Design of Leading Ones or Zeros Counting Circuit

Nebojsa Z. Milenkovic[1], Vladimir V. Stankovic[2]

Abstract – **Design of leading ones or zeros counter in data represented as strings of binary digits is presented in this paper. The proposed design method is applicable to data length of 4k bits, for 4≤ k ≤16. For longer data length, multiple copies of the designed counter can be used with addition of very simple circuits. The proposed design enables very high speed implementation in contemporary technologies.**

Keywords – **Leading zeros/ones count, detection, modular design.**

I. INTRODUCTION

In computer technique, it is frequently necessary to count leading ones or zeros in some data reprezented as strings of binary digits. Normalization of significand in the floating point arithmetic is maybe the most famous example. Techniques that are used for speeding up counting of leading ones or zeros can also be used for encoding of leading zeros anticipator in floating point arithmetic. Counting of leading digits of the divisor may be neded for some fixed point divide algorithms. Some processors, for example the MIPS family processors, in their instruction sets have special instructions for counting leading ones and leading zeros in 32 bit integer data.

Counting or detection of leading zeros and/or leading ones has been considered separately [1,4,5] or in framework of leading zeros anticipation [2,3]. Solution which we have proposed here is improvement of previous solution in including counting leading zeros or leading ones by our choice, and network with lower number of logic levels, with lower propagation time.

In section II we have explained the method of synthesis of leading ones/zeros counter. Section III contains performance evaluation of proposed solution. Section IV contains conclusion, and section V contains used references.

II. SYNTHESYS OF LEADING ONES/ZEROS COUNTER

Leading zeros are zeros in most significant positions of data, up to the position in which first one is present. Analogous, leading ones are ones in most significant positions of data, up to the position in which first zero is present. These definitions may by presented in the forms of 0^k1x^* for k leading zeros, and 1^k0x^* for k leading ones, where x is either 0 od 1, the superscripts reprezent k instances of digit 0 or 1, and * reprezents zero or more instances of digit x. For example, in the binary datum 00001xxx....xxx the count of leading zeros is four, and in the datum 110xxx...xxx the count of leading ones is two.

We will consider the 32 bit data, but the solution that we will present here is applicable to data length of 4k bits, for $4 \le k \le 16$. We will explain the switching network design, which for 32 bit data, under our choice, counts leading ones or zeros.

Let us divide the 32 bit data X(31:0) to 4 bits nibbles. In this data bit 31 is MSBit and bit 0 is LSBit.

X:31÷28	27÷24	23÷20	19÷16	15÷12	11÷8	7÷4	3÷0
p_0,V_0	p_1,V_1	p_2,V_2	p_3,V_3	p_4,V_4	p_5,V_5	p_6,V_6	p_7,V_7
s_0,Z_0	s_1,Z_1	s_2,Z_2	s_3,Z_3	s_4,Z_4	s_5,Z_5	s_6,Z_6	s_7,Z_7

Labels below nibbles, p_i and s_i , $0 \le i \le 7$, denotes logical products (AND operation) and complements of logical sums (NOR operation) of data bits in i^{th} nibble, and V_i and Z_i denotes count of leading ones and zeros in that nibble.

General expression for p_i is

$$p_i = x_{31-4i} \cdot x_{31-(4i+1)} \cdot x_{31-(4i+2)} \cdot x_{31-(4i+3)} \text{ , } i=0,1,...,7 \quad (1)$$

TABLE I Boundary nibble encoding as function of products

Values of logical products	Ordinal numb. of boundary nibble, n	Count of leading ones, m	Selector bits to mux $y_2\ y_1\ y_0$
$p_0=0$	0	0 ÷ 3	0 0 0
$p_0=1,\ p_1=0$	1	4 ÷ 7	0 0 1
$p_0 \cdot p_1=1,\ p_2=0$	2	8 ÷ 11	0 1 0
$p_0 \cdot p_1 \cdot p_2=1,\ p_3=0$	3	12 ÷ 15	0 1 1
$p_0 \cdot p_1 \cdot p_2 \cdot p_3=1,\ p_4=0$	4	16 ÷ 19	1 0 0
$p_0 \cdot p_1 \cdot p_2 \cdot p_3 \cdot p_4=1,\ p_5=0$	5	20 ÷ 23	1 0 1
$p_0 \cdot p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5=1,\ p_6=0$	6	24 ÷ 27	1 1 0
$p_0 \cdot p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 \cdot p_6=1,\ p_7=0$	7	28 ÷ 31	1 1 1
$p_0 \cdot p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 \cdot p_6 \cdot p_7=1$	-	32	0 0 0

