REIHE INFORMATIK
7/95

# Extremal Codes for Speed-Up of Distributed Parallel Arbitration

M. Makhaniok, V. Cherniavsky, R. Männer, K.-H. Noffz
Universität Mannheim
Seminargebäude A5
D-68131 Mannheim

# EXTREMAL CODES FOR SPEED-UP OF DISTRIBUTED PARALLEL ARBITRATION¶

M. Makhaniok[1)], V. Cherniavsky[1)], R. Männer[2,3)], K.-H. Noffz[2)]

1) Institute of Engineering Cybernetics, Academy of Sciences, Surganov St. 6, 220012 Minsk, Republic Belarus
2) Lehrstuhl für Informatik V, Universität Mannheim, A5, D-68131 Mannheim, Germany
3) Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany

**Index terms**: Buses, arbitration priorities, extremal codes, distributed parallel arbitration, Futurebus+

**Abstract**. This paper describes a method that allows the speed up of parallel processes in distributed arbitration schemes as used in Futurebus+. It is based on special arbitration codes that decrease the maximal arbitration time to a specified value. Such codes can be applied with few, if any, minor changes of the hardware. The general structure of these codes is given.

**Introduction**. One important component of every multiprocessor bus is the arbitration system. Many modern multiprocessor buses use arbitration systems principally based on the same distributed parallel arbitration scheme [1]. In this scheme, the time for acquiring bus mastership depends on the maximal arbitration time required. This time is determined by the set of arbitration priorities available [2-4]. Such a set will be called an arbitration code or simply a code below. The maximal arbitration time is reduced, e.g., by using binomial codes as discussed in [4] where these codes were introduced. From the discussion it follows that the achievable speed-up depends on the number of arbitration priorities in the binomial code, i.e., on its capacity. However, it has not been considered whether the binomial codes have the maximal possible capacity, i.e., whether there are other codes with equal or greater capacity that can reduce the maximal arbitration time to the same or a lower limit.

In the present paper we derive the general structure of the codes of maximal capacity that we will call extremal codes. It will be shown that there exist different extremal codes with binomial codes as one example only.

**Arbitration process**. We consider a distributed parallel arbitration process for a single-bus multiprocessor system. To each processor a unique arbitration priority is assigned that is a m-bit binary word. It will be called an arbitration word or simply a word below. If one or more processors request bus mastership, an arbitration process is started. At

¶ This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grants Ma 1150/8-1 and 436-WER-113-1-0 (438 113/117/0).

this time each processor decides whether it participates in the arbitration process or not. The objective of the arbitration process is to find the processor of highest priority among all participating ones.

Arbitration is done in a decentralized way. Each processor has its own local arbitration circuit which is connected to m common arbitration lines. On these lines the logical OR function is formed of the corresponding outputs of the local arbitration circuits. In Fig. 1 this circuit is shown with outputs in positive logic corresponding to expressions (1). In a real implementation negative logic would be used, i.e., the outputs would be inverted so that the OR function could be realized as a wired-OR on open-collector lines. It is important to distinguish between the state of an arbitration line and the state of the corresponding output of an arbitration circuit. Such an output can be changed by a single processor by assertions or withdrawals of bits of its arbitration word. This may or may not influence the state of the arbitration line.

Usually the arbitration circuits are realized as combinatorial logic and the OR functions are computed as a wired-OR using open-collector drivers. The output signals depend on the word assigned to the processor and on the signals currently formed on the arbitration lines.

Any change of the output signals of a processor - assertions as well as withdrawals - will be called a switch of the processor. If a processor is not participating in the process it does not assert its output signals, i.e., keeps them in the "zero" state. When the arbitration process starts, all participating processors assert all bits of their words onto the arbitration lines. Simultaneously they compare their own bit values with the current state of the corresponding lines. If a line is set to a logic "one" and the corresponding bit of the processor´s word is zero, the processor switches. This means it determines the most significant bit that is not equal to the state of the corresponding arbitration line and withdraws this and all its less significant bits from the lines. If the condition is no longer true the processor switches back to the previous state.

Let all processors have the same speed. Then the arbitration process for the j-th participating processor can be described by the following expressions[1].

$$b_{1,j}(t+\tau)=s(t)a_{1,j},$$

$$b_{2,j}(t+\tau)=s(t)a_{2,j}(a_{1,j}\vee\neg B_1(t)),$$

$$b_{3,j}(t+\tau)=s(t)a_{3,j}(a_{1,j}\vee\neg B_1(t))(a_{2,j}\vee\neg B_2(t)),$$

[1] If the local arbitration circuits of the processors taking part in the arbitration process have different delays $\tau$ for switching their signals on the arbitartion lines it is necessary to take into account results from [6] to minimize the maximal arbitration time.

$$\bullet\bullet\bullet \qquad (1)$$

$$b_{m,j}(t+\tau)=s(t)a_{m,j}\prod_{r=1,...,m-1}(a_{r,j}\vee\neg B_r(t)),$$

$$B_i(t)=\mathbf{V}_{j\in P}\, b_{i,j}(t),\ \ i=1,...,m,$$

where $j\in P$,

| | |
|---|---|
| P | is the set of processors participating in the process, |
| t | is the current time ($-\infty<t<\infty$), |
| $a_{i,j}$ | is bit i of the word which is assigned to processor j, ($a_{i,j}\in\{0,1\}$, $a_{1,j}$ is the most significant bit), |
| $b_{i,j}(t)$ | is the output signal asserted by processor j onto arbitration line i, |
| $B_i(t)$ | is the signal formed on arbitration line i, |
| $\tau$ | is the delay introduced by any processor j, ($0<\tau<\infty$), to assert its signals onto the arbitration lines or to withdraw them from there, |
| s(t) | is the start signal (s(t)=0 if t<0, and s(t)=1 otherwise), |
| m | is the number of bits in the arbitration words, i.e., their width. |

The first m equalities in (1) express the individual output signals of all processors. The last one describes the current state of the open collector arbitration lines which form the logic OR function of the signals asserted.

