УДК 519.7

# PIPELINING COMBINATIONAL CIRCUITS

Yu. V. Pottosin, S. N. Kardash

*United Institute of Informatics Problems of NAS of Belarus, Minsk, Belarus*

**E-mail:** pott@newman.bas-net.by

For a multilevel combinatorial circuit, the problem of increasing its performance is considered. The problem is to divide the circuit into a given number of cascades and to connect them via registers providing pipeline-wise development of incoming signals. The frequency of incoming signals is established in the process of dividing the circuit. This frequency must be as high as possible. To solve this problem a model based on the representation of the circuit in the form of a directed graph is used.

**Keywords:** *combinational circuit, pipelining, directed graph.*

## Introduction

Increasing throughput of information processing systems attracts the great attention of specialists in appropriate fields. One of the ways to increase the throughput is application of the pipeline structure [1]. Such a system is formed by several independent processors connected between each other so that the output from one processor is an input for another one. The processors form an information pipeline. The output processor produces results after short time intervals, but the real time for passing the information flow through the pipeline can be rather long.

The principle of pipelining is used effectively when the nature of information processing is a sequence of operations each of which consists of a sequence of stages [2, 3]. To start execution of the next operation one should not wait for finish of the whole previous operation. It is enough to finish only the first stage of the previous operation. In the pipeline with $r$ successive stages, if the $i$-th operation passes the $s$-th stage, then $(i + k)$-th operation can pass the $(s - k)$-th stage where $1 \leqslant s$, $s - k \leqslant r$.

In constructing the systems of digital signal processing in real-time mode, the systolic principle of computing had been widely extended [4, 5]. One unified VLSI element is developed, and an array of elements of that type is built. The elements act concurrently fulfilling the base operation. After fulfilling that operation, output data are passed synchronously from one element to its neighbors through all the local connections.

In this paper, we try to find out the way to increase the throughput of a multilevel combinational circuit by pipelining.

## 1. Statement of the problem

In a multilevel combinational circuit, the delay is summed up of delays of elements in the longest chain. Let a sequence of $p$ sets of binary signals is put at the input of a combinational circuit. If $T$ is the delay time of the circuit, then the time period of changing input signals cannot be less then $T$. So, the response of the circuit to this sequence will appear at least in $pT$ times. Let us partition the circuit into $k$ cascades $C_1$, $C_2$, ..., $C_k$. If $\tau_C$ is the delay time of the most slow cascade, then $T \leqslant k\tau_C$. Let us put memory elements (triggers of $D$ type) at the outputs of every cascade that transmit signals from the cascades by a clock signal. The same clock signal determines the time period of the signal change at the input of the circuit. This time period, $\tau_{\text{clock}}$, cannot be less than the sum of two delay

times: $\tau_C$ and delay time $\tau_D$ of the memory element ($\tau_{\text{clock}} \geqslant \tau_C + \tau_D$). Now, the time of the response to the mentioned sequence of length $p$ is $(k+p)\tau_{\text{clock}}$.

Figure 1 shows an example of partitioning a circuit into three cascades. The borders between cascades are indicated by hatches. The modules used in the circuit at Fig. 1 are in the library of CMOS cells for custom VLSI design [6].

