VLSI DESIGN
2000, Vol. 11, No. 2, pp. 115–128
Reprints available directly from the publisher
Photocopying permitted by license only

A Modular and Scalable Architecture for the Realization of High-speed Programmable Rank Order Filters Using Threshold Logic

İ. HATIRNAZ, F. K. GÜRKAYNAK and Y. LEBLEBICI*

Worcester Polytechnic Institute, Department of Electrical and Computer Engineering, Worcester, MA 01609-2280

(Received 1 June 1999; In final form 10 November 1999)

We present a new scalable architecture for the realization of fully programmable rank order filters (ROF). Capacitive Threshold Logic (CTL) gates are utilized for the implementation of the multi-input programmable majority (voting) functions required in the architecture. The CTL-based realization of the majority gates used in the ROF architecture allows the filter rank as well as the window size to be user-programmable, using a much smaller silicon area, compared to conventional realizations of digital median filters. The proposed filter architecture is completely modular and scalable, and the circuit complexity grows only linearly with maximum window size (m) and with word length (n). A prototype of the proposed filter circuit has been designed and fabricated using double-polysilicon 0.8 μm CMOS technology. Detailed post-layout simulations and test results of the ROF prototype circuit indicate that the new architecture can accommodate sampling clock rates of up to 50 MHz, corresponding to an effective data processing rate of 800 Mb/s for a very large filter with window size 63 and word length of 16 bits.

Keywords: Rank-order filters, median filters, DSP architectures, majority function, voting circuits, capacitive threshold circuits

1. INTRODUCTION

As a generic definition, the rank order filter (ROF) is a non-linear digital filter which determines the i-th ranking element in a given window consisting of (m) binary encoded input words (vectors). In a simple one-dimensional example, the rank-order filter would process a certain number of input vectors contained in a sliding window, and produce an output that corresponds to the i-th ranked vector in the current window, as illustrated in Figure 1. As the sliding window moves by one vector, the overall ranking (rank-ordering) will have to be updated to produce the next output, again corresponding to the i-th ranked vector in the new window. Figure 2 shows the

*Corresponding author.

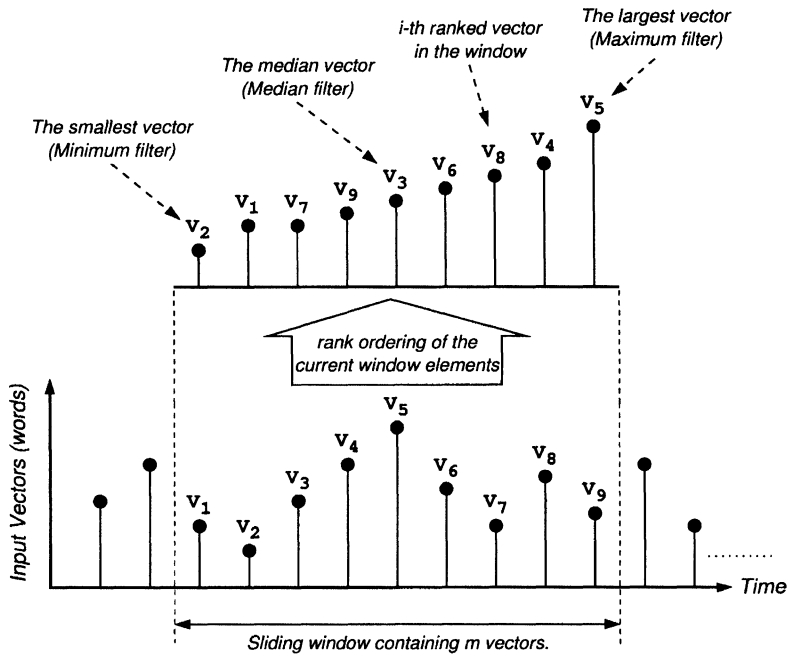


FIGURE 1 One-dimensional illustration of the rank-ordering process.

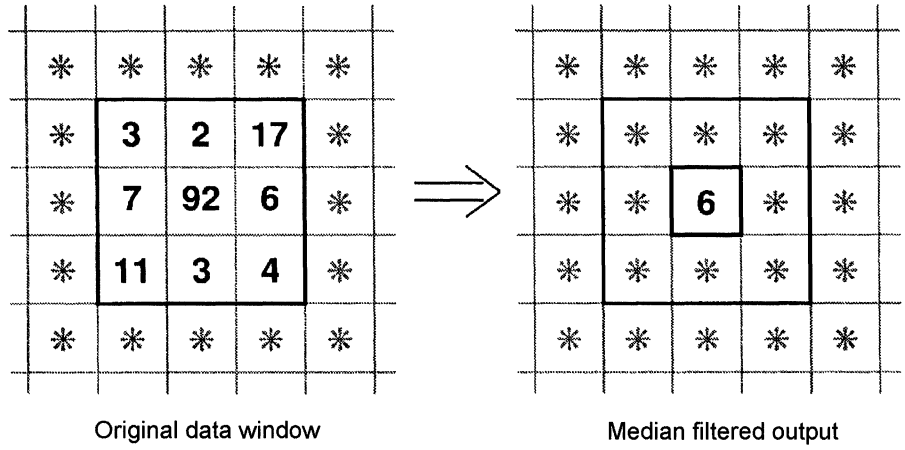


FIGURE 2 Two dimensional application of the rank-order (median) function to a (3 × 3) window.

two-dimensional application of this principle on a simple (3 × 3) window, where the center element (pixel) is being replaced by the median value contained in the current window.

Popular special cases of rank order filters are median, minimum and maximum filters, where the output is determined as the median, the minimum or the maximum value within the input window, respectively [1]. Variants of ROFs are widely used in digital signal and image/video processing because of their non-linear characteristics. Especially, median filters have found many applications in digital image enhancement, such as reducing the high frequency and impulsive noise in digital images without the extensive blurring and edge destruction [2, 3]. Other successful applications of

ROFs include the smoothing of noisy pitch contours in speech signals, data compression in block truncation coding schemes, speckle noise reduction in coherent imaging systems, and preprocessing data for machine vision.

Several algorithms have been proposed for rank order filters that are based on data sorting. Although these algorithms are suitable for software implementations, they usually result in inefficient hardware structures, since they process the input vectors at the word level. Implementations based on stack filters have an area-time complexity of $O(n^2)$, and the hardware complexity increases very rapidly with window size (m) [7].

