MULTI-OPERAND ADDITION WITH CONDITIONAL SUM LOGIC

Kemal Efe

Department of Computer Studies, the University of Leeds,
LEEDS LS2 9JT, England.

ABSTRACT

Column-wise addition schemes involve counting the number of 1's in a column and representing this number in binary weighted form. In Conditional Sum Logic (CSL) partial results are first generated for every column, then one of these results is selected depending on the incoming carry value. A compact scheme for counting the number of 1's and generating partial results for all possible distributions of incoming carries is introduced. Application of such counters to CSL yields a high speed in multi-operand addition.

INTRODUCTION

Sklansky [1] proposed a fast method to find the sum of two operands in two's complement representation. With the philosophy that the higher the parallesim the faster the operation, he produced partial sums for each column simultaneously.

This system is based on the determination of the sums and output carries for a column that can arise from all possible distributions of input carries. Then, one of these output values is selected by a multiplexer dependent on the actual carry value that is received. Due to conditional selection of the outputs, the method is called "Conditional Sum Logic" (CSL). For the addition of two n-bit binary numbers there are only two possibilities for carries coming into a column. They can be either "0" or "1". Two sum-carry pairs are generated to correspond to these incoming carries. Then one of the outputs is selected depending on the carry received by this column. If n=2 then only one multiplexing level is needed. For n>2 nested multiplexing is necessary in order to include the effect of intermediate carries. The number of multiplexer levels is calculated as $\lceil \log_2 n \rceil$.

This method can be applied to multi-operand addition by generating sum-carry pairs for each possible value of carries that can be received by a column. If there are m operands, then m different sum-carry pairs must be generated to correspond to incoming carry values of 0, 1, 2,...,m-1. Then a similar selection mechanism to that used in two operand addition may be employed. Fig. 1 shows the working of CSL for m=4 and n=4.

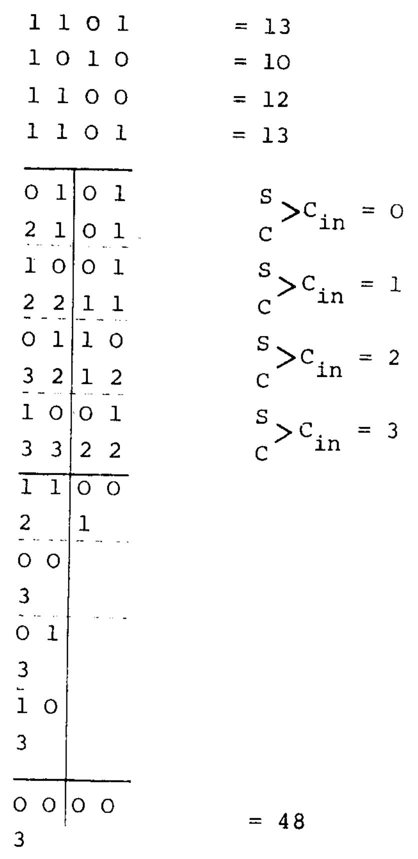


Fig. 1 4x4 Array Addition with CSL.

Implementation of this method for multiple operand addition involves counting the number of 1's in a column, then generating multiple sum-carry pairs. One way of generating partial results corresponding to all possible distributions of input carries may be by employing a seperate counter to generate every sum-carry pair in the output distribution. For addition of an

CH1630-3/81/0000/0251$00.75 © 1981 IEEE

m-bit by n-bit array m X n counters will be required to generate all possible output distributions of n columns; this is highly expensive, if not impractical even with todays VLSI technology. However if we could reduce the price down to the range of other known schemes, this method would be preferrable for the sake of speed.

In this paper, a compact scheme is proposed for producing partial results. The problem of counting the number of 1's at each column has been studied previously [2-6]. Dadda [4] has given a general theory of counters and gave examples with (7,3) (i.e. 7 input, 3 output bits) and (3,2) counters. This idea was extended to the design of general $(2^q-1,q)$ counters using q threshold gates by Irving T. Ho and Tien Chi Chen [5]. The following section of this paper summorizes the threshold gate logic and extends this scheme to produce partial sums. A counter scheme with multiple output vector is introduced. In the later sections, application of this scheme to multiple operand addition is discussed.