**Duration of the arbitration process.** The arbitration process is a sequence of switches. From expressions (1) it follows that every processor can execute at most m switches in one arbitration process. The following example shows that m switches can indeed occur. Let m=4, let three processors participate in the process, and let them have the words ‹1010›, ‹1001›, and ‹0111›. The processors need a first switch to assert these numbers onto the arbitration lines. After the first switch, the lines are in the state ‹1111›. The second switch is needed to withdraw the third bit of word ‹1010›, the forth bit of word ‹1001›, and the last three bits of word ‹0111›. The arbitration lines will therefore be in the state ‹1000› after the second switch. Now the first and the second processors assert their last three bits again. Thus, after the third switch, the arbitration lines will be in the state ‹1011›. Finally, the withdrawal of the last bit by the second processor represents the fourth switch. The word ‹1010› will appear on the arbitration lines and will be stable. By comparing this word with its own arbitration word each processor decides now if it has the maximal priority.

The duration of the arbitration process is proportional to the number of switches executed. Since we consider a situation such as Futurebus+, neither the delays of other modules nor the modules participating in the process nor their arbitration words are known. Because the duration of the process cannot be determined dynamically as a

function of the participants and their delays, every module has always to assume that the maximal number of switches will occur and that they will be executed by the slowest modules, i.e., the worst case. We, therefore, can assume without loss of generality that the delays are identical. Thus to get a correct result of the process for any combination of m-bit arbitration words, the participating processors have to wait the time $m \cdot \tau$ before they make their decision. However, the maximal number of switches can be considerably smaller than m as shown below.

Denote by |K| the number of words in code K, i.e., its capacity.

1. If a code of width m has a capacity $|K|<2^m$, it is possible to reduce the maximal number of switches that can occur if any combination of words from K will participate in the process. Consider, e.g., a code of width 4 and of capacity 15. If the words ‹1010› and ‹1001› are included in the code K and both participate in the process, in the worst case it will require four switches as shown above. However, if we exclude from K either ‹1010› (the only word possibly requiring four switches) or ‹1001› (the only word creating this kind of conflict with ‹1010›), then the arbitration process will require at most three switches for all combinations of the 15 remaining 4-bit arbitration words. In this case a speed-up of 25% is obtained.
2. If a code of width m has the capacity $|K|=2^m$, then we cannot leave out any word because all of them are in use. However, such a possibility exists, if we use an additional arbitration line, i.e., a (m+1)-bit code instead of a m-bit code. Now we can turn to the first case and - by appropriate selection of the words - reduce the maximal number of switches by a factor of approximately 2 as shown in [4].

Therefore, the following problem arises. If the width of a code and its capacity are given, how to select its words from the set of all possible m-bit binary words so that the maximal number of switches is minimized.

**Extremal codes**. To compare the number of switches in different arbitration codes we introduce the following notations.

- $a*b$ is the word that is the concatenation of words a and b,
- $a^{(i)}$ is the word that is the concatenation of i words a,
- $\{a*K\}$ is the code formed by the left concatenation of word a with every word from code K, $\{a*\varnothing\}=\{\varnothing*a\}=\{a\}$,
- $X^{(m)}$ is the code consisting of all possible words of width m.

For example, $0^{(3)}$=‹000›, $X^{(2)}$={11,10,01,00}; and if a=‹10›, b=‹11›, K={111,110} then a∗b=‹1011›, $a^{(2)}$=‹1010›, a∗K={10111,10110}.

**Definition 1.** We define the depth d(K) of an arbitration code K as the maximal number of switches that can occur if any subset Z of K will take part in an arbitration process.

**Definition 2.** We define a code of width m and depth n as the extremal code $Ex^n_m$ if there is no code of the same width and depth, and greater capacity.

Using a code of smaller depth results in a faster arbitration process. According to expressions (1) the depth can vary from 1 to m. Therefore, our task is to find for every depth the code K of maximal capacity.

Let us first focus on the special classes of extremal codes of depth n=0,1,m. The structures of these codes are defined in the following Lemmas.

**Lemma 1**. The code $Ex^0_m$ has the capacity 1 and consists of the single zero word

$$Ex^0_m=\{0^{(m)}\}. \quad (2)$$

The process starts at t=0. For t<0 the arbitration lines are set to zero. Thus, the arbitration process of the single word $0^{(m)}$ does not require any switches. Any other word generates at least one switch during which its non-zero bit values are asserted onto the arbitration lines.

**Lemma 2**. The code $Ex^1_m$ has the capacity (m+1) and is constructed by any bit permutation applied to all words of

$$\cup_{i=0,\ldots,m}\langle 0^{(m-i)}*1^{(i)}\rangle. \quad (3)$$

*Proof*. Let $a_1=\langle a_{1,1}\ldots a_{m,1}\rangle$, $a_2=\langle a_{1,2}\ldots a_{m,2}\rangle$, with $a_1>a_2$, be two words of a code $Ex^1_m$. If $a_{i,1}=0$ ($i\in[1,m]$) then $a_{i,2}=0$ also has to be true. Otherwise, the arbitration process of $\{a_1,a_2\}$ could require two switches, the assertion of $a_1$ and $a_2$ and at least the withdrawal of the bit $a_{i,2}=1$. This is impossible for a code of a depth equal to 1. Therefore, $w_{a1}>w_{a2}$ if $w_{a1}$ and $w_{a2}$ are the number of non-zero bits in the words $a_1$ and $a_2$. Let us arrange all words of the code in increasing order and prove that their number s is equal to m+1. If $w_j$ is the number of non-zero bits in the word $a_j$, then the expression $0\le w_{a1}<w_{a2}<\ldots<w_{as}\le m$ is valid. Obviously, the number of non-zero bits in the word $a_j$ is (j-1) and the maximal possible value of s is m+1.

These conditions are only satisfied for the codes $Ex^1_m$ defined in Lemma 2. For them the signals $B_i(t)$ are settled after the first assertion of the maximal word, i.e., after the first switch. Fig. 2 shows examples of codes $Ex^1_4$ with the corresponding permutation of the words defined by (3).

**Lemma 3**. The code $Ex^m_m$ has the capacity $2^m$ and consists of all possible m-bit words

$$Ex^m_m=X^{(m)}. \quad (4)$$

The code $X^{(m)}$ has the maximal possible capacity according to its definition and can cause a sequence of m switches but not more, as it was shown in [3,4].

