

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing the burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302 and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
----------------------------------	----------------	----------------------------------

4. TITLE AND SUBTITLE OF REPORT <i>Modular implementation of efficient self-checking checkers for the Berger code.</i>	5. FUNDING NUMBERS <i>Grant No. N00014-94-1-0462</i>
---	---

6. AUTHOR(S) <i>P.K. Lala & A. Walker</i>
--

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <i>North Carolina A&T State University Dept. of Electrical Engineering Greensboro, NC 27411.</i>
--

8. PERFORMING ORGANIZATION REPORT NUMBER:

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) <i>Office of Naval Research</i>
--

10. SPONSORING/MONITORING AGENCY REPORT NUMBER:

11. SUPPLEMENTARY NOTES:

12a. DISTRIBUTION AVAILABILITY STATEMENT UNLIMITED
--

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words) <i>The Berger code is the least redundant systematic code available for detecting single and unidirectional multibit errors. In this report, a technique for designing efficient checkers for the Berger code is presented.</i>

14. SUBJECT TERMS 19960625 168
--

15. NUMBER OF PAGES: <i>20</i>

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT:
--

18. SECURITY CLASSIFICATION OF THIS PAGE
--

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT

Progress Report

Grant No: N00014-94-1-0462

Office of Naval Research

~~Self-Checking State Machine Realization in CMOS~~

Reporting Period : January 1 --- March 31, 1996

Dr. P.K. Lala

Principal Investigator

Dr. A. Walker

Co-Principal Investigator

Department of Electrical Engineering

North Carolina A&T State University

Greensboro, NC 27411.

Modular Implementation of Efficient Self-Checking Checkers for the Berger Code

The Berger code is the least redundant systematic code available for detecting single and multibit unidirectional errors [1]. The number of check bits, k , for a conventional Berger code is determined by the equation: $k = \lceil \log_2 (I + 1) \rceil$, where I is the number of information bits. There are two encoding schemes, B0 and B1, used to encode information in conventional Berger code. In the B0 encoding scheme, the check bits are the binary representation of the number of 0's in the information vector, whereas in the B1 encoding scheme, the check bits are the bit by bit complement of the binary representation of the number of 1's in the information vector. If the number of information bits is related to the number of check bits by the relationship, $I = 2^k - 1$, then the resulting code is called the maximal length Berger code.

In this report, a technique for designing efficient checkers for Berger codes is presented. Figure 1 shows the block diagram of the checker. Information bits are passed through the check bit generator. The resulting check bits are then compared to the predicted check bits by a two-rail checker. If there is no fault present in the two-rail checker or the check bit generator, the outputs of the checker are complementary (i.e. 01 or 10), otherwise they are 00 or 11. We will focus only on the implementation of the check bit generator (CBG) circuit. The CBG is designed assuming the B1 encoding scheme for non-maximal length Berger codes, and the B0 encoding scheme for maximal length Berger codes. The basic building block of the proposed CBG is a 4 input 1's counter [see Figure 2]. This 4-input 1's counter receives four inputs and generates at its three output bits the binary representation of the number of 1's in the four input bits.

Two other types of 1's counters, having 2 and 3 inputs, are used to realize checkers for variable length information bits. Figure 3 shows the schematics of these counters. A 1's counter of any size can be constructed from the basic counters. For example, a 5 input 1's counter can be constructed from a 3 input 1's counter, a 2 input 1's counter and a standard 2-bit adder.

A CBG is implemented with two 1's counters and an addition array. The information bits are partitioned into two blocks I_r and I_{r^*} . The I_r block contains $\lceil I/2 \rceil$ bits and the I_{r^*} block contains $I - \lceil I/2 \rceil$ bits. The size of the 1's counters needed to implement a CBG depends on the number of bits in blocks, I_r and I_{r^*} . If the number of information bits in a block are 4 or fewer, an appropriate 1's counter (shown in Figs. 2, 3(a) and 3(b)) in conjunction with the appropriate addition array are used to complete a design. However, if the number of bits in a block is greater than 4, the block is partitioned into sub-blocks. The partitioning process is continued until a sub-block can be implemented with a pre-designed 1's counter and an addition array. The counter corresponding to a sub-block is used as a building block to implement larger size 1's counters. The number of 1's in blocks I_r and I_{r^*} derived by partitioning I information bits is represented by vectors r and r^* , respectively. The number of bits in r and r^* is $\log_2 (\lceil I/2 \rceil + 1)$ and $\log_2 ((I - \lceil I/2 \rceil) + 1)$, respectively. The check bits, k , for I information bits is the sum of r and r^* , and is derived by adding two bits at a time as shown below:

$$\begin{array}{cccccccc}
 r & = & r_n & r_{n-1} & \dots & r_3 & r_2 & r_1 & r_0 \\
 r^* & = & r^*_n & r^*_{n-1} & \dots & r^*_3 & r^*_2 & r^*_1 & r^*_0 \\
 & & \text{-----} & & & \text{-----} & & \text{-----} & \\
 k & = & k_n & k_{n-1} & \dots & k_3 & k_2 & k_1 & k_0
 \end{array}$$

If there are 2^n (where n is an integer) inputs to a block, the block produces a vector r (or r^*) whose most significant bit is considered to be a *special* most significant bit (SMSB). If the SMSB of r or r^* is 1, it is the only '1' present in the vector, and a carry is not produced when r and r^* are added.

To illustrate, let us assume:

$I=7$;
then $I_r = 4$; and $I_{r^*} = 3$;
length of r : 3 bits $\Rightarrow r_2 r_1 r_0$
length of r^* : 2 bits $\Rightarrow r^*_1 r^*_0$

If a vector consisting of all 1's is present at the inputs to the 4 input counter ($I_r = 4$), the most significant bit of r , i.e. r_2 , will be 1 and r_1 and r_0 will be 0. Irrespective of the values of r^*_1 and r^*_0 , summing them with r_1 and r_0 will never produce a carry.

The two least significant bits of vectors r and r^* (i.e. r_0, r^*_0, r_1 , and r^*_1) are grouped together in a set called a *primary* group. A 2-bit adder is used to sum these bits. The outputs of the adder become the least significant check bits, i.e. k_1 and k_0 . The carry-out generated during this addition generally propagates to a secondary group. In some cases, the carry becomes the check bit k_2 . To illustrate, let us assume:

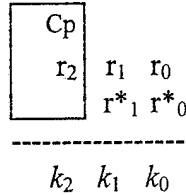
$I \Rightarrow 10110$. $I_r \Rightarrow 101$; $I_{r^*} \Rightarrow 10$; therefore, $r \Rightarrow r_1 r_0 = 10$; $r^* \Rightarrow r^*_1 r^*_0 = 01$.

The check bits, k , are obtained by adding r and r^* as shown in Figure 4. The special 2-bit adder is a derivative of the standard 2-bit adder used in all other CBG designs (Figure 5(a)). Since $r_1 r^*_1 = 11$ does not occur in a CBG design for $(I,k) = (5,3)$, gates with bits $r_1 r^*_1$ in Figure 5(a) are discarded to form the special 2-bit adder (Figure 5(b)). The special 2-bit adder is used in a sub-block which has five information bits. Otherwise, as a default, the standard 2-bit adder is used to sum the bits in the primary grouping.

A *secondary* group, unlike the primary group, may contain as many as 4 bits or as few as 1 bit. Maximal length information bits ($I = \{7,15,31,\dots\}$) will result in an odd number of bits in the secondary group whereas non-maximal length information bits will result in an even number of bits. The type of adder to be used to sum the bits of the secondary group is dependent not only upon the number of bits in the secondary group but also whether or not the most significant bit of r (or r^*) is a SMSB. When the information bits is non-maximal, the most significant bit of r (or r^*) may or may not be a SMSB. However, the most significant bit of r is always a SMSB if the information bits are maximal.

We will consider several cases with different numbers of bits in the secondary group. For simplicity, only a single secondary group will be considered.

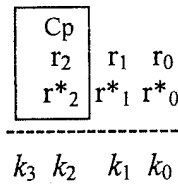
Case 1: One bit in the secondary group.



Note that this case results only when I=7; therefore, r₂ is a SMSB. The carry circuit of the standard 2-bit adder (Figure 5(a)) is modified to accommodate r₂ and is shown in Figure 7. The sum circuit of the modified 2-bit adder is the same as that of the standard 2-bit adder. The CBG block diagram for (I,k) = (7,3) is shown in Figure 6. To illustrate, let us assume:

$$I \Rightarrow 1111101; I_r \Rightarrow 1111; I_{r^*} \Rightarrow 101; \text{ therefore,} \\ r = 100; r^* = 10; k = 110.$$

Case 2: Two bits in the secondary group.