In Table 1 the expression in column "Values of logical products" contains expressions which determine the ordinal number of nibble, counting from zero from left to right, in which continuous string of ones terminates, followed by first zero. Such nibble we will call *boundary nibble*. In this column character "·" designates AND operation. The last table's row refers to the data with number of ones equal to the length of data, in this case 32. In the column "Number of leading ones" are shown the ranges of possible numbers of leading ones for these conditions. These numbers of leading ones can be found as the sum of values 4n and V_n, where $0 \le n \le 7$ is the ordinal number of boundary nibble with one or more zeros, and V_n is the number of terminal ones in this boundary nibble. For example, in datum

1111 1111 1111 1100 1001 1111 0111 1000

[1]Nebojsa Z. Milenkovic is with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia and Montenegro, E-mail: nmilenko@elfak.ni.ac.yu

[2]Vladimir V. Stankovic is with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia and Montenegro, E-mail: svlada@elfak.ni.ac.yu

the first three nibbles contain only ones, while the fourth nibble beside two ones also contains two zeros. That's why the nibble with ordinal number three is boundary nibble, with two terminal ones, thence n = 3 and $V_n = 2$, and the number of leading ones is m=14.

$$m = 4n + V_n \tag{2}$$

Last column of Table I encodes ordinal number of boundary nibble from second column, and will be used as the selector bits to multiplexor MUX 1 and MUX 2 in the network presented in Figure 3.

Because the values of n in range 0 – 7 are multiplied by 4, and V_n can have values in range 0 – 3, multiply operation n by 4 and adding V_n can be substituted by concatenation of three bits of n and two bits of V_n.

From first and fourth columns of Table I we can get following expressions for y_2, y_1, and y_0:

$$\begin{aligned} y_2 &= p_0p_1p_2p_3\overline{p_4p_5p_6p_7} \\ y_1 &= p_0p_1(\overline{p_2p_3} + p_4p_5 \cdot \overline{p_6p_7}) \\ y_0 &= p_0(\overline{p_1} + p_2\overline{p_3}) + p_0p_1p_2p_3p_4(\overline{p_5} + p_6\overline{p_7}) \end{aligned} \tag{3}$$

In fact, columns 1 and 4 of Table I define function of priority encoder with eight input lines and three output lines, as can be seen in standard integrated circuits families.

Dependence of the number of leading ones in the boundary nibble, which contains less then four ones, is presented in Table II. The last table's row is an exception, because it represents the case when the nibble is not boundary. To simplify bit's labels we have introduced index k, k = 31-4i, i=0, …, 7. Value $V_i = \{v^i_1\ v^i_0\}$ represents this number of leading ones in binary. From this table logical expressions for v^i_1 and v^i_0 are:

TABLE II Encoding leading ones in boundary nibble

x_k	x_{k-1}	x_{k-2}	x_{k-3}	v^i_1	v^i_0
0	x	x	x	0	0
1	0	x	x	0	1
1	1	0	x	1	0
1	1	1	0	1	1
1	1	1	1	0	0

$$\begin{aligned} v^i_1 &= x_k \cdot x_{k-1} \cdot \overline{x_{k-2} \cdot x_{k-3}} \\ v^i_0 &= x_k \cdot (\overline{x_{k-1}} + x_{k-2} \cdot \overline{x_{k-3}}) \end{aligned} \tag{4}$$

Number of leading zeros can be found by the same way if for every nibble we take NOR logical operation on their bits:

$$s_i = \overline{x_{31-4i} + x_{31-(4i+1)} + x_{31-(4i+2)} + x_{31-(4i+3)}},\ i = 0,1,..,7 \tag{5}$$

Then s_i =1 indicate that i^{th} nibble contains all zeros. As with counting leading ones, ordinal number of boundary nibble and range of count of leading zeros in relation to values of s_i can be presented as in Table I, with only one difference: in first column, instead of products of p_i, in that table must stand products of s_i, $0 \le i \le 7$. Insufficient space and only that difference are reasons why we don't present such table for leading zeros.

