→ ОБЩЕТЕХНИЧЕСКИЕ ЗАДАЧИ И ПУТИ ИХ РЕШЕНИЯ

A. A. Blyudov
Petersburg State Transport University

ON THE SYNTHESIS OF TEST EQUIPMENT FOR MODULO CODES WITH SUMMATION

In this paper problems of test equipment for binary codes with summation of ones development are viewed. Particularly, an approach for modulo codes generators design is systematized and formula for its redundancy calculation is offered.

comparator, generator, tester, code with summation, complexity.

Introduction

Code with summation (also known as Berger code) [1] is widely used in functional control systems of combinational circuits. Organization of functional control system is carried out by adding to a supervised circuit some special equipment – a block of additional logic and a tester. The object of this paper is designing testers for code with summation.

A tester is a device designed for checking whether the code vector belongs to the certain code. Since the codes with summation ($S(n, m)$-codes, where m is the number of informational bits and $n = m + k$ is the total number of bits, k is the number of check bits of code vectors) refer to the class of partition codes [2], their correctness might be verified by comparing the informational vector with the check vector. All the $S(n, m)$-codes have the same tester structure, which is shown in fig. 1.

The structure of a tester is composed of two blocks: generator and comparator. Variables x_1, x_2,.., x_m corresponding to the informational bits of the code vector are sent to the generator inputs. The generator converts the informational vector to the check vector of the word (which is formed on its outputs $y_1, y_2,.., y_k$). Variables corresponding to check bits of code vector appear on the inputs $x_{m+1}, x_{m+2},.., x_n$ of the comparator. Comparator checks whether the vectors

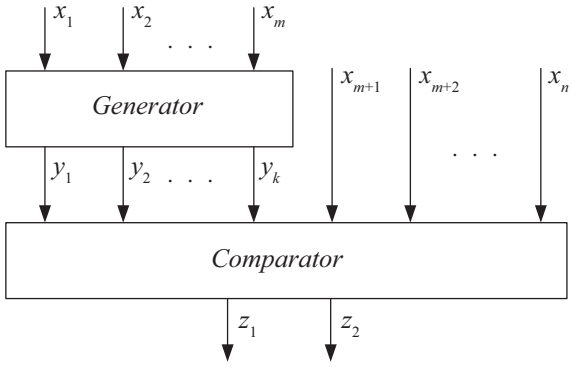

Fig. 1. Tester structure

$y_1, y_2,.., y_k$ and $x_{m+1}, x_{m+2},.., x_n$ are the same. If they match, paraphase signals (0,1) or (1,0) are formed at the outputs z_1 and z_2. Otherwise one can see signals (0,0) or (1,1). Signals (0,1) or (1,0) show that there is a word of the code with summation at the tester inputs.

For all the codes with summation the comparator is designed on the base of the standard two-rail checker module [2]. The generator as a functional block is a counter of ones in the informational vector. Designing such counters by different modulos is described in [3], [4].

1 Generators for Berger codes

In classical Berger code ones are counted by modulo $M = m + 1$. The methods of designing

generators for $S(n, m)$-codes based on standard circuits of Full Adders FA (fig. 2, *a*) and Half Adders HA (fig. 2, *b*) have been developed and described in [5].

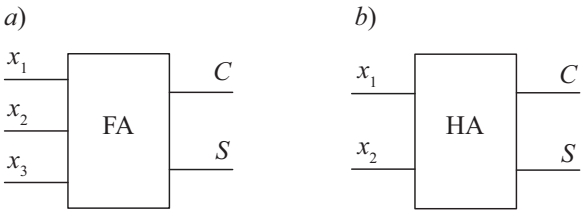

Fig. 2. Full Adder (*a*); Half Adder (*b*)

Full Adder FA has three inputs and two outputs. The binary value corresponding to the number of ones among the informational bits (inputs) is formed at the outputs. S is sum output ($S = x_1 \oplus x_2 \oplus x_3$), C is carry output ($C = x_1x_2 \vee x_1x_3 \vee x_2x_3$). Half Adder HA performs the same functions ($S = x_1 \oplus x_2$; $C = x_1x_2$) but has two inputs.