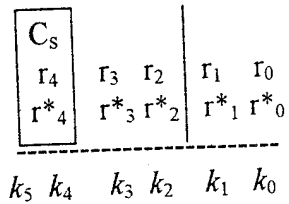
If bits, r₂ and r*₂, are SMSB's [Figure 8], a Type I adder is used to generate check bits, k₃ and k₂. Otherwise, a full adder is used. Figures 9 and 10 show the realizations of the Type I adder and the full adder, respectively.

Let us illustrate Case 2 by designing a CBG for (I,k) = (11,4):

$$I_r = 6 (\lceil I/2 \rceil) \text{ bits and } I_{r^*} = 5 (I - \lceil I/2 \rceil) \text{ bits; therefore } r = r_2 r_1 r_0; r^* = r^*_2 r^*_1 r^*_0; \\ \text{ and } k = k_3 k_2 k_1 k_0.$$

The implementation of the CBG for (I,k) = (11,4) is shown in Figure 11. Note that neither r₂ nor r*₂ is a SMSB; therefore a full adder, rather than a Type I adder is used to sum the bits in the secondary group.

Case 6: Two bits (both of which are SMSB's) in the secondary group2.



Since r_4 and r^*_4 are SMSB's, a Type III adder [Figure 19] is used to generate check bits k_5 and k_4 . Note that in this case, carries, C_0 and C_1 , generated by the parallel 2-bit adder, propagate to the subsequent secondary group. Otherwise, the procedure given in Case 2 requires the full adder to sum bits C_s , r_4 , and r^*_4 . Figure 20 shows a block diagram of the CBG design for $(I,k) = (32, 6)$.