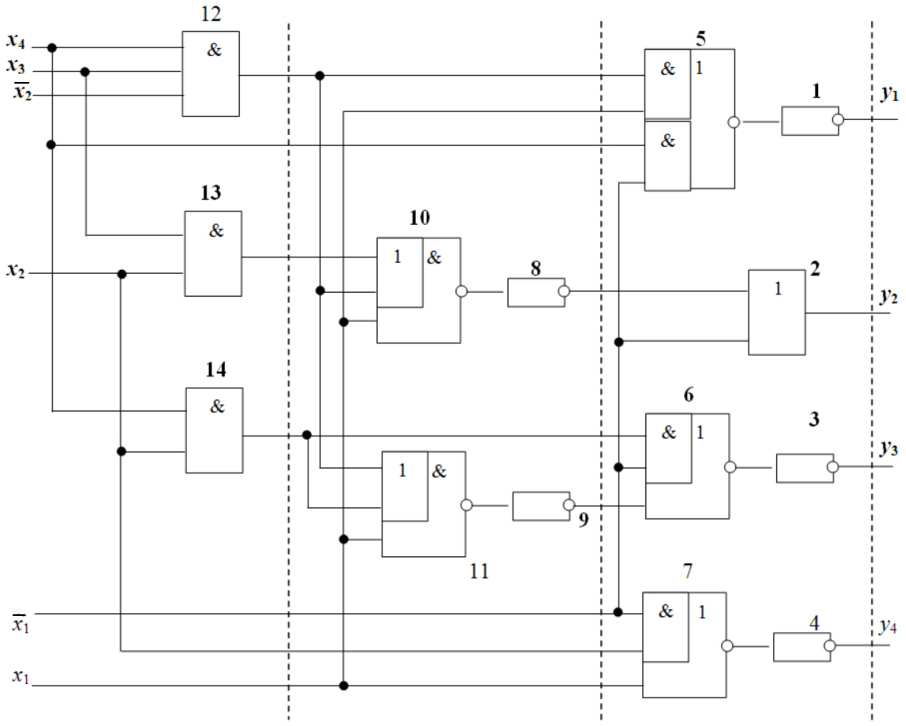


Fig. 1. Circuit with indicated cascades.

The lower bound of the length $p$ of the sequence of binary signals sets at the input of the circuit, at which the signal processing accelerates, is defined by inequality

$$(k+p)\tau_{\text{clock}} < pT, \tag{1}$$

where $T$ is the delay time of the initial circuit. Taking into consideration that $\tau_{\text{clock}} \geqslant \tau_C + \tau_D$, we have

$$p > \frac{k\tau_{\text{clock}}}{T - \tau_{\text{clock}}} \geqslant \frac{k(\tau_C + \tau_D)}{T - \tau_C - \tau_D}. \tag{2}$$

So, the given circuit should be partitioned into a demanded number $k$ of cascades in order to provide maximum throughput at the described pipeline mode.

## 2. Model of a combinational circuit

To represent a combinational circuit, it is convenient to use, as a model, a directed graph (digraph) $G = (V, A)$ with the set $V$ of vertices and the set $A$ of arcs. The vertices of the digraph correspond to the logical elements and input pins of the circuit, and the arcs indicate directions of signals from outputs of certain elements to inputs of other ones. Figure 2 shows the model of the circuit at Fig. 1. The indices at the vertices at Fig. 2 coincide with the numbers of correspondent elements of the circuit at Fig. 1.