The basic structure of the generator presented in [5] is shown in fig. 3. Input variables x_1, x_2,.., x_m are divided into groups of two or three elements, every group appears at one FA or HA module. Adder Σ summarizes all the binary values at the FA and HA outputs. Complexity of

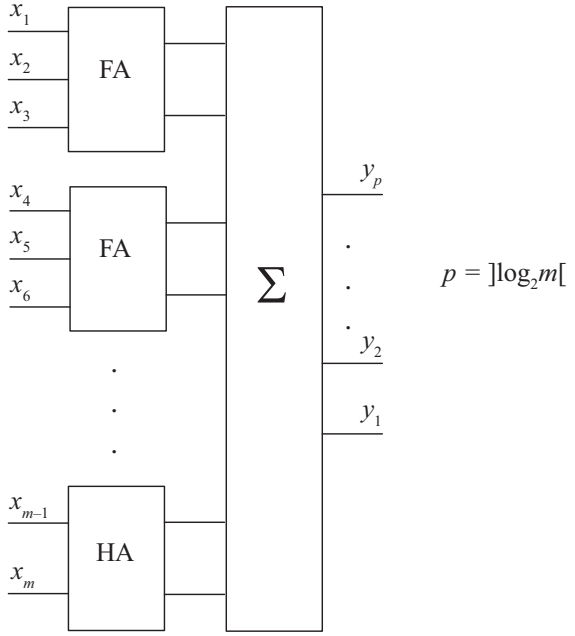

Fig. 3. Basic structure of generator for code with summation

the generator is determined by the way adder Σ is being realized. The most efficient ways of adder realization are offered in [6]. The generator for $S(9,6)$-code is shown in fig. 4.

Modulo codes $SM(n, m)$ [7] are built in the same way as Berger code, but the number of ones is counted by modulo $M \le m$, and M is equal to two in some power. Generator for such codes may be realized according to structure in fig. 3, but in this case only $]\log_2 M[$ lower bits of vector $y_1, y_2,.., y_p$ will be used. For example, for *S4* (n, m)-code, the module Σ will have two outputs.

For the concrete *S4* (n, m)-code the generator is designed as follows: first, the generator for $S(n, m)$-code is built and then its structure is simplified. For example, in circuit in fig. 4 there is no need in variable y_3, therefore the module FA is replaced by two elements M2 (each of them performing the function of nonequivalence or parity); generator for *S4* $(8,6)$-code is shown in fig. 5.

Characteristics, that are very important for a generator, are complexity and high-speed performance [8]. Complexity depends on the way of realization of Σ, which also consists of modules FA and HA. It is offered to determine complexity L as the number of logic elements with two inputs excluding inversions on its pins. It is efficient to use circuits of FA and HA offered in [6], where $L(FA) = 8$, $L(HA) = 3$ and $L(M2) = 3$. High-speed performance is determined by the longest path of the circuit, i. e. by the number of elements that are to come into action for transmitting the signal from inputs to outputs.

The structure of $S(n,m)$-code generator is designated as $g(m \rightarrow k)$ and is shown in [6]. It consists of up to two generators of lower degree (which, in their turn, consist of elementary generators) and block Σ. For example, FA is an elementary generator $g(3 \rightarrow 2)$. Structures of $S(n,m)$-code generators with minimum complexity were developed in [6] and some of them are presented in tab. 1. For example, "generator 11→4 with formula 7+3+1" means, that it consists of one generator with 7 inputs, one generator with 3 inputs and one single input connected directly to Σ (fig. 6). In its turn, generator 7→3 has formula