In recent years, some innovative bit-serial structures for rank-order-filters have been presented, which are mostly based on majority-decision algorithms [4, 6, 9]. Yet, the majority function is typically hard to realize using conventional Boolean building blocks, since it requires a large number of gates and a large logic depth. Consequently, such structures suffer from speed and area limitations, especially if the window size becomes larger than 10 vectors. Also, most of the conventional realizations result in a fixed rank and a fixed window size, which limit the flexibility of their application.

In this paper, we present a new architecture for the realization of fully programmable ROFs based on threshold logic gates, resulting in a very compact and highly modular structure. The architecture consists of a regular array that is composed of only two types of building blocks, and it allows the construction of filter structures of arbitrary size. The processing efficiency of the proposed architecture is significantly increased by fine-grain pipelining in both directions within the array, where the clock frequency remains essentially independent of the window size (m) and word length (n).

The organization of this paper is as follows: The outline of a simple bit-serial algorithm for rank ordering is presented in Section 2. In Section 3, the implementation of the programmable ROF architecture is discussed, and the main building blocks are presented. The structure and operation of the multi-input majority (voting) function blocks are presented in Section 4, followed by a discussion of the prototype ROF circuit and its test results in Section 5. Finally, the conclusions are summarized in Section 6.

2. THE RANK ORDERING ALGORITHM

2.1. Algorithm Description

A bit-serial algorithm first proposed in [6] was chosen as the basis of the programmable rank-order filter architecture implemented in this work. In this algorithm, the problem of finding a rank-order-selection for n-bit long words is reduced to finding "n" rank-order-selections among 1-bit numbers.

The algorithm can be summarized as follows:

```
begin
  for l := 1 to n do
    begin
      – sum[l] is the sum of the
      – l-th bit of all the numbers
      sum[l] := 0;
      for j := 1 to m do
        sum[l] := sum[l] + a_jl;
      if sum[l] < = m – i then
      – selecting the i-th ranked bit
      selector[l] := 0;
    else
      selector[l] :=1;
      – where the selector[l] is the l-th
      – bit of the i-th ranked number
    for j := 1 to m do
      begin
      if a_jl != selector[l] then
        for q:= (l + 1) to n do
          a_jq := a_jl;
      end
    end
    i-th-ranked-number := selector;
    – selector is the concatenation of
    – selector[1], ..., selector [n]
end
```

The algorithm starts by processing the most significant bits (MSB) of the $m=(2N+1)$ words in the current window, through an m-input programmable majority gate (voting gate), to yield the MSB of the desired filter output. The key function performed by the m-input voting gate can be described as $(\geq k-out-of-m)$, which means that the gate output is logic "1" iff (k) or more of the inputs are equal to "1". This output is then compared with the MSBs of the window elements. The vectors whose MSB is *not* equal to the filter output have their MSB propagated down by one position, replacing the less significant bits of the corresponding words. This process is continued for the following bit-planes. Thus, any bit that is not equal to the corresponding stage output is propagated down to the lesser significant positions, until the least significant bit is processed. This process ensures that all numbers that are greater (or less) than the i-th ranked number in the window progressively eliminated, eventually leaving the correct output as the only candidate.

Figure 3 shows an example where, five 8-bit words (denoted P through T with decimal values of 184, 105, 194, 117 and 75 respectively) are being rank-ordered using the algorithm described above. The window size is $m=5$ and the rank is $k=3$, indicating that the third smallest among these five numbers is being found in 8 steps. Note that the main bit-level operation at each step amounts to a majority (rank) decision among n bits of the same bit-plane.

In Step 1, the most significant bit plane is processed, and the output is determined as "0" since only two of the MSBs are equal to "1". Notice that the voting function performed by the programmable majority gate at this bit-plane corresponds to a $(\geq 3-out-of-5)$ function. Immediately following this majority decision, the MSBs that do not coincide with the bit-plane output (*i.e.*, the MSB of vector P and vector R) are propagated down to lesser significant bit-planes, thereby eliminating these two vectors as potential candidates for output.

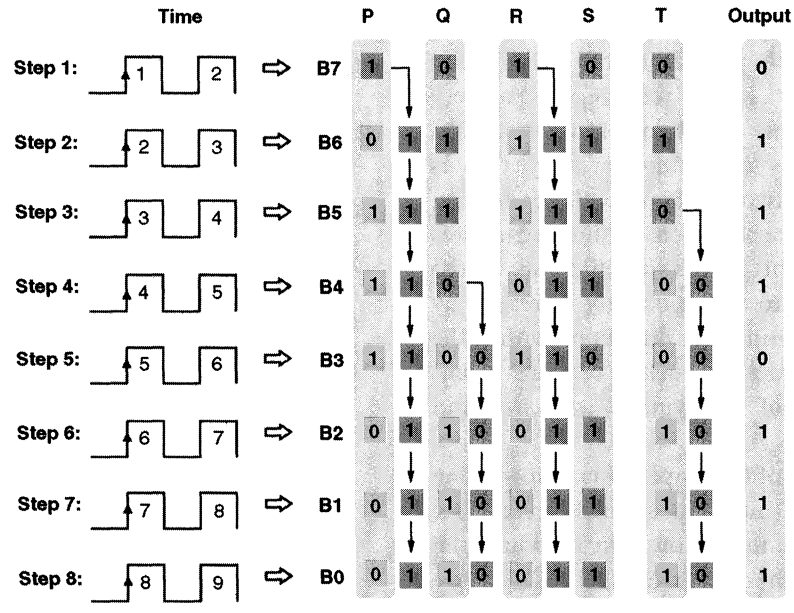


FIGURE 3 An illustration of the rank-ordering algorithm, for five 8-bit words.

In Step 2, the majority output is "1" and all 5 bits at this bit-plane match the output; therefore, the process is continued on to the next bit-plane. In Step 3, vector T is eliminated by propagating its "0"-bit down to lesser significant bit-planes.

It is worth noting that in some cases, the elimination process may determine the correct output before all bit-planes are processed (as in the example, where the output vector S is essentially found after the 4th step). Yet in our implementation, the algorithm is allowed to progress until it reaches the least significant bit-plane, simply to preserve the timing integrity in subsequent runs. Also note that the algorithm allows bit-level pipelining: As the process propagates through lesser significant bit-planes, the more significant bit-planes can start operating on the next input vector set.

2.2. Realization of the Algorithm

The bit-serial operation flow of the algorithm described above suggests a very simple bit-level pipelined data path architecture. Figure 4 shows the conceptual hardware implementation of the operations associated with one bit-plane.