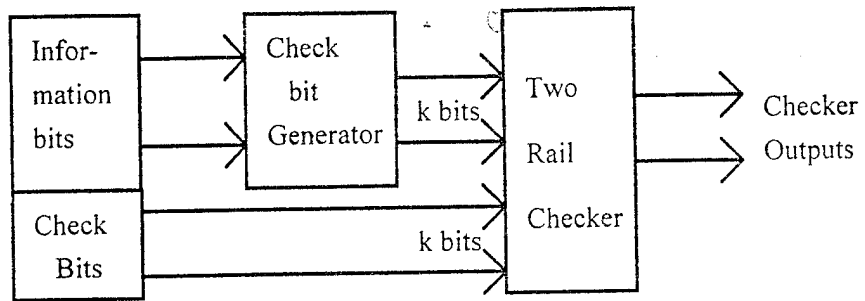


Figure 1

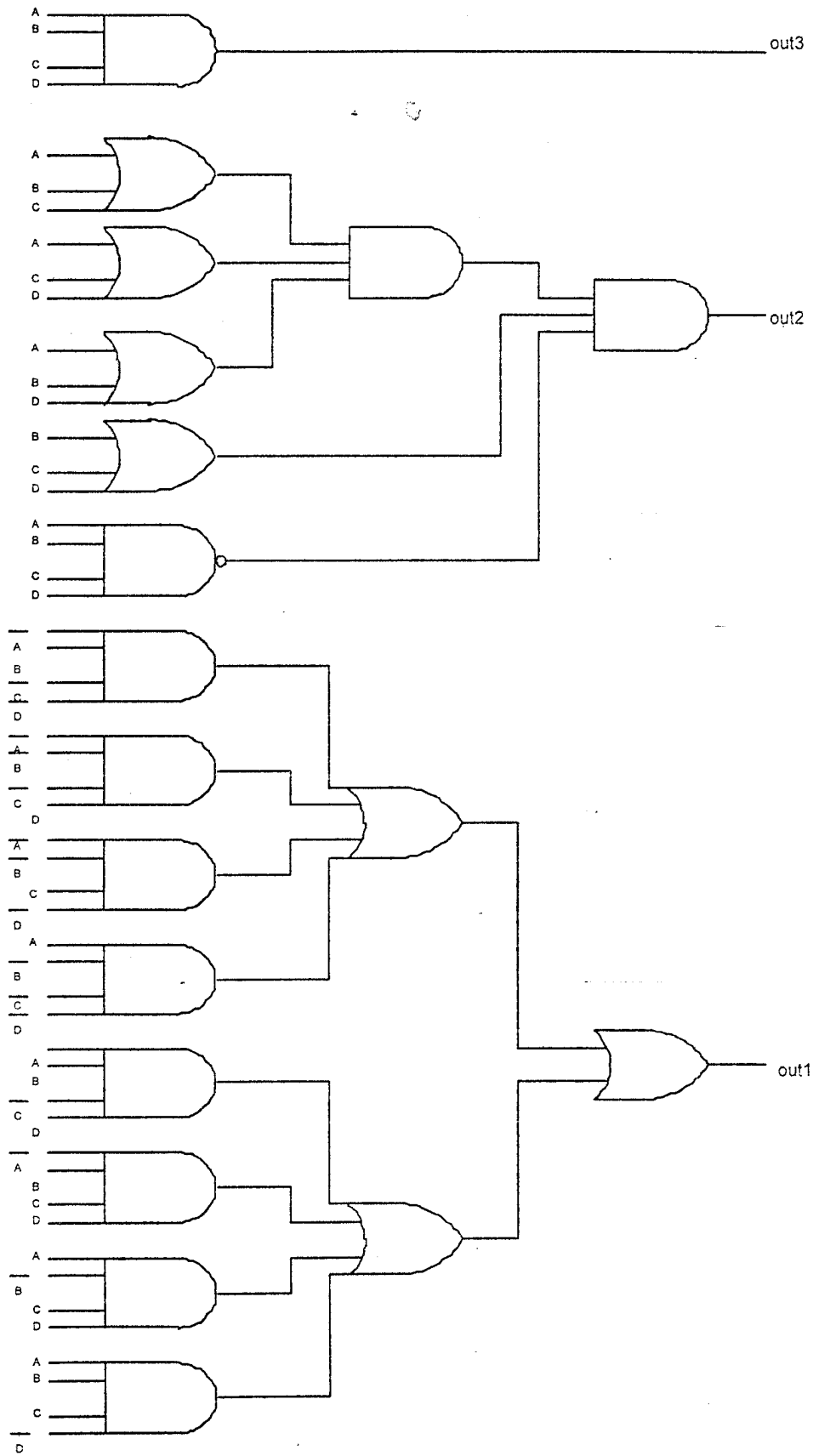


Figure 2 4 input 1's counter

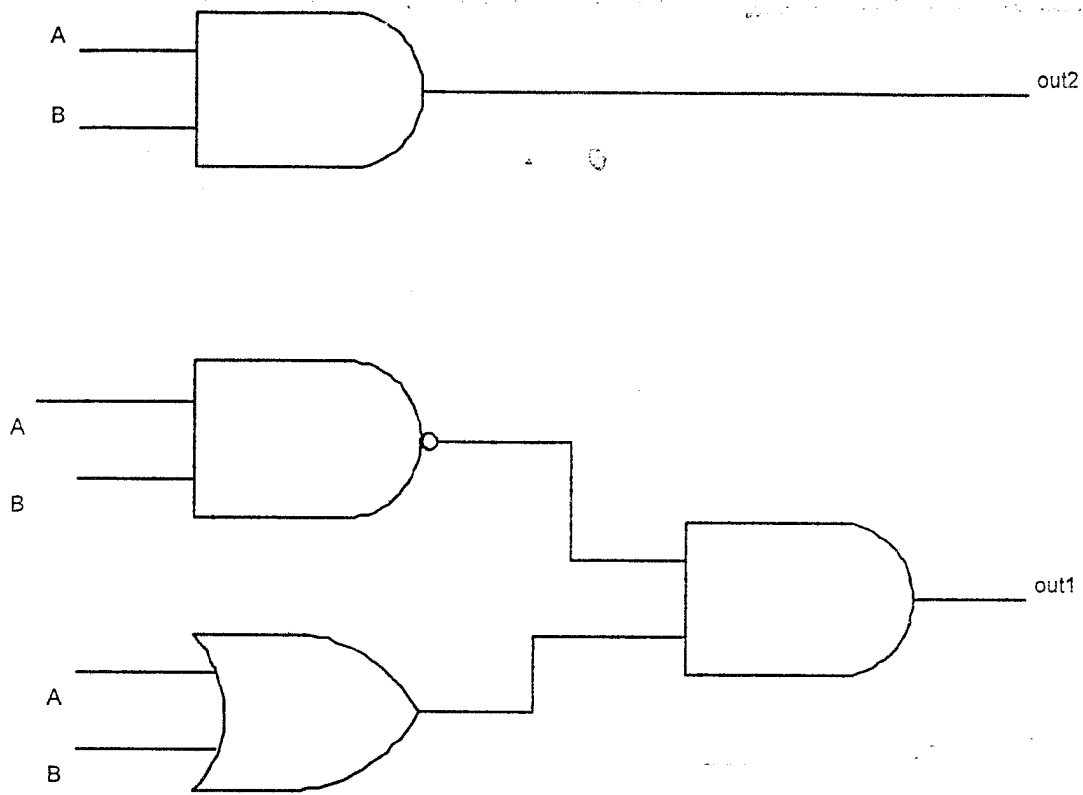


Figure 3(a) 2 input 1's counter

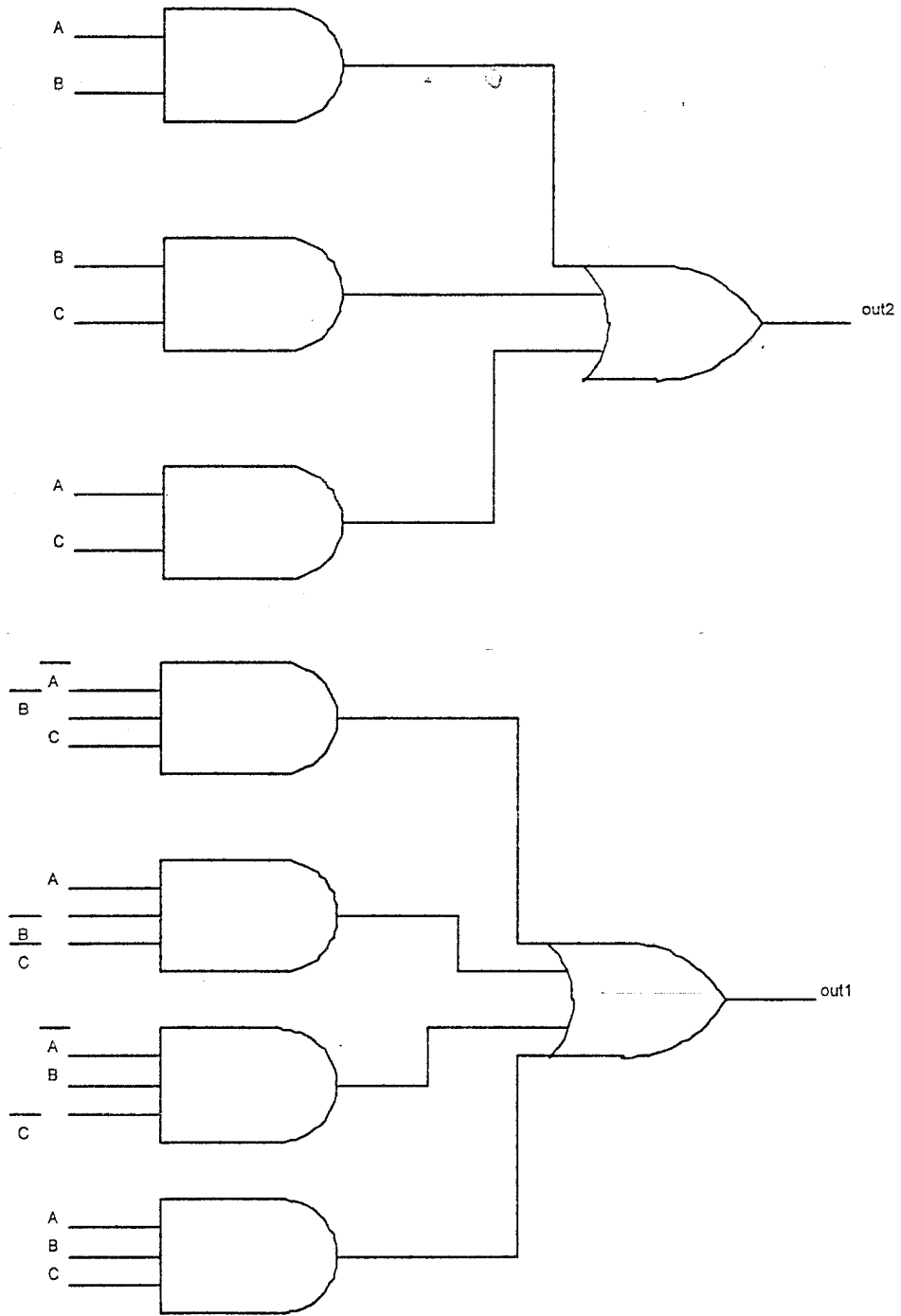


Figure 3(b) 3 input 1's counter

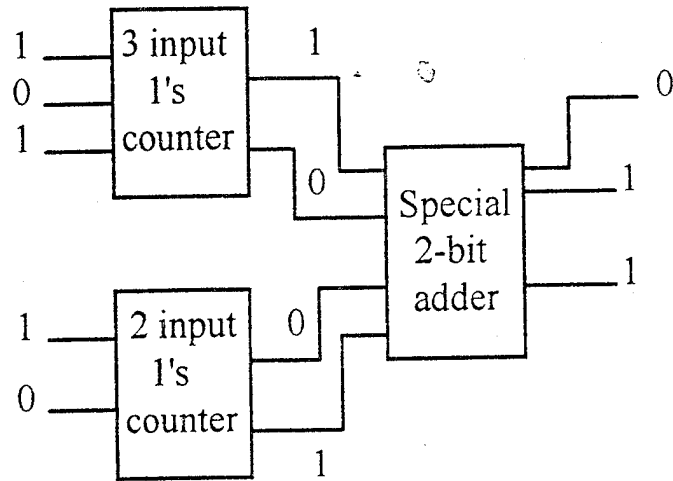


Figure 4

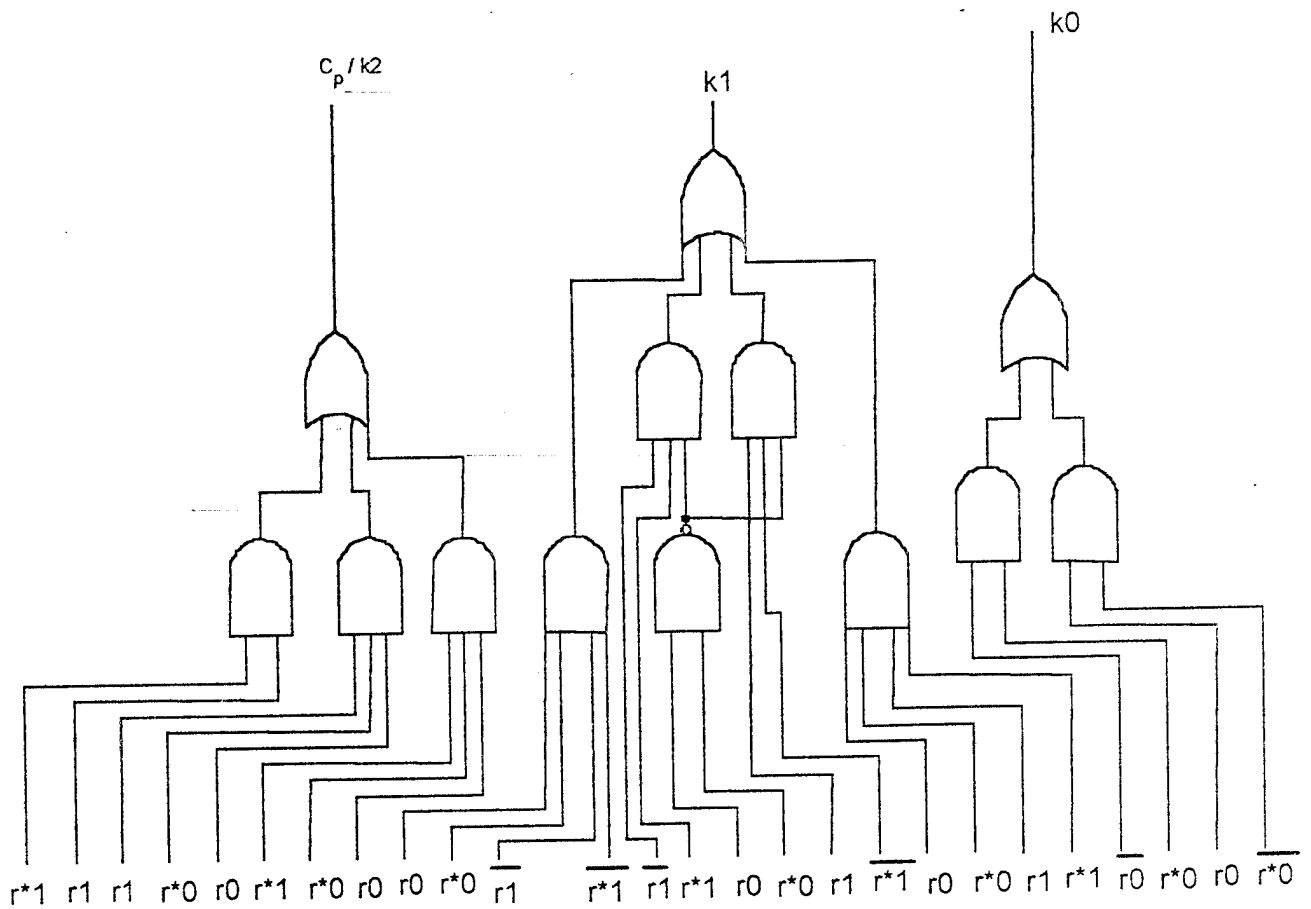


Figure 5(a)