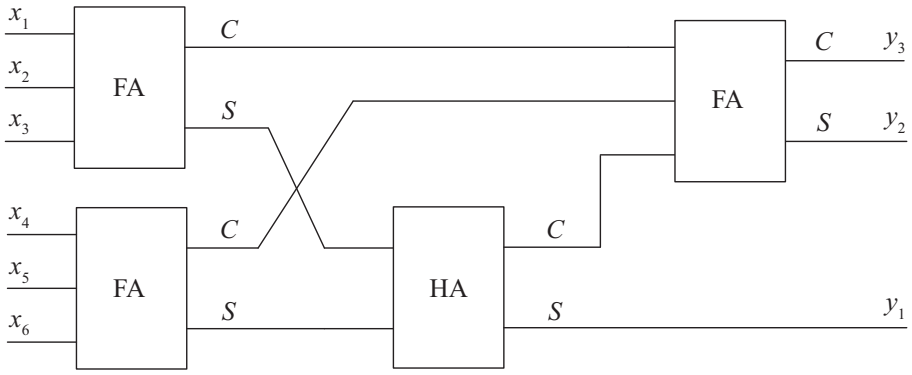

Fig. 4. Generator for *S*(*9*,*6*)-code

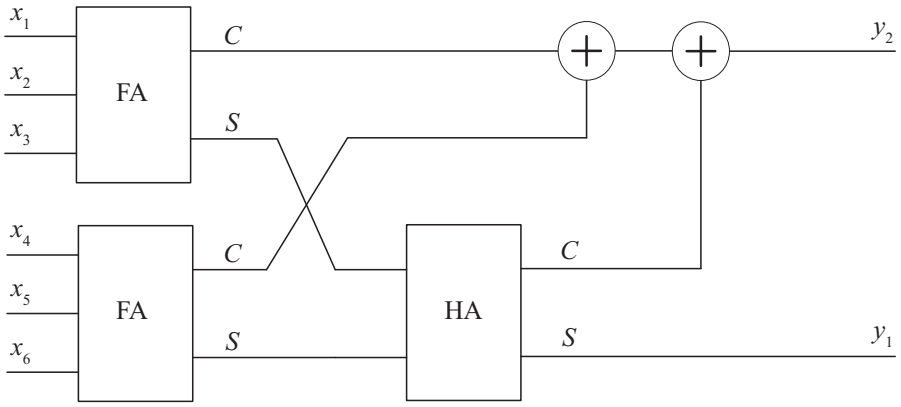

Fig. 5. Generator for *S4*(*8*,*6*)-code

TABLE 1. Formulas for generators *g* (*m*→*k*)

Generator type	Formula	Generator type	Formula
4→3	3+1	11→4	7+3+1
5→3	3+1+1	12→4	6+5+1
6→3	3+3	13→4	7+5+1
7→3	3+3+1	14→4	7+7
8→4	7+1	15→4	7+7+1
9→4	7+1+1	16→5	15+1
10→4	7+3	17→5	15+1+1

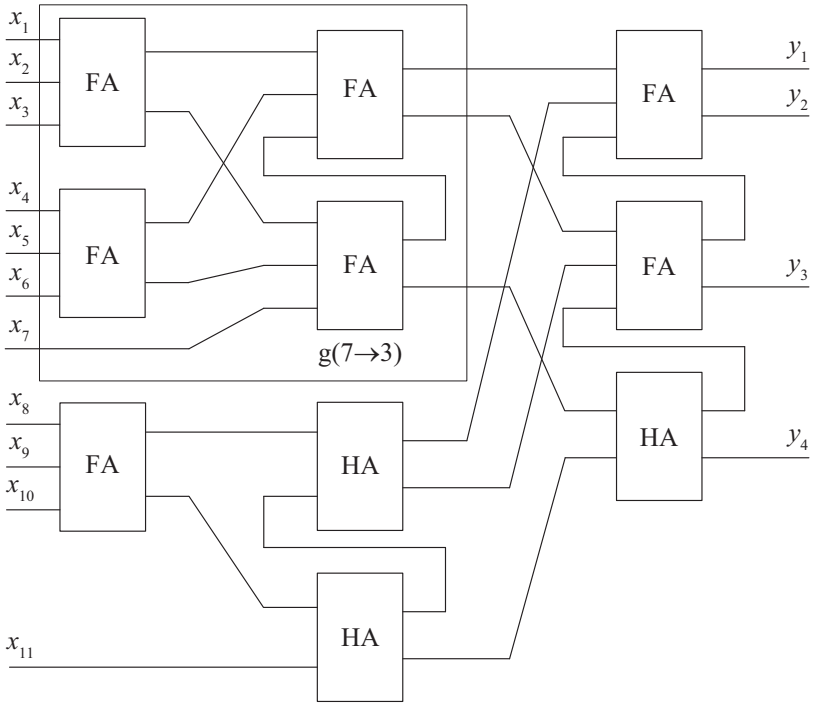

Fig. 6. Generator *g* (*11→4*)

3+3+1. The more "sub-generators" are used, the more "levels" the circuit has and will perform with the lower speed.