COLUMN-WISE COUNTING

The column-wise counting circuits accept an input vector in which the placement of 1's is not important. The output represents, in binary weighted form, the number of 1's in the input vector. Therefore if there are m bits in the input vector, then the number of bits in the output vector will be $\lceil \log_2 m \rceil$.

A threshold function, r, can be adapted to represent the input-output transition of a counter as follows:

$$r = (p, q \mid M) \qquad (1)$$

(Where M = total number of 1's in the input vector).

This means that:

$r = 1$ if $M \bmod q \geq p$

0 otherwise.

If $q = 2p$, this function generates normal binary vectors corresponding to M. For example, if M=6, then:

$$r_0 = (1,2 \mid 6) = 0$$

$$r_1 = (2,4 \mid 6) = 1$$

$$r_2 = (4,8 \mid 6) = 1$$

Thus the binary representation of 6 is easily found as 110.

Ho and Chen proposed a logical circuit which produces an output vector from the input vector whose outputs are calculated by threshold functions. Fig.2.a is an example circuit for a (3,2) counter. This circuit can also be used, with a small modification, to generate outputs for all possible distributions of incoming carries. In Fig.2.b it is shown how this modification is made. In this circuit, the three outputs O_0, O_1 and O_2 correspond to incoming carry values of 0, 1 and 2 respectively. The O_0 output is calculated by the threshold function (1) above. The O_1 output is calculated as :

$$r_0' = (1,2 \mid M+1)$$

$$r_1' = (2,4 \mid M+1)$$

$$r_2' = (4,8 \mid M+1)$$

and the O_2 output :

$$r_0'' = (1,2 \mid M+2)$$

$$r_1'' = (2,4 \mid M+2)$$

$$r_2'' = (4,8 \mid M+2) \quad \text{where } M \leq 3.$$

In the most general case the O_j output vector of a counter is calculated as :

$$r_i^j = (2^i,\ 2x2^i \mid M+j)$$

where $i = 0..q-1$ and $j = 0..p-1$.

Following the notation in [5] such a chip can be represented as a (p,q') chip, where $q'=px(q+1)-1$ and :

q : number of bits in the O_0 output vector,

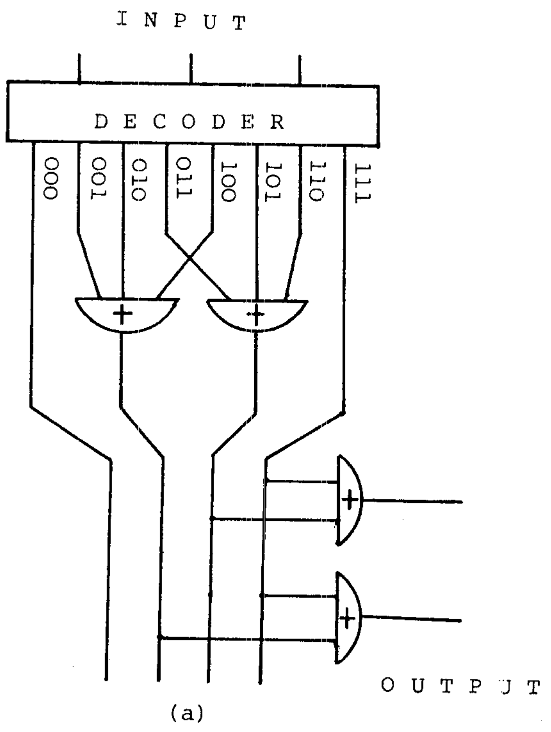
p : number of bits in the input vector such that $p=2^q-1$.

A (p,q') chip will have a delay of three logic levels. A (7,27) chip can be implemented in MSI technology.

ADDITION WITH MULTIPLE OUTPUT COUNTERS

The principle of multiple-operand addition using (p,q') chips is not different from two operand addition which uses Half Adders. In Fig.3.a an example is given for addition of a 3x4 array. This scheme finds the sum of three rows as indicated in Fig. 3.b.