Note that each bit-plane-module consists of two main blocks:

1. The modifier/selector(propagator) block whose function is to store and to shift the actual data and to calculate the selector signal for the next processing block.
2. The majority or rank decision block which determines the output bit as a function of (m) bits in each bit-plane, with a ($\geq k - out - of - m$) operation.

In the modifier/selector block, also called a ROF-cell, the output of the majority function is compared with the corresponding data bit, using an XNOR gate. The result of this XNOR operation is then combined (AND operation) with the select signal originating from the previous block. This provides the information if the data bit taken from the previous block is a propagating one or not. If the data bit is a propagating one, then the new select signal will be "0", indicating that this data bit will continue propagating unchanged through the following stages. Otherwise, the select signal will only depend on the result of the comparison of the filter-slice output with the

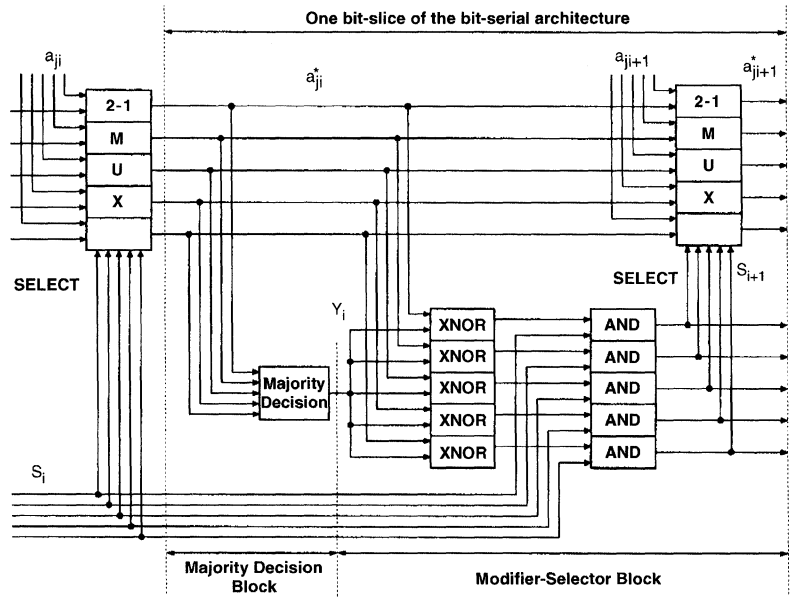


FIGURE 4 A 1-bit slice implementation of the rank ordering algorithm (for $m = 5$).

current data bit. Identical 1-bit filter slices can be used in sequence (cascade configuration) in order to process input vectors of arbitrary bit-length. Thus, the filter throughput can be increased significantly by bit-level pipelining. The modular structure of the one-bit slice described above also allows for scalable realization of the ROFs with different window sizes and word lengths.

3. IMPLEMENTATION OF THE PROGRAMMABLE ROF ARCHITECTURE

3.1. System Components

The proposed modular architecture consists of two types of blocks (cells): The ROF-cell and the majority decision gate. By using these two blocks, a programmable rank-order filter of any window size and word-length can be realized. The word-length dictates the number of the majority decision gates, whereas the window size determines the number of ROF-cells driving one of these majority gates. Figure 5 shows the ROF-cell block realization at gate level. Note that in addition to the simple Boolean operators, three DFFs are utilized for bit-level pipelining in both dimensions. At each positive clock edge, the corresponding select and data signals are fed to the next blocks, and all data transmission occurs between *neighboring* blocks, with the exception of the majority outputs that are transmitted to all ROF-cells in the corresponding bit-plane.

A typical layout of the ROF-cell is also shown in Figure 5. It can be seen that the layout design of this cell permits modular expansion of the array in

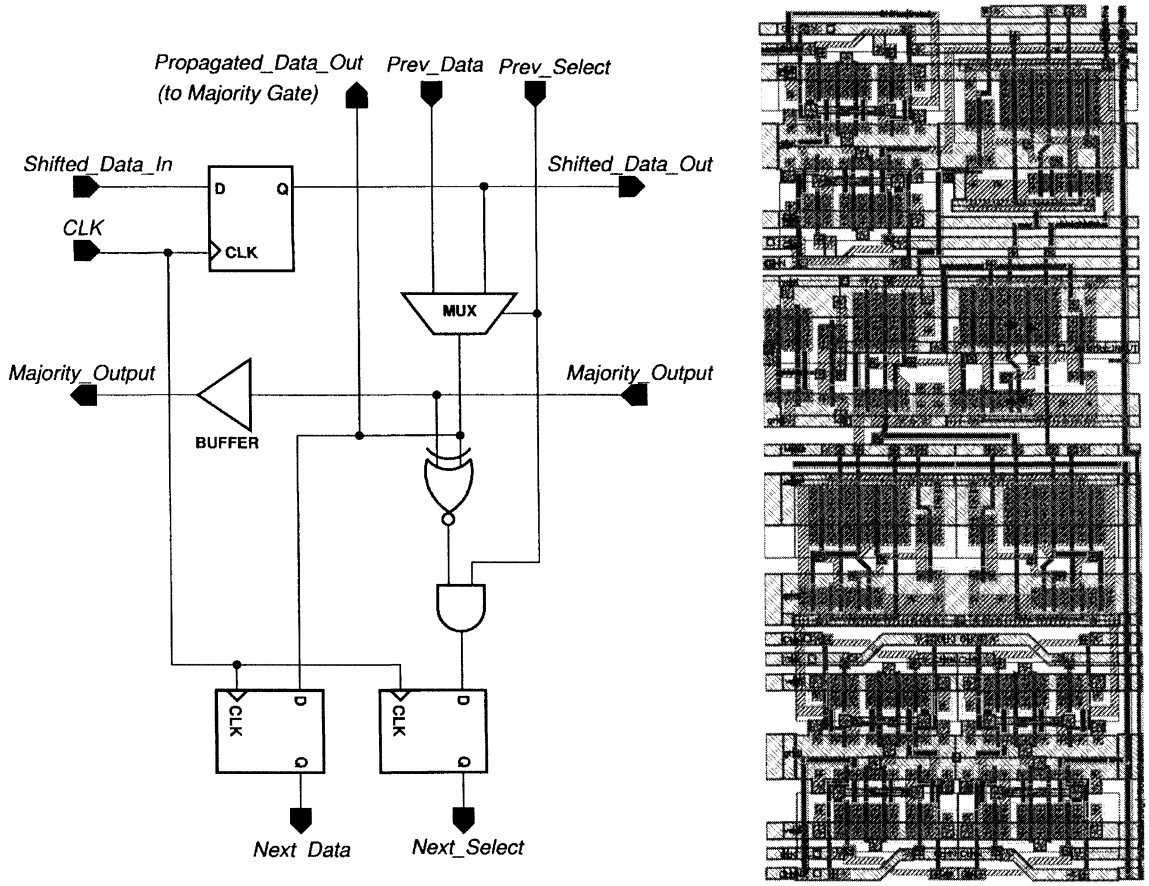


FIGURE 5 Gate-level structure of a ROF-cell and the corresponding layout, allowing modular expansion.

