ELSEVIER

Signal Processing 41 (1995) 101–110

SIGNAL PROCESSING

Decompositional methods for stack filtering using Fibonacci p-codes

S. Agaian[a], J. Astola[b], K. Egiazarian[b,*], P. Kuosmanen[b]

[a]*Department of Electrical Engineering, Tufts University, Medford, MA 021555, USA*
[b]*Signal Processing Laboratory, Tampere University of Technology, P.O. Box 553, SF-33101 Tampere, Finland*

Received 29 April 1993; revised 13 January 1994

Abstract

Stack filters form a wide class of nonlinear filters which has received a great deal of attention during recent years. In this paper new decompositional methods based on Fibonacci p-codes for computing the output for different stack filters are presented. The computational complexities of these methods are also studied and numerical examples illustrating the benefits of using different values of p for different situations.

Zusammenfassung

Stack-Filter bilden eine große Klasse nichtlinearer Filter, die in den letzten Jahren viel Beachtung gefunden hat. In diesem Beitrag wird eine neue Zerlegungsmethode, welche auf Fibonacci p-Codes basiert, zur Berechnung des Ausgangssignals für verschiedene Stack-Filter vorgestellt. Die Komplexität der Methode wird untersucht, und numerische Beispiele zeigen die Vorteile, den Parameter p für verschiedene Situationen zu variieren.

Résumé

Les filtres en pile forment une large classe de filtres non lineaires, qui a reçu beaucoup d'attention ces derniéres années. Dans cet article, de nouvelles méthodes de décomposition, basées sur les codes d'ordre p de Fibonacci, pour le calcul de la sortie de différents filtres en pile, sont présentées. Les complexités de calcul sont aussi étudiées pour ces méthodes, et des exemples numériques illustrant l'avantage de l'utilisation de différentes valeurs de p pour des situations différentes.

Keywords: Stack filters; Fibonacci p-numbers and p-codes; Decompositional methods

1. Introduction

Stack filters are widely used nonlinear filters including Median, Order Statistic and Weighted Order Statistic filters [6]. In median filtering, to be able to use existing hardware as effectively as possible, it is often

*Corresponding author. Tel: + 358-31-3161860. Fax: + 358-31-3161857. E-mail: karen@cs.tut.fi.

SSDI 0165-1684(94)00093-X

the goal to build such an algorithm for finding the median of given data that

$$\varphi(\mu^+, \mu^c, \mu^m) \to \min, \tag{1}$$

where φ is some quality functional and μ^+ denotes the number of arithmetical operations, μ^c the number of comparisons and μ^m the number of needed memory cells. For example, we can use $\varphi = a\mu^+ + \mu^c$ or $\varphi = \mu^m$ as the quality functional, where a is the ratio of the execution time of one comparison operation to the execution time of one arithmetical operation. The same minimization problem is important for other stack filters as well.

Different methods for computing median and median type filters have been developed: Analytical methods, decomposition methods, radix methods, spectral methods, and combined methods [1, 3]. Stack filters exhibit the property of threshold decomposition which allows to express a multi-level signals as a set of binary signals. The importance of this property arises from the fact that binary vectors are easier to analyze than multi-valued vectors. Also filtering operation can be done by parallel realization, and single binary filters are easy to implement. In this paper new decomposition methods, i.e. methods where calculation is decomposed into similar but more easily calculated parts, are introduced for OS, WOS and stack-filtering. These methods are based on Fibonacci p-codes, and give a possibility to obtain a unified approach to decompositional techniques. The well-known binary-tree decomposition (case of $p = 0$) and complete threshold decomposition (case of $p \to \infty$) are the special cases of the proposed method.

2. Fibonacci p-codes

In the following we recall the definitions and some properties of Fibonacci p-numbers and p-codes [5].

Definition 1. Fibonacci p-numbers $\phi_p(i)$, corresponding to a given integer $p \geqslant 0$, are defined by the following recurrent formula:

$$\phi_p(i) = \begin{cases} 0, & i < 0, \\ 1, & i = 0, \\ \phi_p(i-1) + \phi_p(i-p-1), & i > 0. \end{cases} \tag{2}$$

For different values of p, Fibonacci p-numbers form different sequences. For $p = 1$ we have the standard Fibonacci numbers [4] (as shown in Table 1).