**Theorem**. The capacity of any extremal code $Ex^n_m$ is given by the equation

$$|Ex^n_m|=\sum_{i=0,...,n} C^i_m, \quad m\geq 1,\ m\geq n\geq 0 \tag{5}$$

where $C^i_m$ are the binomial coefficients.

*Proof*. Equality (5) is obviously true for the cases n=0,1,m that have been considered in the previous Lemmas.

We will prove (5) for m>n>1 by a double induction with parameters n and m. The hypothesis is that (5) is true for the codes $Ex^n_{m-1}$ and $Ex^{n-2}_{n-1}$, $Ex^{n-2}_{n}$,..., $Ex^{n-2}_{m-2}$. From this we will derive that equation (5) holds true for the code $Ex^n_m$. This induction starts from the cases n=0,1,m confirmed by the Lemmas, and proves the Theorem up to any given m and n in the progression

$$Ex^2_2, Ex^0_m \rightarrow Ex^2_3 \rightarrow Ex^2_4 \rightarrow ... \rightarrow Ex^2_m;$$
$$Ex^3_3, Ex^1_m \rightarrow Ex^3_4 \rightarrow Ex^3_5 \rightarrow ... \rightarrow Ex^3_m;$$
$$\bullet\bullet\bullet$$
$$Ex^n_n, Ex^{n-2}_m \rightarrow Ex^n_{n+1} \rightarrow Ex^n_{n+2} \rightarrow ... \rightarrow Ex^n_m.$$

*Basis*. Assume that the following equations are true.

$$|Ex^n_{m-1}|=\sum_{i=0,...,n} C^i_{m-1}, \tag{6}$$

$$|Ex^{n-2}_{k-1}|=\sum_{i=0,...,n-2} C^i_{k-1}, \ k=n,(n+1),...,(m-1). \tag{7}$$

*Induction*. We intend to prove (5) for the code $Ex^n_m$ for m>n>1 assuming the truth of the basis. First we construct a certain code $R^n_m$ and consider its depth and capacity. The code

$$R^n_m=\cup_{j=n-1,...,m} E_j \tag{8}$$

is the union of the following subcodes (Fig. 3a)

$$E_{n-1}=\{1^{(m-n+1)}*X^{(n-1)}\},$$
$$E_k=\{1^{(m-k)}*0*Ex^{n-2}_{k-1}\},\ k=n,(n+1),...,(m-1),$$
$$E_m=\{0*Ex^n_{m-1}\}.$$

Obviously $E_p \cap E_q = \varnothing$ for any pair of subcodes ($p,q \in [n-1,m]$). Thus, the capacity of the code $R^n_m$ is the sum of the capacities of all subcodes

$$|R^n_m| = |E_{n-1}| + |E_n| + |E_{n+1}| + \ldots + |E_{m-1}| + |E_m| =$$

$$= |X^{(n-1)}| + |Ex^{n-2}_{n-1}| + |Ex^{n-2}_n| + \ldots + |Ex^{n-2}_{m-2}| + |Ex^n_{m-1}|.$$

Taking into account the expressions (6) and (7), assumed to be true as the basis, we can derive (see Appendix)

$$|R^n_m| = 2^{n-1} + \sum_{i=0,\ldots,n} C^i_{m-1} + \sum_{k=n,\ldots,m-1} \sum_{i=0,\ldots,n-2} C^i_{k-1} = \sum_{i=0,\ldots,n} C^i_m. \qquad (9)$$

Below it will be shown that the depth of $R^n_m$ is n. All words of $E_{n-1}$, $E_k$, and $E_m$ consist of two parts, an identical leading part, and a trailing part which is different for every word (Fig. 3a). Thus if only words of one of the subcodes $E_{n-1}$, $E_k$, or $E_m$ participate in the arbitration, switches happen only in the second part. Therefore, the process takes no more than n-1, n-2, or n switches correspondingly.

Below we show that if words from different subcodes participate in the process then it cannot have more than n switches. Consider the case when words from $E_m$ and $E_{m-1}$ are participating in an arbitration process simultaneously. The words from $E_{m-1}$ are headed by the bits ‹1›, and the words of $E_m$ are headed by ‹0›. Thus, after the first switch the logical OR of the words of $E_m$ and $E_{m-1}$ appears on the bus. After the second switch all words of $E_m$ will be withdrawn and only the words of $E_{m-1}$ are left. In the worst case additional n-2 switches are required to determine the maximum word among them. The same argument applies to any other combination of subcodes from $R^n_m$ which proves that the depth of $R^n_m$ is n.

The code $R^n_m$ has depth n, width m, and the capacity defined by (5). Let us show that there exists no other code of the same depth and width which has a capacity greater than $|R^n_m|$, i.e., $R^n_m$ is the extremal code.

Assume the contrary, i.e., that a code $S^n_m$ of m-bit words with $d(S^n_m)=n$ and $|S^n_m| > |R^n_m|$ exists. In order to construct the code $S^n_m$ in a form similar to $R^n_m$ let us introduce the following masks

$$L_{n-1} = \{1^{(m-n+1)} * X^{(n-1)}\},$$
$$L_k = \{1^{(m-k)} * 0 * X^{(k-1)}\},\ k=n,(n+1),\ldots,(m-1),$$
$$L_m = \{0 * X^{(m-1)}\}.$$

The union of these masks is the code consisting of all m-bit binary words $X^{(m)}$. So the code $S^n_m$ is a union of subcodes (Fig. 3b):

$$S^n_m=\cup_{j=n-1,...,m}T_j$$

where

$T_{n-1}=L_{n-1}\cap S^n_m=\{1^{(m-n+1)}*T'_{n-1}\}$
$T_k=L_k\cap S^n_m=\{1^{(m-k)}*0*T'_{k-1}\}$, k=n,(n+1),...,(m-1)
$T_m=L_m\cap S^n_m=\{0*T'_{m-1}\}$.