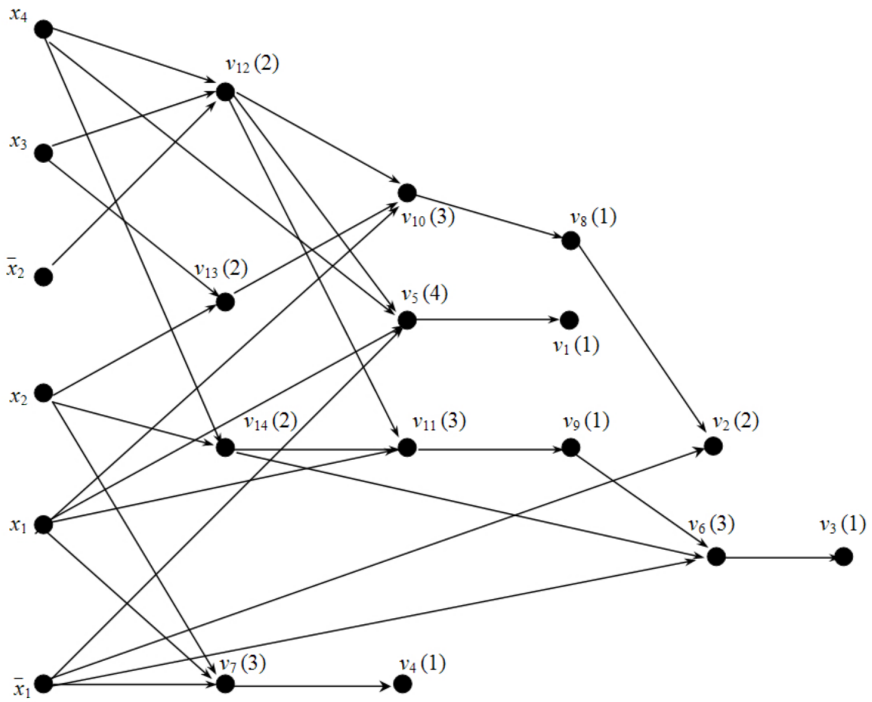


Fig. 2. Digraph $G$.

The digraph $G$ has no circles. Its every vertex $v \in V$ is assigned with the weight $\tau(v)$ that is the delay of the corresponding element. The weights may be integers proportional to element delay times. The vertices corresponding to the circuit input pins have the weight equalled zero. The weights of vertices of $G$ at Fig. 2 are indicated by the numbers in parentheses. Here, we consider that the more complicated the Boolean function implemented by an element, the longer delay that the element has.

## 3. Partitioning a circuit into layers

Let us construct the sequence of layers, $L_1, L_2, \ldots, L_m$, being an ordered partition of the set $V$ in the digraph $G$ with the property that if a vertex $v$ is in the semi-neighborhood of outcome $N^+(u)$ of a vertex $u$, then $v$ and $u$ are in different layers, and the layer containing $u$ precedes (not necessary directly) the layer containing $v$. If the path lengths from the circuit input to its outputs are different, then this partition is not unique. The variant of partitioning the circuit into layers should be selected so that the sum of weights of all layers would be as small as possible. The weight of a layer means the maximum vertex weight in the layer.

Two types of vertices of the digraph $G$ can be distinguished. The vertices of one type lie on the longest paths of $G$. They are distributed strictly among the layers and cannot change their positions. We call such a vertex *immovable*. The positions of the vertices of the other type called *movable* can change within the certain bounds; say from a layer $L_l$ to a layer $L_r$ $(l < r)$. These bounds are rather easy to establish. It is enough to execute Algorithm 1 below for the given digraph $G$ and for the digraph $G^c$ obtained from $G$ by reversing the directions of all the arcs and to change the order of the layers for inverse one in the sequence obtained for $G^c$. The following designations are accepted in Algorithm 1: $N^-(v)$ and $N^+(v)$ are, correspondingly, semi-neighborhood of income and semi-neighborhood of outcome of a vertex $v$; $L_i$ is the $i$-th layer, $m$ is the number of layers.

**Algorithm 1**

1) $L_1 := \{v : N^-(v) = \varnothing\}$, $i := 1$;

2) $i := i + 1$, $L_i := \bigcup\limits_{v \in L_{i-1}} N^+(v)$. If $L_i \neq \varnothing$, go to 2, otherwise $j := i$, $m := i := i - 1$;

3) $i := j := j - 1$. If $j = 1$, go to 5, otherwise

4) $i := i - 1$. If $i = 1$, go to 3, otherwise $L_i := L_i \setminus L_j$, go to 4;

5) End.

Each immovable vertex will be in the same layer of the sequences for $G$ and $G^c$. For a movable vertex, the "left" layer $L_l$ is that obtained by executing Algorithm 1 over digraph $G$ and the "right" layer $L_r$ is that obtained by executing Algorithm 1 over digraph $G^c$.