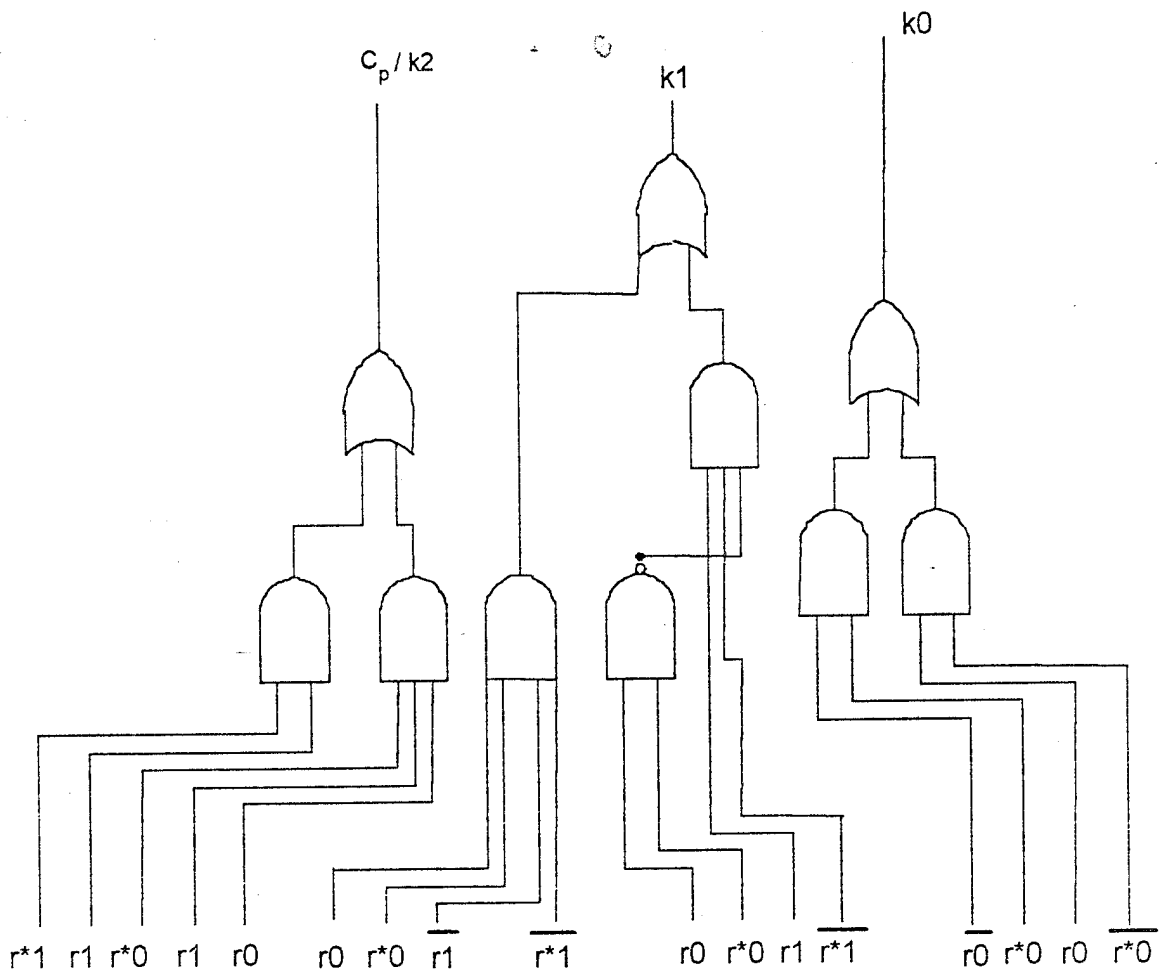


Figure 5(b)

