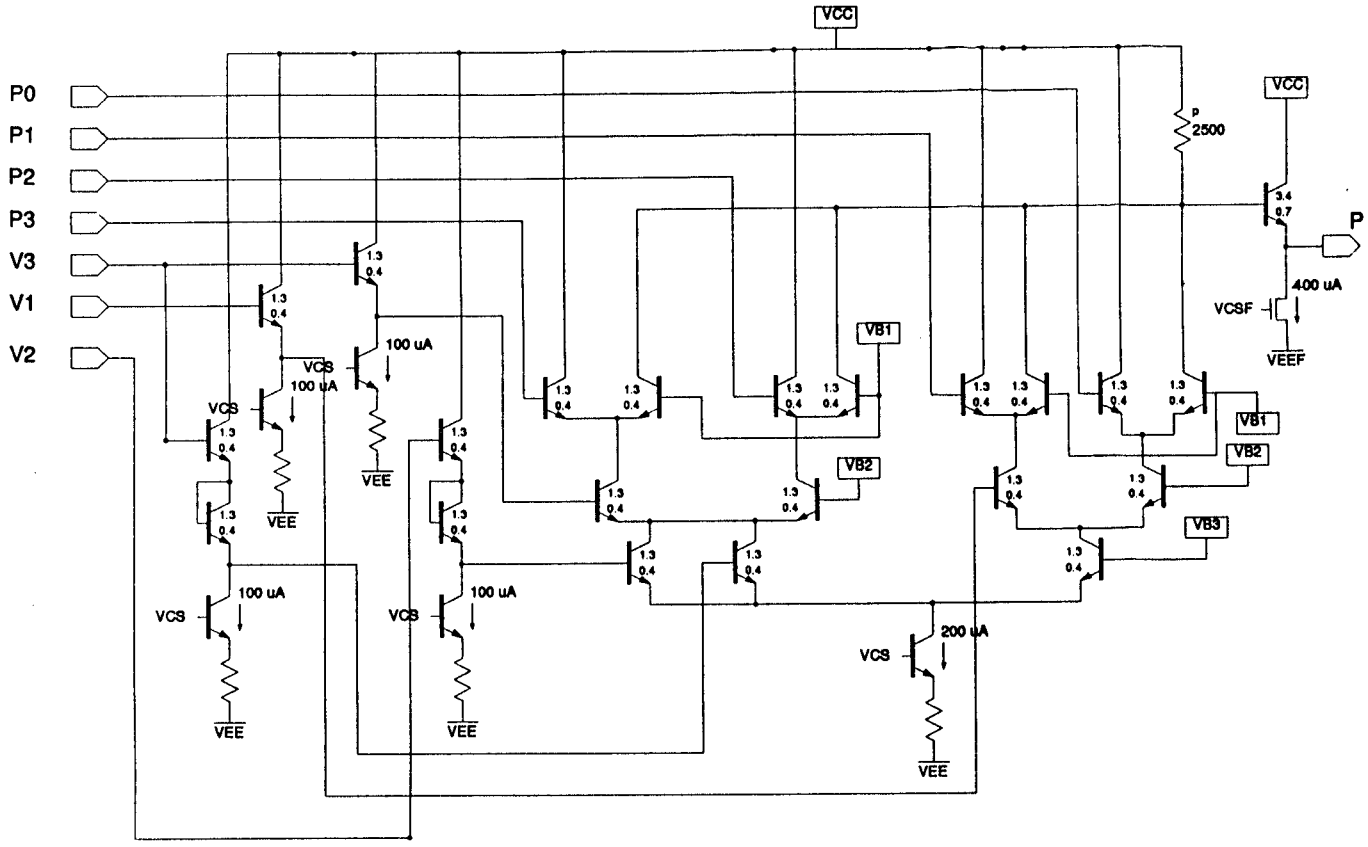


Fig.2. ECL implementation of a complex multiplexer tree as a part of LZD_m

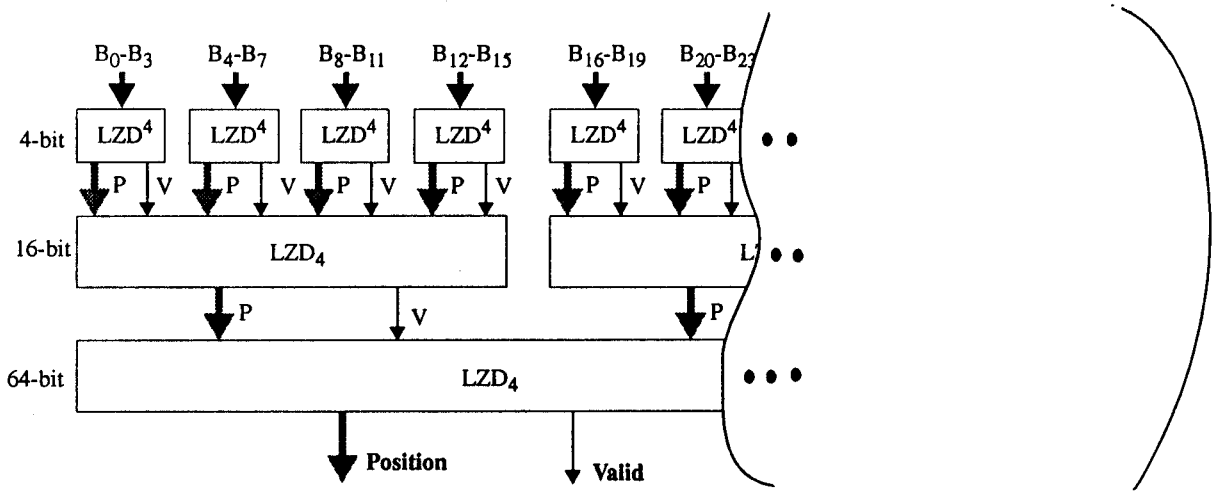


Fig. 3 Example of an 64-bit LZD

