Adder With Distributed Control

ANTONIN SVOBODA, FELLOW, IEEE

Abstract—An adder is described for addition of a large number of binary numbers x_j, $j=1, 2, \cdots, m$, where $x_j=\sum_i x_{ji}\cdot 2^i$, $x_{ji}=0, 1$, $i=0, 1, \cdots, n-1$. The adder's algorithm has two parts: 1) the bits x_{ji} are added independently for each binary order i: $s_i=\sum_j x_{ji}\leqq m$ and the result expressed in the binary form $s_i=\sum_k a_{ik}\cdot 2^k$, $a_{ik}=0, 1$, $k=0, 1, \cdots, p-1$ where $2^{p-1}\leqq m<2^p$); 2) the sum $y=\sum_j x_j$ is formed by adding terms $a_{ik}\cdot 2^{i+k}$ as contributions of the bit s_{ik} to the total y. A hardware implementation of this algorithm is suggested where the sum s_i is obtained by a sequential circuit which reorders the values x_{ji} so that their sum s_i remains unchanged and so that after the reordering the new values x'_{ji} obey the conditions $x_{j+1,i}\geqq x_{ji}$ for every $j=1, 2, \cdots, m-1$. The implementation with integrated circuits should be quite rewarding because the control of the circuit is done with independent control elements distributed all over the chip.

Index Terms—Adder, adder for large number of numbers, distributed control, reordering.

Symbolism

Binary numbers entering addition:

$$x_j = x_{j,n-1}\cdot 2^{n-1} + x_{j,n-2}\cdot 2^{n-2} + \cdots + x_{j,1}\cdot 2^1 + x_{j,0} \quad (1)$$
$$x_{ji}= 0, 1, \quad i = 0, 1, \cdots, n-1, \quad j = 1, 2, \cdots, m.$$

Number of nonzeros (ones) in the order i:

$$s_i = \sum_j x_{ji}. \quad (2)$$

Number s_i expressed as a binary number with p digits s_{ik}:

$$s_i = \sum_k s_{ik}\cdot 2^k, \quad s_{ik} = 0, 1, \quad k = 0, 1, \cdots, p-1. \quad (3)$$

Resulting sum y of numbers x_j:

$$y = \sum_j x_j, \qquad j = 1, 2, \cdots, m \quad (4)$$

In binary form:

$$y = y_{N-1}\cdot 2^{N-1} + y_{N-2}\cdot 2^{N-2} + \cdots + y_1 2^1 + y_0$$
$$y_i= 0, 1, \qquad i = 0, 1, \cdots, N-1. \quad (5)$$

The number of bits of the result is N and to prevent an overflow we have to choose N to obey $2^{N-n}\leqq m<2^{N-n+1}$.

Introduction

There are arithmetic operations which require the addition of a large number of numbers. Multiplication and special function generation are such operations. The following numerical methods have been implemented in operation units of computers.

1) Accumulation through repeated addition (single adder working as an accumulator). This solution is simple but needs large execution time.
2) Addition of numbers x_j by pairs, addition of the resulting sums by pairs, and a repetition of that process until the final sum y is reached. (Implementation by a cascade of adders gives execution time much shorter than in the preceding case. For instance, for 32 numbers case 1 needs 31 addition times, case 2 needing only 4 ($\log_2 32-1$) addition times.)
3) "Carry save addition" which adds a group of three numbers x_j (triplets) and reduces their sum to a sum of two numbers. One of those numbers evaluates the sum modulo 2 of bits in the same binary order, the second number being composed from the carries generated but not transferred. These partial results regrouped in triplets enter a "carry save addition" again and the procedure is repeated until only two numbers remain to be added. (Implementation uses a cascade of full adders. The number of addition cycles is not smaller than in case 2 but the operation time is extremely reduced because the carries are not transferred, although they are formed.)
4) Evaluation of y as a sum of all components $x_{ji}\cdot 2^i$. (Implementation depends on the way in which the components are grouped and on the sequence in which the groups' sums are added [1], [2].) A relay adder for any number of binary numbers has been described by the author [3].

The adder discussed here belongs to the last category. The fundamental idea to count bits of the same binary order is as old as the addition with a pencil on paper. The counting by ordering is believed to be new.