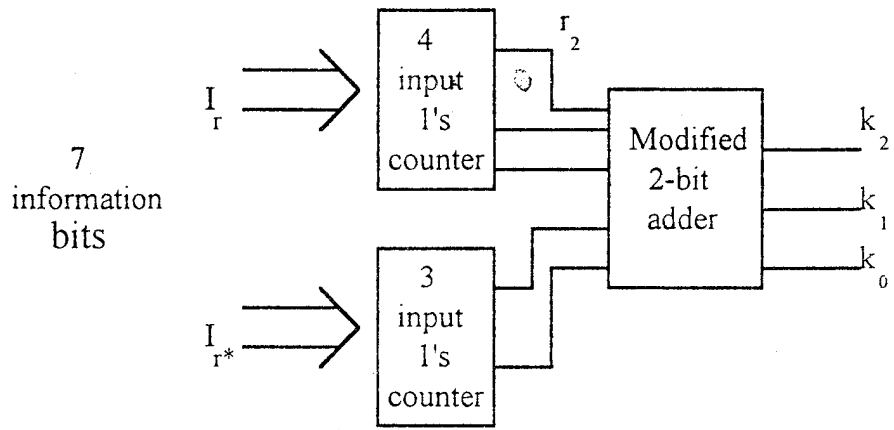


Figure 6

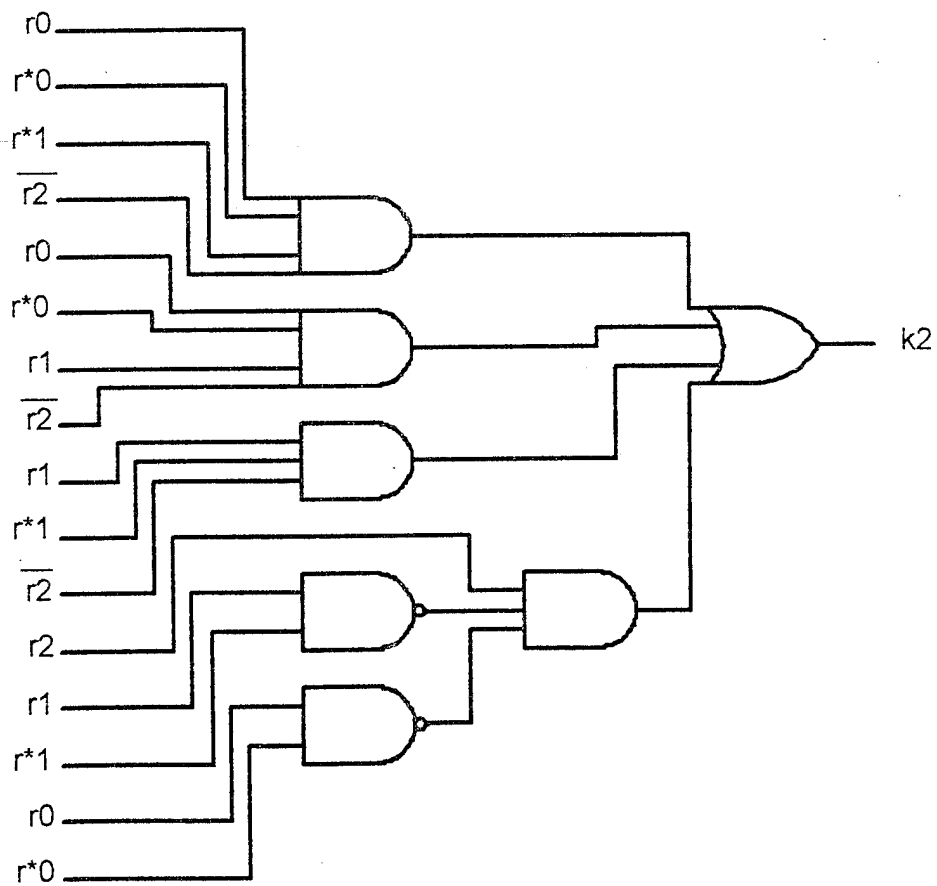


Figure 7

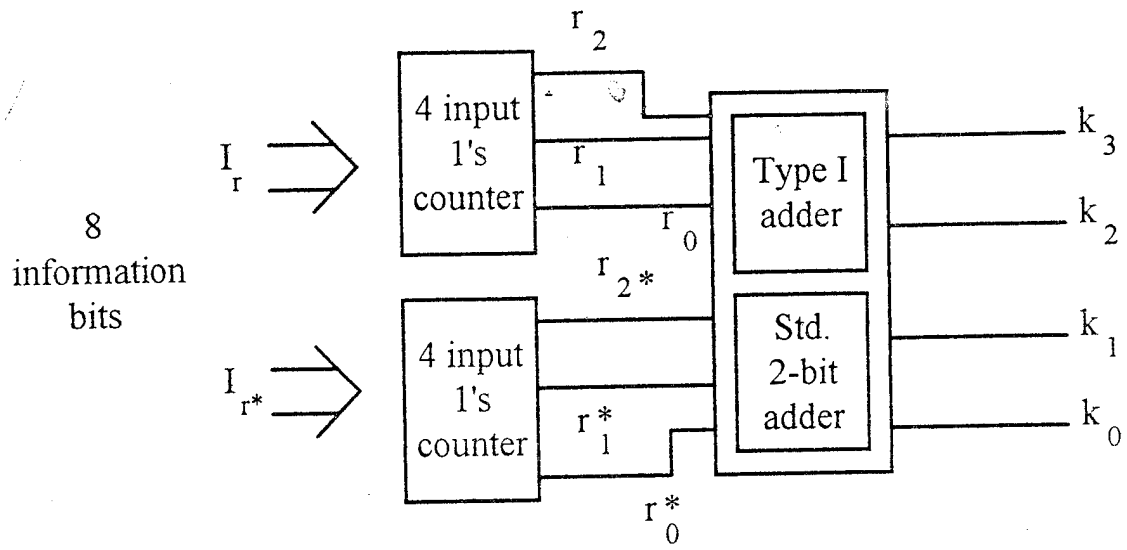


Figure 8

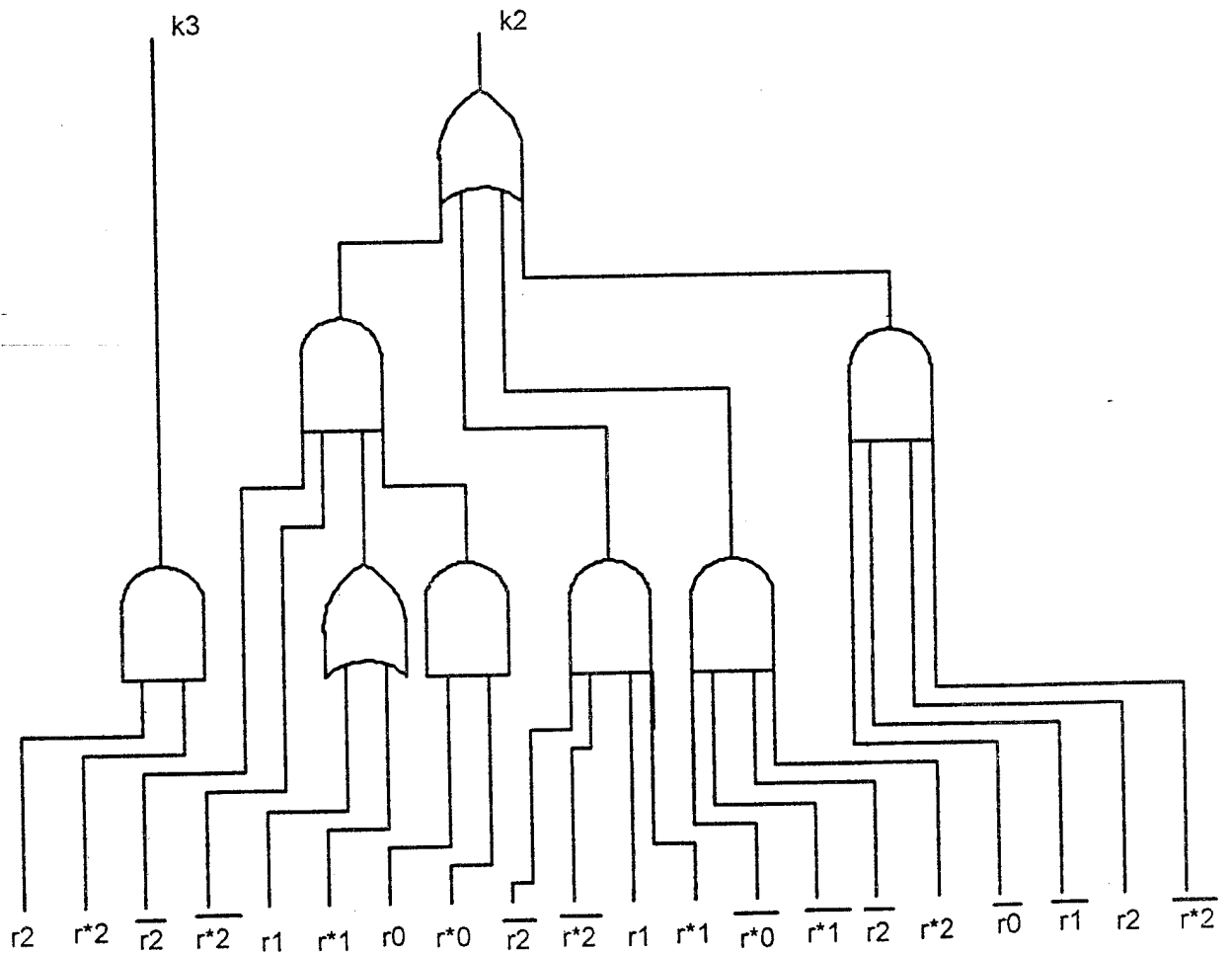


