Zahir, Ali; Ullah, Anees; Reviriego, Pedro; Hassnain, Syed Riaz Ul (2022). Efficient Leading Zero Count (LZC) Implementations for Xilinx FPGAs. *IEEE Embedded Systems Letters*, 14(1), pp.: 35-38.

DOI: https://doi.org/10.1109/LES.2021.3101688

Efficient Leading Zero Count (LZC) Implementations for Xilinx FPGAs

Ali Zahir[1], Anees Ullah[2], Pedro Reviriego[3] and Syed Riaz Ul Hassnain[2]

***Abstract*—Leading Zero Count (LZC) is a fundamental building block in floating point arithmetic and data sketches. These applications are increasingly being implemented on Field Programmable Gate Arrays (FPGAs), however, existing architectures for LZC target ASICs and to the best of authors knowledge specific LZC implementations tailored to FPGA structures have not been presented. In this paper, the implementation of LZC on Xilinx FPGA is considered and it is shown that by carefully adapting the LZC design to the FPGA structure, more efficient implementations can be obtained. In more detail, LZC designs for different bit-widths are presented and evaluated. The results show that significant reductions in the FPGA resources needed are obtained that reach 33% LUTs saving for 32 bit vectors and 20% LUTs saving for 64 bit vectors.**

***Index Terms*—Leading Zero Count, FPGAs, HyperLogLog, Floating point arithmetic**

I. Introduction

Counting the number of leading zeros (or ones) in a binary vector is an important operation to optimize the implementation of floating point arithmetic units by using Leading Zero Count (LZC) and has been studied for decades [1]. The work to implement and further optimize the LZC continues with the use for example of approximate circuits to reduce the complexity or error detection schemes [2], [3]. More recently, counting the number of leading zeros has become an important operation in Big Data applications as it is utilized in some key sketches which are used to process data streams, for example, the HyperLogLog cardinality estimation sketch [4] and its extensions to support multiple streams [5] or in the Hyperminhash sketch for similarity estimation [6].

Therefore, Leading Zero Counting is an important operation and several schemes have been proposed to implement it efficiently, for example, [7], [8]. In [8], a mathematical frame work is presented which uniformly reduces the equations of LZC to carry-lookahead operations. Similarly, [7] modifies the work of [8] by using complex gates to improve the speed of the LZC. An important point is that all previous works on LZC design target Application Specific Integrated Circuit (ASIC) implementations and are not optimized to map efficiently to modern FPGA fabrics.

Field Programmable Gate Arrays (FPGAs) have evolved in to complex System on Chips (SoC) that integrate processors, high speed I/Os and a vast amount of logic and memory resources which makes them attractive for many applications [9]. In particular, big data processing sketches can be implemented on FPGAs [10] and the same applies to floating point arithmetic units [11].

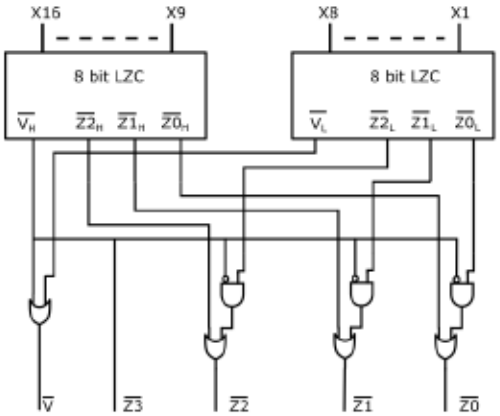

Fig. 1. Structure of the 16 bit LZC [8]

Therefore, it is of interest to implement LZC on FPGAs as specific circuits can be adapted to the FPGA structure to obtain more efficient implementations. This, for example, has been done for Ternary Content Addressable Memory (TCAM) emulation on FPGAs where even reconfiguration can be exploited to achieve better implementations [12]. However, to the best of our knowledge there is no previous work on optimizing the implementation of LZC on FPGAs. In this paper, the implementation of LZC on Xilinx FPGAs is considered and efficient designs are proposed and evaluated. The proposed scheme is tailored to Xilinx's FPGA fabric and thus its evaluation is only relevant for Xilinx FPGAs. However, the results are valid for all series 7 FPGAs that are widely used in real designs. For example, the Hyperloglog acceleration presented in [10] uses Xilinx's FPGAs.