Dependence of the number of leading zeros in the boundary nibble, which contains less then four zeros, is presented in Table III. The last table's row is an exception, because it represents the case when the nibble is not boundary. Value $Z_i = \{z^i_1\ z^i_0\}$ represents this number of leading zeros in binary. From this table logical expressions for z^i_1 and z^i_0 are:

$$\begin{aligned} z^i_1 &= \overline{x_k} \cdot \overline{x_{k-1}} \cdot (x_{k-2} + x_{k-3}) \\ z^i_0 &= \overline{x_k} \cdot (x_{k-1} + \overline{x_{k-2}} \cdot x_{k-3}) \end{aligned} \tag{6}$$

TABLE III Encoding leading zeros in boundary nibble

x_k	x_{k-1}	x_{k-2}	x_{k-3}	z^i_1	z^i_0
1	x	x	x	0	0
0	1	x	x	0	1
0	0	1	x	1	0
0	0	0	1	1	1
0	0	0	0	0	0

Values in columns 2 to 4 of Table I and analogous table for leading zeros are equal, and in column 1 they differ only by using AND or NOR operation on nibble's bits in data. This suggests that logical function y_2, y_1 and y_0 in both tables can be implemented by the same combinatorial network, with inputs p_i for leading ones and s_i for leading zeros. Let q_i be the input signal to that network with values:

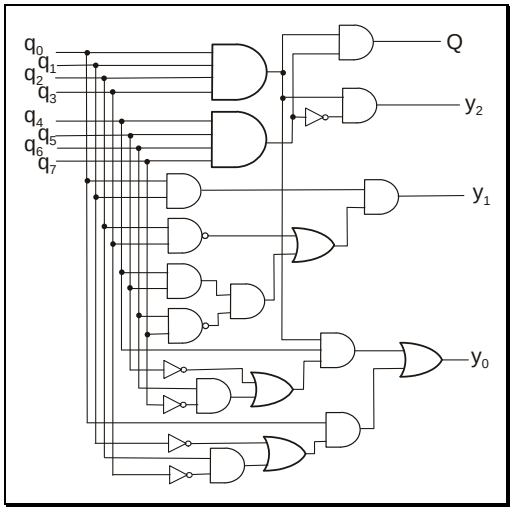

Fig. 1. Scheme of BNE network

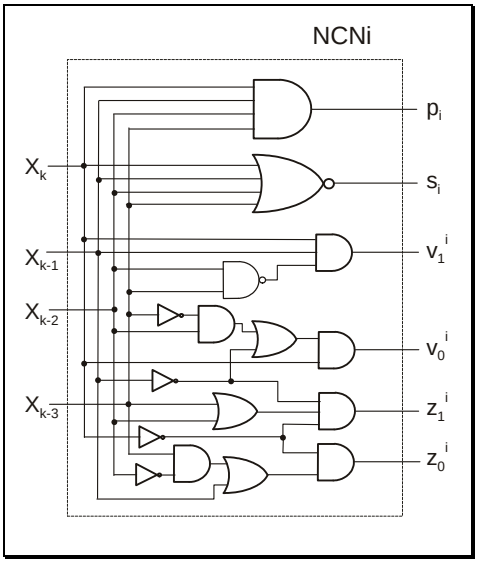

Fig. 2. Scheme of NCNi network

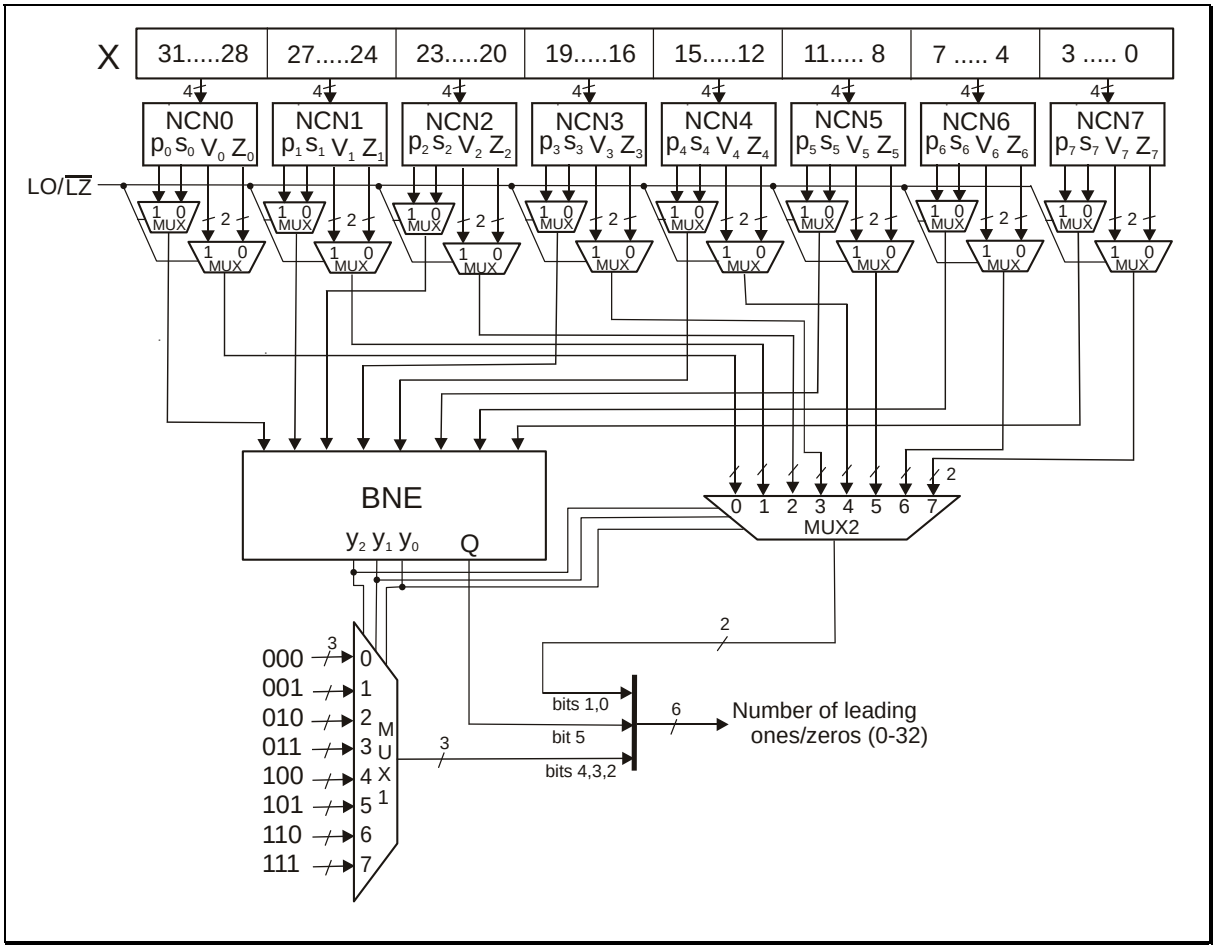

Fig. 3. Block scheme of leading ones/zeros counter in 32-bit data

$q_i = p_i$ for counting leading ones,

$q_i = s_i$ for counting leading zeros.

Combinatorial network which implements the switching functions $y_j = f_j(q_0, q_1, ..., q_7)$, j= 2, 1, 0, is shown in Figure 1. In addition, this network implements the logical product $Q=q_0 \cdot q_1 \cdot q_2 \cdot q_3 \cdot q_4 \cdot q_5 \cdot q_6 \cdot q_7$, which by value 1 signalizes that the datum in all eight nibbles has all ones or zeros.

Figure 2 shows the network which implements switching functions p_i and s_i from expressions 1 and 5 respectively, and z^i_1, z^i_0 and v^i_1, v^i_0 from expressions 4 and 6 respectively.

Finally, Figure 3 shows the complete block scheme of the leading ones/zeros counter. Blocks NCN0 ÷ NCN7 contain logical networks shown in Figure 2, and block BNE (Boundary Nibble Encoder) logical networks shown in Figure 1. $LO/\overline{LZ}$ selects counting leading ones or zeros. Stout line on leading ones/zeros counter's output represents grouping of one output line Q from BNE network, three lines from MUX1 and two lines from MUX2, which carry bits 5, 4:2 and 1:0 of the leading ones/zeros count's six bits.

This solution can be extended for counting leading ones/zeros in 64 bit data as follows. Let the network presented in Figure 3 with outputs "Number of leading ones/zeros (0-32)" be one module with outputs NLO/Z (bits 4:0) and Q. Two such modules are connected to the outputs of 64 bit register X as is presented in Figure 4. Signal Q_H=0 shows that data bits 63 to 32 contains not only ones (zeros), and the count of leading ones (zeros) in range 0÷31 is determined by $00||NLO/Z_H$. Here 00 are two most significant bits, and NLO/Z_H gives five lower bits in the seven bits count of leading ones/zeros. The sign || is concatenation. When Q_H=1 and Q_L=0, most significant 32 data bits are only ones (zeros), and 32 lower data bits contains ones and zeros. The number of leading ones (zeros) is in the range 32÷63, and is determined by $01||NLO/Z_L$. Finally, for Q_H=1 and Q_L=1 all data bits are ones (zeros), and the number of leading ones (zeros) is 64. These values are written in the third column of Table IV. They are put on MUX's data input lines as presented in Figure 4, and MUX's seven output lines give us the number of leading ones/zeros in the range 0-64.