It is clear that

$$y = \sum_j x_j = \sum_j\sum_i x_{ji}\cdot 2^i = \sum_i\left(\sum_j x_{ji}\right)\cdot 2^i = \sum_i s_i\cdot 2^i$$
$$= \sum_i\left(\sum_k s_{ik}\cdot 2^k\right)\cdot 2^i = \sum_k\sum_i s_{ik}\cdot 2^{i+k} = \sum_k a_k,$$

where

$$a_k = \sum_i s_{ik}\cdot 2^{i+k}$$

and

$$s_{ik} = 0 \text{ or } 1. \quad (6)$$

The summation limits are $j=1, 2, \cdots, m$; $i=0, 1, \cdots, n-1$; $k=0, 1, \cdots, p-1$ where $p\ll m$. The integer p can be fixed by the relation

$$\log_2 m < p \leqq 1 + \log_2 m. \quad (7)$$

The sum y of m numbers x_j is transformed in this way into a sum of p ($p\ll m$) numbers a_k. (The new group of p num-

Manuscript received October 19, 1969; revised January 20, 1970.
The author is with the Department of Electrical Engineering, University of California, Los Angeles, Calif. 90024.

as the same sum as the original group of m numbers.) For instance, the addition of $m=31$ numbers is transformed into an addition of $p=5$ numbers. (For $m=3$ we come back to the regular carry save addition.)

The summing up of bits could be done with shift registers and counters as suggested in [2]. We propose a counting of bits based on their reordering implemented by a sequential logical circuit.

Let us suppose that we start with a given sequence of bits $x_{ji}, j=1, 2, \cdots, m$ for a fixed binary order i. Integer j is the ordering parameter of the sequence. Then if

$$x_{ji} \geqq x_{j+1,i} \quad \text{for every } j = 1, 2, \cdots, m-1, \tag{8}$$

the sequence is ordered so that x_{ji} never increases with j (the sequence is monotonic). A given sequence does not have this property in general, but it is quite easy to reorder without changing the sum

$$s_i = \sum_j x_{ji}.$$

To do it we interchange any two bits which follow each other in the wrong order:

$$\begin{aligned}(x_{ji} < x_{j+1,i}) &\Rightarrow (x_{ji} = 0, x_{j+1,i} = 1) \\ &\Rightarrow (x'_{ji} = x_{j+1,i} = 1, x'_{j+1,i} = x_{ji} = 0)\end{aligned} \tag{9}$$

so that $x'_{ji} > x'_{j+1,i}$ after the interchange which is repeated until the final sequence is monotonic and obeys (8).

Addition Algorithm

0 BEGIN
1 IF $(x_{ji} \geqq x_{j+1,i}$ for every $j=1, 2, \cdots, m-1$ and for every $i=0, 1, \cdots, n-1)$ THEN GO TO 4
2 APPLY (9) for every i and for every j for which $(x_{ji} < x_{j+1,i})$
3 GO TO 1
4 For every $i=0, 1, \cdots, n-1$ find the lowest value of j belonging to $x_{ji}=0$ and then make $a_i=j-1$
5 Find $s_{ik}=0$ or 1 so that $s_i = s_{i,p-1} \cdot 2^{p-1} + \cdots + s_{i,1} \cdot 2^1 + a_{i,0}$ for $k=0, 1, \cdots, p-1$
6 Form $a_k = \sum_i s_{ik} \cdot 2^{i+k}$ and evaluate $y = \sum_k a_k$
7 END

Implementation

There are only a few problems of logical design of this adder worth mentioning:

1) sequential circuit for the reordering of x_{ji};
2) logical circuit reading s_i and generating s_{ik};
3) adder for y (line 6);
4) end of reordering strobe generator.

An example of the fundamental version of the sequential circuit for reordering is in Fig. 1. The reset control of the flip-flops for $x_{ji}, j=1, 2, \cdots, 7, i=$ constant as well as the inputs for x_{ji} are not shown to get a clearer picture. We begin with the state where the bits x_{ji} are stored in flip-flops already. The trigger T comes then to test the condition (9) for every j (line 1 of the algorithm). Notice that this condition is never valid for two adjacent values $j, j+1$ at the same

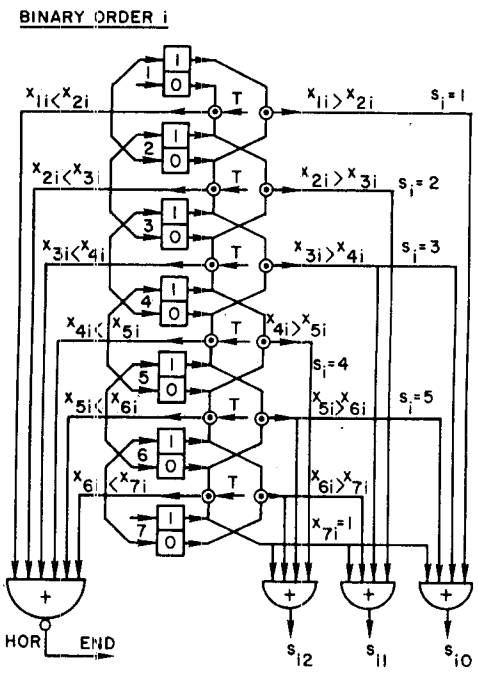

Fig. 1. Sequential circuit for reordering of x_{ji}.

time. If there is a j for which the condition (9) is valid, then the flip-flop x_{ji} is set to "1" and the flip-flop $x_{j+1,i}$ is reset to "0." This process will proceed as long as (9) is valid for at least one value of j. If not then the values of x_{ji} are ordered and (8) holds.