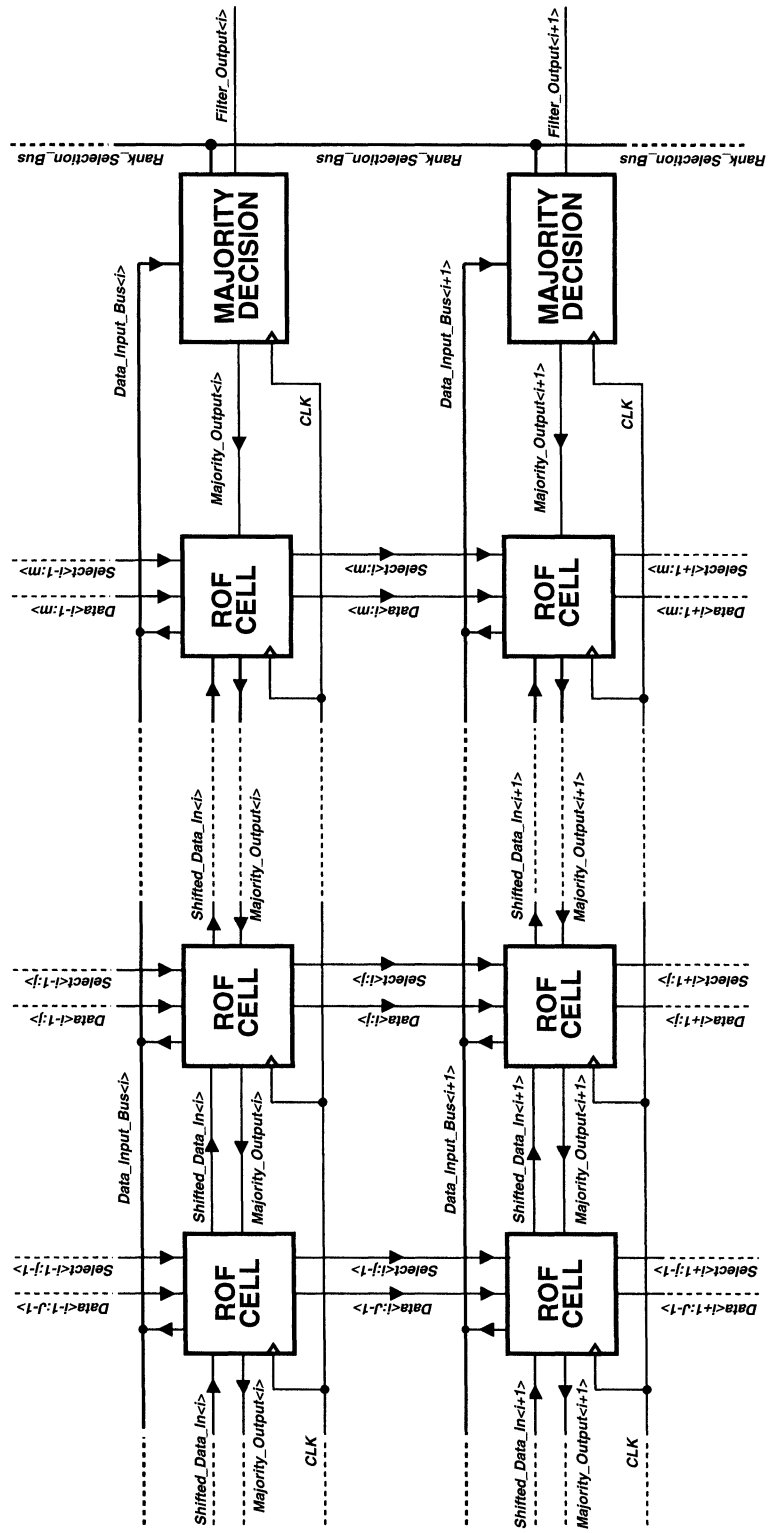


FIGURE 6 Detailed signal flow between modular ROF-cells and majority gates.