The rest of the paper is organized as follows. In section II, the background on LZC implementation and existing designs is covered. The proposed LZC tailored to Xilinx FPGAs is presented in section III and evaluated in section IV. The paper ends with the conclusion and some ideas for future work in section V.

II. Leading Zero Count (LZC) Implementation

Leading zero is defined as the number of consecutive zeros starting from the Most Significant Bit (MSB) up to the first non-zero digit in a binary number. A LZC unit has n bits of input data $X_n, X_{n\ 1}, X_{n\ 2} \ldots\ldots X_1$, where X_n is the MSB. The output of the LZC unit consists of $Log_2(n)$ bits containing the

[1]Ali Zahir is with COMSATS University Islamabad, Abbottabad Campus, Pakistan (e-mail: alizahir@cuiatd.edu.pk).

[2]Anees Ullah and Syed Riaz Ul Hassnain are with University of Engineering and Technology, Peshawar, Abbottabad Campus, Pakistan (Corresponding Author e-mail: aneesullah@uetpeshawar.edu.pk).

[3]Pedro Reviriego is with Universidad Carlos III de Madrid, Leganés 28911, Madrid, Spain. (e-mail: revirieg@it.uc3m.es).

leading zero count Z and a flag V. If all bits of the input data are zero the flag V is set to 1, for the rest of the combinations, the value of Z indicates the number of leading zeros. For example, for 8-bit input data 00001010, the output of LZC ($VZ_2Z_1Z_0$) is $(0100)_2$. Similarly, for the all zero combinations of input data, the output is $(1000)_2$.

The first method for determining the leading-zero count is based on one hot representation [11]. In this representation, the intermediate string S is produced in which the bits are marked as one up to the position of leading one of a word and the remaining most significant bits are kept as zero. The string S increases monotonically from 00 to 01 to 11. It can never have a value equal to 10. Exploiting this monotonically increasing property of the string S, authors in [8] simplify the equations for Z_i and V as

$$\overline{V} = X_8 + X_7 + X_6 + X_5 + X_4 + X_3 + X_2 + X_1 \quad (1)$$

$$\overline{Z_2} = X_8 + X_7 + X_6 + X_5 \quad (2)$$

$$\overline{Z_1} = X_8 + X_7 + \overline{X_6}.\overline{X_5}(X_4 + X_3) \quad (3)$$

$$\overline{Z_0} = X_8 + \overline{X_5}.X_6 + \overline{X_7}.\overline{X_5}.X_4 + \overline{X_7}.\overline{X_5}.\overline{X_3}.X_2 \quad (4)$$

In the above equations, Z_0 determines the leading zero count as an odd or even number. The authors in [8] further exploit this property to design a mathematical framework for a simplified prediction circuit. The computational complexity of the prediction circuit resembles a well known carry-lookahead technique in a unified manner. The work was further modified in [7] by replacing equations (1) and (2) with complex gates (see equations (5) to (8)) to reduce the delay of the LZC structure.

$$\overline{V} = \overline{\overline{X_8 + X_7}.\overline{X_6 + X_5}.\overline{X_4 + X_3}.\overline{X_2 + X_1}} \quad (5)$$

$$\overline{Z_2} = \overline{\overline{X_8 + X_7} + \overline{X6 + X_5}} \quad (6)$$

$$\overline{Z_1} = \overline{\overline{X_7 + X_6}.[(X_5 + X_4) + \overline{X_3 + X_2}]} \quad (7)$$

$$\overline{Z_0} = \overline{\overline{X_8 + \overline{X_5}.X_6}.[X_7 + X_5 + \overline{X_4 + (\overline{X_3}.X_2)}} \quad (8)$$

The two LZC structures are modular and regular. Larger LZC units can be constructed by using several smaller units as building blocks. Fig.1 shows the construction of a 16-bit LZC unit. The 16-bit LZC consists of two smaller 8-bit LZC units and a combining logic (consists of basic gates). In our proposed methodology, we present a new approach to implement the basic 8-bit LZC unit and combining logic by keeping the structure of 7 series Xilinx FPGA into consideration.