Property 1. *For Fibonacci p-numbers $\phi_p(i)$ the following relations hold:*

$$\sum_{i=0}^{n-1} \phi_p(i) = \phi_p(n+p) - 1, \tag{3}$$

$$\sum_{i=n-p}^{n} \phi_p(i) = \phi_p(n+p), \quad n \geqslant p \tag{4}$$

and

$$\phi_p(n) = \binom{p}{0} + \binom{n-p}{1} + \binom{n-2p}{2} + \cdots + \binom{m+r}{m}, \tag{5}$$

where m equals the quotient and r equals the remainder from the division of n by $p + 1$.

Table 1
Fibonacci p-numbers

p	i										
	0	1	2	3	4	5	6	7	8	9	10
0	1	2	4	8	16	32	64	128	256	512	1024
1	1	1	2	3	5	8	13	21	34	55	89
2	1	1	1	2	3	4	6	9	13	19	28
3	1	1	1	1	2	3	4	5	7	10	14
4	1	1	1	1	1	2	3	4	5	6	8
.											
.											
.											
∞	1	1	1	1	1	1	1	1	1	1	1

It is well known that every natural number D can be represented in the following form:

$$D = \sum_{i=0}^{n-1} a_i \phi_p(i), \tag{6}$$

where $a_i \in \{0, 1\}, i = 0, 1, \ldots, n-1$. The sequence $(a_0, \ldots, a_{n-1})_p$ is called the Fibonacci p-code of D, where a_i is the binary digit in code's ith position and $\phi_p(i)$ is the code's ith position weight.

In contrast to the standard binary code, Fibonacci p-codes for $p > 0$ are redundant or nonunique codes, i.e. a natural number D can have more than one representations (6) for a fixed p. For example,

$$20 = (1\,0\,1\,0\,1\,0\,0)_1 = (0\,1\,1\,1\,1\,1\,1)_1 = (1\,0\,0\,1\,1\,1\,1)_1. \tag{7}$$

Property 2. *For given integers $p \geqslant 0$ and $n \geqslant p$ there exists a unique representation of every natural number D in the form*

$$D = \phi_p(n) + r, \tag{8}$$

where $0 \leqslant r < \phi_p(n-p)$.

For $p = 0$ Eq. (8) is the well-known formula $L = 2^n + r$, $0 \leqslant r < 2^n$.

Among all Fibonacci p-representations we choose only the one which is obtained by expanding D and resulting remainders by (8). For that representation it holds that if $a_i = 1$ then $a_{i-1} = a_{i-2} = \cdots = a_{i-p} = 0$, that is, not less than p 0's follow a 1 in Fibonacci p-code. This unique representation is called the *normal representation* of L.

3. Decompositional methods for stack filtering using Fibonacci p-codes

In the following we propose a method for calculating the output of stack filters by using Fibonacci p-codes. We analyze the computational complexities of the presented algorithms and give bounds for the number of required operations.

3.1. OS-filters and Fibonacci p-codes

In the following we present a new approach to the problem of finding the Tth order statistic of the given set of nonnegative integers.

Let $X = \{x_1, x_2, \ldots, x_M\}$ be a given set of nonnegative integers, where $\max\{x_j: j = 1, 2, \ldots, M\} = L - 1$. We aim at finding the Tth order statistic of the set X, i.e. $y = \mathrm{OS}_{(T)}(x_1, x_2, \ldots, x_M)$. We will solve the problem by performing the following decomposition:

$$y = \mathrm{OS}_{(T)}(x_1, x_2, \ldots, x_M) = \sum_{i=1}^{v} c_i \mathrm{OS}_{(T)}\left(g_1^{(i)}(x_1, x_2, \ldots, x_M), \ldots, g_M^{(i)}(x_1, x_2, \ldots, x_M)\right), \tag{9}$$

where c_i are integer constants, and $g_j^{(i)}(x_1, x_2, \ldots, x_M)$, $j = 1, \ldots, M$, $i = 1, \ldots, v$, are integer valued functions chosen in a suitable way.

