EVALUATING "A+B = K" CONDITIONS IN CONSTANT TIME

J. Cortadella, J.M. Llabería

Dept. d'Arquitectura de Computadors, Facultat d'Informàtica
Universitat Politècnica de Catalunya
Pau Gargallo, 5 08028 Barcelona (Spain)

ABSTRACT

One of the most important components of an ALU is the adder. Its response time is mainly determined by the carry propagation delay. Evaluation of conditions between two numbers are usually performed with the ALU by means of a substraction. In this paper we deal with a type of conditions that can be evalauted without requiring a complete ALU operation. The circuit that is presented detects the condition A+B=K (n-bit numbers) in constant time, avoiding the carry propagation delay. Some applications for this circuit are also presented.

INTRODUCTION

The carry computation is one major problem in the response time of parallel adders. Several approaches have been proposed in order to reduce it [1][2][3]. VLSI techniques have been also used to minimize design costs and chip area. The fastest adders, such as lookahead adders, perform additions of n-bit numbers in time $O(\log n)$ and area $O(n \log n)$.

This paper deals with a problem associated with parallel adders. A circuit for detecting when the addition of two n-bit numbers is equal to another n-bit number (A+B=K) is presented. We will prove that the result of this evaluation can be computed in constant time and area $O(n)$, avoiding the problem of the carry propagation delay.

The paper is organized as follows. First, the theoretical basis of the problem is presented. Next, the design of the circuit is described. Finally, some applications for this circuit are discussed.

This work was supported by the Ministry of Education of Spain (CAICYT) under contract number 314-85

THEORETICAL BASIS

Given three n-bit vectors $A=a_n a_{n-1}...a_1$, $B=b_n b_{n-1}...b_1$ and $K= k_n k_{n-1}...k_1$ that represent two's complement integers, we want to design a circuit that can evaluate the conditon A+B=K (arithmetic addition) in constant time. This means that the evaluation does not depend on the lenght (number of bits) of the vectors.

The basic idea of the circuit is the local evaluation of the condition at each bit position i, assuming that the condition is fulfilled in the rest of bits. The last stage computes the global logical OR of all the local evaluations.

The behavior of a full adder can be described with the following expressions:

$p_i = a_i \oplus b_i$	(Carry propagation)
$g_i = a_i \wedge b_i$	(Carry generation)
$c_i = (p_i \wedge c_{i-1}) \vee g_i$	(Carry. We define c_0=0)
$r_i = p_i \oplus c_{i-1}$	(Addition result)

The condition "A+B=K" could be detected by comparing r_i and k_i at each bit position i and performing a global OR of all the comparisons. In this case the response time would be determined by the carry propagation time, since c_i is defined as a function of c_{i-1}. In order to avoid the carry propagation problem, we define the following expressions:

$q_i = (p_i \wedge \overline{k_i}) \vee g_i$	(Predicted carry. We define q_0=0)
$s_i = p_i \oplus q_{i-1}$	(Predicted addition result)
$z_i = s_i \oplus k_i$	($z_i = 0 \Leftrightarrow s_i = k_i$)

$$Z_k = \bigvee_{i=1}^{k} z_i$$

CH2458-8/88/0000-0243$1.00 © 1988 IEEE

The predicted carry (q_i) substitutes the carry for the computation of the predicted result. We can easily prove that $q_i = c_i$ in case that $r_i = k_i$:

$$q_i = (p_i \wedge \overline{k_i}) \vee g_i = (p_i \wedge \overline{r_i}) \vee g_i = (p_i \wedge c_{i-1}) \vee g_i = c_i$$

The design of the circuit is based on the following theorem:

Theorem.

$$Z_n = 0 \quad \Leftrightarrow \quad r_i = k_i \text{ for each } i \in \{1,...,n\}$$

Proof. By induction on n.

First, we will prove that the theorem holds for n=1.

$$Z_1 = z_1 = s_1 \oplus k_1 = p_1 \oplus k_1 = r_1 \oplus k_1 \quad (\text{since } c_0 = q_0 = 0)$$

thus

$$Z_1 = 0 \quad \Leftrightarrow \quad r_1 = k_1$$

Next, we will prove that the theorem holds for n > 1. Let's assume the theorem holds for n-1 bits.

If $Z_{n-1} = 1$ then $Z_n = Z_{n-1} \vee z_n = 1$, and the theorem holds for n bits, since there is $i \in \{1,...,n-1\} \subset \{1,...,n\}$ such that $r_i \neq k_i$ (by induction hypothesis).