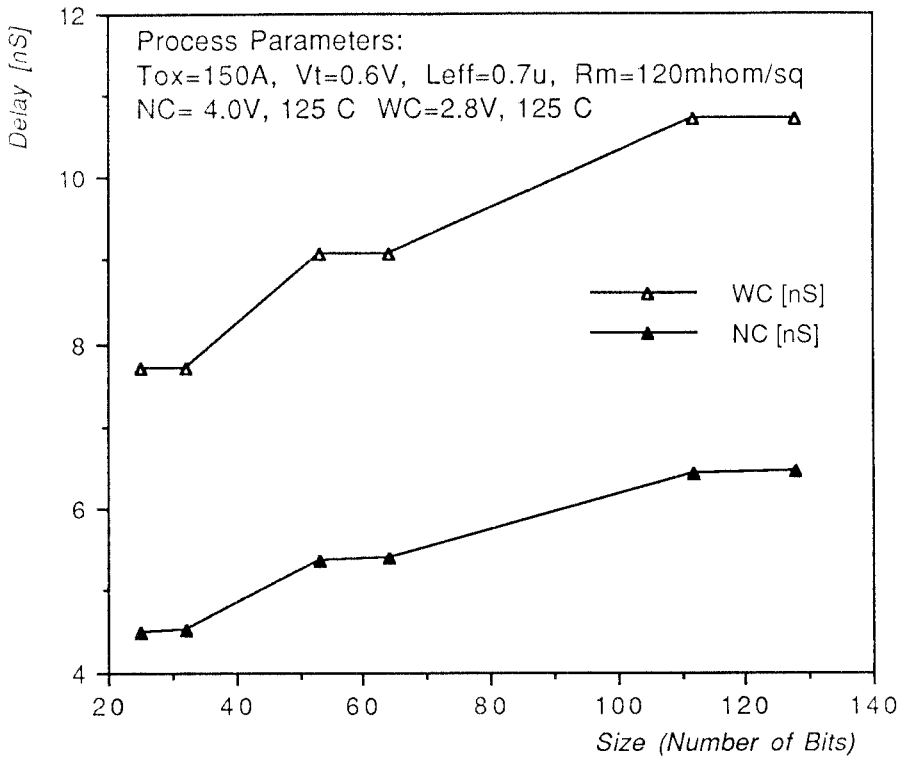


Fig. 5. Speed of the novel LZD for different sizes for CMOS implementation

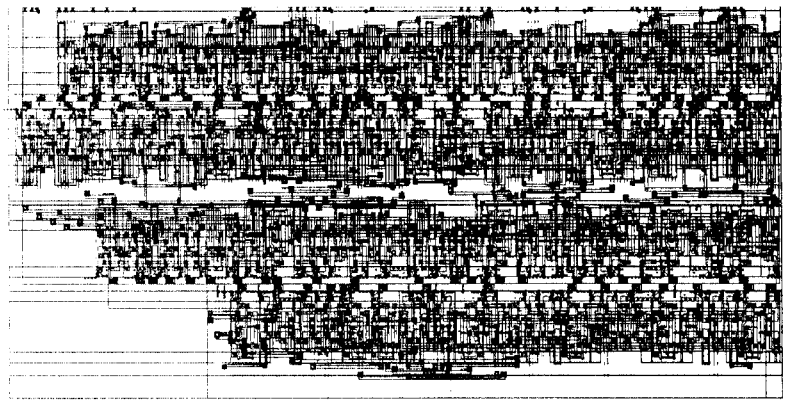


Fig. 6. Layout of novel 32-bit LZD implemented in CMOS