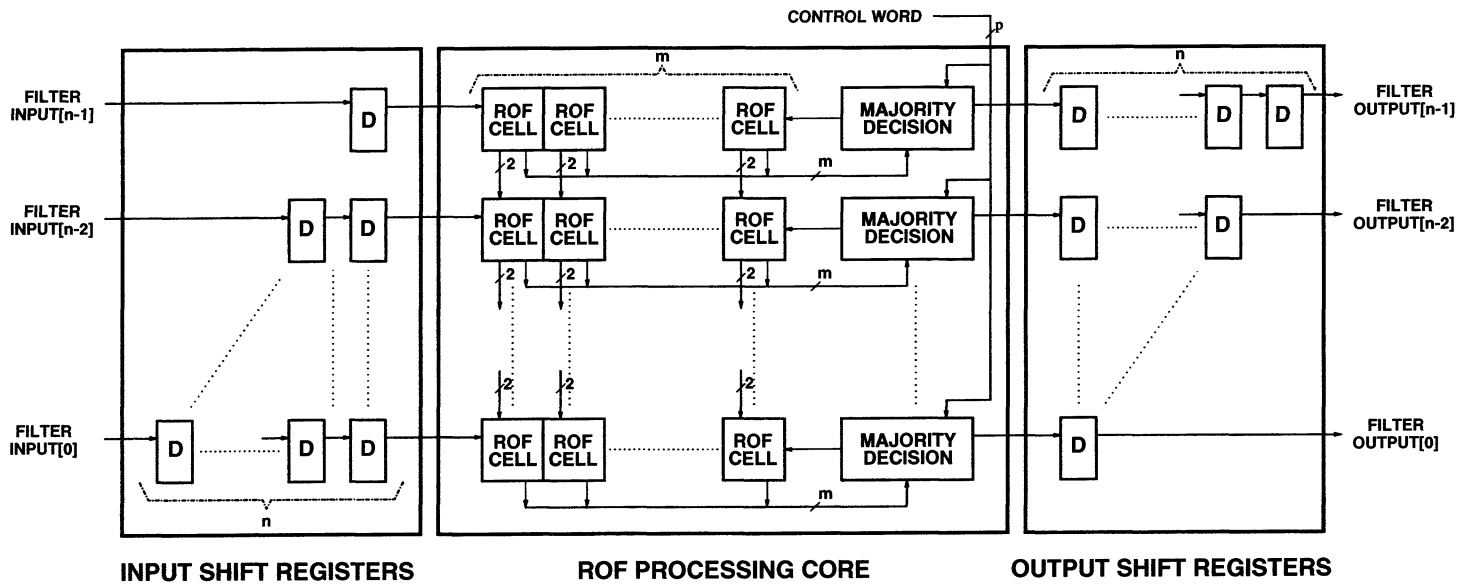


FIGURE 7 The top-level architecture of a $(n \times m)$ programmable CTL based ROF. Here (n) denotes the bit-length of the input vectors, and (m) denotes the maximum window size.

both dimensions, by abutting the input and output signal pins of each cell with those of the neighboring cells. The programmable majority (voting) gate associated with each bit-plane is responsible for processing the propagated data output of all ROF-cells in that bit-plane. This critical system block can be realized as a single capacitive threshold logic gate, which is described in detail in Section 4.

3.2. Overall System Architecture

The signal flow between the ROF cells and the majority gates are shown in Figure 6. The modular architecture consisting of only two major blocks enables fully scalable construction of filter structures of arbitrary size. The ROF core has $(n \times m)$ ROF cells where $m = (2N+1)$ is the window size and (n) is the bit-length of the input words. The ROF cells processing the bits of same significance provide the necessary inputs to the corresponding majority decision block which determines the filter output bit of that level. This output bit is fed to the output shift registers and back to the ROF cells, to be used in determining the select and data signals which will be the inputs of the next stage.

The top level block diagram of the programmable ROF design is shown in Figure 7. The architecture consists of three main blocks: input shift registers, ROF processing core, and output shift registers. To allow bit-level pipelined operation, the input bits are ordered using a staggered shift register array (Fig. 7). As such, the ROF pipeline accepts a new input vector of word length (n) and produces a valid output vector in each clock period. The overall latency of this architecture is $(n-1)$ clock cycles, meaning that the output vector generated at any given clock cycle corresponds to the input window which is completely shifted into the ROF core $(n-1)$ cycles earlier.

3.3. Advantages of the Proposed Architecture

The realization of fully programmable rank order filters has traditionally been a very challenging design problem, mainly due to the fact that the rank selection function (programmable majority function) is extremely hardware-intensive using conventional design approaches. As a result, most of the design efforts so far have either been constrained to median-only filters without any rank selection capability, and/or to relatively small window sizes [7, 8]. The architecture presented here is superior to other ROF implementations, with its following capabilities:

1. The CTL-based realization of the majority gates used in the ROF architecture allows the filter rank and the window size to be fully programmable, using a much smaller silicon area.
2. The rank-ordering algorithm implemented with this architecture does not require the elements of the input window to be pre-ordered, as opposed to other, stack-based ordering algorithms [7]. This increases the processing efficiency.
3. The proposed ROF has a modular architecture which enables easy expandability of the window size and word length of the input vectors without a dramatic change in performance.
4. The overall circuit complexity increases *linearly* both with maximum window size (m) and with word length (n).
5. The clock frequency is essentially independent of the input window size (m), and the overall latency of the pipeline is $(n-1)$ clock cycles.

4. REALIZATION OF THE MAJORITY GATE USING CAPACITIVE THRESHOLD LOGIC

The design of the modifier/selector block shown in Figure 4 is relatively straight-forward using conventional CMOS logic gates, whereas the majority decision block presents a bottleneck with conventional realization methods both in terms of circuit complexity and in terms of logic depth. This well-known limitation has traditionally been a significant impediment for the hardware realization of similar structures [1, 8, 11, 12]. Since the circuit

complexity (number of gates) and depth (number of stages) increase rapidly with the number of inputs, the practical fan-in of voting circuits is usually restricted. For example, a conventional realization of the 63-bit majority gate would require an equivalent of 63 6-bit full-adder circuits, arranged in a network of a logic depth of 64 (synthesized from HDL description).

Since the majority decision is in fact a threshold function, it can be most efficiently implemented with threshold logic. Therefore, capacitive threshold logic (CTL) gates, presented earlier by the authors, offer a more efficient realization of the majority function compared to conventional logic gates [5]. The most significant advantage of the CTL-based realization is that it allows the construction of a very-large input programmable majority (voting) function as a single-level logic gate. As presented in [5], a 31-input programmable majority gate based on CTL is almost three times faster than a full custom standard CMOS realization [10] and occupies approximately one third of the area which results in a area-delay performance increase of nearly an order of magnitude. In addition, the CTL-based majority gate can be easily integrated with the conventional CMOS gates used in the architecture, since the input and output signals are fully CMOS compatible.

The circuit layout of a 15-input single-level programmable majority gate based on CTL is shown in Figure 8, occupying a silicon area of ($320\,\mu m \times 70\,\mu m$) using 0.8 μm CMOS technology. The circuit consists of an array of unit weight capacitors, simple input and threshold switches to control the input variables, and an output voltage comparator to generate the majority output. The layout area of a 31-input majority gate is only twice as big as that of the 15-bit input gate, with ($620\,\mu m \times 70\,\mu m$). The operation of the 31-input majority circuit is demonstrated in Figure 9, where the voting threshold is set to 16 for the first 7 consecutive inputs, and then the threshold is reprogrammed to 5 for the next 8 inputs. This circuit is capable of producing a majority output bit with a typical propagation delay of about 3ns.

A larger version of the CTL-based programmable majority gate with 63 parallel inputs has also been designed for use as the main building block of the ROF prototype chip. The 63-input gate occupies a silicon area of ($625\,\mu m \times 130\,\mu m$), and its worst-case input-to-output propagation delay remains less than 8 ns. Notice that one of the most important advantages of CTL-based realization of majority gates is that the silicon area strictly increases linearly with the input size, while the propagation delay of this one-level structure remains a weak logarithmic function of fan-in.