III. Proposed Implementation

This section presents the proposed method for implementing the Leading Zero Count on 7-series Xilinx FPGAs. The main logic resource used for implementing LZC is the on-chip dual-port LUTs that are located in all the SLICEs of the Xilinx FPGAs. Each slice contains four Look-Up Tables (LUTs) labeled as LUTA, LUTB, LUTC, and LUTD. These LUTs can be configured either as 6-input LUT with single output, or as two 5-input LUTs with two outputs. As in most existing designs, our proposed scheme uses an 8-bit LZC as building block. Therefore, first the proposed 8-bit LZC design is presented and then generalised to 16 bit and 32 bit LZCs. The 8-bit LZC is designed to fit in a single slice of Xilinx FPGAs which makes it efficient and easy to use as a building block for larger LZC units.

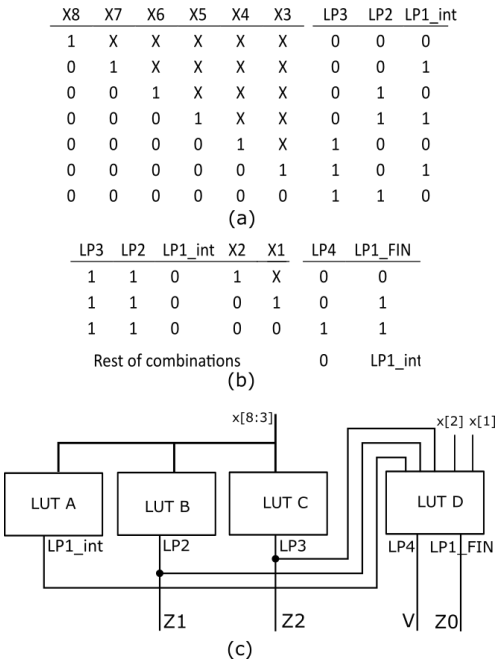

Fig. 2. Structure of the proposed 8 bit LZC: (a) truth table implemented by LUTA,LUTB and LUTC, (b) truth table implemented by LUTD, (c) overall diagram of the proposed 8-bit LZC module

In our approach, the six most significant bits of the input data i.e. $X_8, X_7, X_6, X_5, X_4, X_3$ are used to determine the partial count value bits of LP1_int, LP2 and LP3 as shown in Fig. 2(a). Three LUTs; LUTA, LUTB and LUTC, are configured as 6x1 LUTs to produce the partial count values as shown in Fig.2(c). LUTA is configured with FA, LUTB is configured with FB, and LUTC is configured with FC, as given in (9), (10) and (11), respectively. In this way a six input priority encoder is implemented to calculate the number of consecutive zeros in the input data.

$$LP1_{int} = FA = X_3.\overline{X_4}.\overline{X_6}.\overline{X_8} + X_7.\overline{X_8} + X_5.\overline{X_6}.\overline{X_8} \quad (9)$$

$$LP2 = FB = (\overline{X_3}.\overline{X_4} + X_5 + \overline{X_5}.X_6).(\overline{X_7}.\overline{X_8}) \quad (10)$$

$$LP3 = FC = \overline{X_5}.\overline{X_6}.\overline{X_7}.\overline{X_8} \quad (11)$$

The remaining two least significant bits X_2 and X_1 can only modify the value of the final count when $LP1_{int}LP1LP2$ are 110. To accommodate the effect of X_1 and X_2, we use another table as shown in Fig.2 (b). To implement this table, we use another LUT (LUTD) in 5x2 configuration as shown in Fig.2. Inputs to LUTD are $LP1_int, LP2, LP3, X_2$ and X_1.The O5 output of LUTD is configured with FD1 and O6 with FD2 using (12) and (13), respectively. Finally, we assign V=LP4, Z2=LP3, Z1=LP2 and Z0=LP1. It is worth to mention here that the value of count 8 will be 1111 instead of 1000. However, when the flag V is "1" the count is considered as 8 and the value of Z does not matter.

