Designing With XC5200 Carry Logic

Simple design makes XC5200 carry logic even more flexible than that of XC4000 Series

The XC5200 FPGA's carry logic is deceptively simple. While this architecture only provides a series of multiplexers interconnected by dedicated carry nets, the use of function generators to control these multiplexers makes the XC5200 FPGA's carry logic extremely flexible — more flexible, in fact, than the XC4000 carry logic with its multiplicity of modes.

In the XC5200 architecture, one bit of a simple adder uses two function generators and a carry chain multiplexer, as shown in **Figure 1**. However, not all of the function generator's capability is utilized in the basic adder. In the input function generator, the adder input, b_i, can be any function of the three inputs that are available. For example, one of these inputs could be used as an add/subtract control, inverting b_i in an XOR gate when necessary.

Similarly, the adder output can be combined in any way with the two remaining function generator inputs. In a typical counter application, these might be used for a multiplexer to load the counter or, alternatively, for an AND gate to provide a synchronous clear.

A more complex example uses this additional capability in a carry-select adder that trades additional logic for higher performance (**Figure 2**). The adder is divided into two sections. The lower half operates normally, but in the upper half two carry chains are used in parallel, one initialized with a '0' and the other with a '1.' The carry output of the lower half is used to select which carry chain is used to complete the sum. The carry propagates simultaneously in the upper and lower halves of the adder; thus, the settling time is reduced.

Since a single function generator can both select the carry and complete the sum, the incremental cost over the basic adder is only 25%. The direct access to the carry chain minimizes the timing overhead associated with this technique, which is effective even for relatively short adders.
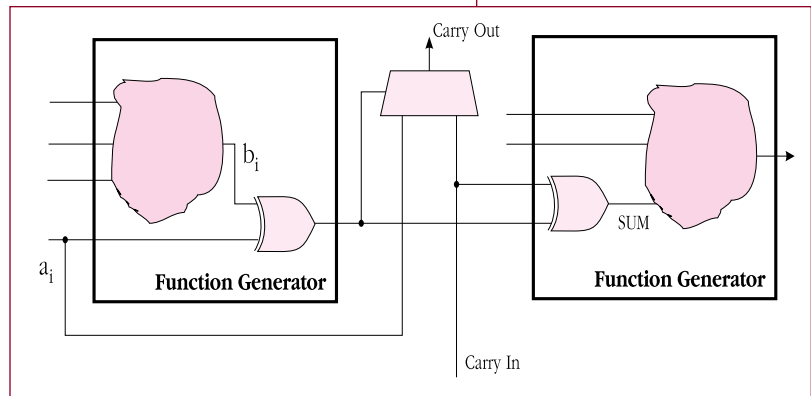
The corresponding acceleration technique for long adders in the XC4000 architecture is the conditional-sum technique. In the upper half of a conditional-sum adder, two complete adders are implemented with '0' and '1' carry inputs, and

Figure 1:
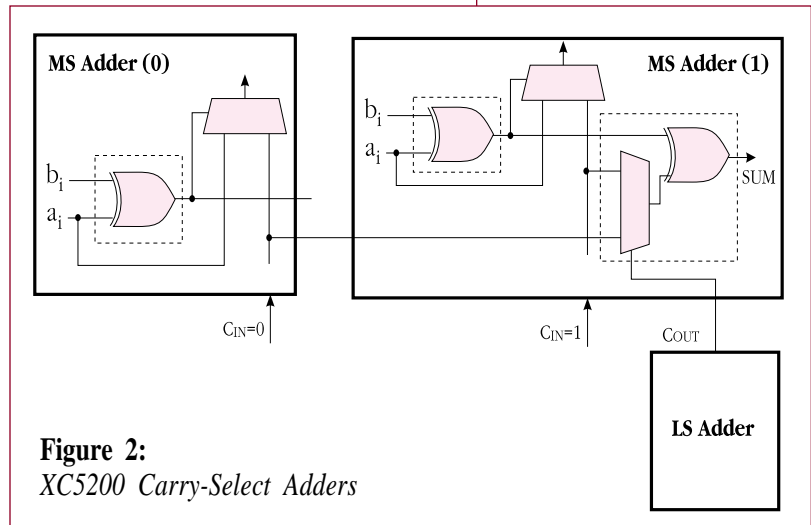*XC5200 Carry Structure*

Figure 2:
XC5200 Carry-Select Adders

23

XC5200 Carry Logic

Continued from the previous page

additional CLBs are required to select between the outputs of these adders according to the carry output from the lower half. The technique is only effective for relatively long adders; the incremental cost is 100%.