5. EXPERIMENTAL RESULTS AND DESIGN VALIDATION

A prototype ROF test circuit has been designed using a conventional 0.8 μm double-polysilicon CMOS process, to validate the main operation principles of the architecture. The prototype blocks consist of four bit-level pipeline stages, each of

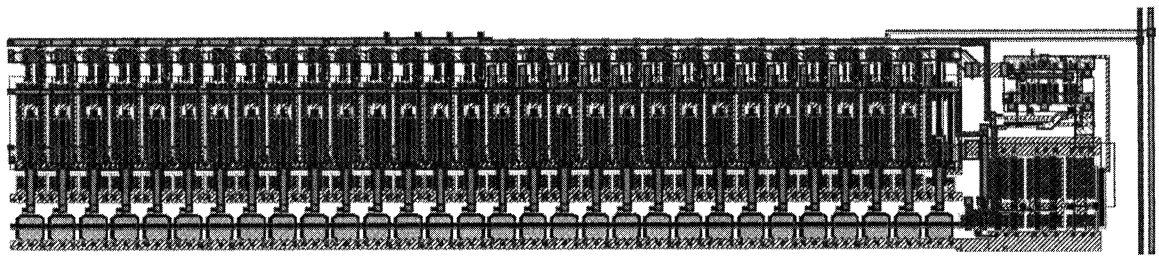


FIGURE 8 Layout of the 15-bit input majority gate. The silicon area is ($320\,\mu m \times 70\,\mu m$) using 0.8 μm CMOS technology.

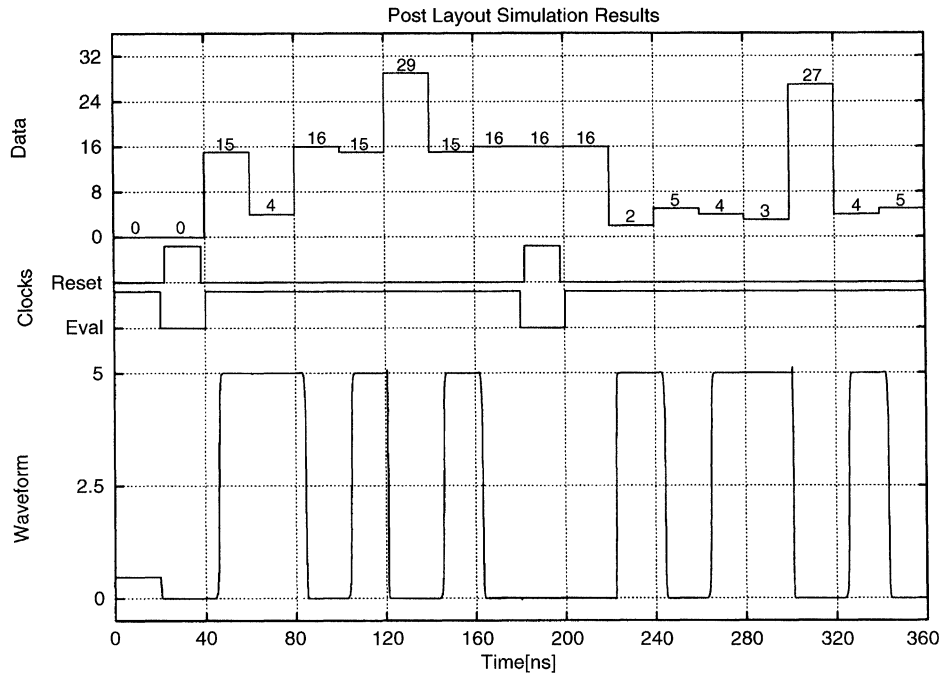


FIGURE 9 The illustration of the 31-input majority gate operation.

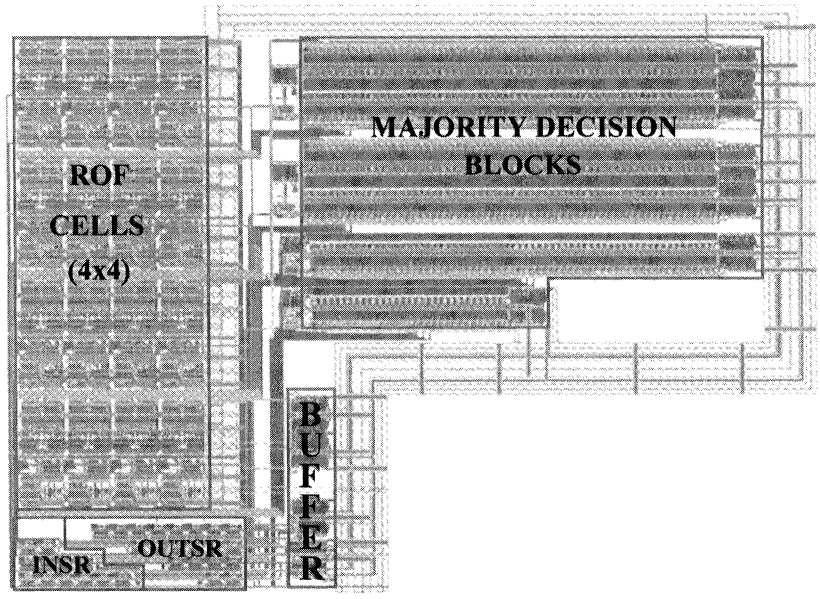


FIGURE 10 Top-level layout of the fabricated prototype circuit.

which contains a 63-bit programmable majority gate to handle the rank selection. To limit overall circuit complexity and to enable easier testing, each stage is designed to process a maximum window size of four samples. The window size capability of the ROF circuit can be easily increased to

accommodate up to 63 samples, since the programmable majority gates already support this window size. The top level layout of this ROF prototype is shown in Figure 10. The circuit occupies a silicon area of (800 μm × 1150 μm).

The operation of the ROF architecture is demonstrated with detailed measurement results in Figure 11 which were obtained using an HP 16500C Logic Analysis System. Here, the window size is 4, and the rank is selected to be 2 – meaning that the second largest input word in the sample window will have to be selected. All four bits of the input and the output are displayed individually. The input words for this sample window are "0101", "0110", "0111" and "1000". The correct output, "0111" appears after 3 clock cycles.