The scheme in Fig. 3.a employs two different types of AND-OR circuits: counters and multiplexers. Allthough the former are generally slower than the latter, they are, however inevitable in today's best known schemes. Once the partial results are generated, an mxn array can be summed in $\lceil \log_2 n \rceil$ multiplexer levels. The delay of an mxn array adder is :



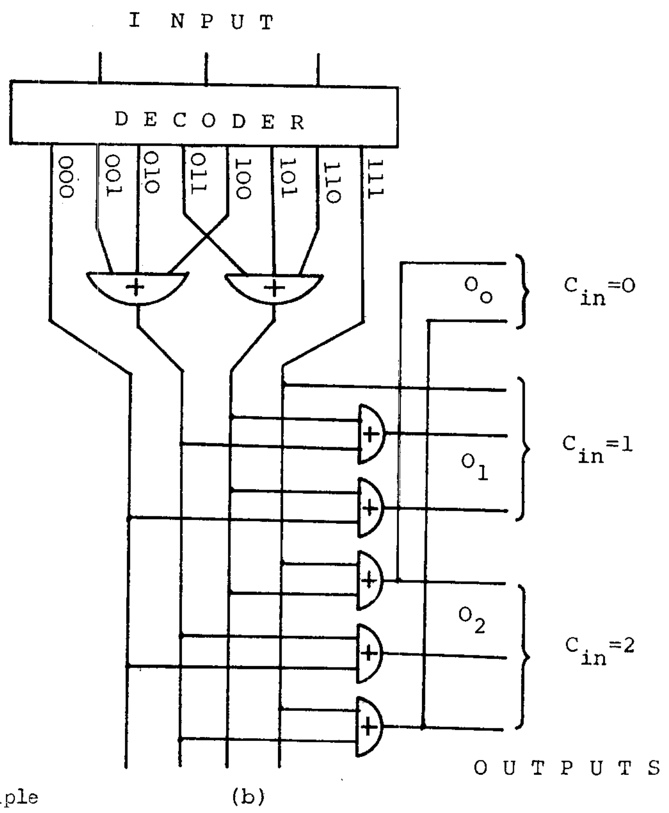


Fig. 2: Counters a) With single output vector, b) With multiple output vector.

