

Hardware accelerators for processing clusters in binary vectors

Iouliia Skliarova*¹ and Valeri Skliarov²

¹ Institute of Electronics and Informatics Engineering of Aveiro (IEETA), Intelligent Systems Associate Laboratory (LASI), Department of Electronics, Telecommunications and Informatics, University of Aveiro, 3810-193 Aveiro, Portugal

² Institute of Electronics and Informatics Engineering of Aveiro (IEETA), Intelligent Systems Associate Laboratory (LASI), 3810-193 Aveiro, Portugal

Abstract. The paper suggests fast hardware accelerators for discovering clusters of zeros and/or ones in binary vectors. Any cluster is composed of successive bits with the same value (either 1 or 0). Search for such segments is required in many practical problems, for example, coding, data, and image processing. The proposed solutions enable, for a given vector, answering such questions as how many one/zero clusters can be found; what is the largest number of consecutive ones/zeros; what is the number of clusters having k consecutive ones/zeros; is the vector only composed of segments with exactly k consecutive ones/zeros; and some others. The relevant practical applications, for which acceleration is required, are also discussed. The paper suggests two core architectural solutions that are based on combinational and iterative networks of gates. Each network is modeled in software (C++ language) and then specified in a hardware-description language (VHDL), synthesized, and implemented in FPGA. Finally, the results of the circuits' evaluations and comparisons are presented.

1 Introduction

Binary vectors are frequently used as a convenient model in many practical problems. Operations over such vectors are profoundly studied in several publications dedicated to Hamming weight (popcount) and distance computation [1, 2], one/zero clusters search [3], etc. The results have been used in such areas as digital signal processing [4], image [5] and data processing [6], encoding and error correction [7], cryptography [8], combinatorial search [9], DNA computing [10], and many others.

Fig. 1 demonstrates tasks that may efficiently be solved in the proposed hardware accelerators. Let's consider a vector $\mathbf{V} = (v_0, \dots, v_{N-1})$, where N is the size of the vector. For certain practical applications, N is a large value. The block S in Fig. 1(a) gives an answer to the question: "is the vector \mathbf{V} composed of only one/zero clusters with exactly k elements?" For example, if $k=2$ and we are interested in only one-clusters, for each of the following two vectors (0011001100; 0000110000) the answer is Yes (1) and for each of the following two vectors (0010111011; 1010101010) the answer is No (0).

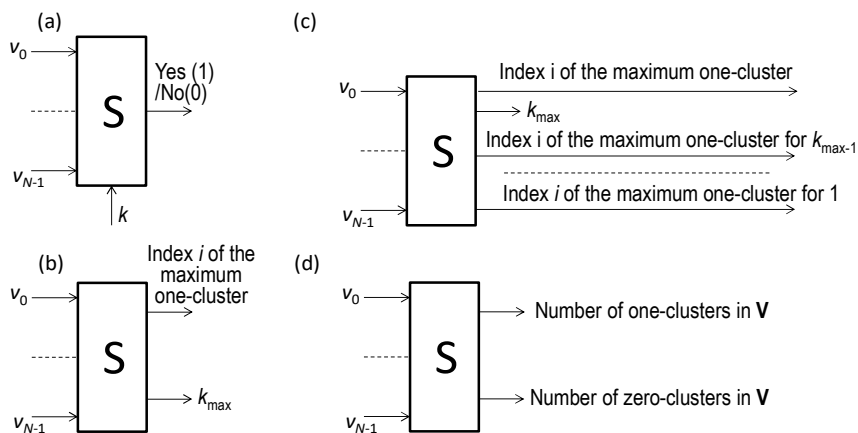


Fig.1. (a) Checking if the vector $\mathbf{V} = (v_0, \dots, v_{N-1})$ is composed of only one/zero clusters with k values 1/0; (b) finding in the vector \mathbf{V} the maximum one-cluster; (c) discovering in the vector \mathbf{V} all one-clusters $1, \dots, k_{\max}$; (d) finding in the vector \mathbf{V} the total number of one/zero clusters.