TABLE IV Forming results with output multiplexer

Q_H Q_L	Number of leading ones/zeros	The way of forming	MUX's select inputs: r_1 r_0	MUX's data input lines
0 x	0 ÷ 31	$00\|\|NLO/Z_H$	0 0	0
1 0	32 ÷ 63	$01\|\|NLO/Z_L$	0 1	1
1 1	64	1000000	1 0	2

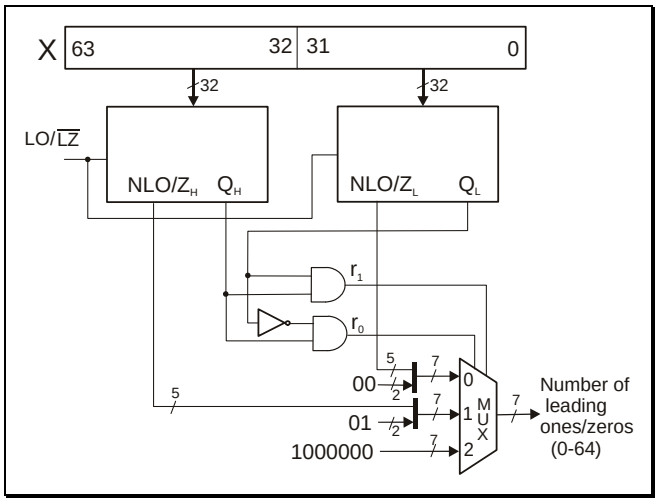

Fig. 4. Block scheme of leading ones/zeros counter in 64 bit data

III. PERFORMANCE EVALUATION

As the building block of microarchitecture of contemporary processors, proposed leading ones/zeros counter must satisfy required performance level. In performance evaluation we are considering only the time delay through the proposed leading ones/zeros counter. In the context of discussions of logic delays, the FO4 metric is used by processor architects as a process neutral metric that can be applied to abstract design and architectural discussions. So, we are using FO4 (equivalent) gate delays as a measure [6], which is simply the delay through an inverter that has to provide the output drive current sufficient to drive 4 other inverters of comparable sizes. In addition, it must be counted the maximal number of level of logical circuits in the considered network. This number of level then must be multiplied with the delay time per logical circuit to find the delay time through the entire network.

In the network in Figure 3 signals propagate through the following blocks: NCNi, MUX, BNE, (MUX1, MUX2 in parallel). From Figures 1 and 2 the maximal numbers of levels for NCNi and BNE are 4 and 5 respectively. For MUX below NCNi blocks we find 3 logic levels, and for MUX1 we find 4 logic levels. Adding these numbers gives 16 logic levels. With FO4 gate delay of 25 ps for 0.18μm process technology [6], the delay time for the proposed leading ones/zeros counter for 32 bit data is 16×25=400 ps. For 64-bit leading ones/zeros counter to this time delay must be added time delay through MUX select logic r_1r_0 and MUX, with 2+3=5 logic level, and additional 5×25=125 ps. For the whole 64-bit leading ones/zeros counter time delay is 525 ps. Of course, with contemporary process technology with shorter FO4 per gate delay time, the performance presented here is even better.

IV. CONCLUSION

Systematic design of leading ones/zeros counter is presented in this paper. Although example design presented here is for 32 bit data, the methodology used for them is applicable to design of leading ones/zeros counter for data length of 4k bits, for $4 \le k \le 16$.

The advantages of this leading ones/zeros counter are:

- choice to count leading ones or zeros,
- relatively little quantity of logic circuits attained by using common logic blocks (BNE, MUX1, MUX2) for counting leading ones and zeros,
- modular design which enables simple upscaling for longer data,
- short time delay, with little additional delay with upscaling.

REFERENCES

[1] V.G.Oklobdzija, "An Algorithmic and Novel Design of a Leading Zero Detector Circuit: Comparison with Logic Synthesis", *IEEE Trans. VLSI Systems*, vol. 2, no1, March 1994., pp. 124-128.

[2] M.S.Schmookler and K.J.Novka, "Leading Zero Anticipation and Detection- A Comparison of Methods" *Proc. 15th IEEE Symposium on Computer Arithmetic*, 2001, pp. 7-12.

[3] F. Arakawa, T. Hayashi, and M. Nishibori, "An Exact Leading Non-Zero Detector for a Floating-Point Unit", *IEICE Trans. Electron.*, vol.E88–C, no.4 April 2005, pp.570-575.

[4] M.Ott,"Optimized method and apparatus for parallel leading zero/one detection", US Patent #6697828B1, Feb. 2004.

[5] S.-H.Yin,M.K.Gowan,"High speed leading or trailing bit value detection", US Patent #7012965B2, Mar. 2006.

[6] D.T.Wang,"Revisiting the FO4 metric", http://www.realworldtech.com/page.cfm?ArticleID=RWT081502231107&p=2