6 ZELDOVICH, B. YA., PILIPETSKII, N. F., and SHKUNOV, V. V.: 'Principles of phase conjugation' (Springer Optical Sciences, Springer-Verlag, Berlin, 1985), Vol. 42

7 HÖÖK, A., and BOLLE, A.: 'Transient dynamics of stimulated Brillouin scattering in optical communication systems', *J. Lightwave Technol.*, 1992, **LT-10**, pp. 493–502

ALGORITHMIC DESIGN OF A HIERARCHICAL AND MODULATOR LEADING ZERO DETECTOR CIRCUIT

V. G. Oklobdzija

Indexing terms: Digital arithmetic, Microprocessors

A novel way of implementing the leading zero detector (LZD) circuit is described. The implementation is based on an algorithmic approach resulting in a modular and scalable circuit for any number of bits. This approach to LZD design yields both speed and area advantages over logic synthesis.

Introduction: In any floating-point processor, normalisation is a common 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 special circuit to detect the number of leading zeros is referred to as the leading zero detector (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 a 64 bit word, consists of 64 inputs. It is obvious that such large fan-in dependences are a problem and that the resulting circuit is likely to be complicated and slow. Designing such a circuit using Boolean minimisation or Karnaugh maps is cumbersome and slow. On the other hand, this circuit is a very good candidate for logic synthesis (LS) because its VHDL description is concise and clear. However, LS tools have not yet reached a level of sophistication at which they can deal with hierarchical structures and create hierarchy in the design. Rather their approach is to expand the logic in one level and optimise it via elaborate and laborious logic minimisation using hours of the CPU time. Therefore the design presented in this Letter results not only in an efficient and fast LZD but also provides an estimate and understanding as to what the logic synthesis tools can and cannot do.

Design procedure: In this approach the inherent hierarchy associated with the leading zero detection process is used and it is translated into a hierarchical and modular design. To understand this design process, let us begin by examining only the first two bits, as shown in Table 1.

It is very straightforward to construct the logic for the two bits representing valid bit V and position bit P as shown in Table 1. The logic for the two bits is trivial.

Extension of the 2 bit case into the 4 bit case is shown in Table 2. We designate the position bits (of four bits total) as P_1 for the left-most two bits and P_r for the right-most two bits. Also we will designate V_1 and V_r as the valid bits for the first two and second two bits, respectively, moving from left to right. The LZ position can be represented as a function of those four bits as shown in the fifth column of Table 2 (the minus sign represents the complement). The 4 bit circuit has two logic levels, and in the second level a valid bit is formed as the 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, that simply means that they are all 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 the P_1 if $V_1 = 1$, or with P_r if $V_1 = 0$ and $V_r = 1$. This is achieved by simply multiplexing P_1 and P_r to the output of the multiplexer. This structure is shown in Fig. 1. However, a 4 bit LZD can be implemented in one level of logic, simply from Table 2. However, this exercise was carried out for tutorial purposes, to illustrate the concept. Naturally, the implementation depends on the technology used; even more bits can be compressed into one level, or one logic tree.

Table 1 TRUTH TABLE FOR TWO BITS

Pattern	Position	Valid
1X	0	Yes
01	1	Yes
00	X	No

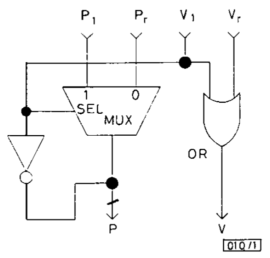


Fig. 1 *Structure of group*

We can take two groups of four bits and form an LZD for an 8 bit word, simply following the same concept as for the example of four bits. From Tables 1 and 2 we can deduce the hierarchical structure for the LZD and arrive at the following algorithm for generating the number of leading zeros:

Table 2 TRUTH TABLE FOR 4 bit LZD

Pattern	Position	Position (binary)	Valid	Position
1011	0	00	Yes	$-V_1/P_1$
0100	1	01	Yes	$-V_1/P_1$
0011	2	10	Yes	$-V_1/P_r$
0001	3	11	Yes	$-V_1/P_r$
0000	4	XX	No	XX

Algorithm for generating LZ count:

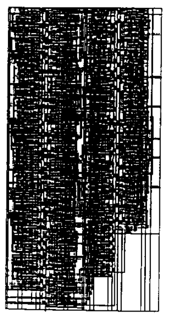
(1) form the pair of bits B_i, B_{i+1} for $i = 0$ to $N - 2$ with bit 0 being the leftmost one

(2) determine P and V bits for each pair

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

```
    $V_g = V_l + V_r$ where $+$ is the logical OR operation
    if $V_l = 1$ then $P_g = 0/P_l$ ('/' is concatenation)
        else if $V_r = 1$ then $P_g = 1/P_r$
            otherwise $V_g = 0$
repeat step 3 log $(N) - 2$ times
```

