JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL.15, NO.4, AUGUST, 2015
http://dx.doi.org/10.5573/JSTS.2015.15.4.477
ISSN(Print) 1598-1657
ISSN(Online) 2233-4866

A Parallel Search Algorithm and Its Implementation for Digital *k*-Winners-Take-All Circuit

Myungchul Yoon

***Abstract*—The *k*-Winners-Take-All (*k*WTA) is an operation to find the largest *k* (>1) inputs among *N* inputs. Parallel search algorithm of *k*WTA for digital inputs is not invented yet, so most of digital *k*WTA architectures have *O*(*N*) time complexity. A parallel search algorithm for digital *k*WTA operation and the circuits for its VLSI implementation are presented in this paper. The proposed *k*WTA architecture can compare all inputs simultaneously in parallel. The time complexity of the new architecture is *O*(log*N*), so that it is scalable to a large number of digital data. The high-speed *k*WTA operation and its *O*(log*N*) dependency of the new architecture are verified by simulations. It takes 290 ns in searching for 5 winners among 1024 of 32 bit data, which is more than thousands of times faster than existing digital *k*WTA circuits, as well as existing analog *k*WTA circuits.**

***Index Terms*—*k*-winners-take-all, digital *k*-WTA circuit, parallel *k*-WTA, parallel search algorithm, scalable *k*-WTA architecture**

I. INTRODUCTION

The Winner-Take-All (WTA) is an operation to search for the largest input among *N* inputs, and the *k*-WTA is an extension of WTA such that searches the largest *k*-inputs among *N* inputs. WTA/*k*WTA operation is one of the most important building blocks in various areas such as machine learning [1], neural networks [2], image processing [3], signal processing [4], pattern recognition [5], filtering [6], vision systems [7], clustering [8], sorting [9], and information retrieval [10], etc.

While many analog *k*WTA circuits [11-13] have been designed, digital *k*WTA circuits have been relatively less researched than their analog counterparts. The analog *k*WTA implementation requires less hardware than the digital implementation, and can find *k*-winners by comparing all inputs simultaneously. However, analog *k*WTA circuits suffer from matching problem [14] and stability-convergence problem [15] for a large number of inputs, which affect the precision of analog *k*WTA circuits. With the development of semiconductor technologies, the number of inputs increases while the input range decreases. Therefore, analog *k*WTA circuits have difficulties in achieving a high-precision operation.

Unlike analog counterparts, digital *k*WTA circuits are free from the mismatch, stability, and convergence problems, and the precision of digital *k*WTA circuits is determined only by the number of digits in digitization process so that it is easy to control the precision. In spite of these advantages, the lack of simultaneous search algorithm is one of the serious drawbacks of digital *k*WTA circuits. Most of analog WTA/*k*WTA circuits find winner(s) by simultaneous competition of all inputs. Although some parallel search architectures are developed for digital WTA operation [16, 17], parallel search algorithm of the digital *k*WTA operation is not yet invented.

A hardware intensive parallel *k*WTA architecture for a small number of digital inputs was proposed in [18], but its $O(N^3)$ hardware complexity limits its usage to applications with a few number of inputs. Other *k*WTA architectures for digital inputs iterate the WTA operation

Manuscript received Apr. 8, 2015; accepted Jun. 9, 2015
The author is with the Department of Electronics Engineering, Dankook University, Cheon-An, Choong Nam, 330-714, Korea
E-mail : myoon@dankook.ac.kr

k-times [19], or compare serially one by one through a pipelined architecture [20]. Most of *k*WTA circuits have $O(N)$ time complexity so that the latency of *k*WTA operation increases greatly along the increase of *N*.

A parallel search algorithm for digital *k*WTA and its circuits for VLSI implementation are presented in this paper. The proposed *k*WTA architecture compares all inputs simultaneously with $O(\log N)$ time complexity. Employing the parallel operations, the latency of *k*WTA is reduced greatly even for a large number of inputs.

The parallel search algorithm for digital *k*WTA is described in Section II, and the architecture and circuits for VLSI implementation are presented in Section III. The latency of the proposed circuits is estimated by simulation and the results are shown in Section IV. Section V concludes this paper.