After the reordering, the last nonzero value along the sequence $x_{ji}, j=1, 2, \cdots, m$ indicates the count s_i of nonzeros in the order i: $s_i=j-1$. Logical circuit producing this information is based on the fact that after the reordering there is never more than just one value of j for which $x_{ji} > x_{j+1,i}$. Then it is possible to find it by the Boolean condition $x_{ji}\bar{x}_{j+1,i}=1$. The corresponding logical circuit is located on the right-hand side of Fig. 1. None (for $x_{0i}=0$) or just one of the signals s_i (horizontal line) will be generated and read out through three OR gates as three binary digits $s_{ik}(s_i = s_{i2} \cdot 2^2 + s_{i1} \cdot 2^1 + s_{i0})$ needed to compile the numbers $a_k, k=0, 1, 2$.

The end of reordering strobe signal is easy to derive because the end means that the condition (9) is invalid for every $j=1, 2, \cdots, 6$. Then the NOR gate on the left-hand side of Fig. 1 has no input signal and generates the end strobe.

In the case of our example ($m=7$), there are only three numbers to be added:

$$\begin{aligned}&a_2 + a_1 + a_0 = y, a_0 = \cdots + s_{30} \cdot 2^3 + s_{20} \cdot 2^2 + s_{10} \cdot 2^1 + s_{00}, \\ &a_1 = \cdots + s_{21} \cdot 2^3 + s_{11} \cdot 2^2 + s_{01} \cdot 2^1 + 0 \\ &\quad = \cdots + s_{21} \cdot 2^2 + s_{11} \cdot 2^1 + s_{01}) \cdot 2^1, \\ &a_2 = (\cdots + s_{22} \cdot 2^2 + s_{12} \cdot 2^1 + s_{02}) \cdot 2^2.\end{aligned}$$

An adder for this task is sketched in Fig. 2. A conventional pattern of full adders is used. It is quite easy to show that a correct result is obtained. (For instance s_{02}, s_{11}, s_{20} which all have the same weight 2^2 enter a full adder with output bits b, a so that $s_{02}+s_{11}+s_{20}=2b+a$.)

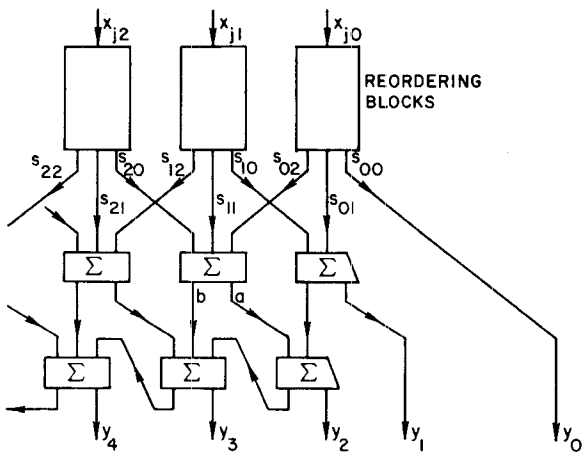

Fig. 2. Conventional adder for $N=3$.

Execution Time

The reordering is executed for all binary orders at the same time. Because of this time sharing we can conclude that the final values of s_{ik} are in the adder of s_k when the reordering has ended for all binary orders i. This moment arrives t_1 seconds after the start of reordering operations when all end of reordering strobe signals are detected. The addition of a_k needs less than t_2 seconds, depending on the structure of the adder for a_k. In the case of our example, t_2 is the time for the longest ripple carry propagation.

Conclusion

The new concept presented in this paper should produce adders with the following desirable properties.

1) It can handle an extremely large number (m) of numbers to add.
2) The hardware cost is smaller than with other concepts.
3) The structure is well suited for circuit integration since the control is distributed over the chip.
4) Very small execution times are expected.

To check the performance of the reordering network a breadboard model was made and tested as a part of an M.S. thesis [4]. It has been found that for $m=7$, $t_1 \leq 90\ n$ seconds.

References

[1] D. Ferrari, "Un moltiplicatore numerico parallelo sperimental," *Rend. LXVII Riunione Ann.*, AEI-II-29/1966.
[2] A. J. Atrubin, "A one-dimensional real-time iterative multiplier," *IEEE Trans. Electronic Computers*, vol. EC-14, pp. 394–399, June 1965.
[3] A. Svoboda, "Releove jednotaktni dvojkove scitacky," *Stroje na Zparacovani Informaci*, vol. 3, 1955. Also, "Parallel relay binary adders," *Information Processing Machines*, vol. 3. Prague, Czechoslovakia: Publishing House of the Academy of Sciences, 1955.
[4] C. Pereida, "Adder with distributed control," M.S. thesis, University of California, Los Angeles.