As before $T_p\cap T_q=\varnothing$; p,q$\in$[n-1,m]; for any pair of subcodes and therefore

$$|S^n_m|=|T_{n-1}|+|T_n|+|T_{n+1}|+...+|T_{m-1}|+|T_m|.$$

To estimate $|S^n_m|$, consider the capacities of the individual subcodes. Since subcode $T'_{n-1}$ consists of (n-1)-bit words its capacity does not exceed $2^{n-1}$, so $|T_{n-1}|=|T'_{n-1}|\le 2^{n-1}=|X^{(n-1)}|$. The depth of the subcode $T'_{m-1}$ cannot be greater than n. Otherwise the depth of $T_m$ and therefore the depth of $S^n_m$ would exceed n. Since $T'_{m-1}$ consists of (m-1)-bit words its capacity has the upper limit $|Ex^n_{m-1}|$. For $T_m$ we therefore have $|T_m|=|T'_{m-1}|\le |Ex^n_{m-1}|$. Since the capacity of $T_{n-1}$ cannot exceed that of $E_{n-1}$, and since the capacity of $T_m$ cannot exceed that of $E_m$, the inequality $|T_k|>|E_k|$ has to hold true at least for one k$\in$[n,m-1] to meet the assumption $|S^n_m|>|R^n_m|$. We define s as the minimal value of the index k for which the inequality $|T_k|>|E_k|$ holds. Since $|T_s|=|T'_{s-1}|>|E_s|$ and $|E_s|=|Ex^{n-2}_{s-1}|$, the depth $d(T_s)=d(T'_{s-1})$ is greater than (n-2).

Let us form the code $D_{s+1}=T_{s+1}\cup T_{s+2}\cup...\cup T_m$. Consider in this code the bits of column (m-s+1) (see Fig. 3b). If in $D_{s+1}$ there exists a word $a_j$ with a non-zero bit (m-s+1), then the arbitration process for the code $Z=T_s\cup\{a_j\}$ can require more than n switches. They are: the assertion of the words of Z (the first switch), the withdrawal of the non-zero bit (m-s+1) of the word $a_j$ (the second switch), and the settlement of the maximal word of the code $T'_{s-1}$ (a sequence of more than (n-2) switches). This is impossible for the code $S^n_m$ of depth n. So, all words of $D_{s+1}$ as well as the words of $T_s$ must also have zeros in column (m-s+1). Let us form the code $D^*_s$ by elimination of this column (m-s+1) from the code $D_s=T_s\cup D_{s+1}$. Obviously the depth $d(D^*_s)$ is still equal to $d(D_s)$. Furthermore, the depth $d(D_s)$ as well as the depth of any other subcode of $S^n_m$ cannot be greater than n. Then we have $|D^*_s|<|Ex^n_{m-1}|$ for the capacity of the code $D^*_s$ of (m-1)-bit words. Therefore, $|D_s|=|D^*_s|<|Ex^n_{m-1}|$.

Now we have upper bounds for the subcodes of $S^n_m$. Taking into account the basis (6) and (7), we can finally estimate $|S^n_m|$ as

$$|S^n_m|=|T_{n-1}|+|T_n|+|T_{n+1}|+...+|T_{s-1}|+|D_s|\le$$

$$\le|X^{(n-1)}|+|Ex^{n-2}_{n-1}|+|Ex^{n-2}_n|+|Ex^{n-2}_{s-2}|+|Ex^n_{m-1}|=$$

$$=2^{n-1}+\sum_{k=n,...,s-1}\sum_{i=0,...,n-2}C^i_{k-1}+\sum_{i=0,...,n}C^i_{m-1}$$

After comparing this expression with (9), we directly obtain

$$|S^n_m| < \sum_{i=1,...,m} C^i_m = |R^n_m| \qquad (10)$$

since s<m. So the assumption $|S^n_m|>|R^n_m|$ leads to the contradiction (10) which proves that a code with capacity greater than $|R^n_m|$ cannot exist.

Let us prove now that any code $S^n_m$ of depth n which is different from $R^n_m$ has a smaller capacity. Assume in contrast that $|S^n_m|=|R^n_m|$ and $S^n_m \neq R^n_m$. If $|T'_k|>|Ex^{n-2}_{k-1}|$ or $d(T'_k)>(n-2)$ holds true for some $k \in [n,m-1]$ then the previous argument brings us back to (10) which contradicts the assumption. On the other hand, $|T'_k|<|E^{n-2}_{k-1}|$ for all k=n,...,m-1 directly yields $|S^n_m|<|R^n_m|$. The only alternative left is $|T_k|=|E_k|$ and $d(T_k)\leq(n-2)$ for all k=n,...,m-1. This is possible only if $T'_{k-1}=Ex^{n-2}_{k-1}$ and $T_k=E_k$. In the same way we have $T_m=E_m$, $T_{n-1}=E_{n-1}$, and therefore $S^n_m=R^n_m$.

So, no code of depth n and width m can have a higher capacity than determined by (5). Q.E.D.

We proved that the capacity of the extremal codes is determined by expression (5) for all m and n. Some values are given in Tab. 1. In [4] it was shown that the capacity of the binomial codes of width m and depth n is also determined by the same expression. Therefore, the binomial codes are extremal.

**Construction of the Extremal Codes.** During the proof of (5) it has been shown that any code $R^n_m$ constructed in accordance with (8) has the depth n and the maximum capacity. It also has been proven that any code different from $R^n_m$ has a lower capacity. In view of this, expression (8) represents the general structure of the extremal codes for any n (n≠0,1,m). We can, therefore, rewrite (8) as

$$Ex^n_m=\{0*Ex^n_{m-1}\}\cup\{1^{(m-n+1)}*X^{(n-1)}\}\cup_{i=1,...,m-n}\{1^{(i)}*0*Ex^{n-2}_{m-i-1}\},\ n\neq 0,1,m. \qquad (11)$$

Using expressions (2)-(4) and (11) the extremal codes of any depth and width can be constructed. For even n (n≠0,m) expression (11) allows to construct the code $Ex^n_m$ from the codes of depth 0, which are determined uniquely by Lemma 1. So for even n only one extremal code exists. Since the binomial code with the same parameters m and n has the capacity given by (5) it is this extremal code. If n is odd (n≠1,m) then the codes $Ex^n_m$ are constructed from the codes of depth 1, which are not unique (Lemma 2). So it is possible to construct different extremal codes of the same odd n. Here the binomial codes are only one example among them.