Thus, the fundamental problem of this section is to find functions $g_j^{(i)}(x_1, x_2, \ldots, x_M)$ and constants c_i for the given set $X = \{x_1, x_2, \ldots, x_M\}$ in such a way that (9) is true and that the complexity of the calculation of the Tth order statistic is reduced by this decomposition.

Theorem 1. *Decomposition (9) is obtained by choosing*

$$g_j^{(i)}(x_1, x_2, \ldots, x_M) = \sigma_{L_i}(x_j), \tag{10}$$

where the threshold operator $\sigma_{L_i}(x_j)$ is defined by

$$\sigma_{L_i}(x_j) = \begin{cases} 1 & \text{if } x_j \geqslant L_i, \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

$0 \leqslant L_i < L$, *L_i are integers*, $i = 1, \ldots, v$.

We call this decomposition *binary threshold decomposition.*

Below, we present an algorithm for finding the Tth order statistic of the given set X of numbers using binary threshold decomposition by Fibonacci p-codes. This algorithm is based on the following theorem, for which the proof is given in Appendix A.

Theorem 2. *The following relation holds:*

$$\sum_{i=1}^{L} f(\sigma_i(x_1), \ldots, \sigma_i(x_M)) = \sum_{i=1}^{r} \phi_p(d_i)\, h(L_i). \tag{12}$$

where $f(\cdot)$ is a positive Boolean function of M variables, $h(t) = f(\sigma_t(x_1), \ldots, \sigma_t(x_M))$, d_1 satisfies the condition $\phi_p(d_1) < L \leqslant \phi_p(d_1 + 1)$, $d_i = d_{i-1} - ph(L_{i-1}) - 1$ $(i = 2, \ldots, r)$, $L_1 = \phi_p(d_1)$, $L_i = \sum_{k=1}^{i-1} \phi_p(d_k) \cdot h(L_k) + \phi_p(d_i)$ $(i = 2, \ldots, r)$, and r is such that $d_{r-1} \geqslant p$ and $d_r < p$.

By taking $f(\cdot) = \mathrm{OS}_T(\cdot)$ in Theorem 2. we obtain the following algorithm.

Algorithm 1

Finding the Tth order statistic of the given set X of numbers using threshold decomposition by Fibonacci p-codes

Step (1) Initialize counters $k = 1$, $z = 0$.

Step (2) Set $N = \phi_p(t_p - k)$, where t_p is such a Fibonacci p-number $\phi_p(t_p)$, that

$$\phi_p(t_p) \geqslant L \quad \text{and} \quad \phi_p(t_p - 1) < L.$$

Step (3) Set $i = z + N$.

Step (4) Compute the threshold $\sigma_i(X) = (\sigma_i(x_1), \ldots, \sigma_i(x_M))$.
Step (5) Compute $g_i = \sum_{j=1}^{M} \sigma_i(x_j)$.
Step (6) If $g_i \geqslant M + 1 - T$, then $z = i$, $k = k + p$.
Step (7) Set $k = k + 1$.
Step (8) If $p \leqslant t_p - k$ then go to the Step (2), otherwise STOP.
At STOP the Tth order statistic is z.

By using Property 2 we can prove the following remark.

Remark 1. The complexity of Algorithm 1 is
– μ^+ addition operations, where

$$M(t_p - p) - (Mp - 1)s_p \leqslant \mu^+ \leqslant Mt_p - (Mp - 1)s_p, \tag{13}$$

– $\mu^c = (M + 1)v_p$ comparison operations, where

$$t_p - p(s_p + 1) \leqslant v_p \leqslant t_p - ps_p \tag{14}$$

and s_p is the number of ones in Fibonacci p-representation of z.

Note, that when $p = 0$, we have $v_p = t_0$ and v_p does not depend on s_p, and when $p > 0$, v_p considerably depends on s_p.

