Carry Look-Ahead Adder

The carry look-ahead adder is based on computing the carry bits C_i prior to the summation. The carry look-ahead logic makes use of the relationship between the carry bits C_i and the input bits A_i and B_i. We define two variables G_i and P_i, named as the generate and the propagate functions, as follows:

$$\begin{aligned} G_i &= A_i B_i \ , \\ P_i &= A_i + B_i \ . \end{aligned}$$

Then, we expand C_1 in terms of G_0 and P_0, and the input carry C_0 as

$$C_1 = A_0 B_0 + C_0(A_0 + B_0) = G_0 + C_0 P_0 \ .$$

Similarly, C_2 is expanded in terms G_1, P_1, and C_1 as

$$C_1 = G_1 + C_1 P_1 \ .$$

When we substitute C_1 in the above equation with the value of C_1 in the preceding equation, we obtain C_1 in terms G_0, G_1, P_0, P_1, and C_0 as

$$C_1 = G_1 + C_1 P_1 = G_1 + (G_0 + C_0 P_0) P_1 = G_1 + G_0 P_1 + C_0 P_0 P_1 \ .$$

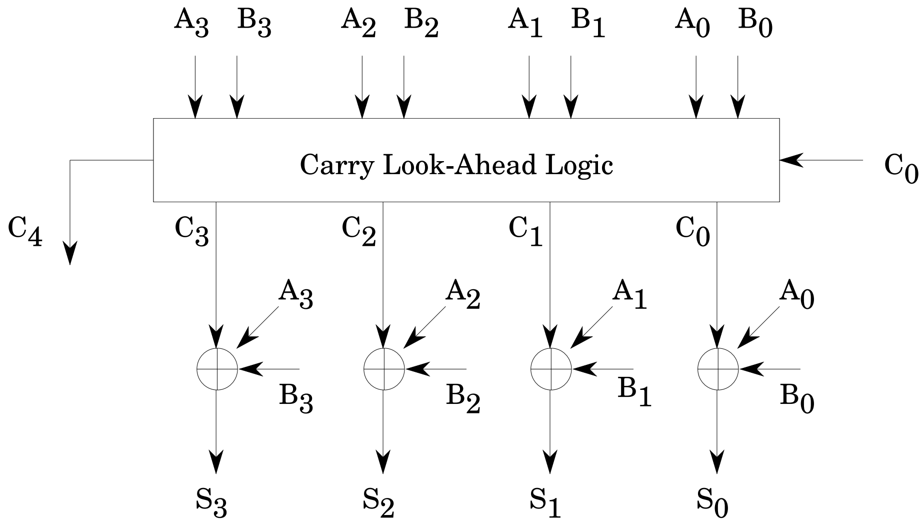
Proceeding in this fashion, we can obtain C_i as function of C_0 and $G_0, G_1, \ldots, G_i$ and $P_0, P_1, \ldots, P_i$. The carry functions up to C_4 are given below:

$$\begin{aligned} C_1 &= G_0 + C_0 P_0 \ , \\ C_2 &= G_1 + G_0 P_1 + C_0 P_0 P_1 \ , \\ C_3 &= G_2 + G_1 P_2 + G_0 P_1 P_2 + C_0 P_0 P_1 P_2 \ , \\ C_4 &= G_3 + G_2 P_3 + G_1 P_2 P_3 + G_0 P_1 P_2 P_3 + C_0 P_0 P_1 P_2 P_3 \ . \end{aligned}$$

The carry look-ahead logic uses these functions in order to compute all C_is in advance, and then feeds these values to an array of EXOR gates to compute the sum vector S. The $i$the element of the sum vector is computed using

$$S_i = A_i \oplus B_i \oplus C_i \ .$$

The carry look-ahead adder for $k = 3$ is illustrated below.



The CLA does not scale up very easily. In order to deal with large operands, we have basically two approaches:

- The block carry look-ahead adder: First we build small (4-bit or 8-bit) carry look-ahead logic cells with section generate and propagate functions, and then stack these to build larger carry look-ahead adders [2, 7, 3].

- The complete carry look-ahead adder: We build a complete carry look-ahead logic for the given operand size. In order to accomplish this task, the carry look-ahead functions are formulated in a way to allow the use of the parallel prefix circuits [1, 4, 5].

The total delay of the carry look-ahead adder is $O(\log k)$ which can be significantly less than the carry propagate adder. There is a penalty paid for this gain: The area increases. The block carry look-ahead adders require $O(k \log k)$ area, while the complete carry look-ahead adders require $O(k)$ area by making use of efficient parallel prefix circuits [5, 6]. It seems that a carry look-ahead adder larger than 256 bits is not cost effective, considering the fact there are better alternatives, e.g., the carry save adders. Even by employing block carry look-ahead approaches, a carry look-ahead adder with 1024 bits seems not feasible or cost effective.

References

[1] R. P. Brent and H. T. Kung. A regular layout for parallel adders. *IEEE Transactions on Computers*, 31(3):260–264, March 1982.

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

[3] I. Koren. *Computer Arithmetic Algorithms.* Prentice-Hall, 1993.

[4] D. C. Kozen. *The Design and Analysis of Algorithms.* Springer, 1992.

[5] R. Ladner and M. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838, October 1980.

[6] S. Lakshmivarahan and S. K. Dhall. *Parallelism in the Prefix Problem.* Oxford University Press, 1994. In press.

[7] S. Waser and M. J. Flynn. *Introduction to Arithmetic for Digital System Designers.* Holt, Rinehart and Winston, 1982.