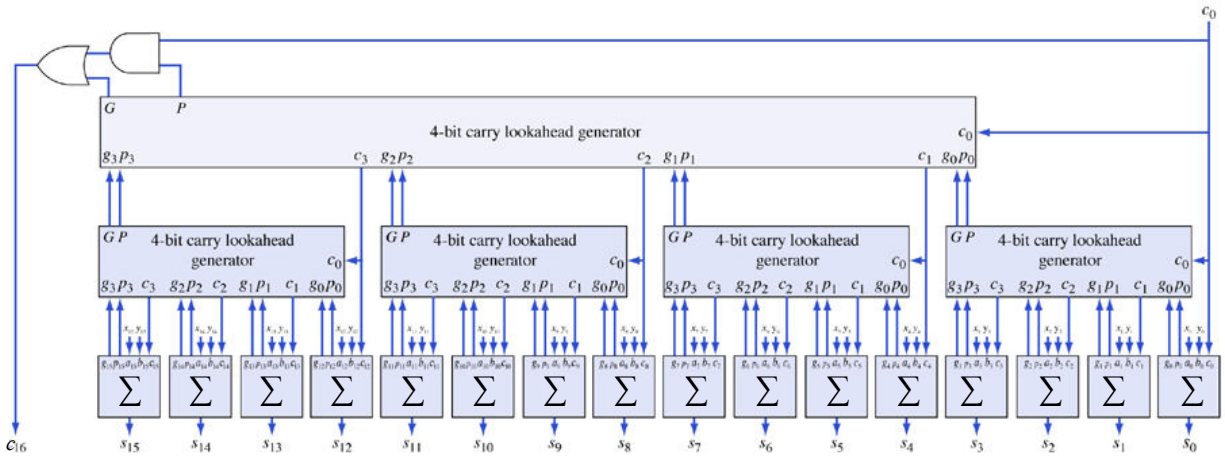
Hierarchical Carry Lookahead Adder

These notes refer to the last slides of lecture 10.



A hierarichal structure of carry lookahead generators is used to reduce the delay (that is, the number of levels of logic gates) in computing the carries, and hence, the summation of two binary numbers.

An adder of two 16-bit numbers contains 3 "floors":

- 16 $\sum$-components in the lower floor.

  The inputs to the *i*-th $\sum$-component are:

  - x_i, y_i : the *i*-th pair of bits to be added
  - c_i : the carry from the summation of $x_{i-1}, y_{i-1}, c_{i-1}$.

  The outputs of the *i*-th $\sum$-component are:

  - p_i : an indicator whether the carry was propagated through the component. Computed using 1 level of logic: $p_i = x_i \oplus y_i$
  - g_i : an indicator whether the carry was generated in the component. Computed using 1 level of logic: $g_i = x_i y_i$
  - s_i : the *i*-th bit of the result. $s_i = p_i \oplus c_i$, therefore, once c_i is available, it takes one more unit of time to compute s_i.

- Four 4-bit carry lookahead generators in the second floor.

This generator is used for a fast computation of the carries. Its inputs are:

- c_0: the carry from the previous generator in the same floor.
- p_0, p_1, p_2, p_3: Four carry-propagated indicators from the four "carry generators"[1] in previous floor that are associated with this generator.
- g_0, g_1, g_2, g_3: Four carry-generated indicators from the four "carry generators" in previous floor.

The outputs of a carry lookahead generator are:

- c_1, c_2, c_3: three carries to be used by the components in the previous floor. They are computed as follows:

  $c_1 = c_0 p_0 + g_0$

  $c_2 = c_0 p_0 p_1 + g_0 p_1 + g_1$

  $c_3 = c_0 p_0 p_1 p_2 + g_0 p_1 p_2 + g_1 p_2 + g_3$

  Note that the carries can be computed using 2 levels of logic once the inputs $c_0, g_0, g_1, g_2, g_3, p_0, p_1, p_2$ of the generator are available.
- P: an indicator whether a carry was propagated through the component. Computed using 1 level of logic: $P = p_0 p_1 p_2 p_3$. In words, a carry is propagated through the entire component if and only if it was propagated through every one of the associated components in the floor below.
- G: an indicator whether a carry was generated in the component. Computed using 2 levels of logic: $G = g_0 p_1 p_2 p_3 + g_1 p_2 p_3 + g_2 p_3 + g_3$. In words, a carry is generated in the entire component if and only if it was generated in one of the associated components in the floor below, and was propagated through the rest of them. For example, the term $g_1 p_2 p_3$ represents a carry that is generated by component 1 and propagated through components 2 and 3.

- The third floor contains one carry lookahead generator. It is used to compute the input carries for the carry lookahead generators in the second floor, and to compute the "carry-propagated" and "carry-generated" bits for the entire 16-bit adder. This bits

[1] Note that the $\sum$-components can also be considered "carry generators" as they produce the carry-propagated and carry-generated indicators.

can then be used either to compute the carry of the entire adder - c_{16}, or as the inputs of a generator in a higher floor of, say, a 64-bit adder.

Let us analyze the delay of this 16-bit adder:[2]

After one unit of time, the output bits p_i, g_i of every ∑-component are available. These values are used to compute the *P* and *G* bits of every generator in the second floor, using 2 levels of logic. The *P* and *G* bits of the second floor are the inputs to the generator in the third floor. Therefore, after additional 2 units of time, all the carries computed by the generator in the third floor are available (note that the input carry for this generator is the input carry for the entire adder and therefore it is available from the beginning).

At this point the input carry of every generator in the second floor is available, and therefore each such generator can compute all of its output carries (after additional two units of time). Finally, the input carry of every ∑-component is now available and thus the result bits (s_i) can be computed (using one more level of logic). Therefore, the total delay of the circuit is 1+2+2+2+1=8 units of time.

Note that every floor causes a delay of at most 4 units of time: at most 2 units when we compute the "carry-generated" and "carry-propagated" bits when "going up" the hierarchy; and at most 2 units when we computed the carries when "going down" the hierarchy. The delay of a hierarchial carry lookahead adder is therefore proportional to the number of "floors" in the adder which is, roughly, $\log_4 n$ for an *n*-bit adder. This is much better than the delays of a ripple carry adder and a chain of carry lookahead adders, which are proportional to the number of bits.

For example, the delay of a 64-bit hierarchical adder would be $\sim 4\log_4 64 = 12$ units of time. This adder would have 64 ∑-components in its first floor; 16 generators in the second floor, 4 generators in the third floor, and one generator in the forth floor.

A 64-bit ripple carry adder would have a delay of ~2·64=128 units of time, while a chain of 16 4-bit carry lookahead adders would have a delay of ~3·64/4=48 units of time.

[2] We assume the delay of every gate is one unit of time.