Example 1. Let us consider finding the 4th order statistic of the set

$$\{7, 26, 30, 29, 32, 25, 33, 0, 4\}. \tag{15}$$

Consider the cases $p = 0$ and $p = 3$.
(a) Let $p = 0$. Then $t_p = 6$, because $\phi_0(6) = 64 > 33$ and $\phi_0(5) = 32 < 33$.
(Cycle 1): $k = 1$, $z = 0$, $M + 1 - T = 9 + 1 - 4 = 6$, $N = \phi_0(5) = 32$, $i = 32$,
$\sigma_{32}(X) = (0,0,0,0,1,0,1,0,0)$, $g_{32} = 2 < 6$.
(Cycle 2): $k = 2$, $N = \phi_0(4) = 16$, $i = 16$, $\sigma_{16}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{16} = 6 \geqslant 6$, $z = 16$.
(Cycle 3): $k = 3$, $N = \phi_0(3) = 8$, $i = z + 8 = 24$, $\sigma_{24}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{24} = 6 \geqslant 6$, $z = 24$.
(Cycle 4): $k = 4$, $N = \phi_0(2) = 4$, $i = z + 4 = 28$, $\sigma_{28}(X) = (0,0,1,1,1,0,1,0,0)$, $g_{28} = 4 < 6$.
(Cycle 5): $k = 5$, $N = \phi_0(1) = 2$, $i = z + 2 = 26$, $\sigma_{26}(X) = (0,1,1,1,1,0,1,0,0)$, $g_{26} = 5 < 6$.
(Cycle 6): $k = 6$, $N = \phi_0(0) = 1$, $i = z + 1 = 25$, $\sigma_{25}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{25} = 6 \geqslant 6$, $z = 25$.
Hence, $z = 25$.
(b) Let $p = 3$. Then $t_p = 13$, because $\phi_3(13) = 36 > 33$ and $\phi_3(12) = 26 < 33$.
(Cycle 1): $k = 1$, $z = 0$, $M + 1 - T = 9 + 1 - 4 = 6$, $N = \phi_3(12) = 26$, $i = 26$,
$\sigma_{26}(X) = (0,1,1,1,1,0,1,0,0)$, $g_{26} = 5 < 6$.
(Cycle 2): $k = 2$, $N = \phi_3(11) = 19$, $i = 19$, $\sigma_{19}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{19} = 6 \geqslant 6$, $z = 19$, $k = 5$.
(Cycle 3): $k = 6$, $N = \phi_3(7) = 5$, $i = z + 5 = 24$, $\sigma_{24}(X) = (0,1,1,1,1,1,1,0,0)$,
$g_{24} = 6 \geqslant 6$, $z = 24$, $k = 9$.
(Cycle 4): $k = 10$, $N = \phi_3(3) = 1$, $i = z + 1 = 25$, $\sigma_{25}(X) = (0,1,1,1,1,1,1,0,0)$,
$g_{25} = 6 \geqslant 6$, $z = 25$, $k = 13$.
Hence, $z = 25$.

From Example 1 we see that finding the 4th order statistic of the example set (15), by using Fibonacci p-codes, requires six cycles, if $p = 0$, and four cycles, if $p = 3$. Thus the complexity of the Algorithm 1 depends on the choice of p; see Remark 1.

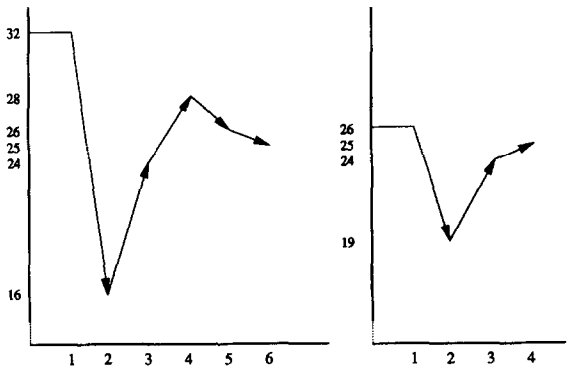

Fig. 1. The convergence of Algorithm 1 in Example 1, $p = 0$ (left) and $p = 3$ (right).

Table 2
The computational complexities of finding the 4th order statistic of the example set (15), by using Algorithm 1 for $p = 0, 1, \ldots, 10$

p	Additions μ^+	Comparisons μ^c	Memory cells μ^m
0	57	60	**15**
1	48	50	16
2	47	50	18
3	39	40	19
4	**29**	**30**	21
5	38	40	22
6	47	50	23
7	56	60	24
8	65	70	25
9	64	70	26
10	38	40	27

Fig. 1. illustrates the convergence of Algorithm 1 in Example 1. The computational complexities of Example 1 are presented in Table 2 for $p = 0, 1, \ldots, 10$.