$$LP1 = FD1 = LP1_{int} + \overline{X_2}.LP2.LP3 \quad (12)$$

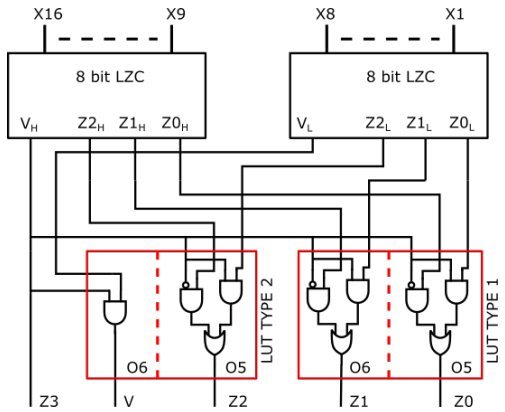

Fig. 3. The proposed structure for a 16-bit LZC

$$LP4 = FD2 = \overline{X_1}.\overline{X_2}.\overline{LP1_{int}}.LP2.LP3 \tag{13}$$

This proposed design of the 8-bit LZC matches perfectly the structure of the FPGA fitting on a single Slice and using all its LUTs.

To implement a 16-bit LZC unit, two 8-bit LZC units are used as shown in Fig.3. One is used for the lower byte input data and the other for higher byte data. Using following Boolean functions ((14) to (18)), outputs from the 8-bit LZC units can be combined to implement a 16-bit LZC. The LUT implementation of these equations is shown in red boxes. LUT type 1 is used in 5x2 configuration to implement (18) and (17). Similarly, LUT type 2 in 5x2 configuration is used to implement (16) and (14). Equation (15) depends only on one variable and does not need any LUT. In this way, only two LUTs are used to implement the extension circuit for 16-bit LZC from two 8-bit LZC units.

$$V = V_H.V_L \tag{14}$$

$$Z3 = V_H \tag{15}$$

$$Z2 = \overline{V_H}.Z_{2H} + V_H.Z_{2L} \tag{16}$$

$$Z1 = \overline{V_H}.Z_{1H} + V_H.Z_{1L} \tag{17}$$

$$Z0 = \overline{V_H}.Z_{0H} + V_H.Z_{0L} \tag{18}$$

Similarly, the implementation of 32-bit LZC is illustrated in Fig.4, where two LUT type1 and one LUT for OR gate are used to implement the following logic equations:

$$V = V_H.V_L \tag{19}$$

$$Z4 = V_H \tag{20}$$

$$Z3 = \overline{V_H}.Z_{3H} + V_H.Z_{3L} \tag{21}$$

$$Z2 = \overline{V_H}.Z_{2H} + V_H.Z_{2L} \tag{22}$$

$$Z1 = \overline{V_H}.Z_{1H} + V_H.Z_{1L} \tag{23}$$

$$Z0 = \overline{V_H}.Z_{0H} + V_H.Z_{0L} \tag{24}$$

From the discussion of the proposed designs, it can be seen that a significant effort has been made to efficiently map the LZC to the FPGA structure. The design has also been extended to 64-bit LZC using similar structures. The aim is to use fewer resources in terms of LUTs than when using generic tools for the mapping such as Vivado. This is in fact the case as will be seen in the evaluation results presented in the following section.