Figure 9

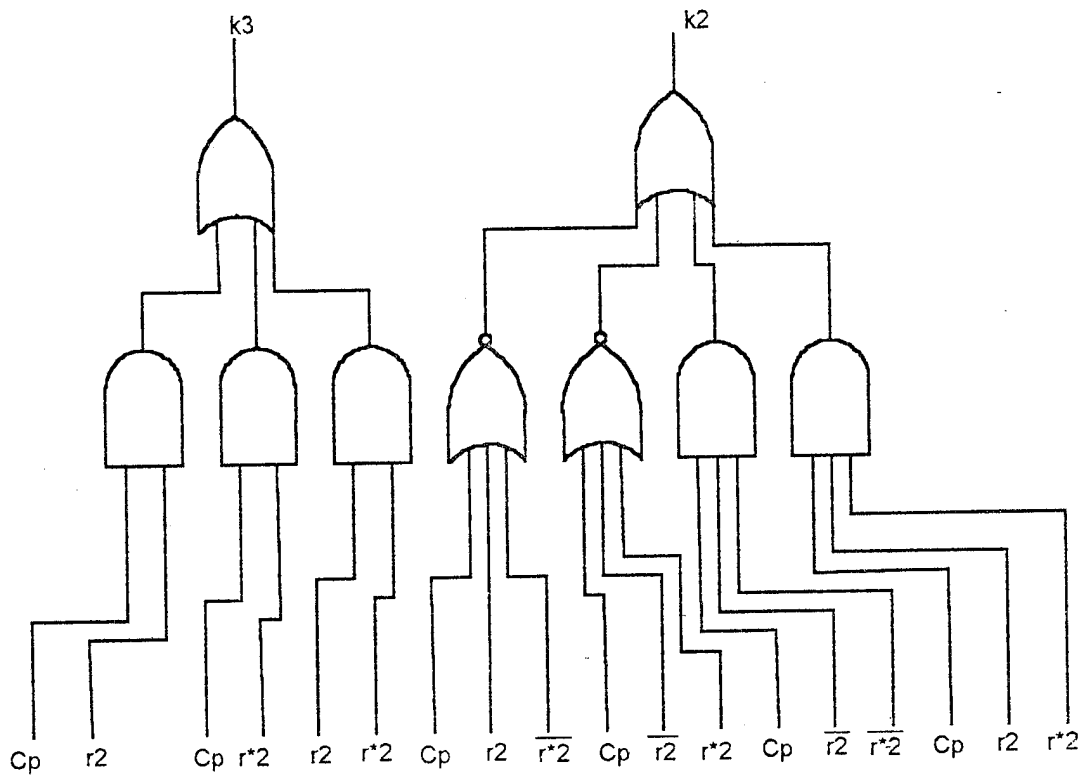


Figure 10

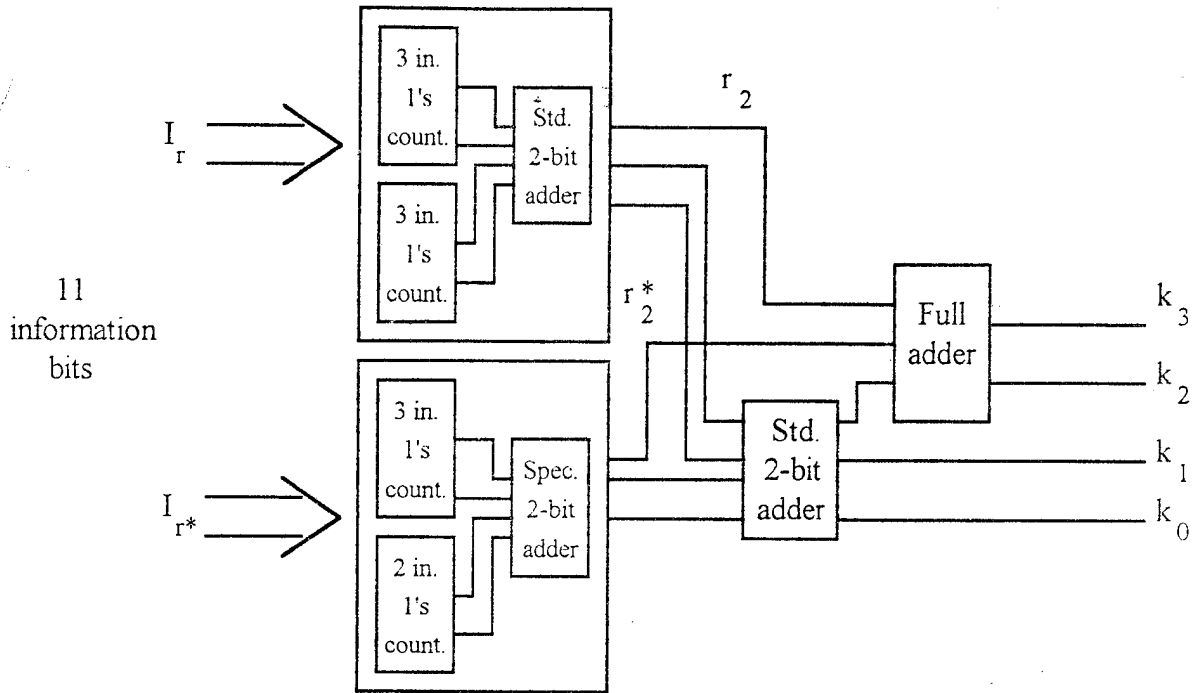


Figure 11

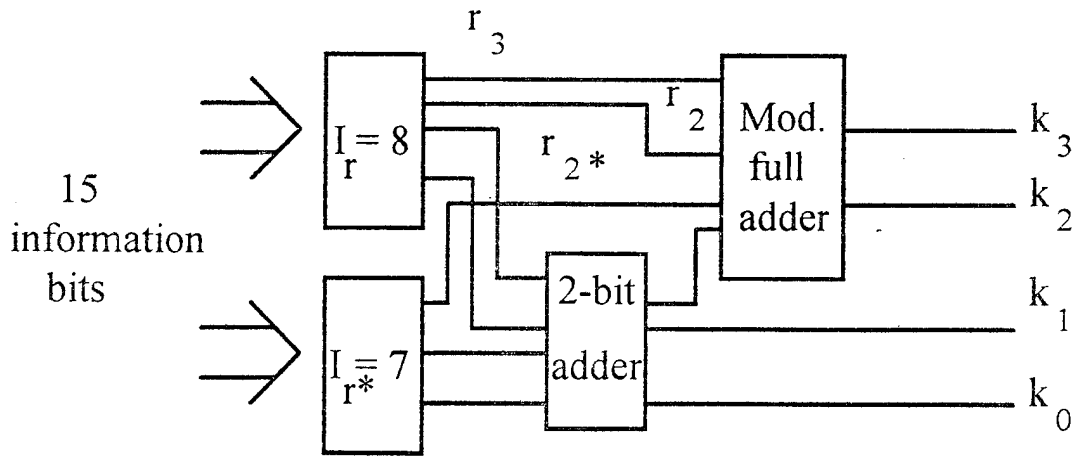


Figure 12

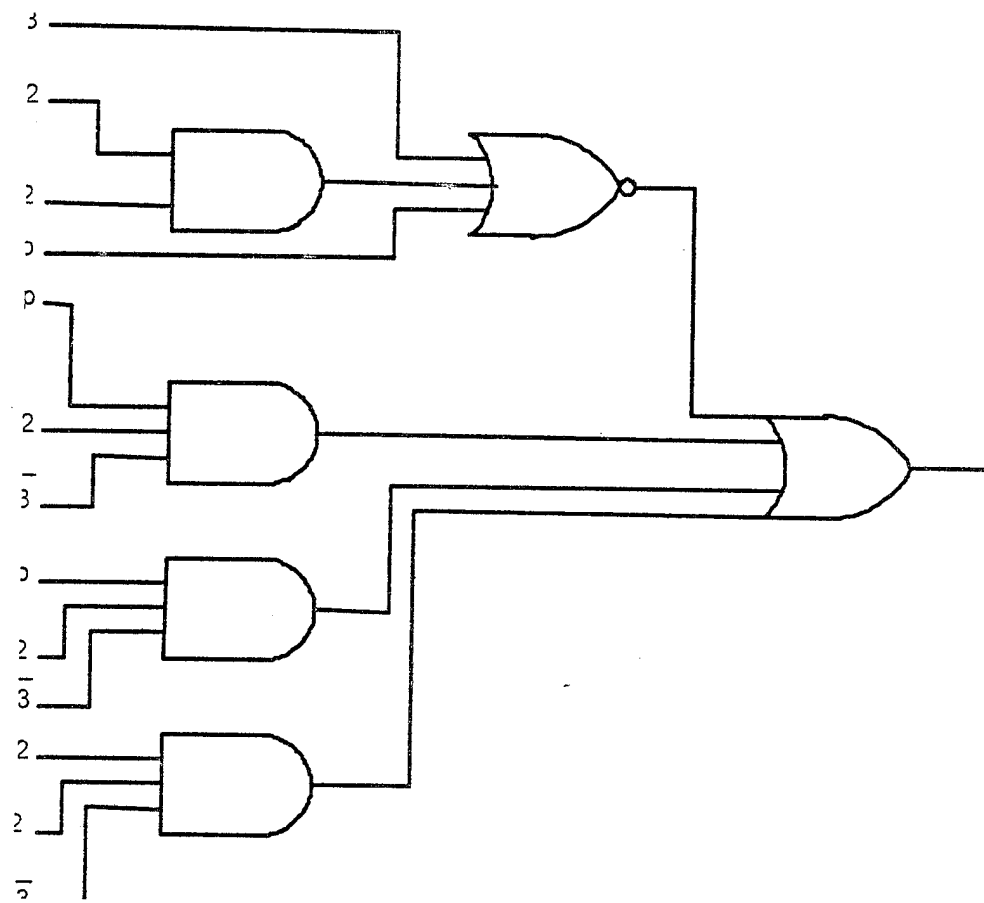