II. ALGORITHM FOR PARALLEL K-WTA

The parallel *k*WTA circuit searches *k*-winners by comparing all inputs simultaneously. From the most significant bit (MSB) to the least significant bit (LSB), it compares all inputs bit by bit until k-winners are identified.

The input data are divided into three groups, a winner group, a competitor group, and a loser group. The inputs in the winner group are already confirmed as winners so that those will be included in the final *k*-winners. The inputs in the competitor group are candidates for winners. The inputs in the loser group have already lost their chance to win.

Let us represent the status of *N* inputs as an *N*-bit bit-vector such that the *i*-th bit of a bit-vector X represents the state of the *i*-th input. For example, the bit-vector C and W are used to store the competitor state and winner state of inputs respectively, and if the *i*-th bit of C is 1, the *i*-th input is in competitor group. Let D[j] ($0 \le j < m$) stand for the bit-vector consists of the *j*-th bits of *N* input data. The **0**-vector (**1**-vector) is the vector that all bits are 0 (1). The notation $n(X)$ is used to represent the number of 1s in a bit-vector X. With the above notations, the peudo-algorithm of the parallel *k*WTA for digital inputs is described in Fig. 1.

At the beginning, all inputs are in the competitor group so that the C is set to **1**-vector and W is reset to **0**. The algorithm consists of *m* iterative steps for *m*-bit digital inputs. From MSB to LSB, only one bit-position is used to distinguish winners in each step. According to the value of bits, competitors are divided as top dogs and underdogs, e.g., if the bit of a competitor is 1, it becomes a top dog, otherwise, it is a underdog. The number of top dogs are calculated and added to the number of winners to obtain the determinant (DET). If DET is greater than *k*, the underdogs are confirmed as losers, and the top dogs become the competitor group of the next step. If DET is less than *k*, the top dogs are inserted to the winner group, and the underdogs become the competitor group of the next step. If DET is equal to k, the top dogs are inserted to winners and the *k*WTA operation is finished.

If DET is greater than k when LSB is checked, it implies that all top dogs have the same magnitude. The function Select(.) is activated to pick $k–n$W inputs among n(T) inputs of the same magnitude. Although the selection rule may be different according to applications, a simple rule is used in this paper; lower-indexed (or higher-indexed) $k–n$W inputs are selected, and added to the winner.

```
1.   // Parallel k-WTA Algorithm for N of m-bit data
2.   Bit_vector  C ;  //  competitior state
3.   Bit_vector  T ;  //  top-dog state
4.   Bit_vector  W ;  //  winner state
5.   Bit_vector D[0:m-1] ; // input data,  m-1: MSB, 0 : LSB
6.   int    DET ;  //  Determinant
7.   int    nW ;   // number of winners
8.
9.   BEGIN
10.  C= 1;  W=0 ;  nW=0 ; //  initialization   of states
11.   for (i=m-1; i >=0; i--)  {
12.       T=  C AND D[i] ;  // Bitwise AND
13.       DET = n(T) + nW  ;
14.       if (DET <= k)  {
15.            W= W OR C ;  // Bitwise OR
16.            nW = DET ;
17.             If (DET == k) { DONE=1 ; exit; }
18.               else  C= C XOR T ;  // Bitwise XOR
19.             }
20.          else  C=T ;
21.        } // end of for-loop
22.   if ( nW < k )  {
23.       // select  k–nW  inputs  from C, and insert to W
24.       R=Select ( C,  k–nW ) ;
25.       W= W OR R ;  // Bitwise OR
26.       }
27.   END
```

Fig. 1. Pseudo-algorithm of parallel *k*WTA for digital inputs.

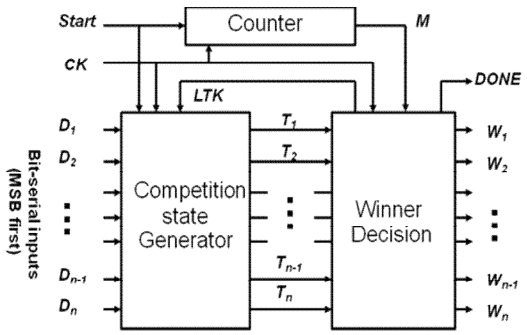

Fig. 2. Block diagram of the parallel *k*WTA module.