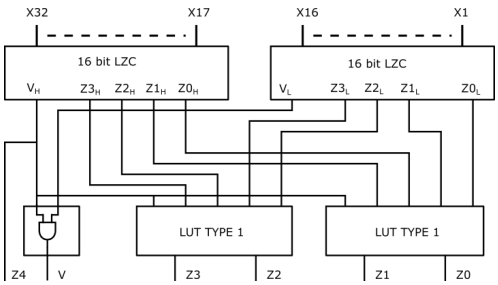

Fig. 4. The proposed structure of 32 bit LZC

IV. Evaluation

The proposed LZC architecture has been implemented on a Xilinx Artix 7-series 28nm xc7a100tcsg324 FPGA device with -2 speed grade using Vivado HLx 16.3 design suite. The results are collected for 8, 16, 32 and 64 bit sizes. All the reported results are based on post-place and post-route implementation. For comparison, several existing designs were also implemented. For example, [8] and [7] are also implemented using same FPGA device. The reasons for reproducing the work of [8] and [7] are threefold: 1) the reported results in [7] are implemented on Xilinx SPARTAN-3E XC3S250E device which has 4-input LUT structure, whereas the 7-series FPGA has 6-input LUTs. 2) the SPARTAN-3E XC3S250E device is based on 90nm technology, whereas 7-series is based on 28nm technology. 3) The reported results for the larger circuits (32 bit and 64 bit) are ASIC based, whereas we implement all the structures on FPGA. Thus for fair comparison the work of [8] and [7] are reproduced on same FPGA. We have also implemented the built-in HLS functions (_builtin_clz and _builtin_clzll) for 32 bits and 64 bits inputs [14]. The generated Verilog code uses if-else statements in a progressive manner from MSB to LSB and assigns the number of leading zeros to the output whenever a 1 is encountered. As there are no built- in functions for 8 and 16 bit inputs, utilizing the _builtin_clz and _builtin_clzll for those cases, results in the same LUT utilization. Therefore, for those cases, we have taken the generated Verilog codes and manually converted them to 8 and 16 bit equivalent ones. Finally, the design in [13] was also implemented and mapped to the FPGA. It is worth mentioning that we used Vivado default area optimizer for getting the implementation results of all the designs.

The comparison of the proposed LZC for different configurations with [7], [8], [13] and [14] is presented in Table I. In this table the comparison is made in terms of LUTs/Slice utilization, power, delay and their product PADP. Our proposed scheme shows significant improvements in LUTs and Slice utilization. For the largest 64 bit configuration, the maximum saving is 15 LUTs (20%) w.r.t to [7] with a minimum of 11 LUTs (15.5%) saving with respect to [8] and [13]. Similarly, Slice saving vary from 17% to 7% w.r.t [7] and [14], respectively. Likewise, the proposed method consumes less power than the other methods. For example, the saving in power consumption varies from 15% to 3% w.r.t [7] and [14], respectively, for a64 bit configuration, Therefore, the proposed designs provide significant gains both in terms of resource usage and power consumption.

Comparing with other existing methods, the proposed method

TABLE I
RESOURCE UTILIZATION, POWER IN MW, SPEED IN ŁOWERCASENS AND POWER X LUTS X DELAY PRODUCT (PADP) FOR DIFFERENT CONFIGURATION OF LZC

	bits	[7]	[8]	[13]	[14]	Proposed
LUTs	8	6	5	–	5	4
	16	16	14	–	14	10
	32	39	33	36	36	26
	64	75	71	71	73	60
SLICEs	8	2	2	–	2	1
	16	6	5	–	5	4
	32	16	12	12	12	9
	64	29	28	28	26	24
Power	8	4.41	4.32	–	3.41	3.86
	16	6.36	6.02	–	6.49	5.03
	32	6.83	6.77	7.063	6.77	6.36
	64	8.84	8.04	7.71	8.17	7.45
Delay	8	1.92	1.62	–	2.2	1.87
	16	1.98	2.06	–	2.38	2.71
	32	2.76	2.84	2.94	2.92	3.03
	64	3.52	3.40	3.70	3.66	3.83
PDAP	8	50.8	34.99	–	37.51	28.87
	16	201.48	173.62	–	216.25	136.31
	32	735.18	634.48	747.55	711.66	501.04
	64	2333.76	1940.86	2025.41	2182.86	1712.01

shows degradation in speed. However, it must be noted that the delay is in all cases below 4ns so that a frequency of operation of at least 250 Mhz can be supported even with 64 bits. This means that for many designs that operate at that frequency or smaller ones, the LZC would meet the timing requirements. For practical designs, what is needed is that the LZC meets the delay needed to support the target operating frequency. For sketches, that is commonly the case. For example in [10] that accelerates Hyperloglog, a target frequency of 322Mhz was used. In that case, the proposed LZC for the 32-bit values used in Hyperlolog can be used as its delay even if slightly larger than existing designs is 3.03 ns.

The Power Area Delay Product (Power x LUTs x Delay (PADP)) shows improvement in performance. For example, this improvement varies from 26% to 15% w.r.t [7] and [8], respectively, for 64 bit configurations. Similarly, for a 32 bit configuration the improvement is more than 20% over all other methods. Thus the proposed method outperforms all state of the art implementations in PADP. The main contributing factors to this improved performance is lower resource usage.

