An Implementation Algorithm and Design of a Novel Leading Zero Detector Circuit

Vojin G. Oklobdzija
Department of Electrical and Computer Engineering
University of California
Davis, CA 95616
vojin@eecs.ucdavis.edu
(916) 752-5634

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. This implementation is compared with the results obtained using modern Logic Synthesis (LS) tools in the same 0.9u CMOS technology. Our approach to LZD design yields both speed and area advantages over LS.

1. Introduction

In any Floating-Point processor *normalization* is a common operation. It consists of an appropriate left shift until the first non-zero 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 non-zero digit is reached. The exponents are appropriately decremented for the shift amount. The normalization is normally performed before storing the numbers in the register file (memory), commonly referred as *post-normalization*, and also before the operation is performed, *prenormalization*. In both cases the special circuit implemented (in hardware) to detect the number of leading zeroes is referred to as *Leading Zero Detector* (LZD).

1.1 Implementation of LZD

Implementation of the LZD circuit is rather complicated. This is 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 64-bit LZD circuit would consist of 6 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. These types of circuits are very good candidates for *Logic Synthesis* (LS) tools because their VHDL description is on the contrary concise and clear. However, the sophistication of LS tools has not reached a level at which LS can deal with hierarchical structures and create 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 often hours (even days) of the CPU time. Therefore the design presented in this paper is resulting not only in an efficient and fast LZD but is also providing an estimate and understanding what LS tools can and what they can not do.

2 Design Approach

In our approach we use the inherent hierarchy associated with the Leading Zero detection process and map it into a modular design. In order to understand this design process, let us examine first only the first two bits, as shown in the Table 1.

Truth Table for two bits		
pattern	position	valid
1X	0	yes
01	1	yes
00	X	no

Table 1. Two bit Truth Table for LZD

It is very straightforward to construct the logic for the two bits representing valid bit (V) and position bit (P) for the two bit case shown in the Table 1. The logic for the two bit case is shown in Fig.1. Let us now extend the two bit case into 4-bit case shown in the Table 2. Let us designate the position bit for the first two bits (of 4-bits) as P0 and P1 for the second two bits.

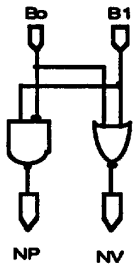

Fig 1. Logic Diagram for P and V bits over a group of two bits

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 LZ position can be represented as a

function of those four bits as shown in the fifth column of the Table 2. (minus sign represents complementation)

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

Table 2. Truth Table for 4-bit LZD

The depth of the circuit is log(N) levels and in each level valid bit is formed as a logical OR of the valid bits from the previous level. The left valid bit Vl is inverted and concatenated with the Pl if Vl=1, or with Pr if Vl=0 and Vr=1. This is achieved by simply multiplexing Pl and Pr to the output of the multiplexer. This structure is shown in Fig.2.

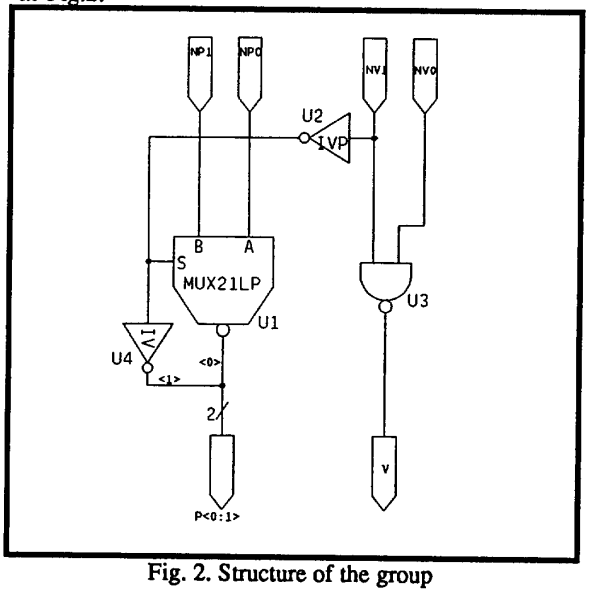

Fig. 2. Structure of the group

From Tables 1-3 we can deduce the hierarchical structure for the LZD and arrive at the following algorithm for generating the number of Leading Zeroes:

Algorithm for generating LZ count:

(1) Form the pair of bits Bi, Bi+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 Pg and Vg bits as function of two pairs of inputs P and V in this level in the following way:

Vg = Vl + Vr where + is logical OR operation of the left and right inputs

if Vl=1 then Pg = 0,Pl where (,) is concatenation

else if Vr = 1 then Pg = 1, Pr

Repeat the step (3) for log(N) - 1 times

It can be easily concluded that the logical depth of this circuit is log(N) stages where the pass through each stage is of the complexity of the multiplexer or one level of logic. It turns out that in many CMOS circuit families 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 extraordinary speed of this scheme for LZD.

In addition we might want to save on one level and instead of log(N) levels implement this scheme in log(N) - 1 levels. This is achieved by starting with the groups of 4 bits and proceeding in the way described by the algorithm. In CMOS this is usually the most we can do in terms of the logic levels, because any further compression of the number of levels would pass the point

Truth Table for 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

Table 3. Truth Table for 8-bit LZD

of diminishing returns as far as the speed of this scheme is concerned. However in technologies such as ECL we can take further advantage and implement this scheme in log(N)/2 levels.

3. Implementation

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 (LS), produced layout and simulated the results. This provided enough data for performance analysis and evaluation of the LS tool.

3.1 The Layout

The regularity of the novel LZD design can also be used to produce 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 a better performance and facilitates the inclusion of LZD in the regular data-path of the floating-point or any other unit that needs LZD circuit.

We have also tried to explore the regular and hierarchical structure of this design by applying this to the layout using the approach described by Vuillemin and Guibas [1]. The idea here 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 bigger increasing on their driving capability so that the signal can drive more inputs in shorter time. We carried out this idea and laid the 32-bit LZD circuit such that the X dimension of the circuit was kept constant. However, we obtained mixed results and overall this approach did not improve the performance. 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 reason why the performance was not affected is because the input capacitance of each also 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 slightly negative gain. Therefore this idea [1] was not found to work in our case. However, the 32-bit LZD with rectangular layout did provide much better circuit when driving larger output loads. For a 1pF output load, we achieved speedup of 16%under the nominal operating conditions and 12% for the worse case.

Simulation Conditions Tox=150A, VT=0.6V, Weff=0.7u, Leff=0.6u Rmt1=120 mohm/sq		
NC	4.0 V, 125 C	Nominal
WC	2.8 V, 125 C	Worse Case

Table 4. Simulation Conditions

Layout of the 32-bit LZD is shown in Fig. 3. It has 4 rows of cells which were placed using Timberwolf package resulting in an area of 163×340 microns.

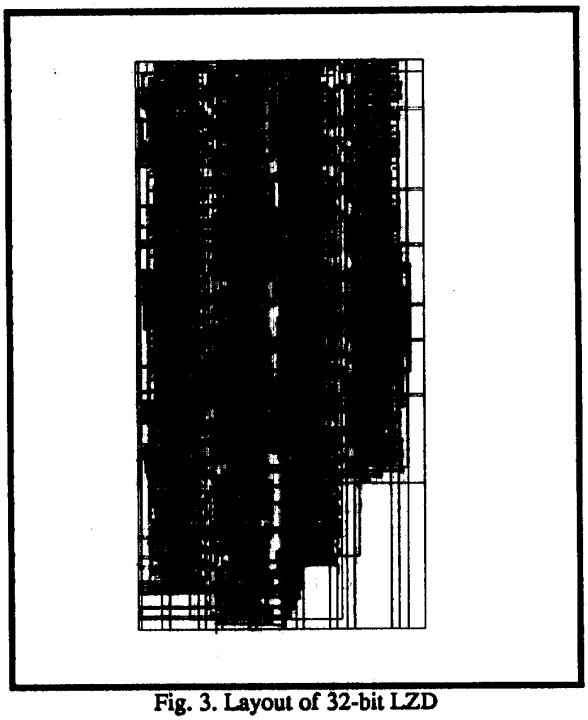

Fig. 3. Layout of 32-bit LZD

3.2 Performance

The performance of the novel LZD was simulated under nominal and worse case conditions, NC and WC respectively. The NC and WC conditions are characterized in the Table 4. together with the parameters typical for this CMOS process.