In Fig. 4 some examples of codes $Ex^n_5$ are shown. The code $Ex^3_5$, e.g., is formed in accordance with (11) as

$$\{0*Ex^3{}_4\}\cup\{1^{(3)}*X^{(2)}\}\cup\{1^{(2)}*0*Ex^1{}_2\}\cup\{1*0*Ex^1{}_3\}$$

where $Ex^3{}_4=\{0*Ex^3{}_3\}\cup\{11*X^{(2)}\}\cup\{10*Ex^1{}_2\}$ is also given by (11).

It is easy to determine the number $\gamma^n{}_m$ of different codes $Ex^n{}_m$. From Lemmas 1-3 we have $\gamma^0{}_m=\gamma^m{}_m=1$ and $\gamma^1{}_m=m!$. For other n

$$\gamma^n{}_m=\gamma^n{}_{m-1}\cdot\gamma^{n-2}{}_{m-2}\cdot\gamma^{n-2}{}_{m-3}\cdot\ldots\cdot\gamma^{n-2}{}_{n-1} \quad \text{for } m>n>1$$

follows directly from (11).

It allows to calculate $\gamma^n{}_m$ recursively. Some values of $\gamma^n{}_m$ are given in Tab. 2.

**Application.** Having derived the expressions for the capacities of extremal codes we can calculate the achievable speed-up.

Consider a system that uses a code K of width m, $|K|<2^m$. If the arbitration words of this code are chosen without taking into account its depth, then the chances are high to get a depth equal to m. Since the system uses not all possible binary words of width m, it is possible to replace the original code K by an extremal code in order to reduce the maximal number of switches. To achieve the maximal speed-up one has to find a code $Ex^n{}_m$ of capacity $|Ex^n{}_m|\geq|K|$ and minimal depth n. Now if K is replaced by $Ex^n{}_m$, the maximal number of switches is reduced from m to n. This results in a speed-up of the arbitration process by factor of m/n.

Obviously the maximal speed-up s is obtained if the depth n of the code K is minimal and the number of processors p is maximal. For arbitration the minimal n is equal to 1. Here the maximal number of processors is the width of the arbitration words plus 1. Thus, the highest speed-up is obtained with a relatively small number of processors. Nowadays this is the most relevant case. Consider, e.g., a Futurebus+ system with nine processors. Since Futurebus+ uses a priority word of width m=8, it is possible to achieve a speed-up s=8 by applying the extremal code of depth n=1. For a higher number of processors the achievable speed-up decreases because codes $Ex^n{}_m$ of increasing depth are required to assign arbitration words to all processors.

Tab. 3 shows the achievable speed-up for a system where all arbitration words are used. In this case the speed-up is obtained by providing additional arbitration lines. This allows to apply extremal codes of higher width and smaller depth. Note that the arbitration time can be reduced by a factor of ≈2 by adding a single line only [4]. Higher speed-ups can only be obtained using additional arbitration lines.

Note that after the maximal word is finally asserted onto the arbitration lines switches in the local arbitration circuits of processors can still occur. Consider, e.g., an arbitration between the two words {‹1111›,‹1101›}. The arbitration process takes only one switch to assert the word ‹1111›. After this moment the processor having the word ‹1101› with-

draws its least significant non-zero bit. Depending on the technology used this withdrawal might or might not influence the arbitration line which is already set to a logic "one". If, e.g., open collector lines are used to implement wired OR logic, the withdrawal of a signal from a line in the logic "one" state may result in a "zero" glitch [5]. It is possible to avoid this glitch by using the code

$$\cup_{i=1,\ldots,m}\{1^{(i)}*0^{(m-i)}\}$$

as the $Ex^1_m$ code. In this case the code $Ex^n_m$ is uniquely defined for any m and n. For even n it coincides with the binomial code.

**Conclusions**. In this paper we derived the structure of codes which reduce the maximal arbitration time to a defined limit. We obtained a recursive formula that allows one to construct codes of highest capacity for all possible numbers of switches. No higher speed-up can be achieved by using any other code. It was shown that, in some cases, a considerable number of extremal codes exist. Binomial codes are only one representative of these codes. The recursive formula (11) that gives us the general structure of extremal codes is quite complicated. However, one has to use it only once when the arbitration system is set up. For most arbitration systems the codes can be applied directly without changes of the hardware.

## APPENDIX.

Here we present the detailed derivation of (9). The properties

$$1)\ \sum_{i=0}^{m} C_m^i = 2^m, \quad m\geq 0,$$

$$2)\ \sum_{i=n}^{m} C_i^n = C_{m+1}^{n+1}, \quad m\geq n\geq 0,$$

$$3)\ C_{m-1}^{i+1} + C_{m-1}^{i} = C_m^{i+1}, \quad m-2\geq i\geq 0$$

are used below. For any m and n (m>n>1) we can derive

$$2^{n-1}+\sum_{j=n-1}^{m-2}\sum_{i=0}^{n-2} C_j^i+\sum_{i=0}^{n} C_{m-1}^i = 2^{n-1}+\sum_{i=0}^{n-2}\sum_{j=n-1}^{m-2} C_j^i+\sum_{i=0}^{n} C_{m-1}^i = 2^{n-1}+\sum_{i=0}^{n-2}\left(\sum_{j=i}^{m-2} C_j^i-\sum_{j=i}^{n-2} C_j^i\right)+\sum_{i=0}^{n} C_{m-1}^i =$$

$$=2^{n-1}+\sum_{i=0}^{n-2}\sum_{j=i}^{m-2} C_j^i-\sum_{i=0}^{n-2}\sum_{j=i}^{n-2} C_j^i+\sum_{i=0}^{n} C_{m-1}^i = 2^{n-1}+\sum_{i=0}^{n-2}\sum_{j=i}^{m-2} C_j^i-\sum_{j=0}^{n-2}\sum_{i=0}^{j} C_j^i+\sum_{i=0}^{n} C_{m-1}^i \overset{1,2)}{=}$$

$$= 2^{n-1} + \sum_{i=0}^{n-2} C_{m-1}^{i+1} - \sum_{j=0}^{n-2} 2^{j} + \sum_{i=o}^{n} C_{m-1}^{i} = 2^{n-1} + \sum_{i=0}^{n-2} C_{m-1}^{i+1} - 2^{n-1} + 1 + \sum_{i=0}^{n} C_{m-1}^{i} = \sum_{i=0}^{n-2} \left( C_{m-1}^{i+1} + C_{m-1}^{i} \right) +$$

$$+ C_{m-1}^{n-1} + C_{m-1}^{n} + 1 \overset{3)}{=}$$

$$\overset{3)}{=} \sum_{i=0}^{n-2} C_{m}^{i+1} + C_{m-1}^{n-1} + C_{m-1}^{n} + 1 \overset{3)}{=} \sum_{i=0}^{n-2} C_{m}^{i+1} + C_{m}^{n} + 1 = \sum_{i=0}^{n} C_{m}^{i} - C_{m}^{0} + 1 = \sum_{i=0}^{n} C_{m}^{i} .$$

