Average propagation delay in a ripple adder

Feodor Vainstein[1] and Lev B. Levitin[2]
[1] Georgia Tech Regional Engineering Program
School of Electrical and Computer Engineering
College of Engineering
Georgia Institute of Technology
6001 Chatham Center Dr., Suite 350
Savannah, GA 31405, USA
Tel: 912.790.3440, Fax: 912.651.7279
Email: fvainste@gtrep.gatech.edu, vainstei@ece.gatech.edu
[2] Department of Electrical and Computer Engineering
College of Engineering
Boston University
Boston, Massachusetts, USA
Tel: 617.353.4607, Fax: 617.353.6440
Email: levitin@bu.edu

Abstract: An expression for the average carry propagation delay in a ripple carry adder is obtained which is exact up to terms of the order $O(n^{-1}\ln n)$. The case of several adders working in parallel is also considered.

1. Introduction

A synchronous self-timed system consists of a number of functional units equipped by completion detectors. Initially, with a new clock pulse, completion its are cleared by the completion detectors circuitry when the operation being performed by units is completed.

An AND gate is used to signal to the clock generator that all units completed their operations and a new clock pulse can be issued. With this approach, the throughput of a system becomes considerably higher compared with the systems using traditional approach, when the rate of clock pulses is determined by the longest possible propagation delay in the functional units. In this paper we evaluate the potentials in speeding up for the case when we have only one functional unit that is a ripple carry adder. A more general case of m ripple carry adders operating in parallel is also considered.

An estimate of the average carry propagation delay was first obtained in [1]. A tighter upper bound was obtained in [2]. This paper presents a much simpler analytical approach that yield an expression for the average delay which is asymptotically exact with the remainder of $O(\frac{\ln n}{n})$, where n is the number of full adders in the ripple adder.

By definition, the propagation delay in a combinational circuit is the duration of the time interval between the moment when all the input signals are stabilized and the moment when all output signals are stabilized.

Propagation delay depends on both the present inputs and on the inputs immediately preceding to the present ones. Often the term "propagation delay of a circuit" is used to denote the upper bound of propagation delays for all possible inputs.

Denote by τ the maximum propagation delay in one full adder. It is clear that in the worst case the propagation delay of n-bit carry ripple adder is equal to $n\tau$. This value is used to determine the clock frequency of a conventional (not self-timed) system. In this paper we will show that in the average case the propagation delay is much smaller than $n\tau$. In fact it is equal to $\tau(\log_2 n - c)$, where c is a constant approximately equal to 0.668.

It follows from this result that a 64-bit ripple carry adder can work on average almost 9 times faster if the self-timed approach is used.

2. Stabilization time determined by stoppers and runners.

Consider a ripple carry adder that consists of n full adders. We assume that the input bits

$$A_{n-1},\ldots,A_0;B_{n-1},\ldots,B_0;C_0$$

are independent random variables taking on values of 0 and 1 with equal probabilities. At time τ, the output carry for some full adders will stabilize.

In particular, if $A_i = B_i = 1$ then C_i=1 in τ and C_i will not change thereafter unless a new set of inputs is applied.

Similarly c then C_i=0 in τ and C_i will not change thereafter unless a new set of inputs is applied.

Hence if, for example, for every i $A_i = B_i$, then the carry propagation delay will be less than or equal to τ and all the outputs of the adder will be stable at time 2τ.

In fact, the actual time depends on the values of carries that appeared as a result of computation with inputs immediately preceding the present ones. In the most unfavorable case, however, the carry propagation delay is determined by the maximum number of consecutive bits of the input for which $A_i \neq B_i$.

Let us call input values $A_i = B_i$ "stoppers", and values $A_i \neq B_i$ "runners". Obviously, for completely random inputs, stoppers and runners occur independently with equal probabilities.

Denote stoppers by 0 and runners by 1. Then the total stabilization time $T = \tau(L+2)$, where L is the random variable that is the maximum length of a run of consecutive 1's in a random sequence of 0's and 1's.

3. The average propagation delay in a ripple carry adder.

As known in previous section, the propagation delay in a ripple carry n-bit adder is a linear function of the maximum length of a run of 1's in a random sequence of zeros and ones of length n, where the bits take on values of 0 and 1 with equal probabilities.

Thus the problem is reduced to the following: find the distribution of the maximum length of a run of 1's in a random sequence of zeros and ones.

Denote by $W(l,n)$ the number of such sequences where the maximum length of a run of 1's is smaller l.