The distributions of vertices among layers being the results of applying Algorithm 1 to digraph $G$ and to digraph $G^c$ are shown in Fig. 2 and in Fig. 3, correspondingly. The layers are represented in these figures by vertical rows of vertices. The layers $L_1 = \{x_1, \overline{x_1}, x_2, \overline{x_2}, x_3, x_4\}$, $L_2 = \{v_7, v_{12}, v_{13}, v_{14}\}$, $L_3 = \{v_4, v_5, v_{10}, v_{11}\}$, $L_4 = \{v_1, v_8, v_9\}$, $L_5 = \{v_2, v_6\}$, $L_6 = \{v_3\}$ are obtained for digraph $G$, and $L_1 = \{x_2, \overline{x_2}, x_3, x_4\}$, $L_2 = \{x_1, v_{12}, v_{14}\}$, $L_3 = \{v_{11}, v_{13}\}$, $L_4 = \{\overline{x_1}, v_9, v_{10}\}$, $L_5 = \{v_5, v_6, v_7, v_8\}$, $L_6 = \{v_1, v_2, v_3, v_4\}$ for digraph $G^c$. The vertices $\overline{x_2}$, $x_2$, $x_3$, $x_4$, $v_3$, $v_6$, $v_9$, $v_{11}$, $v_{12}$ and $v_{14}$ are immovable. They form sublayers $L'_1 = \{x_2, \overline{x_2}, x_3, x_4\}$, $L'_2 = \{v_{12}, v_{14}\}$, $L'_3 = \{v_{11}\}$, $L'_4 = \{v_9\}$, $L'_5 = \{v_6\}$ and $L'_6 = \{v_3\}$ with weights 0, 2, 3, 1, 3 and 1, correspondingly.

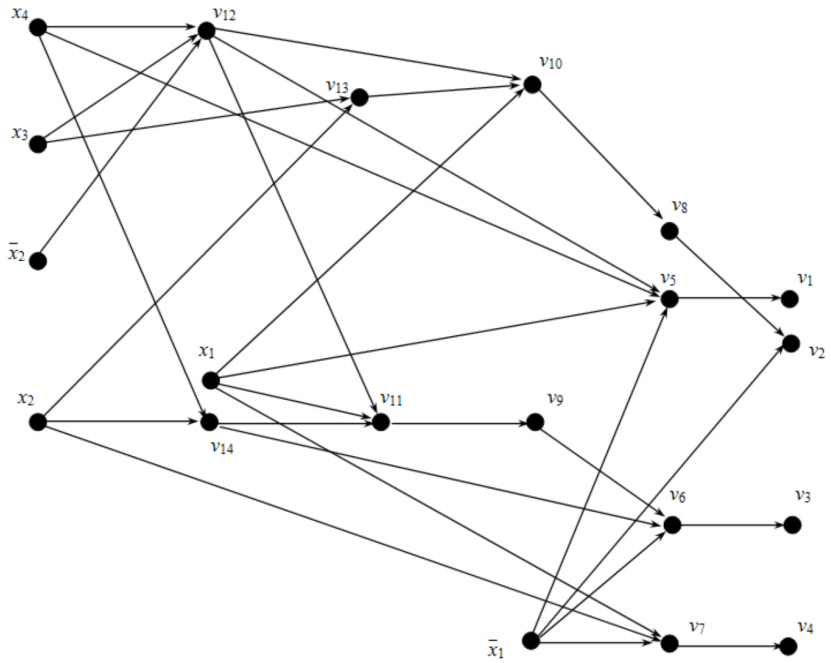


Fig. 3. Distribution of vertices among layers obtained by applying Algorithm 1 to $G^c$.

The rest of vertices ($x_1$, $\overline{x_1}$, $v_1$, $v_2$, $v_4$, $v_5$, $v_7$, $v_8$, $v_{10}$, $v_{13}$) are movable, and as it is shown above for each of them there exists the nonempty set of layers where the vertex can be. Vertex $x_1$ can be in layers $L_1$ and $L_2$; $\overline{x_1}$ in $L_1$, $L_2$, $L_3$ and $L_4$; $v_1$ in $L_4$, $L_5$ and $L_6$; $v_2$ in $L_5$ and $L_6$; $v_4$ in $L_3$, $L_4$, $L_5$ and $L_6$; $v_5$ in $L_3$, $L_4$ and $L_5$; $v_7$ in $L_2$, $L_3$, $L_4$ and $L_5$; $v_8$ in $L_4$ and $L_5$; $v_{10}$ in $L_3$ and $L_4$; $v_{13}$ in $L_2$ and $L_3$. But the positions of some vertices may depend on the positions of other ones. In particular, the vertices connected with an arc cannot be in the same layer.

