Comments

Comments on "Evaluation of $A + B = K$ Conditions Without Carry Propagation"

Behrooz Parhami

Abstract—**A special carry-free circuit for the evaluation of conditions of the type $A + B = K$ is proposed by cortadella and Llaberia, and its usefulness for reducing the negative effects of conditional branches in pipelined architectures is noted. It is shown that the same advantages are offered by another equally simple circuit based on carry-save redundant numbers and (3, 2)-counters. This alternative circuit has other potential applications and much of it may in fact be already present in some ALU's.**

Index Terms—**Addition, carry propagation, carry-save numbers, comparators, conditional branches, parallel counters, pipelined architectures, redundant number representation.**

I. Introduction

A special add-on circuit, comparable in area complexity to a ripple-carry adder, has recently been proposed for evaluating conditions of the type $A + B = K$ without the need for performing a carry-propagate addition [1]. Internal "propagate" and "generate" signals $p_i = a_i \oplus b_i$ and $g_i = a_i \wedge b_i$ from the addition $A + B$ are combined with the bits k_i of K_i, in a local way, and the resulting n-bit vector Z is fed to an n-input AND circuit computing the final test result $E = z_n \wedge z_{n-1} \wedge \cdots \wedge z_1$. The proposed circuit is positioned between the logic for producing the p and g signals and the ALU's main adder. Usefulness of the circuit, named "fast adder-comparator," is demonstrated for reducing conditional branch penalties in a pipelined CPU.

In this note, I utilize the simple equivalence of $A + B = K$ and $A + B + (2^n - 1 - K) = 2^n - 1$ to convert the problem into a 3-operand carry-save addition [4] and a subsequent all-ones detection for the resulting carry-save redundant number.

II. The Alternative Circuit

Let A, B, and K be n-bit 2's-complement binary numbers. To determine if $A + B - K = 0$, one can add A, B, and $K^{\text{comp}} = 2^n - 1 - K$ (bitwise complement of K) and check if the sum is equal to $2^n - 1$ which is represented by the n-bit all-ones vector. Instead of completely evaluating $A + B + K^{\text{comp}}$, one can use a row of full adders with no carry propagation, also known as (3, 2)-counters [4], to find two numbers S and C satisfying $A + B + K^{\text{comp}} = S + C$. The problem then reduces to checking the condition $S + C = 2^n - 1$ or $S = 2^n - 1 - C = C^{\text{comp}}$. This is accomplished by a row of n two-input XOR gates feeding an n-input AND circuit that produces the final comparison result E.

The conversion of the sum $A + B + K^{\text{comp}}$ to the two numbers S and C, that together can be viewed as a single redundant radix-2 number with the digit set $\{0, 1, 2\}$, is a standard technique in computer arithmetic [4]. The verification of $S = C^{\text{comp}}$ is also a special case of a reduction method that I have used previously for zero, sign, and overflow detection in generalized signed-digit arithmetic [2], [3].

Manuscript received December 31, 1992; revised June 16, 1993.

The author is with the Department of Electrical & Computer Engineering, University of California, Santa Barbara, CA 93106-9560.

IEEE Log Number 9213764.

III. Comparison

The alternative circuit defined in Section II consists of n full adders, n two-input XOR gates, and an n-input AND circuit. It can be positioned before the inputs to the ALU's main adder and implies logarithmic delay when the n-input AND is implemented as a tree of smaller AND gates. With respect to speed, the two alternatives are virtually identical. The critical path in the design of [1] goes through 3 XOR gates (one to generate p_i and two in the adder-comparator cell) and n-input AND circuit. In my design, the critical path contains a full-adder cell (2 XOR delay), an XOR gate, and an n-input AND circuit. While p_i still needs to be generated for the ALU's fast adder, the corresponding circuit is no longer in the path of the add-on comparator.

The hardware complexity of my design is higher than that of [1] by about one XOR gate per bit slice. However, there are two redeeming factors. First, I use standard full adder cells that have been fully optimized for many different technologies. Second, my circuit is potentially more useful in that the 3-to-2 reduction can also be used to speed up multiple-operand addition and multiplication in some applications. If such a 3-to-2 reduction circuit is already present in the ALU to facilitate multiplication, then the cost of my approach is indeed very small.

References

[1] J. Cortadella and M. Llaberia, "Evaluation of $A + B = K$ conditions without carry propagation," *IEEE Trans. Comput.*, vol. 41, no. 11, pp. 1484–1488, Nov. 1992.
[2] B. Parhami, "Generalized signed-digit number systems: a unifying framework for redundant number representations," *IEEE Trans. Comput.*, vol. 39, no. 1, pp. 89–98, Jan. 1990.
[3] ——, "On the implementation of arithmetic support functions for generalized signed-digit number systems," *IEEE Trans. Comput.*, vol. 42, no. 3, pp. 379–384, Mar. 1993.
[4] S. Waser and M. J. Flynn, *Introduction to Arithmetic for Digital Systems Designers.* New York: Holt, Rinehart & Winston, 1982.

Comments on "Decomposition of Complex Multipliers Using PolynomialEncoding"

Rajendra Katti

Abstract— **We present a better way of decomposition of complex multipliers using polynomial encoding than the method presented in the paper, "Decomposition of Complex Multipliers Using Polynomial Encoding." The decomposition described in this paper makes use of smaller multipliers which results in smaller ROM's if ROM table look-ups are used to implement multipliers.**

Manuscript received March 15, 1993.

The author is with the Department of Electrical Engineering, North Dakota State University, Fargo, ND 58105.

IEE Log Number 9213766.