Except the function *n*(T) in line 13, all operations in Fig. 1 can be performed in parallel. Finding the value, *n*(T) is the most critical operation which determines the latency of the algorithm. Since the *n*(T) is obtained by counting the number of 1s in *N*-inputs, the counting operation can be performed in parallel by a tree structure. For example, *N* inputs are partitioned into *p* block of *q* inputs ($p \times q = N$), the number of 1s in each block is counted in parallel, and the results are accumulated in the upper level. If *N* is very large so that *p* becomes large, the accumulation step itself can be performed in parallel with a multi-level adder-tree. With a multi-level tree architecture of counting circuits, time complexity of the parallel *k*WTA algorithm becomes $O(\log N)$. The circuits for counting *n*(T) are described in the next section.

III. Implementation of Parallel k-WTA Circuits

1. Block Diagram

Fig. 2 shows the block diagram for the proposed digital *k*WTA module. The circuit consists of two main parts: competition-state generator (CG) circuit and winner decision (WD) circuit. The module starts with the START signal. With the START signal the counter is reset to 0 and increased for every clock cycle. The DONE signal which informs the completion of *k*WTA operation becomes 1 when *k*-winners are identified.

At the beginning, all inputs are set as competitors, and no winner exists. In each cycle, the CG part identifies top-dogs (T) by C and input bits (D[i]), and the results are fed to the WD part to count the number of 1s in T.

The WD part calculates *n*(T)+*n*W, and compares it with *k*. If it is equal to *k*, the top-dogs are inserted to winner register and the search operation is finished with the generation of the DONE signal. If it is not equal to *k*, the state of C is updated by the LTK signal which becomes 1 only when it is less than *k*.

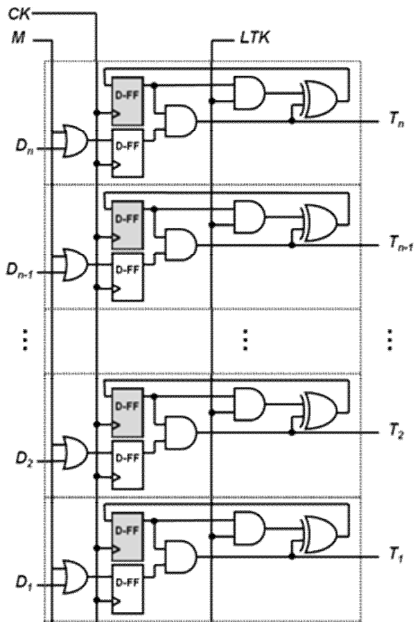

Fig. 3. The competition state generator circuit. The shaded D-FFs are preset to 1 while the unshaded D-FFs are reset to 0.

The circuit of the CG parts is depicted in Fig. 3. Note that every block works independently so that the CG part operates in parallel. Unlike the CG part, all input-states are involved in WD operations so that the most of delay takes place in the WD part. For high speed operation, therefore, the design of WD part is the key of *k*WTA circuit.

2. Single Level Implementation

The overall structure of the WD part is depicted in Fig. 4. The WD is composed of winner-counter, tie-breaker and winner-registers. The winner-counter calculates *n*(T)+*n*W, compares the result with *k*, and generates control signals according to the comparison results. The tie-breaker performs the function Select(.) in Fig. 1. It is used to select lower- indexed *k*–*n*W inputs. The winner-registers are used to store the winner-state.