* Corresponding author: iouliia@ua.pt

Similar operations can be executed for detecting zero-clusters. The block S in Fig. 1(b) finds the maximum one-cluster (i.e. such cluster that contains the maximum number of values 1) in a given vector \mathbf{V} and outputs the index of the first bit in such a cluster. For example, for the vector 0010111011 the answer is $k_{\max}=3$ (number of ones in the maximum one-cluster) and $i=4$ (indices are counted from 0 for the left-hand bit of the vector). The block S in Fig. 1(c) discovers the cluster with k_{\max} values 1 and all other clusters with $1, \dots, k_{\max}-1$ values 1. For the example above: $k_{\max}=3$ for the cluster with the first bit index 4; $k_{\max}=2$ for the cluster with the first bit index 8; and $k_{\max}=1$ for the cluster with the first bit index 2. The last block S in Fig. 1(d) finds the total number of one/zero clusters in a given vector \mathbf{V} . For example, for the vector **0010111011** the number of one-clusters (they are shown in **bold**) and the number of zero-clusters (they are shown in *italic*) is the same and is equal to 3.

Let us discuss now some practical applications, for which solving the tasks listed above is needed. One of such applications is frequent item computations in a given set of data $\mathbf{D} = \{d_0, \dots, d_N\}$ [11-12]. The objective is finding the item(s) repeated the maximum number of times in the set \mathbf{D} . To accomplish this task the following sequence of steps may be applied. First of all, the items are sorted (for instance, with a parallel sorting network [13-15]); afterward, all pairs of adjacent items are compared one to another, and an N bit binary vector \mathbf{V} is constructed whose element i , $i = 0, 1, \dots, N-1$, is either set to one (if items of the sorted set \mathbf{D} with indices i and $i+1$ are equal) or to zero (if items of the sorted set \mathbf{D} with indices i and $i+1$ are not equal). Finally, the largest cluster of ones in the vector \mathbf{V} has to be found and its Hamming weight (HW) has to be calculated. These steps are summarized in Fig. 2. There are three basic operations in Fig. 2: sorting, clustering, and computing the results. The proposed in this paper circuits allow the execution of the clustering operation to be accelerated.

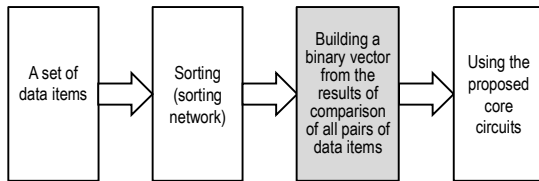


Fig. 2. Using the proposed accelerators for discovering the most repeated items in a data set.

Let us consider an example. Suppose the set $\mathbf{D} = \{10, 0, 21, 41, 89, 7, 21, 22, 14, 10, 21, 7, 10, 21, 24, 28\}$ of integers is given. Sorting permits the following sorted set $\{0, 7, 7, 10, 10, 10, 14, 21, 21, 21, 21, 22, 24, 28, 41, 89\}$ to be produced. In the next step, all pairs of adjacent items are compared, and the vector \mathbf{V} is built: $\mathbf{V} = (0101100**1110**0000)$, where the left 0 is the result of the comparison of 0 and 7, the next 1 is the result of the comparison of 7 and 7, the next 0 is the result of comparison of 7 and 10, etc. The longest cluster of ones is marked with **bold** font in the vector \mathbf{V} above; the

cluster's Hamming weight is 3. Therefore, the most frequent item appears four times (the computed Hamming weight plus 1) in the original set and this is item 21 (the item whose index is 7 in the sorted set). Other practical applications are described in [16-24]. Since many systems have hard real-time constraints, performance is important and hardware accelerators may provide significant assistance.