We suggest the following technique to distribute the vertices among layers so that the sum of the layer weights becomes possibly minimal. Having removed the immovable vertices and their incident arcs from $G$, we obtain digraph $H$ where we choose a vertex of maximum

weight in each component of $H$. We put this vertex $v$ in one of the layers of maximum weight that is acceptable for $v$. After this, the bounds of the vertex positions will change, and some movable vertices will become immovable. The further distribution of the vertices among the layers can be made for each component of $H$ in the same way.

Figure 4 shows the digraph $H$ with one component from the example on consideration. The vertices $x_1$ and $\overline{x_1}$ with zero weights can be put in any layer. According to the above technique, we put the vertex $v_5$ of maximum weight in the layer $L_3$. Finally, we obtain $L_1 = \{x_1, \overline{x_1}, x_2, \overline{x_2}, x_3, x_4\}$, $L_2 = \{v_{12}, v_{13}, v_{14}\}$, $L_3 = \{v_5, v_7, v_{10}, v_{11}\}$, $L_4 = \{v_1, v_4, v_8, v_9\}$, $L_5 = \{v_2, v_6\}$ and $L_6 = \{v_3\}$.

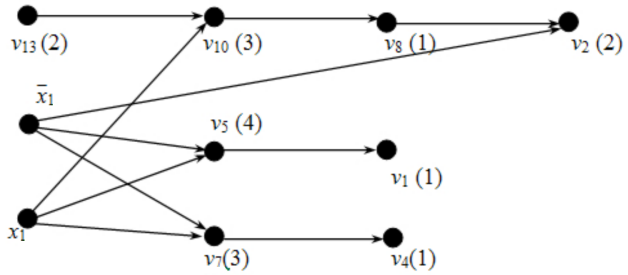


Fig. 4. Digraph $H$.

## 4. Equalization of paths in a digraph

Let us make all the lengths of paths in a digraph be equal by adding new vertices of zero weight. As a result, the initial digraph $G = (V, A)$ is transformed into digraph $G' = (V', A')$. New vertices are added to layers in $G$ so that $N^+(v) \subseteq L_{i+1}$ for any $v \in L_i$, $i = 1, 2, \ldots, m$–1, the vertex accessibility being kept. Here the vertex accessibility means that in any path from any vertex $v \in V$ to $u \in V$ in digraph $G'$, the sequence of vertices from $V$ is the same as the sequence of vertices in the corresponding path in $G$. The new vertices are assigned with zero weight. To do it, Algorithm 2 is used. The digraph $G'$ obtained from $G$ in such a way is shown in Fig. 5 where the new vertices are indicated by light circles.

**Algorithm 2**

1) $i := 0$, $j := 0$; $U := \varnothing$;
2) $i := i + 1$. If $i = m$, go to 4, otherwise $L := L_i$;
3) If $L = \varnothing$, go to 2, otherwise choose $v \in L$, $L := L \setminus \{v\}$, $A := N^+(v) \setminus L_{i+1}$.
   If $A = \varnothing$, go to 3, otherwise $j := j+1$, $U := U \cup \{u_j\}$, $N^+(v) := (N^+(v) \cap L_{i+1}) \cup \{u_j\}$, $N^+(u_j) := A$, $L_{i+1} := L_{i+1} \cup \{u_j\}$, go to 3;
4) End.

## 5. Partitioning a circuit into cascades

Every layer of a circuit has a set of weights assigned to vertices belonging to the layer. The maximum weight in the layer is the delay of passing signal through it. A given combinational circuit must be partitioned into a fixed number $k$ of *cascades*, the delay in the slowest cascade being minimal as possible. Each cascade is an ordered set of layers. The following problem arises. A similar problem is considered in [7].