The cumulative distribution function (cdf) of the maximum length L of a run is given by

$$F_L(l) = \Pr\{L < l\} = W(l,n)\cdot 2^{-n} \quad (1)$$

It is easy to see that $W(l,n)$ satisfies the following recurrence equation:

$$W(l,n) = 2\cdot W(l,n-1) - W(l,n-l-1) \quad (2)$$

with the boundary condition

$$W(l,l-1) = 2^{l-1} \quad (3)$$

It is well known that for large n the solution of equation (2) is asymptotically equal to

$$W(l,n) = a(l)\cdot X_l^n \quad (4)$$

where X_l is the largest real root of the characteristic equation:

$$X^{l+1} - 2X^l + 1 = 0 \quad (5)$$

Substituting into the bounding condition (3), we obtain:

$$a(l) = 2^{l-1} X_l^{-(l-1)} \quad (6)$$

Hence,

$$W(l,n) = 2^{l-1} X^{n-l+1} \quad (7)$$

The largest root of (5) can be expressed asymptotically for $2^{l+1} >> 1$, which gives a good approximation of X_l:

$$X_l = 2 \cdot (1 - \frac{1}{2^{l+1} - l}) + O((2^{l+1} - l)^{-2}) \quad (8)$$

Hence, asymptotically,

$$W(l,n) = 2^n (1 - \frac{1}{2^{l+1} - l})^{n-l+1} \quad (9)$$

Thus, from (1) and (9) the cdf of L is

$$F_L(l) = (1 - \frac{1}{2^{l+1} - l})^{n-l+1} \quad (10)$$

The expected value $E[L]$ is given by

$$E[L] = \sum_{l=1}^{n} (1 - F_L(l)) \quad (11)$$

For large n, the function $F_L(l)$ changes very rapidly from 0 to 1 in a narrow region. Therefore, a good estimate of $E[L]$ is given by

$E[l] \approx l_{1/2}$, where $l_{1/2}$ is the root of the equation:

$$F_L(l) = \frac{1}{2} \quad (12)$$

Solving (12), we obtain:

$$l_{1/2} = \log_2 \frac{n}{2 \ln 2} + O(\frac{\ln n}{n}) \quad (13)$$

Note that for large n,

$$F_L(l_{1/2} \pm k) = 2^{-2^{\mp k}} (1 + O(\frac{\ln n}{n} 2^{\mp k})) \quad (14)$$

Expression (14) allows us to obtain an accurate evaluation of $E[L]$ using (11).

$$E[L] = \log_2 n - c + O(\frac{\ln n}{n}) \quad (15)$$

where c is a constant $c \approx 0.66751\ldots$
Finally, we obtain that the average propagation delay in a ripple carry adder is approximately

$$\bar{T} = \tau(\log_2 n + 1.332) \quad (16)$$

Let us consider now another version of the problem. Suppose, we have m such n-bit adders working in parallel. Then the delay is equal to the maximum delay among all m adders. The latter is proportional to the maximum length L_{max} of the runs of $1's$ in all m adders. The cumulative distribution function of L_{max} is

$$L_{max}(l,n) = F_L^m \quad (17)$$

As a result, the expected value of $E[L_{max}]$ is:

$$E[L_{max}] = \log_2 mn - 0.668 \quad (18)$$

4. Conclusion

The results obtained above give the exact value of the average carry propagation delay up to terms of order $O(\frac{\ln n}{n})$ for randomly distributed input values. The important fact is that, because of the narrowness of the delay distribution, the time variations of the delay are rather limited. In the case of m adders working in parallel, the average delay increases quite modestly by $\tau \log_2 m$.

It is interesting to compare the results obtained under the assumption random input statistics with empirical data. There exists experimental evidence [3] that data arithmetic operations carry propagation chains can be very long, exceeding $n/2$.

One possible explanation of this effect is that very often in such computations a small number (that has a long run of 0's starting with the leftmost digit) is subtracted from another small number, which creates a long sequence of runners. If this is the case the situation can be corrected by special means with small hardware overhead.

References.

[1] A. W. Burks, H. H. Goldstine, and J. von Neumann, "Preliminary disscussion of the logical design of an electronic computing instrument", in Collected Works of John von Neumann", A. H. Taub, ed., vol.5, New York, Macmillan, 1963, pp.34-79

[2] B. E. Briley, ;Some new results on average worst case carry', IEEE Trans. on Computers, vol C-22, No 5, May 1973, pp. 459-463.

[3] K. Y. Yun, P. A. Beerel, V. Vakilotjar, A. E. Dooply, J. Arceo, "The design and verification of a high-performance low-control-overhead asynchronous differential equation solver", IEEE Conference, 1997, pp. 140-153