3.2. *WOS-filters and Fibonacci p-codes*

As above, let $X = \{x_1, x_2, \ldots, x_M\}$ be a given set of nonnegative integers, where $\max\{x_j: j = 1, 2, \ldots, M\} = L - 1$ and let $W = \{w_1, w_2, \ldots, w_M\}$ be a set on nonnegative integer weights, where $\sum_{i=1}^{M} w_i$ is odd. We aim at finding the Tth weighted order statistic of the set X, i.e. $y = \mathrm{WOS}_{(T)}(x_1, x_2, \ldots, x_M)$. Again, we will solve the problem by finding the same Tth weighted order statistic by the decomposition

$$y = \mathrm{WOS}_{(T)}(x_1, x_2, \ldots, x_M) = \sum_{i=1}^{v} c_i \mathrm{WOS}_{(T)}(g_1^{(i)}(x_1, x_2, \ldots, x_M), \ldots, g_M^{(i)}(x_1, x_2, \ldots, x_M)), \tag{16}$$

where c_i are integer constants $g_j^{(i)}(x_1, x_2, \ldots, x_M)$, $j = 1, \ldots, M$, $i = 1, \ldots, v$, are integer valued functions chosen in a suitable way.

Using Theorem 2 with $f(\cdot) = \mathrm{WOS}_{(T)}(\cdot)$ we obtain the following algorithm.

Algorithm 2
Finding the Tth weighted order statistic of the given set X of numbers using threshold decomposition by Fibonacci p-codes
Step (1)–Step (4) As in Algorithm 1.
Step (5) Compute $g_i = \sum_{j=1}^{M} \sigma_i(x_j) w_j$.
Step (6) If $g_i \geqslant \sum_{j=1}^{M} w_j + 1 - T$, then $z = i$, $k = k + p$.
Step (7)–Step (8) As in Algorithm 1.
At STOP the Tth weighted order statistic is z.

Similarly, by using Property 2 we can prove the following remark.

Remark 2. The complexity of Algorithm 2 is
- μ^+ addition operations, where

$$M(t_p - p + 1) - (Mp - 1)s_p \leqslant \mu^+ \leqslant M(t_p + 1) - (Mp - 1)s_p - 1. \tag{17}$$

- $\mu^\times = Mv_p$ multiplication operations,
- $\mu^c = (M + 1)v_p$ comparison operations, where

$$t_p - p(s_p + 1) \leqslant v_p \leqslant t_p - ps_p \tag{18}$$

and s_p is the number of ones in Fibonacci p-representation of z.

Example 2. Let us consider the finding the fourth weighted order statistic of the set

$$\{7, 26, 30, 29, 32, 25, 33, 0, 4\}, \tag{19}$$

when weights are

$$\{1, 1, 2, 2, 3, 2, 2, 1, 1\}. \tag{20}$$

Consider the cases $p = 2$ and $p = 3$.

(a) Let $p = 2$. Then $t_p = 11$, because $\phi_2(11) = 41 > 33$ and $\phi_2(10) = 28 < 33$.

(Cycle 1): $k = 1$, $z = 0$, $\sum_{i=1}^{m} w_i + 1 - T = 15 + 1 - 4 = 12$, $N = \phi_2(10) = 28$, $i = 28$, $\sigma_{28}(X) = (0,0,1,1,1,0,1,0,0)$, $g_{28} = 9 < 12$.

(Cycle 2): $k = 2$, $N = \phi_2(9) = 19$, $i = 19$, $\sigma_{19}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{19} = 12 \geqslant 12$, $z = 19$, $k = 4$.

(Cycle 3): $k = 5$, $N = \phi_2(6) = 6$, $i = z + 6 = 25$, $\sigma_{25}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{25} = 12 \geqslant 12$, $z = 25$, $k = 7$.

(Cycle 4): $k = 8$, $N = \phi_2(3) = 2$, $i = z + 2 = 27$, $\sigma_{27}(X) = (0,0,1,1,1,0,1,0,0)$, $g_{27} = 9 < 12$.

(Cycle 5): $k = 9$, $N = \phi_2(2) = 1$, $i = z + 1 = 26$, $\sigma_{26}(X) = (0,1,1,1,1,0,1,0,0)$, $g_{26} = 11 < 12$.