A sequence $(a_1, a_2, \ldots, a_n)$ of positive numbers is given. A part of it in the form $(a_p, a_{p+1}, \ldots, a_q)$ where $1 \leqslant p < q \leqslant n$ is called segment. A given sequence must be partitioned into a fixed number $k$ of segments $B_1$, $B_2$, …, $B_k$ where $B_i = (a_{n_{i-1}+1}, \ldots, a_{n_i})$, $i = 1, 2, \ldots, k$, $n_0 = 0$, $n_k = n$, with the following properties. Segments $B_1$, $B_2$, …, $B_k$ correspond to $S_1$, $S_2$, …, $S_k$ where $S_i = \sum\limits_{a_j \in B_i} a_j$, and $\max\{S_1, S_2, \ldots, S_k\}$ must be minimal.

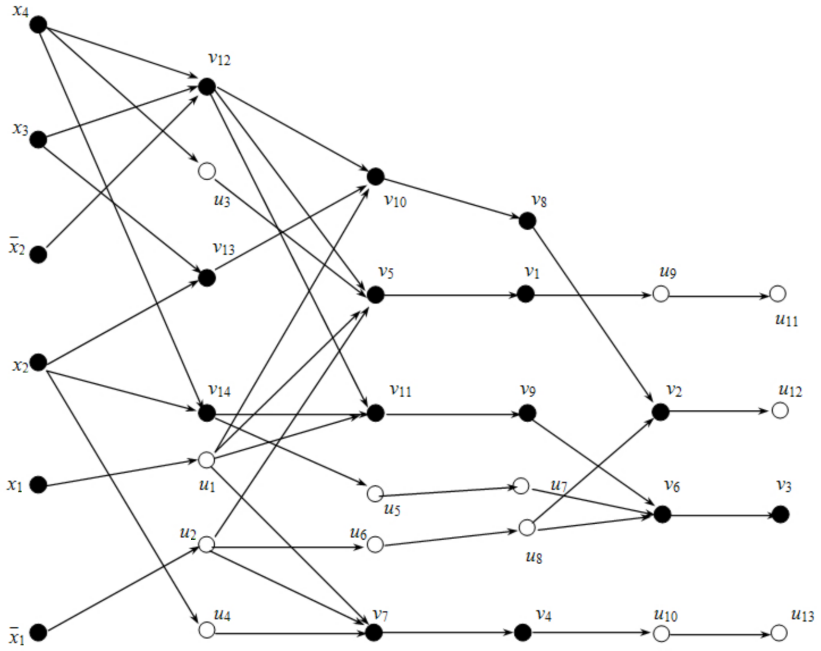


Fig. 5. Digraph $G'$ with equalized paths.

The elements $S_1, S_2, \ldots, S_k$ correspond to the desired cascades of the given circuit, and each of them is proportional to the delay of the corresponding cascade.

We suggest the following method to obtain a solution for this task next to optimal.

First, the lower bound of the largest delay in a cascade is determined as $b = \left(\sum_{j=1}^{n} a_j\right)/k$. The current $i$-th segment is formed by accumulation of the sum $S_i = \sum_{a_j \in B_i} a_j$ comparing it with the bound $b$. When $S_i > b$ after adding next $a_j$ to the sum $S_i$, the rightmost element in $B_i$ is $a_j$ if $b - (S_i - a_j) > S_i - b$, and $a_{j-1}$ otherwise. The procedure is repeated for the rest of the sequence $(a_1, a_2, \ldots, a_n)$.

Below, Algorithm 3 is given. It solves this problem when $\left(\sum_{j=1}^{k} S_j\right)/k > a_i$ for any $i = 1, 2, \ldots, n$. The sequence $n_1, n_2, \ldots, n_k$ represents a solution of the problem where $n_i$ is the number of the rightmost element of $i$-th segment. In Algorithm 3, the symbols $\tau$, $S$ and $B$ denote the maximal delay in a cascade, average value of $S_i$ and the current value of $S_i$, correspondingly.