The main contributions of this paper are the following:

- Suggesting two core circuits allowing the maximum one-cluster in a binary vector to be found.
- Illustrating how the proposed circuits may be employed in higher-level systems, solving more complex tasks (such as depicted in Fig. 1).
- Experiments and comparisons that prove that the suggested circuits are parameterizable, occupy reasonable hardware resources and may operate at high clock frequencies.

The remainder of this paper is organized as follows. Section 2 suggests two basic core architectures for solving the described above tasks, which are: a fully combinational circuit; and an iterative circuit (with a sequentially reusable combinational fragment). Section 3 gives implementation details in hardware and demonstrates the results of experiments and comparisons. The conclusion is given in section 4.

2 Basic architectures of hardware accelerators

Fig. 3 depicts the basic architecture of the proposed combinational circuit. It consists of fragments, each of which takes on the inputs the current vector (for example, \mathbf{V}) and outputs the new vector (for example, \mathbf{V}_1). The latter is formed on outputs of AND gates executing operations over adjacent bits.

For example, $\mathbf{V}_1 = (v_0 \& v_1, v_1 \& v_2, \dots, v_{N-3} \& v_{N-2}, v_{N-2} \& v_{N-1})$. Similarly, the other vectors ($\mathbf{V}_2, \mathbf{V}_3, \dots$) are built. If any vector \mathbf{V}_j ($j=1, \dots, N-1$) contains only zero bits then the previous vector \mathbf{V}_{j-1} (that is not composed of only zero bits) indicates clusters with j values 1. Thus, if $\mathbf{V} \neq 0$ (i.e. \mathbf{V} does not contain all 0 values) and $\mathbf{V}_1 = 0$ (i.e. \mathbf{V}_1 contains all 0 values) then \mathbf{V} is composed of isolated 1 values (i.e. any one-cluster contains only one value 1) separated by one or more values 0. If $\mathbf{V}_j \neq 0$ ($j \geq 1$) and $\mathbf{V}_{j+1} = 0$ then the initial vector \mathbf{V} includes one or more ($j+1$)-cluster(s), i.e. such clusters that have ($j+1$) values 1. If the initial vector \mathbf{V} contains exactly N ones, then propagation of signals through the circuit in Fig. 3 will never give the result equal to 0. Thus, if the final value on the output of Fig. 3 is 1 then this indicates that the vector \mathbf{V} contains only 1 values and the single one-cluster has N values 1.

Propagating signals from vector \mathbf{V} through the circuit in Fig. 3 until the vector $\mathbf{V}_{j+1} = 0$ permits the maximum number $k_{\max}=(j+1)$ of successive ones in the vector \mathbf{V} to be found. Besides, all the clusters $1, \dots, k_{\max}-1$ may also be found through analyzing the previous pairs of the vectors (i.e. \mathbf{V} and $\mathbf{V}_1, \mathbf{V}_1$ and \mathbf{V}_2 , etc.).