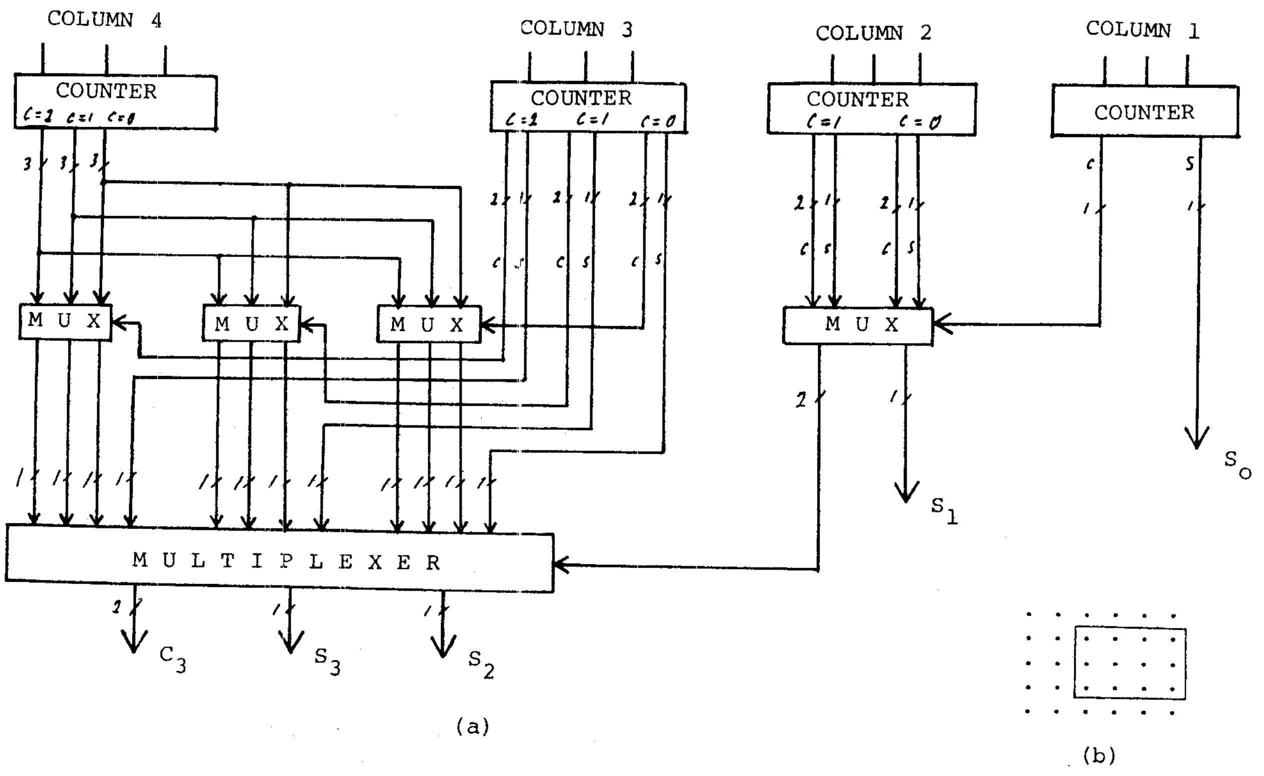


Fig. 3: 3x4 array adder using multiple output counters.

$$t_o = t_c + \lceil \log_2 n \rceil \times t_m \qquad (2)$$

where t_c : counter delay

n : number of bits in a row

t_m : delay of one multiplexer level.

Once an mxn adder is implemented as a basic building block, a kxn array (k≥m) can be summed in

$$N_s = \lceil \log_m k \rceil \qquad (3)$$

stages.

In the most general case the total delay for addition of a kxn array by using mxn basic adders is :

$$T \le [t_c \times (\lceil \log_2 n \rceil + 1)] \times \lceil \log_m k \rceil \qquad (4)$$

Here the number of multiplexer levels is incremented by one in order to include the effect of additional columns obtained in the later stages of addition process.

It can be seen from (2) that t_o is not affected by m. Therefore, in order to maximize the speed m must be chosen to be as large as technology permits. Note from (3) that $N_s \to 1$ as $m \to k$. This means that if m=k then k-operand addition can be made as fast as 2-operand addition in CSL.

SIMULATION RESULTS OF SPEED ESTIMATIONS

The speed of the scheme introduced here depends on the choice of m. It is possible to implement an 8 x 8 array adder on a single chip with m=k in todays technology. For larger arrays it may be necessary to divide the array into smaller blocks.

An 8-bit by 8-bit multiplication scheme was implemented in NMOS technology, using 4x8 and 3x16 adder blocks [7]. Reduction steps were as in Fig. 4. A simulation program, written in SPICE, has shown that this multiplication could be done in less than 50 ns.

Analytically, the time needed for generating the result of 8-bit by 8-bit multiplication is $t_8 + t_{16}$, where :

$$t_8 = t_c + t_m (\log_2 8) = t_c + 3\, t_m$$

$$t_{16} = t_c + t_m (\log_2 16) = t_c + 4\, t_m$$

Multiple output counters were implemented with 13 ns delay [7]. Assuming t_m=3 ns delay at each multiplexer level (when implemented on the same chip as the counters), the total delay will be 47 ns.

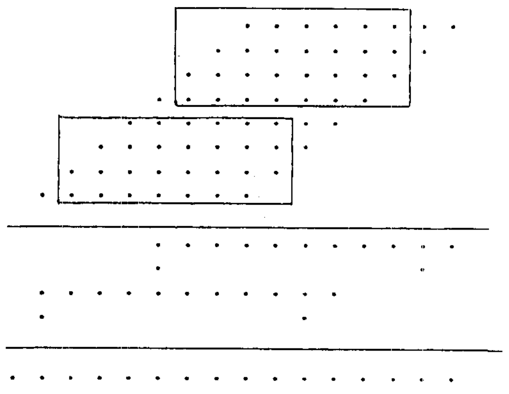


Fig. 4. Reduction steps of 8 x 8 multiplication scheme in CSL.