Hence $z = 25$.

(b) Let $p = 3$. Then $t_p = 13$, because $\phi_3(13) = 36 > 33$ and $\phi_3(12) = 26 < 33$.

(Cycle 1): $k = 1$, $z = 0$, $\sum_{i=1}^{m} w_i + 1 - T = 15 + 1 - 4 = 12$, $N = \phi_3(12) = 26$, $i = 26$, $\sigma_{26}(X) = (0,1,1,1,1,0,1,0,0)$, $g_{26} = 10 < 12$.

(Cycle 2): $k = 2$, $N = \phi_3(11) = 19$, $i = 19$, $\sigma_{19}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{19} = 12 \geqslant 12$, $z = 19$, $k = 5$.

(Cycle 3): $k = 6$, $N = \phi_3(7) = 5$, $i = z + 5 = 24$, $\sigma_{24}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{24} = 12 \geqslant 12$, $z = 24$, $k = 9$.

(Cycle 4): $k = 10$, $N = \phi_3(3) = 1$, $i = z + 1 = 25$, $\sigma_{25}(X) = (0,1,1,1,1,1,1,0,0)$, $g_{25} = 12 \geqslant 12$, $z = 25$.

Hence, $z = 25$.

Again, from the above example we can see that the complexity of Algorithm 2 depends on p. This can also be seen from Remark 2 (see Table 3).

3.3. Stack filters and Fibonacci p-codes

Let $X = \{x_1, x_2, \ldots, x_M\}$ be the given signal segment in the window of length M, x_i are nonnegative integers, and $\max\{x_j : j = 1, 2, \ldots, M\} = L - 1$. Let $f(\cdot)$ be a positive Boolean function and $S(\cdot)$ be the stack filter corresponding to $f(\cdot)$.

By the definition of stack filters, the output of $S(\cdot)$ is

$$y = S(X) = \sum_{i=1}^{L-1} f(\sigma_i(X)) = \sum_{i=1}^{L-1} f(\sigma_i(x_1), \ldots, \sigma_i(x_M)). \tag{21}$$

Table 3
The computational complexities of finding the 4th weighted order statistic of the example set (19) and weights (20), by using Algorithm 2 for $p = 0, 1, \ldots, 10$

p	Additions μ^+	Multiplic. $\mu^\times$	Comparisons μ^c	Memory cells μ^m
0	65	54	60	**15**
1	56	45	50	16
2	55	45	50	18
3	47	36	40	19
4	**37**	**27**	**30**	21
5	46	36	40	22
6	55	45	50	23
7	64	54	60	24
8	73	63	70	25
9	72	63	70	26
10	46	36	45	27

Table 4
The computational complexities of finding the output of the stack filter in Example 3, when input signal is (24), by using Algorithm 3 for $p = 0, 1, \ldots, 10$

p	Additions μ^+	Comparisons μ^c	Calc. of PBF v_p	Memory cells μ^m
0	8	30	5	5
1	7	24	4	5
2	8	42	7	5
3	8	36	6	5
4	8	36	6	5
5	9	42	7	5
6	8	42	7	5
7	**5**	**18**	**3**	5
8	**5**	**18**	**3**	5
9	6	24	4	5
10	7	30	5	5

We aim at finding the output of $S(\cdot)$ for the input X, i.e. $S(x_1, x_2, \ldots, x_M)$ by the decomposition

$$y = S(x_1, x_2, \ldots, x_M) = \sum_{i=1}^{v} c_i f(g_1^{(i)}(x_1, x_2, \ldots, x_M), \ldots, g_M^{(i)}(x_1, x_2, \ldots, x_M)), \tag{22}$$

where c_i are integer constants, and $g_j^{(i)}(x_1, x_2, \ldots, x_M), j = 1, \ldots, M$, $i = 1, \ldots, v$, are integer valued functions chosen in a suitable way.

Note, that the decompositions (9) and (16) are special cases of the decomposition (22).

As a result of Theorem 2 we obtain Algorithm 3.