The performance achieved with novel LZD is shown in Tables 5-7. Table 5. shows the speed of the LZD for different sizes starting from N=25 to N=128 bits. This is also shown in the Fig. 4. We are showing the speed of the unbuffered LZD (without the output buffers) for the nominal and worse-case conditions.

Performance of novel LZD		
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

Table 5. Performance of the new LZD circuit under nominal and worse-case condition

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]
163X340 86mils	7.7	4.5	12.8	7.95	186X403 116mils	12	5.8	181X473 133mils	13.6	7.7

Table 6. Comparison of the Algorithmic LZD and Logic Synthesis results

In Table 6. we compare the results for a 32-bit LZD using our approach and 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 no load conditions.
Under 1.0pF load LS obtained LZD is slightly faster under nominal conditions and this is because we are comparing the regular layout with the rectangular (buffered) one. In any case it has been demonstrated that our Algorithmic LZD circuit outperforms the LS in each of the cases and the improvement in performance ranges from 10% (NC rectangular layout) to 36% (WC regular layout). The improvement in the area is from 13% (rectangular layout) up to 26% (regular layout). This clearly demonstrates superiority of the Algorithmic approach. In Table 7. we compare the results for a 32-bit LZD using rectangular layout versus one with unconstrained layout under load and no-load conditions. We observe that under the no load conditions the LZD circuit obtained via regular layout performs better. However with 1.0pF 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.

4. Conclusion

In this paper we have described an Algorithmic approach to design Leading Zero Detector. This circuit has been implemented in 0.6μ CMOS technology and compared to the results obtained using Logic Synthesis under various conditions and for different layout approaches. The Algorithmic approach outperformed LS consistently ranging from 10% - 36% in terms of the performance and 13% - 26% in terms of the layout area. We have clearly demonstrated on 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, LS 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

I gratefully acknowledge Vincent Chang for running the simulation and timberwolf program. The idea was conceived 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 Monty Denneau from IBM for his ideas and discussion during the project.

References

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

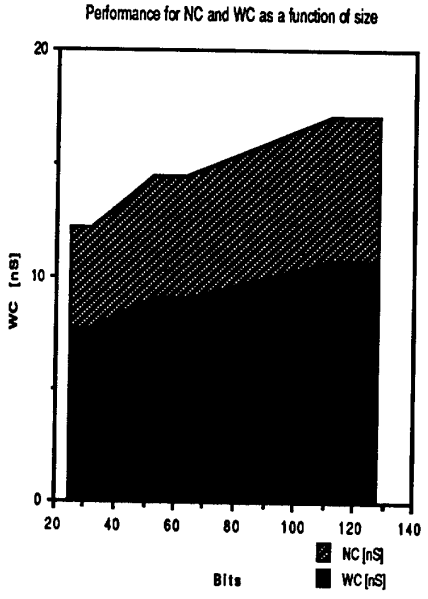

Fig. 4. Speed of the novel LZD vs size

Comparison of the Algorithmic LZD circuit obtained via regular vs rectangular layout									
Algorithmic LZD (regular layout)					Algorithmic LZD (rectangular layout)				
	no load		1.0 pF load			no load		1.0 pF load	
Area [u]	WC [nS]	NC [nS]	WC [nS]	NC [nS]	Area [u]	WC [nS]	NC [nS]	WC [nS]	NC [nS]
163X340 86mils	7.7	4.5	12.8	7.95	206X363 116mils	9.5	4.93	11.4	6.9

Table 7. Performance comparison of the regular vs rectangular layout of the algorithmic LZD