If $Z_{n-1} = 0$ then $Z_n = z_n$
By the definition of z_i we have that

$$z_n = s_n \oplus k_n = p_n \oplus q_{n-1} \oplus k_n$$

since $Z_{n-1} = 0$ then $r_{n-1} = k_{n-1}$ and $q_{n-1} = c_{n-1}$ (by induction hypothesis). Therefore,

$$z_n = p_n \oplus c_{n-1} \oplus k_n = r_n \oplus k_n$$

and

$$z_n = 0 \quad \Leftrightarrow \quad r_n = k_n$$

Since $r_i = k_i$ for each $i \in \{1,...,n-1\}$, we have that

$$Z_n = z_n = 0 \Leftrightarrow r_n = k_n \Leftrightarrow r_i = k_i \text{ for each } i \in \{1,...,n\}$$

□

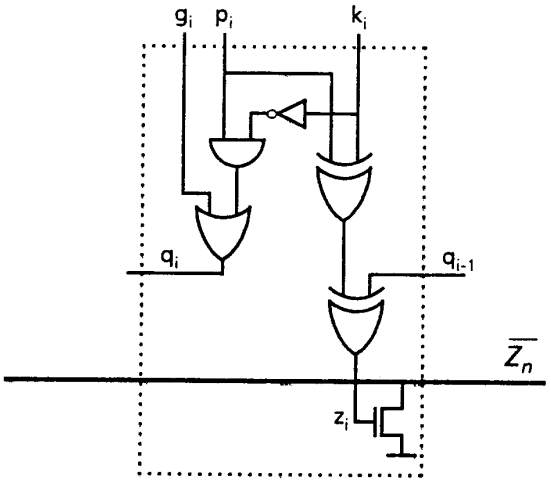

Figure 1. Circuit design

CIRCUIT DESIGN

From the previous section we can observe that function q_i (predicted carry) does not depend on any information in the other stages. So functions z_i can be computed in parallel. Figure 1 depicts a diagram of the circuit that computes Z_n. We assume that each gate and the $\overline{Z_n}$ line discharge (n-input NOR) take constant time. So we can state that the computation of Z_n also takes constant time. Since the circuit is made of n identical cells, the chip area is $O(n)$.

The circuit design can be simplified if any of the operands is constant. Figure 2 shows a diagram of the circuit when vector K is constant.

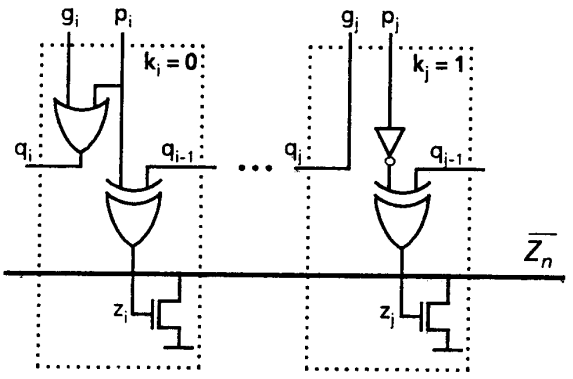

Figure 2. Circuit design with K constant

SOME APPLICATIONS

The condition evaluation is one of the most important operations performed in the execution of conditional branches. In most architectures, the condition is evaluated as a function of the condition codes. Their value depends on the result of ALU operations. Figure 3 depicts a widely used ALU structure [4]. It consists of an Operand Modifier Unit (OMU) and an adder. The OMU computes the carry propagation and generation functions (p_i and g_i) depending on the data input (a_i and b_i) and the operation.

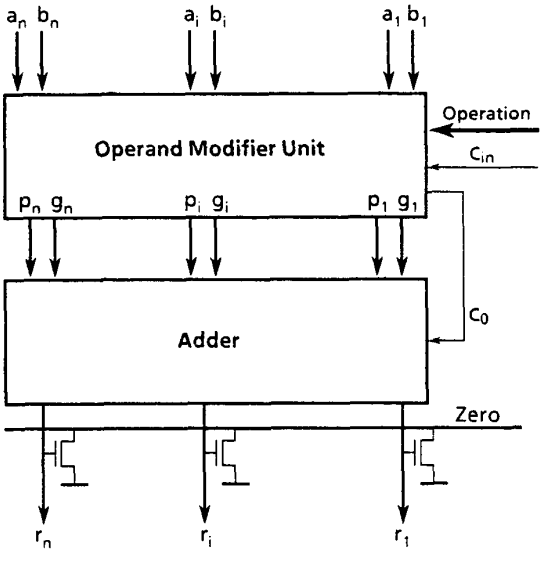

Figure 3 ALU structure

The dependencies produced by the execution of branches have been extensively studied by many authors [5][6]. In pipelined processors, the condition evaluation has to be delayed until all the instructions that modify the condition codes and precede the branch have finished its execution. The condition evaluation is the most important dependency that restricts the execution of branches with zero delay [7].

Katevenis observed that 80% of conditional branches involve tests for equality, inequality and any relation with zero (fast comparisons) [8]. Equality and inequality tests are determined by the zero condition code (Z), usually after a substraction operation (comparison). Relations with zero are determined by Z and the sign bit of the operand that is tested.

