

Carry Lookahead Adder

Chris Calabro

May 17, 2006

The standard method for adding 2 n -bit natural numbers $x = (x_{n-1} \cdots x_0)_2, y = (y_{n-1} \cdots y_0)_2$, ripple carry addition (RCA), is not optimal for parallel computation, at least in the bounded-fanin circuit model, using $\Theta(n)$ depth and $\Theta(n)$ size. Carry lookahead addition (CLAA), to be described shortly, requires less depth ($\Theta(\lg n)$), but more size ($\Theta(n^2)$).

The computation model used throughout will be fanin 2 circuits with arbitrary gates at each node.

1 Ripple carry addition

We can construct a 3-bit adder $+_3$ to compute the sum s and carry bit c' of the sum of c, x, y as follows:

$$\begin{aligned} s &= c \oplus x \oplus y \\ c' &= \text{majority}(c, x, y) = (c \wedge x) \vee (c \wedge y) \vee (x \wedge y). \end{aligned}$$

A circuit of depth 3, size 7 suffices to compute s, c' . Similarly, we can construct a 2-bit adder $+_2$ of depth 1, size 2 as follows:

$$\begin{aligned} s &= x \oplus y \\ c' &= x \wedge y. \end{aligned}$$

To compute the sum $s = (s_n \cdots s_0)_2$ of $x = (x_{n-1} \cdots x_0)_2, y = (y_{n-1} \cdots y_0)_2$, we can use the recursive equations

$$\begin{aligned} (s_0, c_1) &= +_2(x_0, y_0) \\ (s_i, c_{i+1}) &= +_3(c_i, x_i, y_i), \text{ for } i = 1, \dots, n-1 \\ s_n &= c_n. \end{aligned}$$

This yields a circuit of depth $\Theta(n)$, size $\Theta(n)$.

2 Carry lookahead adder

We could compute s_i with a shallow circuit if c_i were given as an input in addition to x_i, y_i . Our goal will be to compute the c_i in parallel using only log depth. There are 2 key observations that allow us to do this:

- $c_i = 1$ iff the (x_j, y_j) pairs preceding (x_i, y_i) are of the form $(0, 1)$ or $(1, 0)$ and then are followed by a $(1, 1)$; i.e.

$$\exists 0 \leq j < i \ (\forall j < k < i \ x_k \oplus y_k = 1 \wedge x_j = y_j = 1).$$

- both \wedge, \vee are associative and so a chain of $O(n)$ of them can be computed by a circuit of $O(\lg n)$ depth.

We use the following (nonrecursive) equations to compute s :

$$\begin{aligned} z_i &= x_i \oplus y_i, & \text{for } i = 0, \dots, n-1 \\ c_i &= \bigvee_{j=1}^{i-1} \left(\bigwedge_{k=j+1}^{i-1} z_k \wedge x_j \wedge y_j \right), & \text{for } i = 0, \dots, n \\ s_i &= z_i \oplus c_i, & \text{for } i = 0, \dots, n-1 \\ s_n &= c_n. \end{aligned}$$

The big AND and the big OR can each be computed in $\Theta(\lg n)$ depth.

To compute the size of the circuit, notice that there are $\Omega(n^2)$ expressions of the form $\bigwedge_{k=j+1}^{i-1} z_k$, the value of which must appear at some node of the circuit. Each of these expressions is a distinct function of the inputs and so each must correspond to a distinct node. So the circuit has size $\Omega(n^2)$. On the other hand, we can compute the $\bigwedge_{k=j+1}^{i-1} z_k$ expressions using $O(n^2)$ nodes; and from these, we can compute the big OR expressions using only $O(n^2)$ more nodes. So the circuit has size $\Theta(n^2)$.