**References.**


[1] Borrill P. "A Comparison of 32-bit Buses", IEEE Micro, Dec. 1985, pp. 71-79

[2] Taub D.M. "Corrected Settling Time of the Distributed Parallel Arbiter", IEE Proc. E, Vol. 139, No. 4, 1992, pp. 348-354

[3] Taub D.M. "Improved Control Acquisition Scheme for the IEEE 896 Futurebus", IEEE Micro, Vol. 7, No. 3, 1987, pp. 52-62

[4] Makhaniok M., Cherniavsky V., Männer R., Stucky O.: "Binomial Coding in Distributed Arbitration Systems", Electr. Lett., Vol. 25, No. 20, 1989, pp. 1343-1344

[5] Gustavson D.B., Theus J.: Wire-OR Logic on Transmission Lines; IEEE Micro, Vol. 3, No. 3, 1983, pp. 51-55

[6] Makhaniok M., Männer R.: "Synchronization of Parallel Processes in Distributed Systems; in Bougè L., Cosnard M., Robert Y., Trystram D. (Eds.): Proc. CONPAR ´92 - VAPP V, Lyons, France; Lect. Notes in Comp. Sci. 634, Springer, Berlin (1992) pp. 157-162

**Captions.**

Fig. 1. Local arbitration circuit.
Fig. 2. Examples of extremal codes $Ex^1_4$.
Fig. 3. The structure of the sets $R^n_m$ and $S^n_m$.
Fig. 4. Examples of extremal codes $Ex^n_5$.

Tab. 1. The capacity $Ex^n_m$ of extremal codes of depth n and width m.
Tab. 2. The number $\gamma^n_m$ of extremal codes of depth n and width m.
Tab. 3. The speed-up s for different word widths m and number of processors $p=2^m$ achieved by adding dm arbitration lines.

| m | d 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 4 | | | | | | |
| 3 | 1 | 4 | 7 | 8 | | | | | |
| 4 | 1 | 5 | 11 | 15 | 16 | | | | |
| 5 | 1 | 6 | 16 | 26 | 31 | 32 | | | |
| 6 | 1 | 7 | 22 | 42 | 57 | 63 | 64 | | |
| 7 | 1 | 8 | 29 | 64 | 99 | 120 | 127 | 128 | |
| 8 | 1 | 9 | 37 | 93 | 163 | 219 | 247 | 255 | 526 |
| 9 | 1 | 10 | 46 | 130 | 256 | 382 | 466 | 502 | 511 |
| 10 | 1 | 11 | 56 | 176 | 386 | 638 | 848 | 968 | 1013 |
| 11 | 1 | 12 | 67 | 232 | 562 | 1024 | 1486 | 1816 | 1981 |
| 12 | 1 | 13 | 79 | 299 | 794 | 1586 | 2510 | 3302 | 3797 |
| 13 | 1 | 14 | 92 | 378 | 1093 | 2380 | 4096 | 5812 | 7099 |

Table 1.

| n | m 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 6 | 24 | 120 | 720 | 5040 | 40320 | $4 \cdot 10^5$ |
| 2 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | | | 1 | 2 | 24 | 6912 | $2 \cdot 108$ | $6 \cdot 10^{15}$ | $8 \cdot 10^{26}$ |
| 4 | | | | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | | | | | 1 | 2 | 96 | $3 \cdot 10^7$ | $2 \cdot 10^{21}$ |

Table 2.

| m | p | dm | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1.0 | | | | | | | | |
| 2 | 4 | 2.0 | | | | | | | | |
| 3 | 8 | 1.5 | 1.5 | 1.5 | 3.0 | | | | | |
| 4 | 16 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| 5 | 32 | 1.7 | 1.7 | 2.5 | 2.5 | 2.5 | 2.5 | 2.5 | 2.5 | 2.5 |
| 6 | 64 | 2.0 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 7 | 128 | 1.8 | 2.3 | 2.3 | 2.3 | 2.3 | 2.3 | 2.3 | 2.3 | 3.5 |
| 8 | 256 | 2.0 | 2.0 | 2.0 | 2.7 | 2.7 | 2.7 | 2.7 | 2.7 | 2.7 |
| 9 | 512 | 1.8 | 2.2 | 2.2 | 2.2 | 2.2 | 3.0 | 3.0 | 3.0 | 3.0 |
| 10 | 1024 | 2.0 | 2.0 | 2.5 | 2.5 | 2.5 | 2.5 | 2.5 | 2.5 | 3.3 |
| 11 | 2048 | 1.8 | 2.2 | 2.2 | 2.2 | 2.8 | 2.8 | 2.8 | 2.8 | 2.8 |
| 12 | 4096 | 2.0 | 2.0 | 2.4 | 2.4 | 2.4 | 2.4 | 3.0 | 3.0 | 3.0 |
| 13 | 8192 | 1.9 | 2.2 | 2.2 | 2.6 | 2.6 | 2.6 | 2.6 | 2.6 | 3.2 |

Table 3.