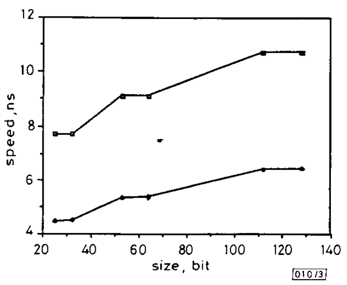
The logical depth of this circuit is $\log_2(N)$ stages where the pass through each stage is of the complexity of the multiplexer or equivalent to one level of logic. In many CMOS circuits the multiplexer is actually implemented using a pass-transistor structure and therefore is even faster than a regular CMOS gate. This is the reason for the extraordinary speed of this scheme for the LZD. In CMOS any further compression of the number of levels would pass the point of diminishing returns as far as speed is concerned. Using the same concept on groups of four bits instead of two bits, in ECL technology, this concept shows an additional advantage. In this case the speed of this LZD implementation is of depth $\log_4(N)$ stages and is even faster.



010/2

Fig. 2 *Layout of 32 bit LZD for the algorithmic design*

Implementation and performance: 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 an LZD in the regular data path of the floating-point or any other unit that needs an LZD circuit. The layout of the 32 bit LZD is shown in Fig. 2 for the algorithmic design. The algorithmic layout has four rows of cells which were placed using the Timberwolf package resulting in an area of $163 \times 340\,\mu m^2$. The results obtained using LS has a 35% larger area, $186 \times 403\,\mu m^2$. In terms of speed, the algorithmic LZD introduces a delay of $T_{al} = 4{\cdot}5$ nS for the typical case in $0{\cdot}6\,\mu m$ CMOS technology. The LZD resulting from the logic synthesis has a delay $T_{ls} = 5{\cdot}8$ ns (both for the typical case) which is 29% slower. Therefore, we can say that the algorithmically designed LZD is roughly 1/3 smaller and faster than the equivalent LZD resulting from the LS. The performance of the novel LZD is shown for the nominal and worse case conditions, NC and WC, respectively, in Fig. 3.



010/3

Fig. 3 *Speed of novel LZD against size*

—■— WC (nS)
—◆— NC (nS)

Conclusion: This circuit has been implemented in $0{\cdot}6\,\mu m$ 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 12 to 56% in terms of the performance and 14·5 to 35% in terms of the layout area. We have clearly demonstrated with this circuit the superiority of the algorithmic approach.

Acknowledgment: I thank V. Chang for running the simulation and the 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 M. Denneau of IBM for his ideas and discussion during the project.

16th November 1992

V. G. Oklobdzija (*Department of Electrical and Computer Engineering, University of California, Davis, CA 95616, USA*)

PHOTONIC 2 × 2 PACKET SWITCH WITH INPUT BUFFERS

J. Spring and R. S. Tucker

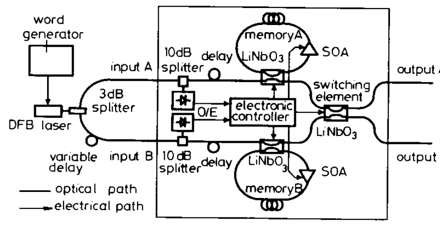
Indexing terms: Optical switching, Optical communications

A self-routing 2 × 2 photonic packet switch with two fibre-loop input buffers that provide output contention resolution is demonstrated. The switch uses high-speed electronic control that prioritises all switching and buffer operations and guarantees packet integrity while maximising throughput.

Introduction: Packet switching is established as a telecommunications standard. Self-routing photonic packet switches will be important components for future all-optical networks. There have been a number of recent demonstrations of self-routing photonic packet switches [1, 2]. These demonstrations have highlighted the fact that the information content (payload) of a packet is not limited by electronics in an optically transparent switch [1].

A major limitation of previously reported photonic packet switches is that they are unable to resolve output contention events that occur when input packets are simultaneously targeted for the same output. Eiselt *et al.* [3] demonstrated output contention resolution using a fibre loop at one input of a 2 × 2 packet switch. However, when a packet is stored in the fibre loop the associated input cannot receive new packets. The Eiselt experiments could only handle synchronous packet arrivals. Photonic switches will, in general, have asynchronous packet arrivals, and for this reason it is critical that the switching node be able to switch input packets and resolve output contention regardless of the arrival time of the packets and regardless of previous packet flow through the switch. This Letter demonstrates a new 2 × 2 photonic packet switch with two fibre-loop input buffers (memories) that enable output contention to be resolved. The new 2 × 2 switch handles fully asynchronous traffic. It uses high-speed electronic control that prioritises all switching and memory operations and guarantees packet integrity while maximising throughput. We believe that this is the first demonstration of a self-routing photonic packet switch that resolves output contentions and operates asynchronously.

Experimental setup: Fig. 1 shows the architecture of the experimental 2 × 2 packet switch. A directly modulated DFB



2477/1

Fig. 1 *Architecture of experimental 2 × 2 packet switch*