Algorithm 3
Finding the output of a given signal X for a Stack filter $S(\cdot)$ defined by Boolean function $f(\cdot)$.
Step (1)–Step (4) As in Algorithm 1.
Step (5) Compute $g_i = f(\sigma_i(X))$.
Step (6) If $g_i = 1$, then $z = i$, $k = k + p$.
Step (7)–Step (8) As in Algorithm 1.
At STOP the output of $S(\cdot)$ is z.

Again, by using Property 2 we can prove the following remark.

Remark 3. The complexity of Algorithm 3 is
- $\mu^+ = v_p + s_p$ addition operations,
- $\mu^c = (M + 1)v_p$ comparison operations,
- v_p times the calculation of the positive Boolean function $f(\cdot)$, where the number of iterations v_p satisfies

$$t_p - p(s_p + 1) \leqslant v_p \leqslant t_p - ps_p \tag{23}$$

and s_p is the number of ones in Fibonacci p-representation of z.

Example 3. Let us consider finding the output of the stack filter defined by the positive Boolean function $f(X) = x_1x_2 \vee x_3x_4 \vee x_5$, when the input signal is

$$X = \{7, 26, 2, 29, 28\}. \tag{24}$$

Consider the cases $p = 1$ and $p = 3$.

(a) Let $p = 1$. Then $t_p = 8$, because $\phi_1(8) = 34 > 29$ and $\phi_2(7) = 21 < 29$.
(Cycle 1): $k = 1,\ z = 0,\ N = \phi_1(7) = 21,\ i = 21,\ \sigma_{21}(X) = (0,1,0,1,1),\ g_{21} = 1,\ z = 21,\ k = 2.$
(Cycle 2): $k = 3,\ N = \phi_1(5) = 8,\ i = z + 8 = 29,\ \sigma_{29}(X) = (0,0,0,1,0),\ g_{29} = 0.$
(Cycle 3): $k = 4,\ N = \phi_1(4) = 5,\ i = z + 5 = 26,\ \sigma_{26}(X) = (0,1,0,1,1),\ g_{26} = 1,\ z = 26,\ k = 5.$
(Cycle 4): $k = 6,\ N = \phi_1(2) = 2,\ i = z + 2 = 28,\ \sigma_{28}(X) = (0,0,0,1,1),\ g_{28} = 1,\ z = 28,\ k = 7.$
Hence, $z = 28$.

(b) Let $p = 3$. Then $t_p = 13$, because $\phi_3(13) = 36 > 29$ and $\phi_3(12) = 26 < 29$.
(Cycle 1): $k = 1,\ z = 0,\ N = \phi_3(12) = 26,\ i = 26,\ \sigma_{26}(X) = (0,1,0,1,1),\ g_{26} = 1,\ z = 26,\ k = 4.$
(Cycle 2): $k = 5,\ N = \phi_3(8) = 7,\ i = z + 7 = 33,\ \sigma_{33}(X) = (0,0,0,0,0),\ g_{33} = 0.$
(Cycle 3): $k = 6,\ N = \phi_3(7) = 5,\ i = z + 5 = 31,\ \sigma_{31}(X) = (0,0,0,0,0),\ g_{31} = 0.$
(Cycle 4): $k = 7,\ N = \phi_3(6) = 4,\ i = z + 4 = 30,\ \sigma_{30}(X) = (0,0,0,0,0),\ g_{30} = 0.$
(Cycle 5): $k = 8,\ N = \phi_3(5) = 3,\ i = z + 3 = 29,\ \sigma_{29}(X) = (0,0,0,1,0),\ g_{29} = 0.$
(Cycle 6): $k = 9,\ N = \phi_3(4) = 2,\ i = z + 2 = 28,\ \sigma_{28}(X) = (0,0,0,1,1),\ g_{28} = 1,\ z = 28,\ k = 12.$
Hence, $z = 28$.

From the above example we can see that the complexity of Algorithm 3 depends on p. This can also be seen from Remark 3 (see Table 4).

Remark 4. If in Algorithm 3, $p \geqslant L$, then we are using complete binary threshold decomposition. Now, $s_p = 1$ and the complexity of Algorithm 3 is
- $\mu^+ = L - z + 1$ addition operations,
- $\mu^c = (M + 1)(L - z)$ comparison operations,
- $L - z$ times the calculation of the positive Boolean function $f(\cdot)$.