V. CONCLUSIONS AND FUTURE WORK

In this paper, new Leading Zero Count (LZC) designs optimized for Xilinx FPGAs have been presented. The new designs take advantage of the architectural features of the FPGAs to better map the LZC to the FPGA fabric. The 8 bit vector of is divided into two blocks and then implemented on a single slice. Similarly, the extension circuit to build larger LZCs from 8-bit ones uses LUTs in 5x2 LUT configuration to minimize the number of LUTs. This means that not only the logic utilization density but also the power consumption is reduced. The delay is slightly larger than that of alternative approaches but the overall Power Area Delay Product is lower. Therefore, the proposed designs are of interest to reduce resource usage or power consumption and also to minimize the PADP.

ACKNOWLEDGMENT

The work of Pedro Reviriego was supported in part by the ACHILLES project (PID2019-104207RB-I00) funded by the Spanish Ministry of Science and Innovation, in part by the Madrid Government (Comunidad de Madrid-Spain) through the Multiannual Agreement with Universidad Carlos III de Madrid (UC3M) in the line of Excellence of University Professors under Grant EPUC3M21 in the Context of the V Plan Regional de Investigación Científica e Innovación Tecnológica (V PRICIT)

REFERENCES

[1] E. Hokenek and R. K. Montoye, "Leading-zero anticipator (LZA) in the IBM RISC System/6000 floating-point execution unit," in IBM Journal of Research and Development, vol. 34, no. 1, pp. 71-77, Jan. 1990, doi: 10.1147/rd.341.0071.

[2] B. Mathis and J. Stine, "A Novel Single/Double Precision Normalized IEEE 754 Floating-Point Adder/Subtracter", VLSI (ISVLSI) 2019 IEEE Computer Society Annual Symposium on, pp. 278-283, 2019.

[3] S. Gandhi, M. S. Ansari, B. F. Cockburn and J. Han, "Approximate Leading One Detector Design for a Hardware-Efficient Mitchell Multiplier," 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 2019, pp. 1-4, doi: 10.1109/CCECE.2019.8861800.

[4] P. Flajolet, E. Fusy, O. Gandouet, and et al.,"Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm," in Proceedings of the International Conference on Analysis of Algorithms (AOFA), 2007.

[5] D. Ting, "Approximate distinct counts for billions of datasets", Proc. Int. Conf. Manage. Data (SIGMOD), pp. 69-86, 2019.

[6] Y. W. Yu and G. M. Weber, "HyperMinHash: MinHash in LogLog space," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2020.2981311.

[7] J. Miao and S. Li, "A design for high speed leading-zero counter," 2017 IEEE International Symposium on Consumer Electronics (ISCE), Kuala Lumpur, 2017, pp. 22-23, doi: 10.1109/ISCE.2017.8355536.

[8] G. Dimitrakopoulos, K. Galanopoulos, C. Mavrokefalidis and D. Nikolos, "Low-Power Leading-Zero Counting and Anticipation Logic for High-Speed Floating Point Units," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 7, pp. 837-850, July 2008, doi: 10.1109/TVLSI.2008.2000458.

[9] S. Ahmad, V. Boppana, I. Ganusov, V. Kathail, V. Rajagopalan and R. Wittig, "A 16-nm Multiprocessing System-on-Chip Field-Programmable Gate Array Platform," in IEEE Micro, vol. 36, no. 2, pp. 48-62, Mar.-Apr. 2016, doi: 10.1109/MM.2016.18.

[10] A. Kulkarni, M. Chiosa, T. B. Preußer, K. Kara, D. Sidler, and G. Alonso, "HyperLogLog Sketch Acceleration on FPGA" in proceedings of the 30th International Conference on Field Programmable Logic and Applications (FPL), 2020.

[11] Y. J. Chong and S. Parameswaran, "Configurable Multimode Embedded Floating-Point Units for FPGAs," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 11, pp. 2033-2044, Nov. 2011.

[12] P. Reviriego, A. Ullah and S. Pontarelli, "PR-TCAM: Efficient TCAM Emulation on Xilinx FPGAs Using Partial Reconfiguration," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 8, pp. 1952-1956, Aug. 2019, doi: 10.1109/TVLSI.2019.2903980.

[13] N.Z. Milenković, V. V. Stanković, M.L. Milić. "Modular Design Of Fast Leading Zeros Counting Circuit" in Journal of Electrical Engineering, 2015 Nov 1,66(6):329-33.

[14] Xilinx, "Vivado Design Suite User Guide: High Level Synthesis", UG902 (v2017.4) February 2, 2018.