Figures 12(a) and 12(b) show the measured filter output for longer input sequences, again demon-

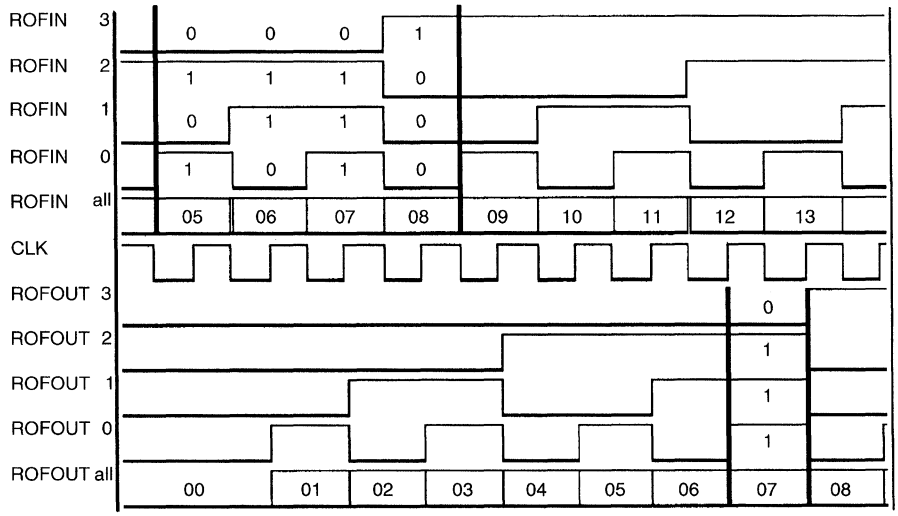
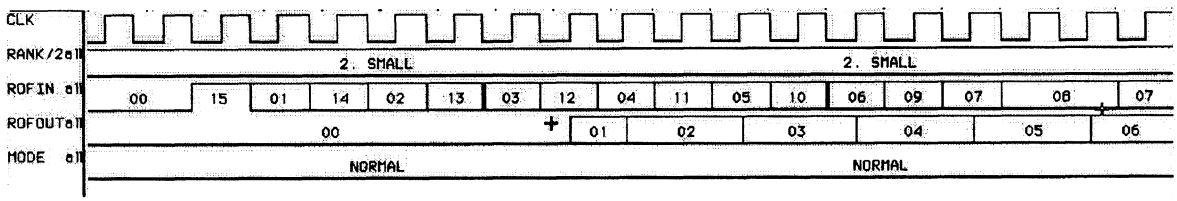
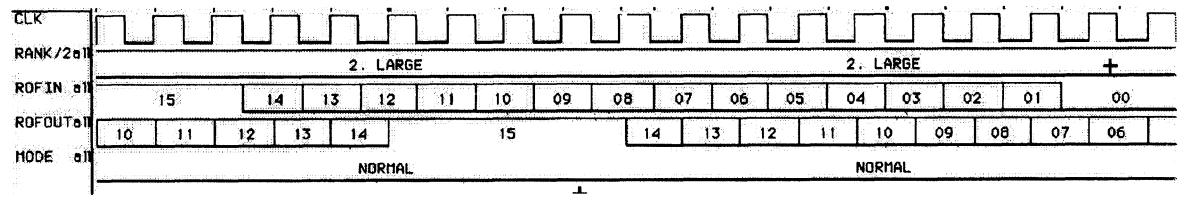


FIGURE 11 Measured output sequences of the prototype circuit.



(a) Searching for the second smallest vector in the window.



(b) Searching for the second largest vector in the window.

FIGURE 12 Measurement results for longer sequences.

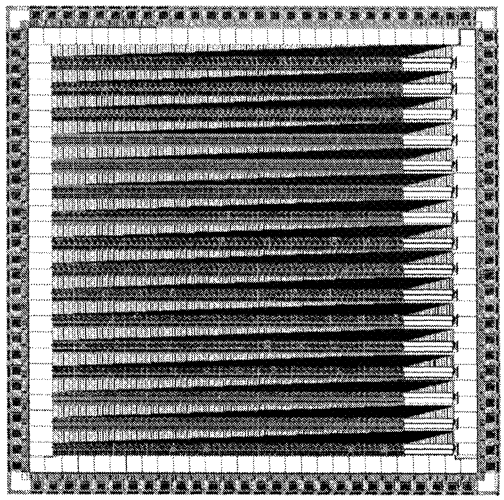


FIGURE 13 Top-level layout of the ROF circuit with a window size of 63 and word-length of 16 bits. The silicon area is approximately (5 mm × 5 mm).

strating that the prototype circuit produces the correct output sequence with a latency of 3 clock cycles.

Finally, to serve as a dramatic demonstration of the proposed filter architecture, the top-level layout of a programmable ROF core with a maximum window size of 63 samples and a word-length of 16 bits is shown in Figure 13. To our knowledge, the design of a rank order filter of this complexity has not been attempted before. The circuit occupies a silicon area of approximately (5 mm × 5 mm) with 0.8 μm double-poly-silicon CMOS technology, and operates with a latency of 15 clock cycles at the clock frequency of 50 MHz, which results in an effective data rate of 800 Mbits/s. The silicon area as well as the performance of this structure compare very favorably not only with any of the existing filter architectures proposed so far [7, 8], but also with any extrapolated characteristics thereof for larger window size and word lengths.

6. CONCLUSION

In this paper, we have presented a scalable architecture for the realization of fully programmable rank-order filters. The bit-serial realization of the rank ordering algorithm allows a simple fine-grained pipelined filter architecture which is highly modular and easily expandable. The overall performance and processing efficiency of the architecture are also boosted significantly by the bit-serial pipelining. Capacitive Threshold Logic (CTL) gates are utilized for the implementation of the multi-input programmable majority (voting) functions required in the architecture. The CTL-based realization of the majority gates used in the ROF architecture allows the filter rank as well as the window size to be user-programmable, using a much smaller silicon area, compared to conventional realizations of digital median filters.

The circuit complexity of the proposed architecture grows only linearly both with window size (m) and with the word length (n) of the input vectors. The clock frequency remains essentially independent of the window size and word length, and the ROF core is capable of producing a new output vector (corresponding to the input window) at each clock period, with a latency of $(n-1)$ cycles. The operation and performance of the proposed circuit architecture has been experimentally demonstrated with a prototype chip. Results indicate that the new architecture can accommodate sampling clock rates of up to 50 MHz, corresponding to an effective data processing rate

of 800 Mbits/s for a very large filter with window size 63 and word length of 16 bits.

References

[1] Richards, D. S., "VLSI median filters", *IEEE Trans. Acoust. Speech, Signal Processing*, **38**, 145–152, January, 1990.

[2] Fitch, J. P., Coyle, E. L. and Callagher, N. C. (1984). "Median filtering by threshold decomposition", *IEEE Trans. Acous. Speech, Signal Proc.*, **32**, 1183–1188.