2 Generators for modulo codes

As to *S4*(*n*, *m*)-codes, the similar approach may be used. All the possible structures have been considered and those with less complexity are presented in tab. 2. In some cases there are more than one formula with the least complexity. For example, *g*(*6→2*) may have the form of four different modifications. Generator structure [3+3] coincides with the one shown in fig. 5, where HA and two M2 modules form adder Σ by modulo 4, which sums two two-bit binary values, and every FA is a generator *g*(*3→2*).

Generator *g*(*6→2*) [4+1+1] is shown in fig. 7. Two adders Σ(M4) included in its structure sum one two-bit and two one-bit binary values. Generator *g*(*4→2*) is based on formula [2+1+1]. Therefore, the structure of this generator may be presented as [2+1+1]+1+1, or as a diagram given below:

$$\overbrace{4+1+1}^{\downarrow}$$
$$2+1+1$$

One important property follows from tab. 2: for every *m* there is a structure with formula [(*m*–2)+1+1], which has the least complexity. Due to this fact it is possible to offer the standard structure of generator *g*(*m→2*). Such structure for even *m* is shown in fig. 8. To modify it for odd *m* HA with inputs x_1, x_2 should be replaced by FA with inputs x_1, x_2, x_3.

Complexity of such structures may be counted as:

$$L = L(HA) + \frac{m-2}{2} L(\Sigma(M4)) \text{ – for even } m,$$

or

$$L = L(FA) + \frac{m-3}{2} L(\Sigma(M4)) \text{ – for odd } m.$$

Conclusion

Design of generators for codes with summation is of widely applied importance. Al-

TABLE 2. Formulas and complexities of generators $g(m\rightarrow 2)$

Generator type	Formula	L	Generator type	Formula	L
4→2	3+1 2+1+1	14	11→2	5+5+1 7+3+1 9+1+1	52
5→2	3+1+1	19	12→2	6+5+1 9+2+1 11+1 10+1+1	58
6→2	3+3 3+2+1 5+1 4+1+1	25	13→2	7+5+1 11+1+1	63
7→2	3+3+1 5+1+1	30	14→2	7+7 13+1 12+1+1	69
8→2	4+3+1 5+2+1 7+1 6+1+1	36	15→2	7+7+1 13+1+1	74
9→2	7+1+1	41	16→2	8+7+1 15+1 14+1+1	80
10→2	5+5 7+3 7+2+1 9+1 8+1+1	47	17→2	15+1+1	85

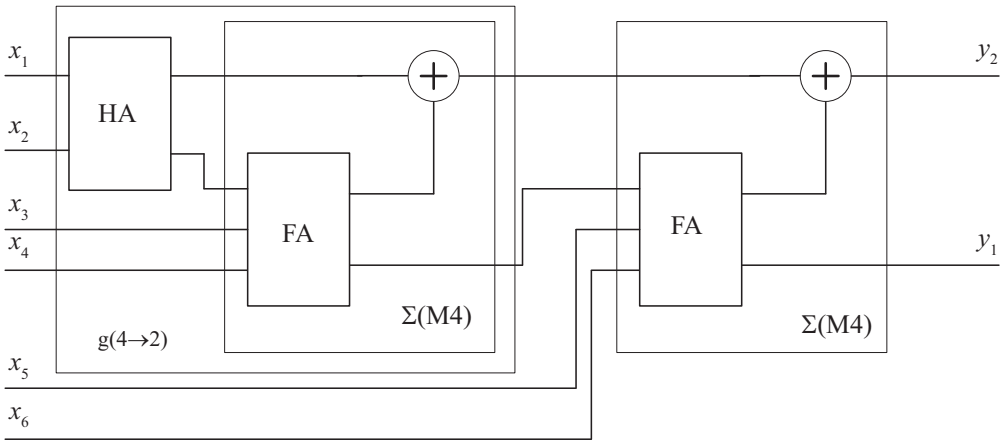

Fig. 7. Generator $g(6\rightarrow 2)$ [4+1+1]