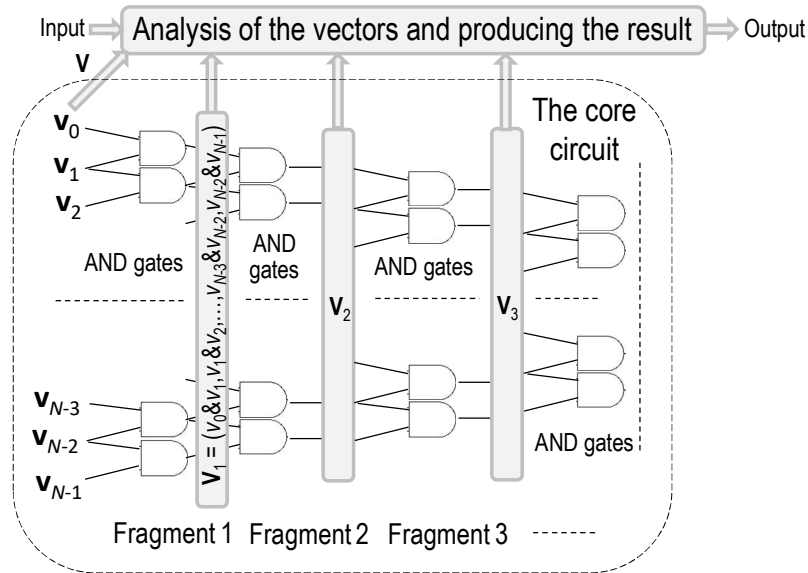


Fig. 3. The architecture of the combinational circuit

Let us consider the same set of integers as in the introduction section: {10, 0, 21, 41, 89, 7, 21, 22, 14, 10, 21, 7, 10, 21, 24, 28} for which the following vector V is built: $V = (010110011100000)$. The vector V is handled by the circuit in Fig. 3. Consequently, the vectors $V_1=(000100011000000)$, $V_2=(000000010000000)$, $V_3=(000000000000000)$ are generated and $HW(V_1)=3$, $HW(V_2)=1$, $HW(V_3)=0$. Since the vector V_3 contains all zeros, the maximum number of repetitions in the initial data items is $3+1=4$. The result 3 is incremented because comparators in the block of Fig. 3, filled in with grey color, form one-clusters corresponding to the number of repeated items minus 1.

For instance, for four repeated integers 21, the result of comparison contains 3 values 1. Thus, to get the correct result it is necessary to add 1 to 3. Since $HW(V_2)=1$ only one item is repeated 4 times. The index of the value 1 in V_2 is 7. The integer with the index 7 is 21 and this item is repeated 4 times. Similarly, $HW(V_1) = 3$, which indicates that there are three items repeated 3 or more times. The indices of these items are 3, 7, and 8, which correspond to the indices of values 1 in V_1 . Thus, the respective repeated items are 10, 21, and 21. In the last case there are three repetitions of the value 21 beginning from the index 7 and there are three repetitions of the value 21 beginning from the index 8. If necessary, such items as 21 (which have already been taken as more repeated) may easily be excluded (comparing values in the first {21} and in the second {10, 21} sets).

The core circuit in Fig. 3 is composed of only AND gates with two inputs and may be easily parameterized and automatically generated. A supplementary circuit for analysis of the vectors and producing the result may execute different functions that depend on the target requirements (see Fig. 1). For example, if we consider the circuit S shown in Fig. 1(a) then the "Input" in Fig.

3 is k and the supplementary circuit looks like it is illustrated in Fig. 4.

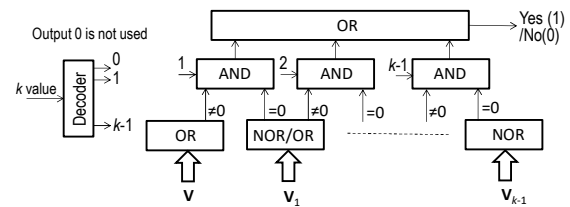


Fig. 4. Analysis of vectors and producing the result for the circuit in Fig. 1(a).

The binary decoder activates only one output indicating the desired cluster k . Thus, an active value on the output 0 says that we are interested in only such vectors that do not have any one-cluster (i.e. a test of V containing only zero bits has to be done). All the other outputs of the decoder $1, \dots, k-1$ indicate the relevant one-clusters, which we would like to check; i.e. we have to answer whether they are available or not.

Note that the number of segments in the architecture of Fig. 3 depends on the maximum desired value of k_{max} (the maximum value of consecutive ones in one cluster). For large-size vectors, the number of fragments might be considerable, and the propagation delay becomes extensive. Fig. 5 depicts the basic architecture of the iterative circuit, for which the propagation delay in the combinational part is very small and clock frequency to support iterations may be relatively high. Thus, the performance of the circuit in Fig. 3 is comparable to the circuit in Fig. 5, but the resources of the circuit in Fig. 5 are significantly reduced. The main ideas are taken from [14, 25], where a similar technique is used for sorting and HW counting.