**Algorithm 3**

1) $i := 0$, $R := 0$, $\tau := 0$, $l := k$, $n_k := n$;
2) If $i = n$, go to 3, otherwise $i := i + 1$, $R := R + a_i$, go to 2;
3) $S := R/l$, $i := 0$, $B := 0$, $j := 1$;
4) $i := i + 1$. If $i = n$, go to 6,
   otherwise $C := B$, $B := B + a_i$. If $S > B$, go to 4,
   otherwise $D := S - C$, $E := B - S$.
   If $D < E$, then $j := j + 1$, $nj := i - 1$, $B := a_i$, go to 5,
   otherwise $j := j + 1$, $n_j := i$, $C := B$, $B := 0$;
5) $l := l - 1$, $R := R - C$, $S := R/l$. If $\tau < C$, then $\tau := C$, go to 4,
   otherwise go to 4;
6) End.

In the considered example, the sequence of numbers (0, 2, 4, 1, 3, 1) represents the delays in the layers. Suppose this sequence must be partitioned into three segments ($k = 3$, $n = 6$). The result of execution of Algorithm 3 is the sequence $(n_1, n_2, n_3) = (2, 3, 6)$, and the maximal cascade delay is 5.

The triggers must be in the circuit after the second, third and sixth layers at each of the rightmost element of every cascade. The digraph transformed appropriately is shown in Fig. 6 where the vertices corresponding to triggers are marked with $d_i$. The vertices representing fictive elements with zero delay must be removed from the digraph.

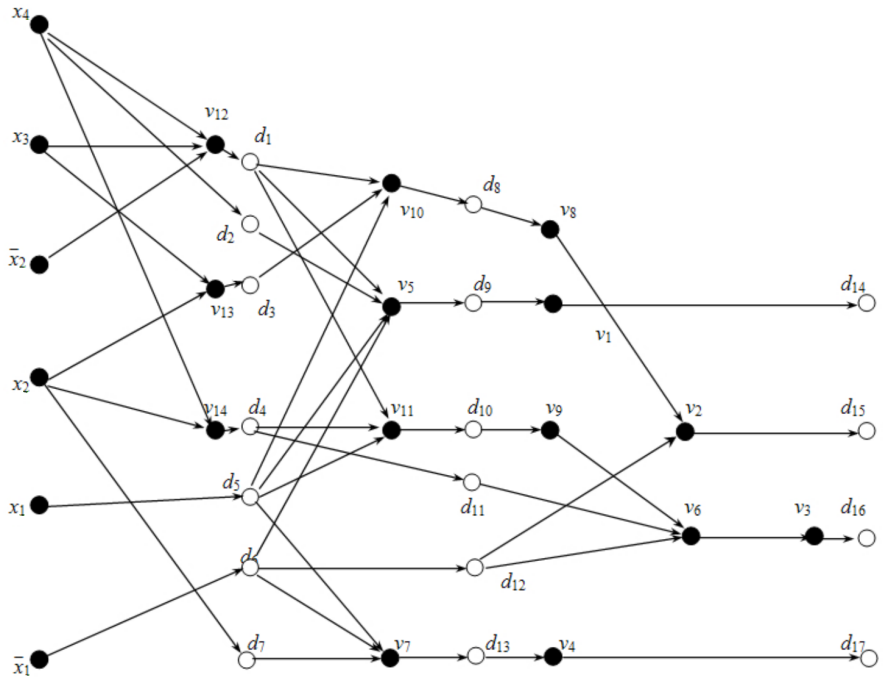


Fig. 6. Digraph with vertices corresponding to triggers.

Figure 7 shows the circuit working in the pipeline mode as a result of transformation of the circuit at Fig. 1. The inverse outputs of triggers are used instead of invertors in Fig. 7.

Suppose, the trigger delay is four conventional units of time that are used for estimate of cascade delay in the circuit from the considered example. The delay of the slowest cascade is equal to 5. So, the clock time period must be at least 9. According to the formula (2) the acceleration of signal processing can be reached when the length of the input sequence is at least 14. The complete reaction of the initial circuit to such a sequence will be after 154 conventional units of time, while the pipelined circuit will produce the reaction after 153 units. It is seen from the formula (1) that the difference between these values will increase if the length of input sequence increases.