DISCUSSION

A comparison between the other methods and the method presented in this paper may be made from two different aspects :

a. The choice of the strategy for exploiting parallelism,

b. The choice of the parallel counters for generating partial results.

The conditional Sum Addition technique maximizes parallelism and this parallelism is not obstructed by any carry propagation in the classical sense. The only delay incurred after generation of the partial results is in the selection the correct outputs among the partial results. Most of the selection here is done in parallel. Extension of this method to multiple operand addition is straightforward and requires no modification. In the literarure carry save addition techniques were used in order to exploit parallelism [11]. In this method the parallelism is maximized in the early stages. However the final stage of the addition process, which is the stage when the "saved" carries are taken into consideration, requires carry propogation. This carry propagation may reduce the speed down to an intolerable level as stated in an earlier study [12]. It is felt that a suitable combination of the two systems could yield an interesting scheme. The feasibility of such a technique is now under investigation.

A number of parallel counter schemes were proposed in the literature [3-6]. In general, it is possible to modify any parallel counter scheme to generate the multiple output vector needed in Conditional Sum Logic. All one needs to do for this is to design a logic array which generates the multiple output vector from the basic C_{in}=0 output which is generated by the counter. Such an extension was found to be most natural in Ho and Chen's parallel counters. Therefore the counters used in this paper were derived from these counters.

Implementation of the method presented is not limited to the way described. The multiplexer network could be implemented by a PLA. ROM table look-up techniques could be utilized for counter design. It is not, however, the primary interest of this paper to investigate such considerations in detail.

CONCLUSION

Using the Conditional Sum Logic, a scheme for multiple operand addition is introduced. For generating the partial results, a modified version of $(2^q-1, q)$ counters are used.

A comparison among some schemes has been given in [6,8,9,10]. The scheme introduced in this paper maximizes parallelism and therefore yields a higher speed in comparison with the other schemes.

The cost is comparable to the other known schemes. The counters are implemented by only a small modification of the schemes in [5].

REFERENCES

1- Slansky, J., "Conditional-Sum Addition Logic", IRE Trans. on Electronic Computers, June 1960, pp.749-731.

2- Svoboda, A., "Adder with Distributed Control", IEEE Trans. on Computers, Vol. C-19, No. 8, Aug. 1970, pp. 749-751.

3- Ferrari, D., "Un Motiplicatore Numeriko Parallelo superimentale", in Rend. 67th Riunione Ann., 1966, pp.1-4.

4- Dadda, L., "Some Schemes for Parallel Multipliers", Alta Frequenze, Vol. 34, March 1965, pp.349-356.

5- Ho, I.T. and Chen, C.T.,"Multiple Addition by Residue Threshold Functions and Their Representation by Array Logic", IEEE Transactions on Computers, Vol. C-22, No. 8, Aug. 1973, pp. 762-767.

6- Swartzlander, E.E., JR., "Parallel Counters", IEEE Trans. on Comp., Vol. C-22, No. 11, Nov. 1973, pp. 1021-1024.

7- Efe, K., "A High Speed Multiplication Scheme", Final Report - Fall Quarter, UCLA, Dec. 1979, Private Communic.

8- Habibi, A. and Wintz, P.A. "Fast Multipliers", IEEE Trans. on Comp., Vol. C-19, No. 2, Feb. 1970, pp. 153-157.

9- Stenzel, W.J. "A Class of Compact High-Speed Parallel Multiplication Schemes", Dept. of Comp. Sci., Univ of Illinois at Urbana, Tech. Rep. Sep. 1975.

10- Agrawal, J.P. and Reddy, V.U., "Log-Sum Multiplier", National Computer Conference 1976, pp.783-787.

11- Anderson, S.F., Earle, J.G., Goldschmidt, R.E., Powers, D.M. "The IBM System/360 Model 91 : Floating-Point Execution Unit", IBM Journal, January 1967, pp.34-53.

12- Hwang, K., "Computer Arithmetic : Principles, Architecture and Design" John Wiley and Sons, 1979.