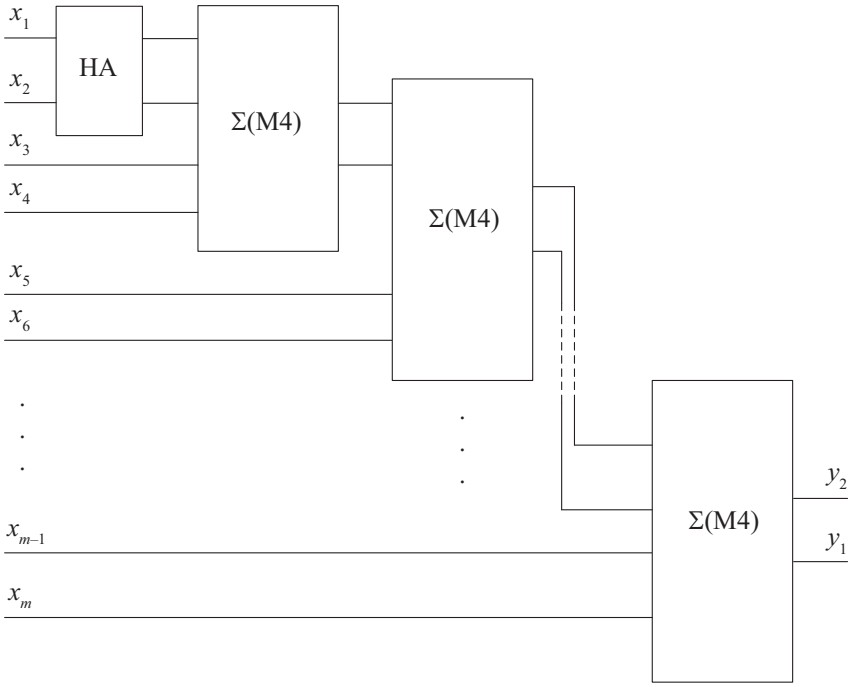

Fig. 8. Standard structure of generator $g(m \rightarrow 2)$

gorithmization of their development process reduces the probability of errors in it and saves time of the designer.

References

1. **A note** on error detecting codes for asymmetric channels / J. M. Berger // Information and Control. – 1961. – 4, № 3. – P. 68–73.

2. **Self-checking** digital devices / V. V. Sapozhnikov, Vl. V. Sapozhnikov – St. Pb. : Energoatomizdat, 1992. – 224 p. – ISBN 5-283-04605-2.

3. **Method** of constructing testers of code vectors / V. V. Sapozhnikov, Vl. V. Sapozhnikov, D. I. Urganskov // Electronic modeling. – 2000. – № 6 (22). – P. 66–76.

4. **Universal** structures of binary counters of ones by free counting modulo / V. V. Sapozhnikov, Vl. V. Sapozhnikov, D. I. Urganskov // Electronic modeling. – 2002. – № 4 (24). – P. 65–81.

5. **Design** of Self-Checking Checkers for Berger Codes / M. A. Marouf, A. D. Friedman // Proc. 8th Annual Intern. Conf. on Fault-Tolerant Computing. – Toulouse, France. – 1978. – P. 179–183.

6. **Investigation** of combinational self-checking devices with independent and monotonically independent outputs / M. Goessel, A. A. Morozov, V. V. Sapozhnikov, Vl. V. Sapozhnikov // Automation and remote control. – 1997. – № 2. – P. 180–193.

7. **Theoretical** studies of sum binary modular codes / A.A. Blyudov // Bulletin of scientific research result: electronic scientific publication. 2011 [Electronic resource]. – Edition 1. System requirements: Adobe Acrobat Reader. URL: http://sampgups.ru/brni/index.php (Date accessed: 11.11.2011).

8. **Characetristics** of self-checking testers for equilibrium codes / V. V. Sapozhnikov, Vl. V. Sapozhnikov, A. F. Shpak // Electronic modeling. – 1989. – № 5. – P. 39–44.