Figure 13

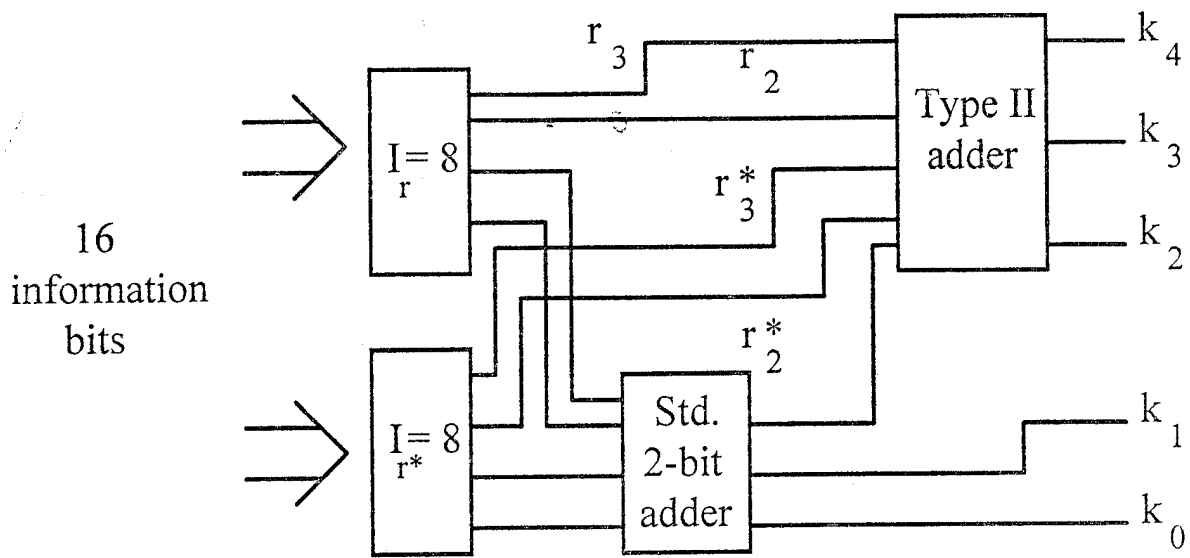


Figure 14

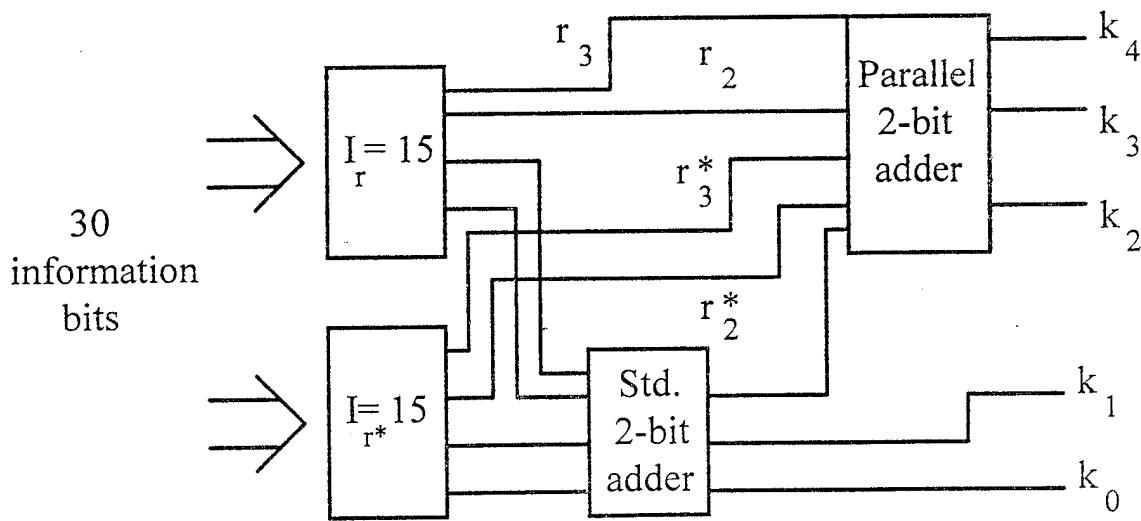


Figure 15

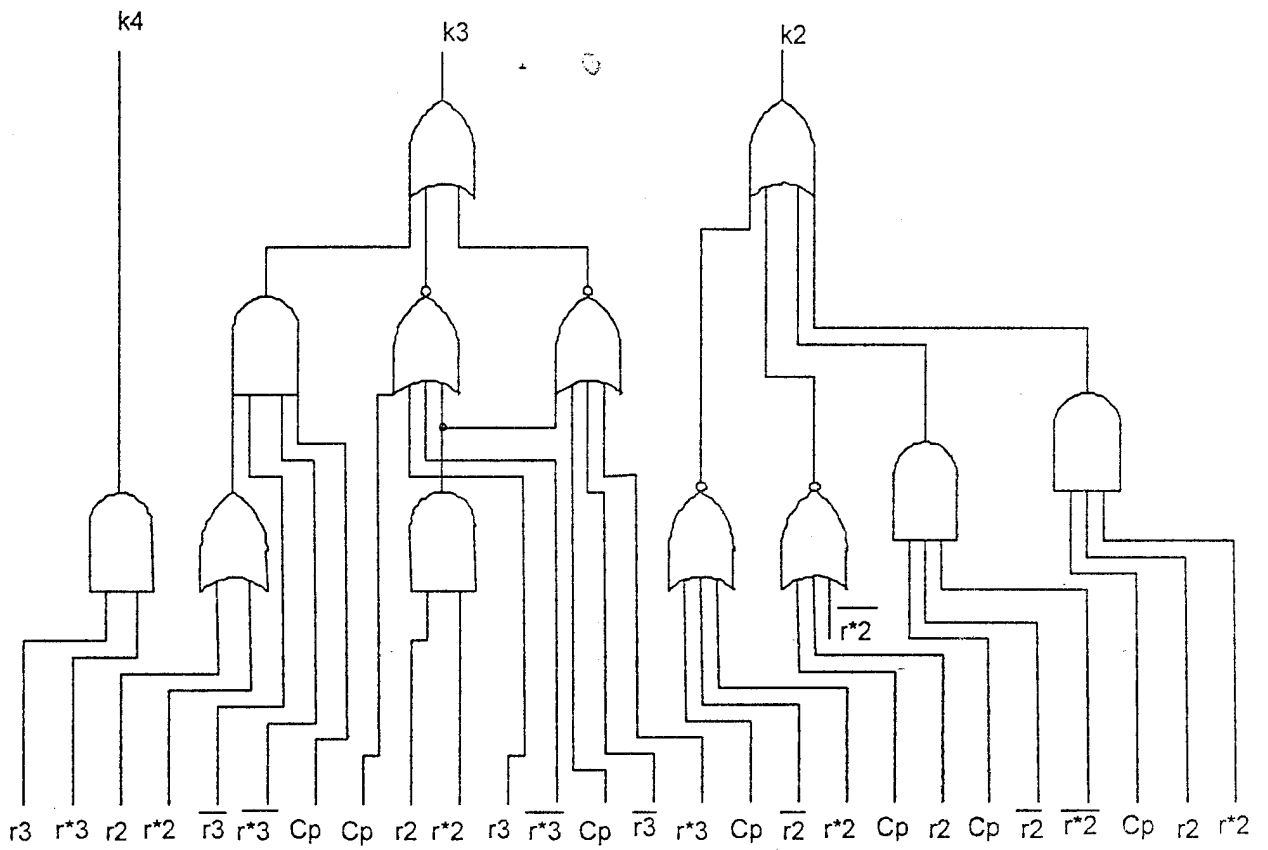


Figure 16