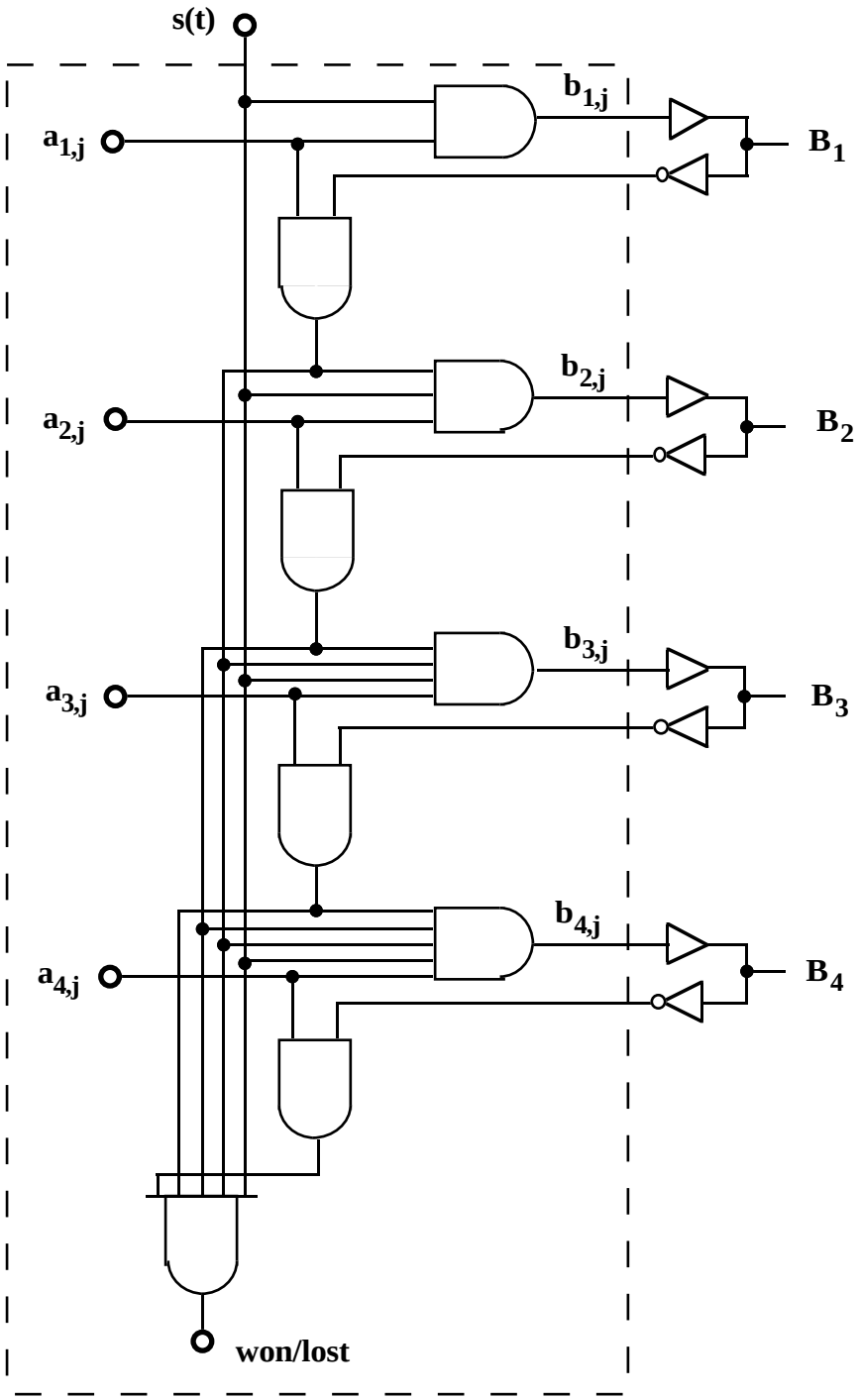


Fig. 1

| | | | | |
|---|---|---|---|---|
| 1111 | 1111 | 1111 | | 1111 |
| 0111 | 0111 | 0111 | | 1110 |
| 0011 | 0011 | 0101 | • • • | 1100 |
| 0001 | 0010 | 0001 | | 1000 |
| 0000 | 0000 | 0000 | | 0000 |