The main body of the winner-counter is composed of $N \times (k+1)$ array of 1×2 multiplexer (MUX) which is made by two NMOS transistors. Each column of the array is called a path and indexed from p_0 to p_k. The array called 1-counter is used to count the number of 1s in T_i's ($1 \le i$

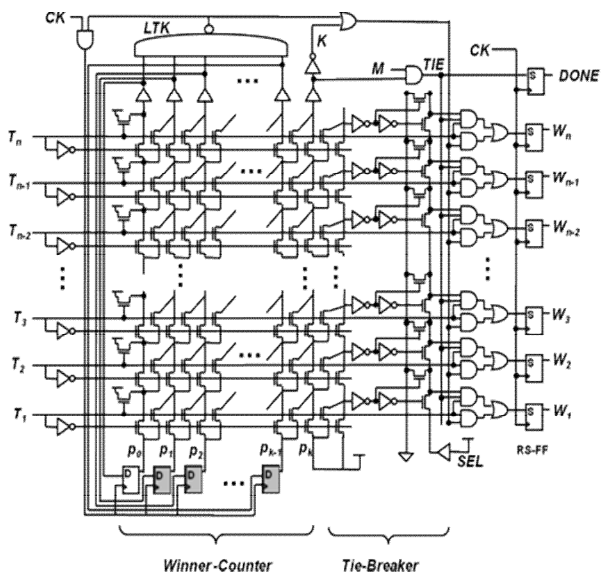

Fig. 4. Single level implementation of the winner-decision circuit. The shaded flip-flops are preset to 1 while the unshaded flip-flops are reset to 0.

≤ n). A D-flip-flop (D-FF) is attached to each path, and the only one D-FF attached to p_{nW} stores 0 while the other D-FFs store 1.

Let us think a worm called zero-worm nested at the D-FF attached to p_{nW}. For each cycle, the worm starts from the bottom of p_{nW} and proceeds toward the top of the array. The passage of the worm is determined by the value of T_i's. If T_i is 0, the worm goes straight-up so that it stays on the same path. If T_i is 1, the worm crosses-up to the right path. If n(T)+nW is greater than k, the worm exits on the right edge of the array so that it cannot reach the top. In this case, the C–state is replaced by T-state and nW is not changed. If n(T)+nW is not greater than k, the worm reaches the top of the array in which cases all top-dogs are inserted to the winner-registers on the right of Fig. 4. If n(T)+nW is equal to k, the signal K is generated to finish the kWTA operation. If n(T)+nW is less than k, the signal LTK is generated to update the value nW by n(T)+ nW, and to update the stored values in D-FFs.

The tie-breaker circuit is activated when n(T)+nW is still greater than k, even though m cycles are finished. It happens when the final n(T) (>1) inputs have the same magnitude. The tie-breaker circuit selects lower-indexed k–nW inputs from n(T) inputs, and adds them to the winner-register.

In the worst case, the delay of winner-counter circuit is

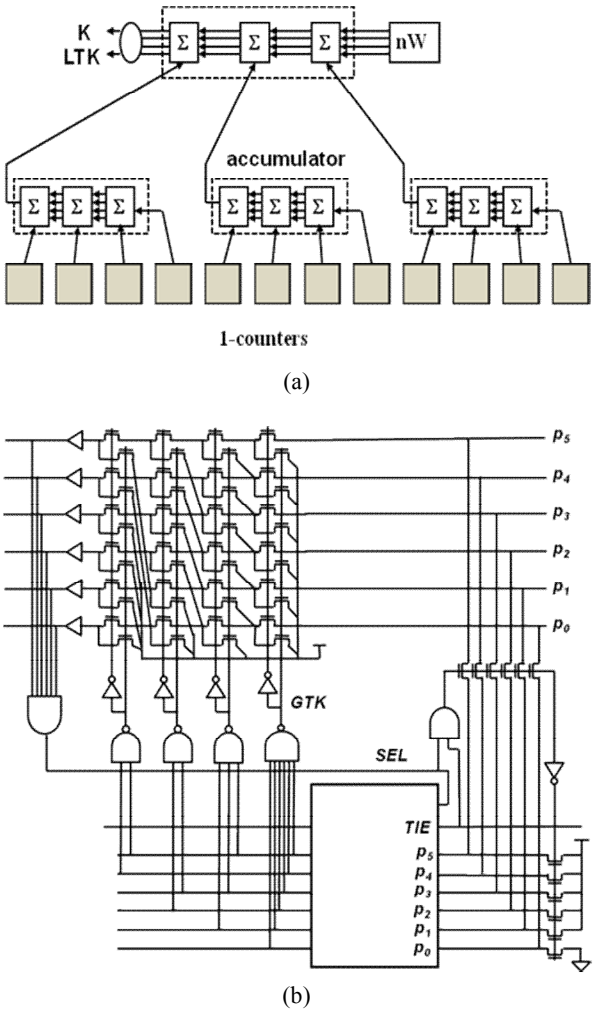

Fig. 5. Multi-level implementation of the winner-counter circuit (a) the structure of parallel counting circuit, (b) an example of Σ-circuit for k=5.