The fragment 1 in Fig. 5 is sequentially reused. At the beginning, the vector V is recorded in the Register and the first vector V_1 is formed on outputs of Fragment 1 (see example on the right-hand side of Fig. 5). Two

vectors \mathbf{V} and \mathbf{V}_1 are checked for " $\neq 0$ " (there are bits not equal to 0) and " $=0$ " (all the bits are equal to 0). If both vectors (\mathbf{V} and \mathbf{V}_1) contain bits not equal to 0 then the vector \mathbf{V}_1 is extended to the size N recording a 0 value in the bit $N-1$ and the next iteration is executed, i.e. the extended vector \mathbf{V}_1 is recorded in the Register and a new vector \mathbf{V}_2 is formed on the outputs of Fragment 1.

Similarly, other iterations are executed until all the bits on the outputs of Fragment 1 contain only 0 values.

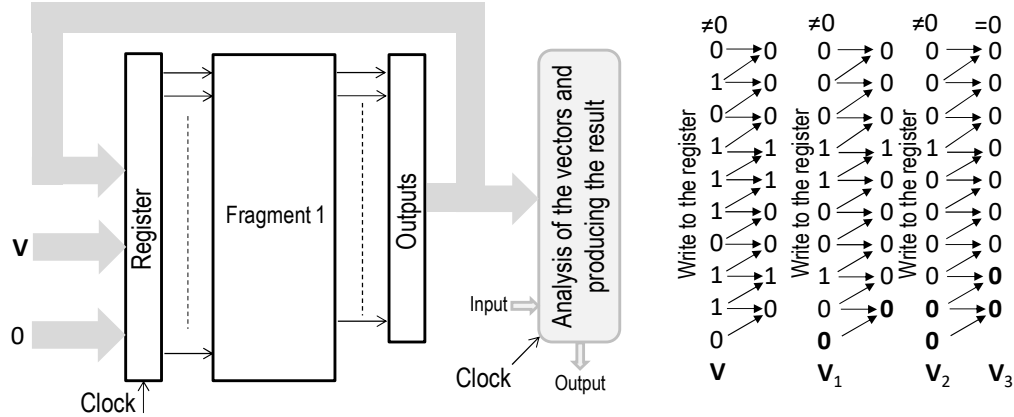


Fig. 5. The architecture of the iterative circuit, functionally equivalent to the circuit from Fig. 3.

Let $HW(\mathbf{V})$ be the Hamming weight of \mathbf{V} . It is easy to see that $HW(\mathbf{V})=6$ (for our example) is the number of values 1 in the initial vector \mathbf{V} ; $HW(\mathbf{V}_1)=3$ is the number of one-clusters in the initial vector \mathbf{V} containing two values 1. Indices of values 1 in \mathbf{V}_1 give the initial indices of such clusters in the vector \mathbf{V} . They are 3, 4, and 7, pointing to the following clusters (indicated by **bold** font): 010**11**0110; 010**11**0110; and 01011**10**110 (the left-hand bit of $\mathbf{V} = 0101110110$ has the index 0 and the right-hand bit of \mathbf{V} has the index 9). $HW(\mathbf{V}_2)=1$ indicating that there is just one cluster in \mathbf{V} with three values 1. The following three clock cycles are required in the circuit of Fig. 5 to obtain the final result: 1) to record the initial vector \mathbf{V} to the Register; 2) to record vector \mathbf{V}_1 to the Register; and 3) to record vector \mathbf{V}_2 to the Register. The combinational circuit is only composed of one level of AND gates and the propagation delay from inputs to outputs is very small.