## Conclusion

The suggested approach to increasing the throughput of a combinational circuit is intended for its application to ready circuits, and it solves this task at the level of circuit technology. Perhaps, the more effect can be reached at the functional level. But in this case, it is necessary to design the circuit from the start, having its functional description.

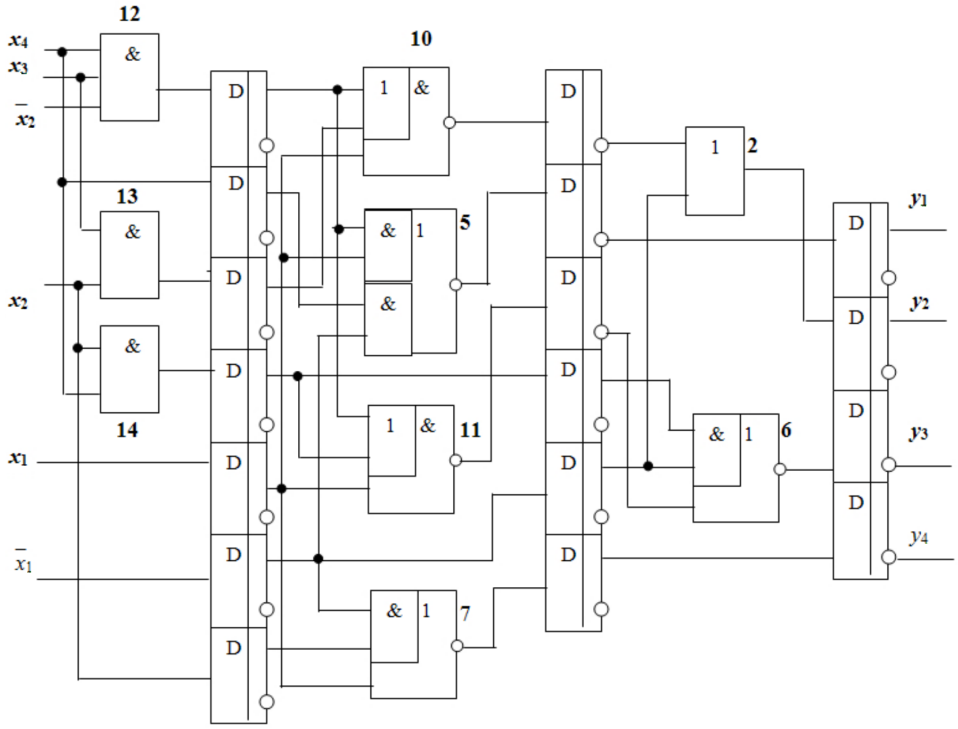


Fig. 7. Circuit working in pipeline mode.

## BIBLIOGRAPHY

1. *Kagan B. M. and Kanevskij M. M.* Digital Computers and Systems. Moscow: Energiya, 1973. 680 p. (in Russian).
2. *Voevodin V. B.* Mathematical Models and Methods in Concurrent Processes. Moscow: Nauka, 1986. 296 p. (in Russian).
3. *Kapitonova Yu. V. and Letichevskij A. A.* Mathematical Theory of Computing System Design. Moscow: Nauka, 1988. 296 p. (in Russian).
4. *Kukharev G. A., Tropchenko A. Yu., and Shmerko V. P.* Systolic Processors for Signal Processing. Minsk: Belarus, 1988. 127 p. (in Russian).
5. *Kukharev G. A., Shmerko V. P., and Zaitseva E. N.* Algorithms and Systolic Processors for Multivalued Data Processing. Minsk: Navuka i Tekhnika, 1990. 296 p. (in Russian).
6. *Lukoshko G. and Konnov E.* CMOS-base array chips of K1574 series // Radiolyubitel. 1997. No. 9. P. 39–40 (in Russian).
7. *Agibalov G. P. and Belyaev V. A.* Technique for Solving Combinatorial Logical Tasks by the Method of Shortcut Bypassing of a Search Tree. Tomsk: TSU, 1981. 126 p. (in Russian).