$O(N)$, and the whole kWTA operation needs (m+1) cycles, therefore, the overall delay is $O(mN)$. The single level implementation of the winner-counter circuit is suitable only for a small number of inputs. For a large number of inputs parallel implementation of winner-counter circuit is necessary for a high speed operation.

3. Parallel Multi-level Implementation

The delay of the single-level winner-counter greatly increases for a large number of inputs. To reduce the delay, parallelism can be employed in counting the number of 1s. As shown in Fig. 5(a), the 1-counter is divided into multiple blocks and the number of 1s in each block is counted in parallel with multiple smaller 1-counters. The results of each block are added in the upper

level by accumulators. An accumulator is a series connection of Σ-circuits. The structure of Σ-circuit for k=5 is shown in Fig. 5(b). For a large number of inputs, the accumulators can have a multi-level tree structure as in Fig. 5(a). The *n*W is stored, added, and updated at the top level.

To balance latencies of the 1-counter and the accumulator, the number of links of the tree (l), i.e. the number of Σ-circuits in an accumulator is chosen as

$$l \approx \mathrm{N}_1/\mathrm{N}_\Sigma \quad (1)$$

where N_1 is the number of inputs to an 1-counter block, and N_Σ to a Σ-circuit. N_Σ is given by

$$N_\Sigma = \lceil 1+\log_2(k+1) \rceil \quad (2)$$

With the multi-level implementation, the delay for counting n(T) is O(log N), and the worst case required cycle is $m+h$, where h is the number of levels. About one cycle per level is required to identify the boundary between the k-th and the (k+1)-th largest inputs of the same magnitude. If $m \gg h$, the overall delay of the multi-level parallel kWTA is $O(m \log N)$.

IV. EXPERIMENTS

The delay of single-level *k*WTA (SL*k*WTA) and multi-level parallel *k*WTA (ML*k*WTA) are estimated by SPICE simulation for various numbers of inputs. The simulation is performed by HSPICE with IBM's "1.2V-0.13μm 8RF-LM" model parameter. Through the simulations, the minimum clock periods (T_c) are obtained for the SL*k*WTA architecture and 2-level and 3-level ML*k*WTA architectures.

The results of the simulations with k=5 are shown in Table 1. For the 2-level and 3-level ML*k*WTA, various configurations are simulated and the best results are listed in Table 1. The delay of SL*k*WTA is almost independent to k, unless k varies widely. The delay of ML*k*WTA is affected by k through N_Σ (Eq. (2)), but its effect is negligible. Table 1 shows that the delay of SL*k*WTA (1-level) is proportional to N, while that of well-balanced ML*k*WTA increases along log$N$. The ML*k*WTA architecture can operate the *k*WTA module for 2048 digital inputs with a 100 MHz clock.

Table 1. Simulation Results for k=5

Number of Inputs	Minimum Clock Period				
	1-level	2-level		3-level	
	T_c (ns)	Configuration ($N_1 \times l$)	T_c (ns)	Configuration ($N_1 \times l \times l$)	T_c (ns)
8	1.3	-	-	-	-
16	2.0	-	-	-	-
32	3.3	-	-	-	-
64	6.1	**16×4**	3.6	-	-
128	12.1	**16×8**	5.1	**8×4×4**	4.2
256	23.1	**32×8**	6.6	**16×4×4**	5.1
512	48.4	**64×8**	9.7	**32×4×4**	6.4
1024	99.1	**64×16**	12.7	**16×8×8**	8.3
2048	193.9	**128×16**	18.3	**32×8×8**	9.6
4096	371.4	**128×32**	24.5	**64×8×8**	12.8