$\binom{1234}{1234}\binom{1234}{1243}\binom{1234}{1324}\quad\binom{1234}{4321}$

Fig. 2

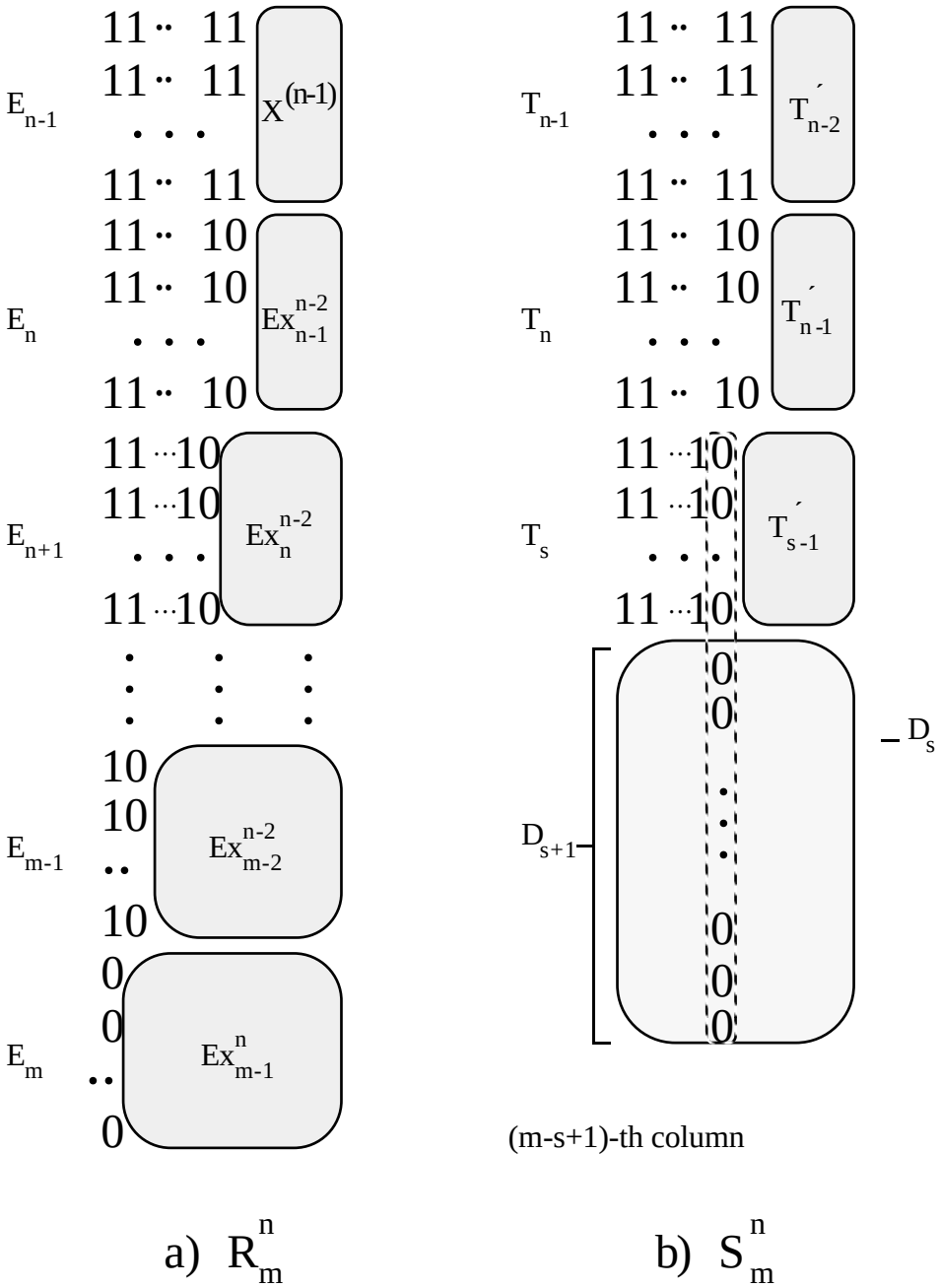


Fig. 3

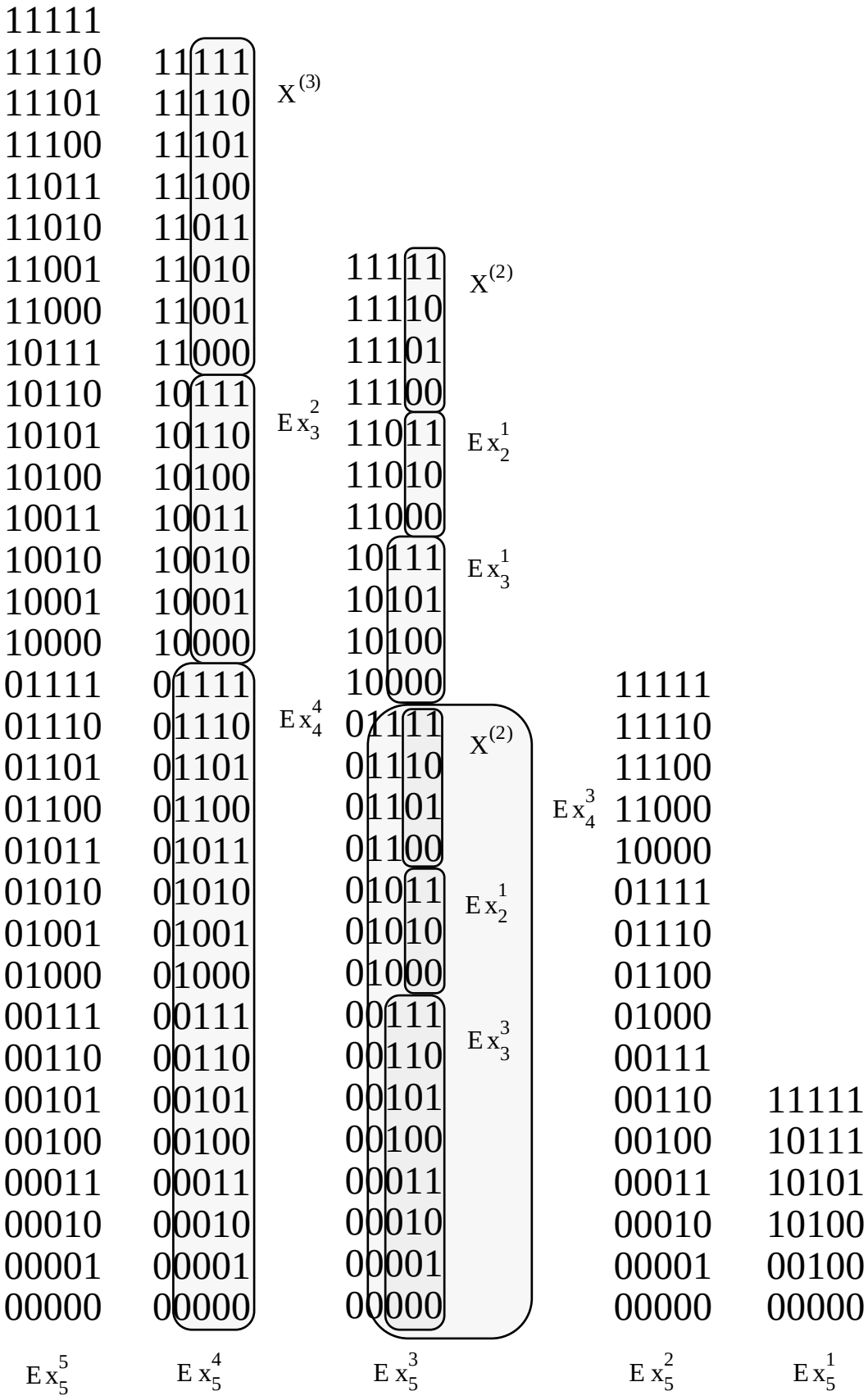
11111
11110
11101
11100
11011
11010
11001
11000
10111
10110
10101
10100
10011
10010
10001
10000
01111
01110
01101
01100
01011
01010
01001
01000
00111
00110
00101
00100
00011
00010
00001
00000
11111
11110
11101
11100
11011
11010
11001
11000
10111
10110
10101
10100
10011
10010
10001
10000
01111
01110
01101
01100
01011
01010
01001
01000
00111
00110
00101
00100
00011
00010
00001
00000
X(3)
Ex23
Ex44
11111
11110
11101
11100
11011
11010
11001
11000
10111
10101
10100
10000
01111
01110
01101
01100
01011
01010
01001
01000
00111
00110
00101
00100
00011
00010
00001
00000
X(2)
Ex12
Ex13
X(2)
Ex12
Ex13
Ex34
11111
11110
11100
11000
10000
01111
01110
01100
01000
00111
00110
00100
00011
00010
00001
00000
11111
10111
10101
10100
00100
00000
Ex55
Ex45
Ex35
Ex25
Ex15

Fig. 4