[3] Huang, T. S. and Yang, G. J., "Median filters and their applications to image processing", *School of Elec. Eng.*, Purdue Univ., West Lafayette, IN, TR-EE 80-1, January, 1980.

[4] Gasteratos, A., Andreadis, I. and Tsalides, Ph. (1997). "Realization of rank order filters based on majority gate", *Pattern Recognition*, **30**(9), 1571–1576.

[5] Leblebici, Y., Gurkaynak, F. K. and Mlynek, D., "A compact 31-input programmable majority gate based on capacitive threshold logic", In: *Proc. IEEE Int. ASIC Conference 1998*, pp. 281–285.

[6] Kar, B. K. and Pradhan, D. K., "A new algorithm for order statistic and sorting", *IEEE Trans. on Signal Processing*, **41**, 2688–2694, August, 1993.

[7] Lin, C. C. and Kuo, C. J., "Fast response 2-D rank order algorithm by using max-min sorting network", *International Conference on Image Processing 1996*, **1**, 403–406.

[8] Chen, C., Chen, L., Chiueh, T. and Hsiao, J., "An efficient pipelined VLSI implementation of rank order filter", *ISSIPNN 1994*, **2**, 630–633.

[9] Lee, C. L. and Jen, C. W. (1992). "Bit-sliced median filter design based on majority gate", In: *Proc. Ins. Elec. Eng. G*, **139**, 63–71.

[10] Swartzlander, E. E. Jr., "Parallel counters", *IEEE Trans. Comput.*, **34**, 1021–1024, November, 1973.

[11] Wendt, P. *et al.* (1986). "Stack filters", *IEEE Trans. Acoust. Speech, Signal Processing*, pp. 898–911.

[12] Gu, Q. and Swamy, M. N. S., "A binary logic synthesis approach to the bit-level implementation of generalized rank-order filters", *Proc. ISCAS92*, pp. 109–112.

Authors' Biographies

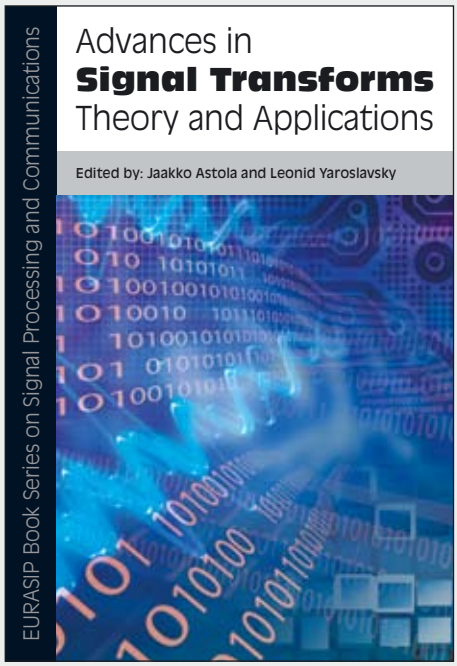
Ilhan Hatirnaz received his B.Sc. degree in electrical and electronics engineering from Istanbul Technical University in 1996. From 1996 to 1998, he worked as a design engineer at the Marmara Research Institute of the Turkish Scientific and Technological Research Council. Since Fall 1998, he has been a graduate student at the Electrical and Computer Engineering Department of Worcester Polytechnic Institute. His research interests include VLSI design, novel charge-based circuit structures for data processing, and digital signal processing architectures.

Frank K. Gurkaynak received his B.Sc. and M.Sc. degrees in electrical and electronics engineering from Istanbul Technical University in 1994 and 1998, respectively. From May, 1997 to December, 1997, he also worked as a research assistant at the Integrated Systems Laboratory of the Swiss Federal Institute of Technology (EPFL) in Lausanne, Switzerland. Currently, he is a graduate student at the Electrical and Computer Engineering Department of Worcester Polytechnic Institute. His research interests include full-custom design methodologies, layout design automation for VLSI, digital arithmetic building blocks, and data-path synthesis methodologies.

Yusuf Leblebici received the B.Sc. and M.Sc. degrees in electrical engineering from Istanbul Technical University in 1984 and 1986, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 1990. He worked as Visiting Assistant Professor at UIUC between 1991 and 1993, as Associate Professor of Electrical Engineering at Istanbul Technical University between 1993 and 1998, and as Invited Professor at the Swiss Federal Institute of Technology in Lausanne, Switzerland between 1996 and 1998. Currently, he is Associate Professor of Electrical and Computer Engineering at Worcester Polytechnic Institute. His research interests include digital and mixed-signal integrated circuit design, VLSI design automation and layout synthesis methodologies, and novel architectures for digital signal processing applications.

Advances in Signal Transforms: Theory and Applications

Edited by: J. Astola, and L. Yaroslavsky



Digital signal transforms are of a fundamental value in digital signal and image processing. Their role is manifold. Transforms selected appropriately enable substantial compressing signals and images for storage and transmission. No signal recovery, image reconstruction, and restoration task can be efficiently solved without using digital signal transforms. Transforms are successfully used for logic design and digital data encryption. Fast transforms are the main tools for acceleration of computations in digital signal and image processing.

The volume collects in one book most recent developments in the theory and practice of the design and usage of transforms in digital signal and image processing. It emerged from the series of reports published by Tampere International Centre for Signal Processing, Tampere University of Technology. For the volume, all contributions are appropriately updated to represent the state of the art in the field and to cover the most recent developments in different aspects of the theory and applications of transforms.

The book consists of two parts that represent two major directions in the field: development of new transforms and development of transform-based signal and image processing algorithms. The first part contains four chapters devoted to recent advances in transforms for image compression and switching and logic design and to new fast transforms for digital holography and tomography. In the second part, advanced transform-based signal and image algorithms are considered: signal and image local adaptive restoration methods and two complementing families of signal and image resampling algorithms, fast transform-based discrete sinc-interpolation and spline-theory-based ones.

Topics and Features:

- The subject of the book is of a fundamental importance in digital signal and image processing.
- A variety of signal and image processing tasks are considered and treated on the common methodological base of transform domain processing.
- Theoretical results are strongly application-oriented.

EURASIP Book Series on SP&C, Volume 7, ISBN 977-5945-55-0.
Please visit http://www.hindawi.com/spc.7.html for more information about the book. To place an order while taking advantage of our current promotional offer, please contact books.orders@hindawi.com