3 Implementation in hardware, experiments, and comparisons

The core architectures in Fig. 3 and 5 were first modeled in C++. After modeling in software, the circuits for architectures of hardware accelerators discussed in section 2 have been described in VHDL, synthesized, implemented, and tested in a low-cost and low-power xc7a100tcs324-1 FPGA [26] from Artix-7 family (available on the prototyping board Nexys-4 of Digilent [27]). Synthesis, implementation, and bitstream generation were done in Xilinx Vitis/Vivado 2021.2. Networks composed of AND gates have been coded in structural VHDL. So, they repeat exactly the structures that are shown in Fig. 3, 5.

This enables us the maximum one-cluster and all the other one-clusters to be found.

A complete example is shown on the right-hand side of Fig. 5. The vector \mathbf{V}_3 contains only zeros. Thus, the maximum number of consecutive ones in \mathbf{V} is 3: 010**11**0110 (see three ones in **bold**). The first index in \mathbf{V} for the relevant cluster is indicated by the index of the value 1 in the previous vector \mathbf{V}_2 recorded in the register (which is 3).

For example, the following VHDL code describes a combinational circuit that enables answering if an input vector contains at least one cluster with exactly K consecutive ones. In this case the output "led" is on. If there are just clusters with less than K consecutive ones or also clusters with more than K consecutive ones then the answer is no and the "led" is off. The N -bit vector is supplied on inputs "sw" and the result (the answer) is shown on the output "led".

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity Exact_K_ConsGen is
generic ( N       : integer := 16;
         K       : integer := 5 );
port (sw: in std_logic_vector(N-1 downto 0);
     led : out std_logic_vector(0 downto 0));
end Exact_K_ConsGen;
architecture Behavioral of Exact_K_ConsGen is
    type iter is array (0 to K) of
        std_logic_vector(N-1 downto 0);
    signal steps : iter;
begin
    steps(0) <= sw;
    -- generation of AND gates (see Fig. 3)
    repeat_steps: for j in 0 to K-1 generate
        first_and_gates: for i in 0 to N-2
            generate
                steps(j+1)(i) <=
                    steps(j)(i) and steps(j)(i+1);
            end generate first_and_gates;
            steps(j+1)(N-1) <= '0';
        end generate repeat_steps;
    led <= "1" when
        (steps(K-1) /= 0) and (steps(K) = 0)
        else "0";
end Behavioral;
    