Thus, if the output of S is near L, the complexity of Algorithm 3 is very low, when $p \geqslant L$.

Remark 5. If in Algorithm 3, $p = 0$, then we are using the well-known binary-tree threshold decomposition [2].

Appendix A. Proof of Theorem 2

Let us considered the following sum:

$$S = \sum_{i=1}^{L} f(\sigma_i(x_1), \ldots, \sigma_i(x_M)) = \sum_{i=1}^{L} h(i). \tag{A.1}$$

Represent S in the form

$$S = \sum_{i=1}^{\phi_p(d_1)} h(i) + \sum_{i=\phi_p(d_1)+1}^{\phi_p(d_1+1)} h(i). \tag{A.2}$$

Note, that because $f(\cdot)$ is a positive Boolean function, $h(j) = 0$ implies $h(k) = 0$ for all $k \geqslant j$, and $h(j) = 1$ implies $h(k) = 1$ for all $0 \leqslant k \leqslant j$. Therefore, if $h(L_1) = 0$, where $L_1 = \phi_p(d_1)$, then the second sum on the right-hand side of (A.2) equals 0, and if $h(L_1) = 1$, then the first sum on the right-hand side of (A.2) equals

$L_1 = \phi_p(d_1)$. In the last case

$$S = \phi_p(d_1) + \sum_{i=1}^{\phi_p(d_1+1)-\phi_p(d_1)} h(\phi_p(d_1) + i) = \phi_p(d_1) + \sum_{i=1}^{\phi_p(d_1-p)} h(\phi_p(d_1) + i). \tag{A.3}$$

Now, we can rewrite (A.2) as

$$S = h(L_1)\phi_p(d_1) + \sum_{i=1}^{\phi_p(d_2+1)} h(i + h(L_1)\phi_p(d_1)). \tag{A.4}$$

Continuing in this manner we obtain

$$S = h(L_1)\phi_p(d_1) + h(L_2)\phi_p(d_2) + \sum_{i=1}^{\phi_p(d_3+1)} h(i + h(L_1)\phi_p(d_1) + h(L_2)\phi_p(d_2)), \tag{A.5}$$

and finally

$$S = \sum_{i=1}^{r} h(L_i)\phi_p(d_i) + \sum_{i=1}^{\phi_p(d_r - ph(L_r))} h\left(i + \sum_{j=1}^{r} h(L_j)\phi_p(d_j)\right). \tag{A.6}$$

The second sum on the right-hand side of (A.6) is always equal to 0. Really, if $h(L_r) = 1$, then $\phi_p(d_r - ph(L_r)) = 0$, since $d_r < p$. On the other hand, if $h(L_r) = 0$, then the sum under consideration consists only on one summand $h(1 + \sum_{j=1}^{r} h(L_j)\phi_p(d_j))$, which is equal to 0, since $\phi_p(d_r) = 1$, and, hence,

$$h\left(1 + \sum_{j=1}^{r} h(L_j)\phi_p(d_j)\right) = h\left(\phi_p(d_r) + \sum_{j=1}^{r} h(L_j)\phi_p(d_j)\right) = h(L_r) = 0. \tag{A.7}$$

Thus, (A.6) proves the theorem's statement. □

References

[1] S. Agaian, J. Astola and K. Egiazarian, "Mixed methods for nonlinear filtering", *Proc. SPIE, Vol. 1902, Nonlinear Image Processing IV*, San Jose, CA, 1–3 February 1993, pp. 140–146.

[2] M. Juhola, J. Katajainen and T. Raita, "Comparison of algorithms for standard median filtering", *IEEE Trans. Signal Process.*, Vol. 39, January 1991, pp. 204–208.

[3] D. S. Richards, "VLSI median filters", *IEEE Trans. Acoust. Speech Signal Process.*, Vol. 38, January 1990, pp. 145–153.

[4] J. Riordan, *An Introduction to Combinatorical Analysis*, Wiley, New York, 1958.

[5] A. P. Stakhov, "Algorithmic measurement theory", Moscow, *Znanie*, No. 6, 1979, 64 p. (in Russian).

[6] P. Wendt, E. Coyle and N. Gallagher, "Stack filters", *IEEE Trans. Acoust. Speech Signal Process.*, Vol. 34, August 1986, pp. 898–911.