Table 2. Comparison to Existing Digital *k*WTA Architectures

Parameter	Architecture in			Proposed architecture
	[18]	[19]	[20]	
Technology	–	035 m-3.3V	FPGA	0.13 m-1.2V
Time	$O(N)$	$O(N)$	$O(N)$	$O(\log N)$
Area	$O(N^3)$	$O(N)$	$O(N)$	$O(N \log N)$
Latency	–	15 ns	7.9 ms	8.3 ns
		(N=4)	(N=64, k=4)	(N=1024, k=5)

In Table 2, the proposed architecture is compared to several existing digital *k*WTA architectures. The time complexity of the ML*k*WTA architecture is O(log$N$), while that of the existing *k*WTA architectures are $O(N)$. Therefore, the ML*k*WTA architecutre is superior in speed to existing architectures for a large number of inputs. The *k*WTA architecture in [19] operates with 66 MHz clock (T_c=15ns) in sorting 4 digital inputs with the *k*WTA circuits, while the SL*k*WTA can operate with 770 MHz clock (T_c=1.3ns) which is 11-times faster than the former architecture. Although it is only 11-times faster for 4-input kWTA operation, the discrepancy of speed would greatly increase along the number of inputs due to the difference of their time complexities. The architecture in [20] is implemented in FPGA (Altera Cyclone III EP3C120), so that it is not appropriate for the direct comparison. But, the ML*k*WTA is 64000 times faster than that anyway.

Not only digital *k*WTA circuits but also most of analog *k*WTA circuits [11-13] have $O(N)$ time complexity. Although it has been considered that the analog *k*WTA implementation is faster than its digital counterparts, the proposed *k*WTA architecture is much faster than its

analog counterparts. For example, the analog *k*WTA circuit in [13] takes 2 ms for *k*WTA of 1000 inputs with *k*=20, but the 3-level ML*k*WTA for 1000 (*m*=32) data takes only 291 ns, which is 6800 times faster than the former circuit.

Although the area complexity for ML*k*WTA is $O(N \log N)$, the concise structure of the 1-counter and the Σ-circuit diminishes its overheads, and the merit from $O(\log N)$ time complexity can overcome the overheads.

V. CONCLUSIONS

A parallel searching algorithm and its implementation circuits for digital *k*WTA are presented in this paper. In the new algorithm, all inputs are compared at the same time until *k*-winners are identified. The parallel operation of digital *k*WTA is possible by counting the number of winners in parallel. The single level counting circuit for a small number of inputs, and the multi-level counting circuit for a large number of inputs are presented. With the multi-level counting structure, the proposed *k*WTA has $O(\log N)$ time complexity.

The performance of the proposed architecture is verified by SPICE simulations. According to the simulation results, the proposed architecture takes 290ns in searching 5-winners among 1024 of 32-bit data. This high speed operation makes the new architecture attractive for searching *k* winners among a large number of inputs.

ACKNOWLEDGMENTS

This work was conducted by the research fund of Dankook University in 2015.

REFERENCES

[1] C. A. Marinov and J. J. Hopfield, "Stable computational dynamics for a class of circuits with $O(N)$ interconnections capable of KWTA and rank extractions". *IEEE Trans. Circuit. Syst.*, vol.52, no.5, pp. 949–959, 2005.

[2] M. Rahman, K. L. Baishnab, and F. A. Talukdar, "A high speed and high resolution VLSI Winner-take-all circuit for neural networks and fuzzy systems" *IEEE ISSCC2009,* pp. 1-4, 2009.

[3] A. Fish, D. Akselrod, and O. Yadid-Pecht, "High precision image centroid computation via an adaptive k-winner-take-all circuit in conjunction with a dynamic element matching algorithm for star tracking applications". *Analog Integ. Circuit. Signal Process*. vol. 39, pp. 251–266, 2004.

[4] A. K. J. Hertz and R. G. Palmer, *Introduction to the Theory of Neural Computation,* Redwood City, Addison-Wesley, 1991.

[5] D. Tian, Y. Liu, and D. Wei, "A Dynamic Growing Neural Network for Supervised or Unsupervised Learning," *Intelligent Control and Automation, WCICA 2006,* vol.1, pp. 2886-2890, 2006.