```

If we access the elaborated design in Vivado then we can see that the schematic is very similar to the circuit

shown in Fig. 3. An iterative circuit (see Fig. 5) has been built by analogy to [14].

For the experiments, the size N of input vectors was chosen to vary from 16 to 16,384 and $K = 5/10$. The input vectors have been pseudo-randomly generated in a MicroBlaze processor core instantiated in the FPGA and then stored in an external cellular 16MB RAM. Then the vectors have been processed separately in software running on MicroBlaze and in hardware accelerators proposed in this paper, and the results have been compared. Both processing cores (software and hardware) receive a stream of 32-bit data through DMA from the cellular 16MB RAM. 32-bit vector chunks are accumulated and concatenated until the desired length (N -bit) vector is constructed, which is further used for parallel processing in the suggested core architectures. Concatenation of 32-bit sub-vectors arriving from DMA is done at the speed of AXI DMA transfers. The hardware accelerator provides memory-mapped interface to the MicroBlaze for communicating the result. The outcomes of both cores are displayed on Vitis serial terminal communicating with the MicroBlaze through UARTLite module.

The hardware costs reported by the synthesis and implementation tools for the combinational circuit from Fig. 3 are summarized in Table 1 for different values on N and K . These costs do only refer to implementing the core circuit and do not account for the framework required for the experiments, i.e. the MicroBlaze processor and the respective interfaces. The core circuit latency is given by the network depth, which is equal to K . Albeit the resulting latency is relatively low, the maximum throughput will never exceed the interface bandwidth divided by the number of memory accesses needed to get the source data. All the tested circuits run at 100 MHz. We expect that increasing the value K will lead to reducing the supported clock frequency, as the core circuit latency will exceed the targeted minimum clock period. This problem can however be solved by changing to the sequential circuit from Fig. 5, which would require several (K) clock cycles to produce the result and can function at K times (minus register's setup and propagation delays) higher clock frequency. This sequential circuit uses much fewer logical resources (just one level of AND gates is reutilized, so the number of LUTs reported in Table 1 can be roughly divided by K) but requires additional N flip-flops to implement the register for keeping intermediate results between the network levels.

Table 1. Hardware costs of implementing the combinational network of AND gates in xc7a100tcs324-1 FPGA for $N=16/256/1024/16384$ and $K=5/10$.

K	N	LUTs	% of FPGA logic resources
5	16	16	0.03%
	256	255	0.4%
	1024	1195	1.9%
	16384	32068	50.6%
10	16	17	0.03%
	256	371	0.6%
	1024	2578	4.1%
	16384	30323	47.8%

As the hardware execution time is bounded by AXI DMA transactions, and, assuming 100 MHz clock frequency, this amounts to 110 μ s for processing $N=16384$ bits of randomly generated data, which is 10-23 times faster (depending on the input vector V and the value of $K=5/10$) than producing the result in software running on MicroBlaze. The exact speedup over software depends on the vector length as the critical path is bounded by K and does not depend on the value N (which is not true for software where the bigger N value, the longer the processing time).

4 Conclusion

The paper describes hardware accelerators for processing clusters in binary vectors and points to a number of practical applications in which the proposed technique may effectively be used. The described architectures have the following distinctive features: 1) high performance because of very fast combinational and iterative circuits (signals propagate through only simple AND gates); 2) regularity, simplifying parameterization; 3) extended functional capabilities (numerous types of clusters can be discovered); 4) low consumed resources (for the sequential circuit). Software models and hardware implementations are also presented and discussed which allows embedding the respective designs in more complicated systems. The results of experiments and comparison confirm adequate performance and reasonable resources for the proposed hardware accelerators.

This work was supported by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UIDB/00127/2020.

References

1. B. Parhami, "Efficient Hamming weight comparators for binary vectors based on accumulative and up/down parallel counters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 2, pp. 167–171 (2009)
2. I. Skliarova, "Accelerating Population Count with a Hardware Co-Processor for MicroBlaze," *Journal of Low Power Electronics and Applications*, vol. 11, no. 2 (2021)