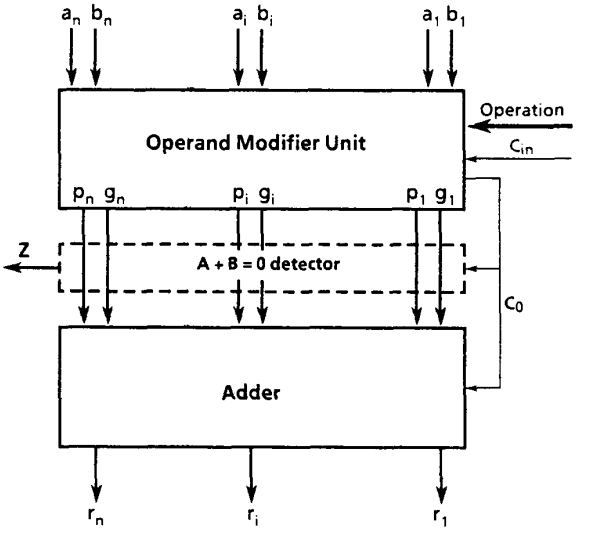

Figure 4 Evaluation of condition code Z

By designing a circuit that detects the condition A+B=0, the computation of Z can be advanced and the delay produced by the condition evaluation reduced (see figure 4). This improvement can increase the processor performance substantially since about one in every six instructions are branches that have to evaluate this kind of conditions.

Another application can be found for DO-like loops. This kind of loops are very used in numerical programs (figure 5.a). The compiler generates a code similar to the one shown in figure 5.b. As Katevenis also observed, some comparisons can be converted to fast comparisons [6]. This is the case of "I ≤ N", which can be converted to "I ≠ N+1" without modifying the loop behavior. By considering an ALU structure such as the one shown in figure 6, and introducing a new instrucion in the machine language (NEXT), the compiler can generate a code similar to the one in fig. 5.c. The

do i=1,n . . . end do	Ri ← 1 do: . . . Ri ← Ri + 1 if Ri ≦ n goto do	Ri ← 1 Rlimit ← n+1 do: . . . NEXT Ri, do
(a)	(b)	(c)

Figure 5. Do-like loops

instruction NEXT increments Ri and compares the new value with Rlimit. In case that Ri≠Rlimit, control is transferred to the branch target address. Again, the improvement is based on the fact that the condition can be evaluated before computing the new value of Ri, avoiding pipeline delays in the execution of conditional branches.

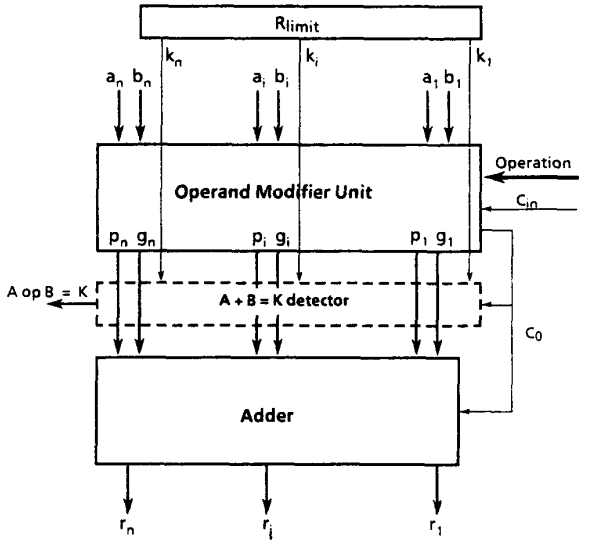

Figure 6. ALU structure for DO-like loops.

CONCLUSIONS

In this paper we have presented a circuit that detects the condition "A+B=K" in constant time. Its area is proportional to the number of bits of the vectors. The theoretical basis and several design approaches have been described.

This circuit can be used to detect a wide spectrum of conditions in branch instructions. It can improve the processor performance by advancing the evaluation of conditions and eliminating the pipeline delays produced by these operations.

REFERENCES

[1] K. Hwang, "*Computer Arithmetic. Principles, Architecture and Design*, John Wiley & Sons, 1979.

[2] R.P. Brent and H.T. Kung, "*A Regular Layout for Parallel Adders*", IEEE Transactions on Computers, Vol. C-31, No. 3, March 1982, pp. 260-264.

[3] M.Lehman and N.Burla, "*Skip Techniques for High-Speed Carry-Propagation in Binary Arithmetic Units*", IRE Transactions on Electronic Computers, Dec. 1961, p. 691.

[4] M. Pomper et al., "*A 32-bit Execution Unit in an Advanced nMOS Technology*", IEEE Journal of Solid State Circuits, Vol. SC-17, No. 3, June 1982, pp. 533-538.

[5] E.M. Riseman and C.C. Foster, "*The Inhibition of Potencial Parallelism by Conditional Jumps*", IEEE Transaction on Computers, Vol. C-21, No. 12, Dec. 1972, pp. 1405-1411.

[6] Scott McFarling and J.L. Hennessy, "*Reducing the Cost of Branches*", Proc. 13th. Annual Symposium on Computer Architecture, June 1986, pp. 396-403.

[7] D.R. Ditzel and H.R. McLellan, "*Branch Folding in the CRISP Microprocessor: Reducing Branch Delay to Zero*", 14th. Ann. Int. Symp. on Computer Architecture, June 1987.

[8] M.G.H. Katevenis, "*Reduced Instruction Set Computer Architectures for VLSI*", Ph. D. dissertation, University of California, Berkeley, October 1983.