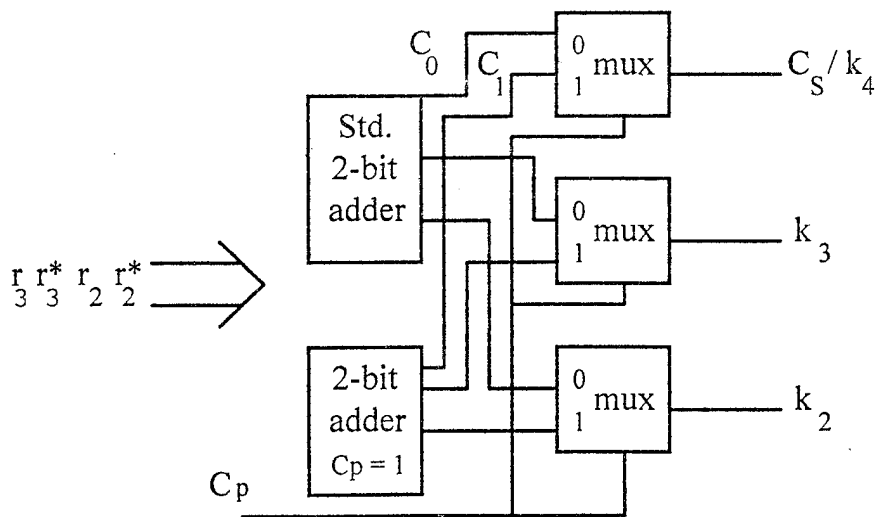


Figure 17

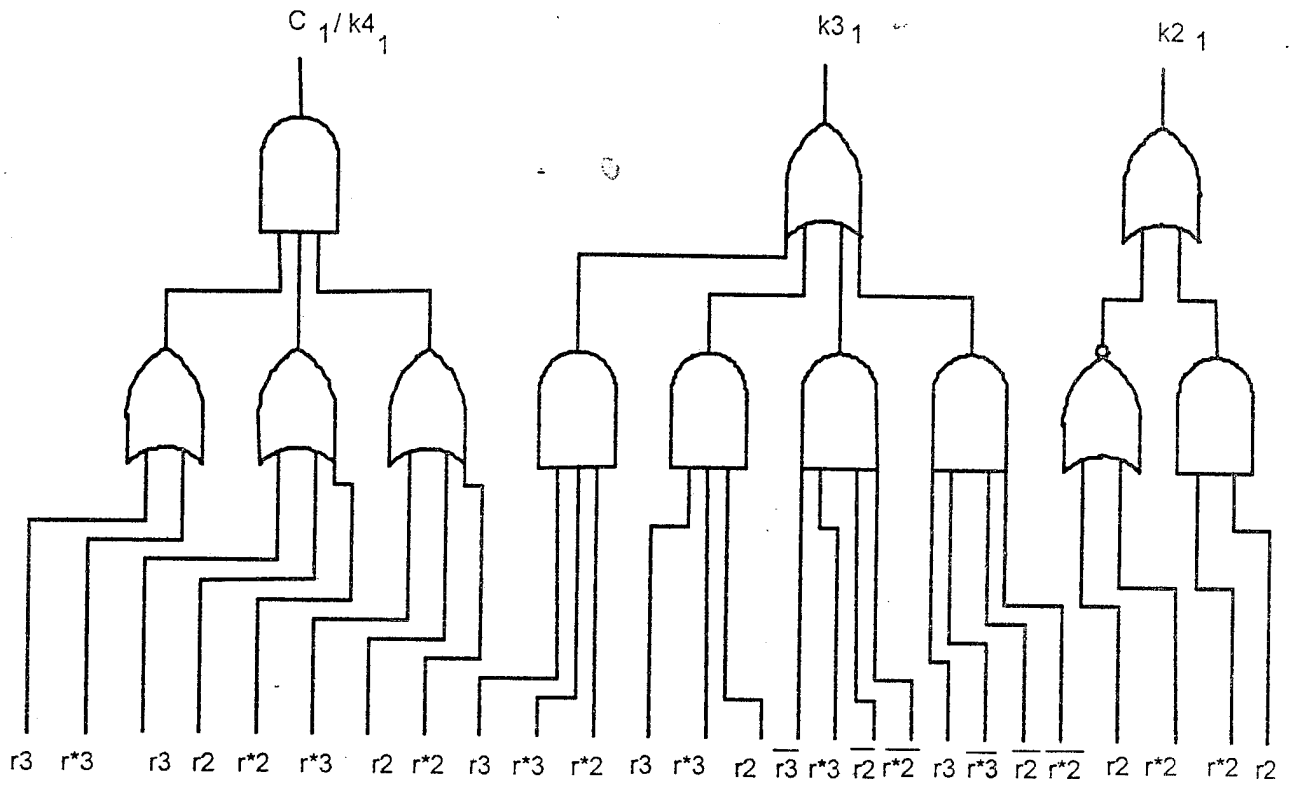


Figure 18

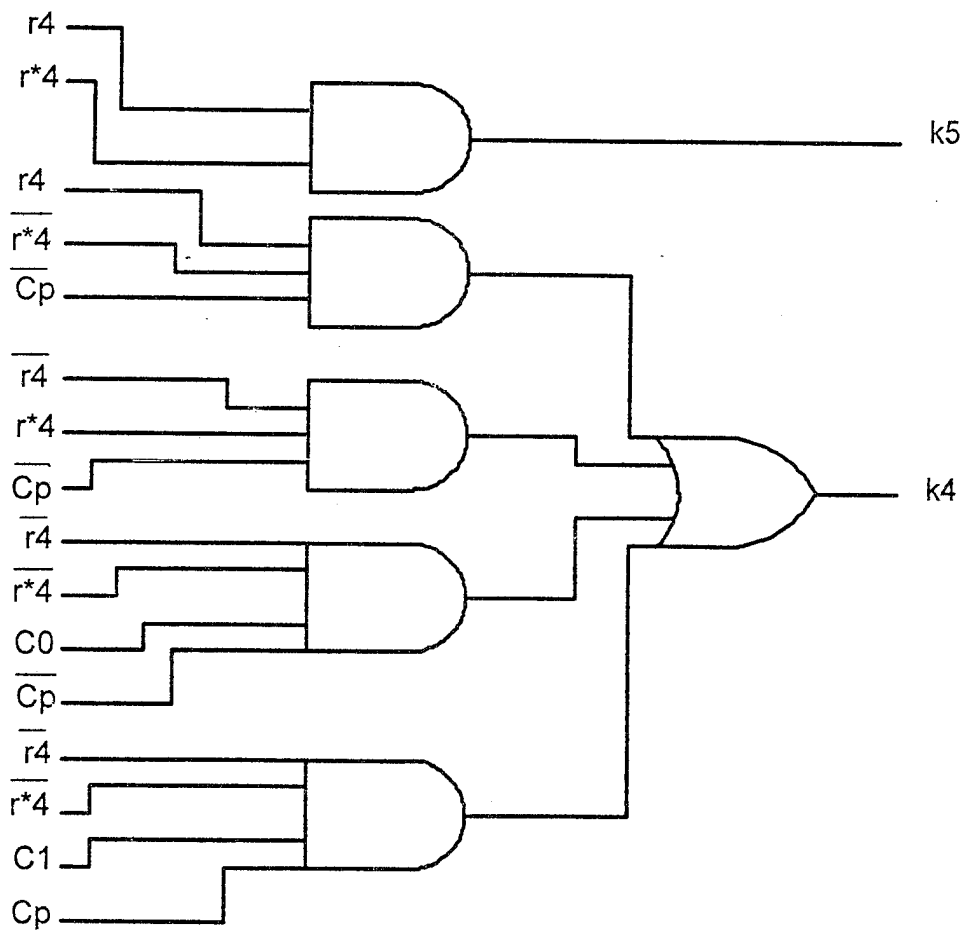


Figure 19

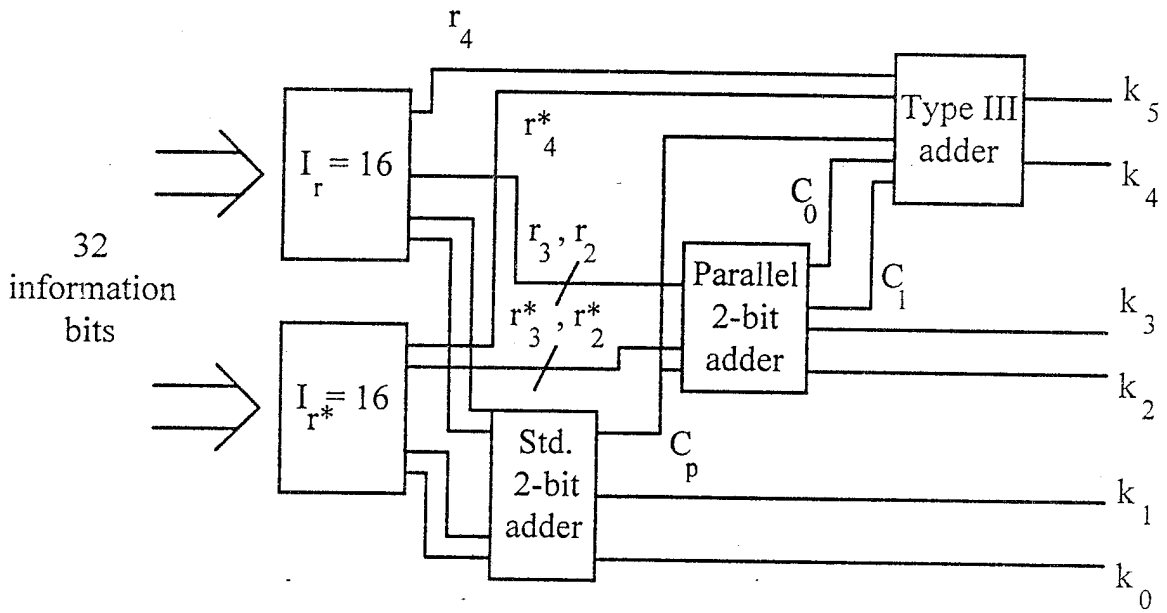


Figure 20