3. V. Sklyarov and I. Skliarova, "Digital Hamming weight and distance analyzers for binary vectors and matrices," *International Journal of Innovative Computing, Information and Control*, vol. 9, no. 12, pp. 4825-4849 (2013)
4. V. Sklyarov and I. Skliarova, "Multi-core DSP-based vector set bits counters/comparators," *Journal of Signal Processing Systems*, vol. 80, no. 3, pp. 309–322 (2015)
5. C. Qin, C.C. Chang, and P.L. Tsou, "Perceptual Image Hashing based on the Error Diffusion Halftone Mechanism," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 9, pp. 6161-6172 (2012)

6. V. Sklyarov, I. Skliarova, and I. Utepbergenov, "Hardware Accelerators for Data Processing in High-performance Computing Systems," tutorial, Proceedings of the 15th International Conference on Application of Information and Communication Technologies, October, virtual mode, pp. 1-4 (2021)
7. I.N. John, P.W. Kamaku, D.K. Macharia, and N.M. Mutua, "Error Detection and Correction Using Hamming and Cyclic Codes in a Communication Channel," Pure and Applied Mathematics Journal, vol. 5, no. 6, pp. 220-231 (2016)
8. J.S. Coron and A. Gini, "Improved cryptanalysis of the AJPS Mersenne based cryptosystem," J. Math. Cryptol, vol. 14, pp. 218–223 (2020)
9. I. Skliarova and A.B. Ferrari, "Reconfigurable Hardware SAT Solvers: A Survey of Systems," IEEE Transactions on Computers, vol. 53, no. 11, pp. 1449-1461 (2004)
10. A. Sarkar, Z. Al-Ars, C.G. Almudever and K.L.M. Bertels, "QiBAM: Approximate Sub-String Index Search on Quantum Accelerators Applied to DNA Read Alignment," Electronics, vol. 10, no. 19 (2021)
11. K. Yamamoto, M. Ikebe, T. Asai and M. Motomura, "FPGA-Based Stream Processing for Frequent Itemset Mining with Incremental Multiple Hashes," Circuits and Systems, vol. 7, no. 10 (2016)
12. C.H. Chee, J. Jaafar, I.A. Aziz, M.H. Hasan, and W. Yeoh, "Algorithms for frequent itemset mining: a literature review," Artificial Intelligence Review (2018)
13. S.W. Aj-Haj Baddar and K.E. Batcher, Designing Sorting Networks, A New Paradigm. Springer (2011)
14. V. Sklyarov and I. Skliarova, "High-performance implementation of regular and easily scalable sorting networks on an FPGA," Microprocessors and Microsystems, vol. 38, no. 5, pp. 470-484 (2014)
15. I. Skliarova and V. Sklyarov, FPGA-based Hardware Accelerators. Springer, Switzerland (2019)
16. S. Sun, Analysis and acceleration of data mining algorithms on high performance reconfigurable computing platforms, Ph.D. thesis, Iowa State University. Available online: <http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=1421&context=etd> (2011)
17. X. Wu, V. Kumar, J.R. Quinlan, et al., "Top 10 algorithms in data mining," Knowledge and Information Systems, vol. 14, pp. 1-37 (2007)
18. M. Yasir, A. Ashraf, M.U. Chaudhry, F. Hassan, J.H. Lee, M. Jasiński, Z. Leonowicz, and E. Jasińska, "D-GENE-Based Discovery of Frequent Occupational Diseases among Female Home-Based Workers," Electronics, vol. 10, no. 11 (2021)
19. D. Zmaranda, H. Silaghi, G. Gabor, and C. Vancea, "Issues on Applying Knowledge-Based Techniques in Real-Time Control Systems," International Journal of Computers, Communications and Control, vol. 8, no. 1, pp. 166-175 (2013)
20. L. Field, T. Barnie, J. Blundy, R.A. Brooker, D. Keir, E. Lewi, and K. Saunders, "Integrated field, satellite and petrological observations of the November 2010 eruption of Erta Ale," Bulletin of Volcanology, vol. 74, no. 10, pp. 2251–2271 (2012)
21. W. Zhang, K. Thurow, and R. Stoll, "A Knowledge-based Telemonitoring Platform for Application in Remote Healthcare," International Journal of Computers, Communications and Control, vol. 9, no. 5, pp. 644-654 (2014)
22. M. Fularz, M. Kraft, A. Schmidt, and A. Kasiński, "A High-performance FPGA-based Image Feature Detector and Matcher Based on the FAST and BRIEF Algorithms," International Journal of Advanced Robotic Systems, vol. 12, no. 10 (2015)
23. B. Parhami, "Computer architecture for big data", in: Encyclopedia of Big Data Technologies. S. Sakr and A. Zomaya (eds.), Springer (2018)
24. V. Sklyarov, A. Rjabov, I. Skliarova, and A. Sudnitson, "High-Performance Information Processing in Distributed Computing Systems," International Journal of Innovative Computing, Information and Control, vol. 12, no. 1, pp. 139-160 (2016)
25. V. Sklyarov and I. Skliarova, "Design and implementation of counting networks," Computing, vol. 97, no. 6, pp. 557–577 (2015)
26. AMD, Inc., 7 Series FPGAs Data Sheet: Overview. Available online: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf (2020)
27. Diligent, Inc., Nexys-4 Reference Manual, available online: <https://reference.digilentinc.com/reference/programmable-logic/nexys-4/reference-manual>.