[6] U. Cilingiroglu and T. L. E. Dake, "Rank-order filter design with a sampled-analog multiple-winners-take-all core," *IEEE J. Solid-State Circuits,* vol. 37, no. 2, pp. 978-984, Aug. 2002.

[7] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254 – 1259, Nov. 1998.

[8] T. Eltoft and R. I. P. deFigueiredo, "A new neural network for cluster-detection-and-labeling," *IEEE Trans. Neural Networks,* vol. 9, no. 5, pp. 1021-1035, 1998.

[9] K. Urahama and T. Nagao, "*K*-winners-take-all circuit with 0(N) complexity," *IEEE Trans. Neural Networks,* vol. 6, pp. 776-778, May 1995.

[10] Z. Guo and J.Wang, "Information retrieval from large data sets via multiple-winners-take-all," *Circuits and Systems (ISCAS2011)* pp. 2669-2672, May 2011.

[11] B. Sekerkiran, and U. Cilingiroglu, "A CMOS K-winners-take-all circuit with O(n) complexity," *IEEE Trans. Circuits and Systems II,* vol. 46, no. 1, pp. 1-5, 1999.

[12] Y. Hung, C. Y. Tsai, and B. D. Liu, "1-V rail-to-rail analog CMOS programmable winner-take-all chip with two-side searching capability for neurocomputing applications," in *Proc. Neural Networks and Signal Processing,* vol.1, pp. 337-340, 2003.

[13] P.V. Tymoshchuk, "A fast analogue K-winners-take-all neural circuit," in *Proc. Neural Networks (IJCNN),* pp.1-8, 2013.

[14] N. Kumar, P. O. Pouliquen, and A. G. Andreou,

"Device mismatch limitations on the performance of a Hamming distance classifier" in *Proc. Defect and Fault Tolerance in VLSI Systems*, pp. 327-334 1993.

[15] P. V. Tymoshchuk, and M. P. Tymoshchuk, "Stability and convergence analysis of model state variable trajectories of analogue KWTA neural circuit," in *Proc. Direct and Inverse Problems of Electro-magnetic and Acoustic Wave Theory,* pp. 26-35, 2011.

[16] A. Kapralski, "The maximum and minimum selector SELRAM and its application for developing fast sorting machines," *IEEE Trans. Computers,* vol. 38, no. 11, pp. 1572-1577, 1989.

[17] M. Ogawa, K. Ito, and T. Shibata, "A general-purpose vector- quantization processor employing two-dimensional bit-propagating winner-take-all" *IEEE Sym. VLSI Circuits Digest of Tech. Papers,* vol. 35, no.11, pp. 244-247, 2002.

[18] T. C. Hsu and S. D. Wang, "k-Winners-take-all neural net with O(1) time complexity". *IEEE Trans. Neural Networks.* vol. 8, no. 6, pp. 1557–1561, 1997.

[19] C. S. Lin, P. Ou, and B. D. Liu, "Design of k-WTA/Sorting Network Using Maskable WTA/MAX Circuit". In *Proc. VLSI Symposium on Technology, Systems and Applications*, pp. 69–72, June 2001.

[20] H. Y. Li, C. M. Ou, Y.T. Hung, W. J. Hwang, and C. L. Hung, "Hardware Implementation of k-Winner-Take-All Neural Network with On-chip Learning," in *Proc. Computational Science and Engineering,* pp. 340-345, Dec. 2010.

Myungchul Yoon received the BS and MS degrees in electronics engineering from Seoul National University, Korea, in 1986 and in 1988 respectively, and the Ph.D. degree in Electrical and Computer Engineering from the University of Texas at Austin in 1998. From 1988 to 2002, he was with Hynix Inc. Icheon, Korea as technical research staff at Semiconductor R&D Lab. and Mobile Communication R&D Lab. From 2005 to 2006, he was with DGIST, Korea as technical staff at the Information Technology R&D Division. Since 2006, he has been with the Department of Electronics Engineering, Dankook University, Cheonan, Korea, where he is a professor. His research interests are in low-power VLSI design, embedded systems, mobile communication, and wireless personal area networks.