It is interesting to note that, in the XC5200 architecture, the carry-select and conditional-sum techniques cannot always

In a simple multiplier, the product is computed using a cascade of gated adders together with appropriate shifts (**Figure 3**). A more detailed view of a single bit slice through the first two adders of this multiplier is shown in **Figure 4**. One of the two AND gates that precedes the first adder is combined into the function generator that controls the carry multiplexer. The other
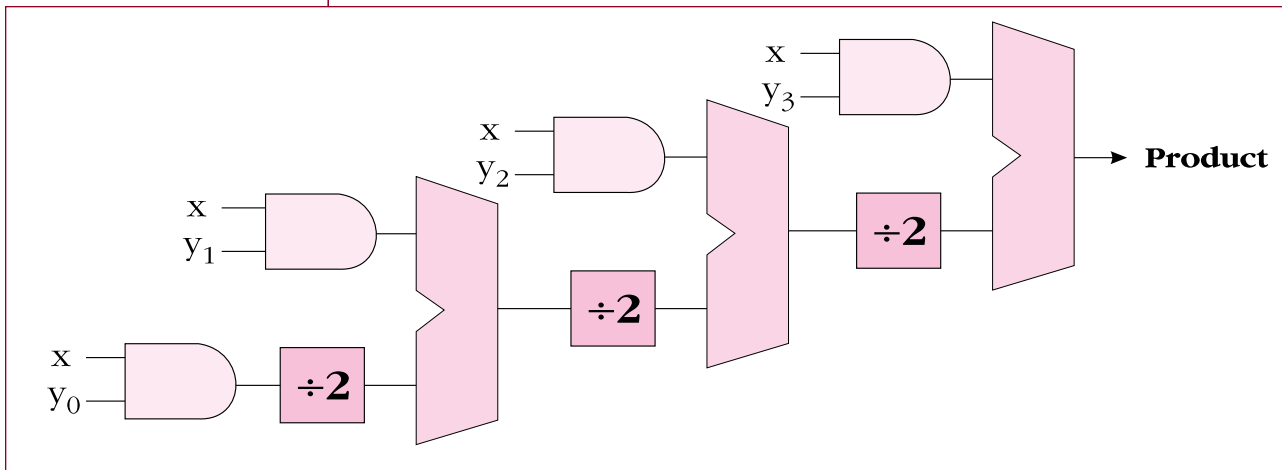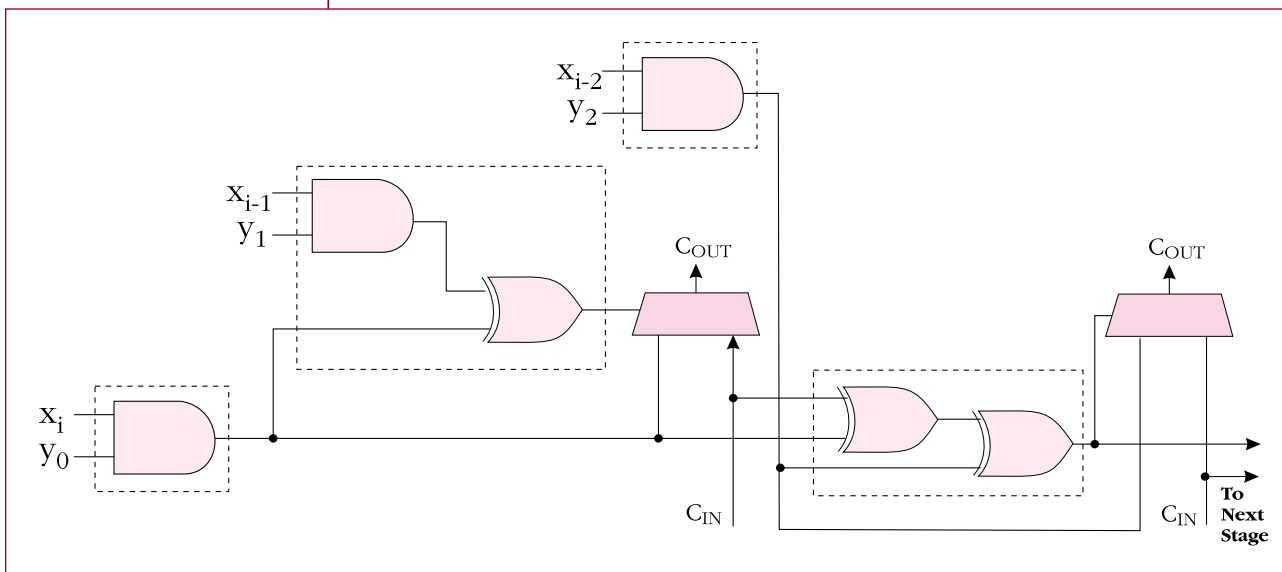

Figure 3:
4-Bit Cascade Multiplier

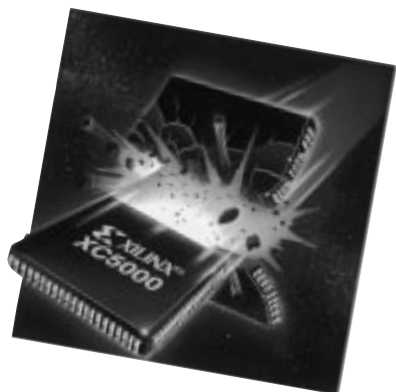**Figure 4:**
Bit-Slice of First Two Adders

be distinguished. Both sums of the conditional-sum adder can be completed in the same function generator. This function generator can be further used to select between them. If this is done, all the function-generator truth tables and all the interconnections become identical to the carry-select adder. This should not be surprising, however, since both techniques implement the same uniquely defined adder function.

AND gate, however, must use a separate function generator because there is a fanout point at the second input to the adder.

In the second and subsequent adders, the AND gates again use separate function generators. This permits the XOR gates that complete the previous sums to be merged adder inputs, thus saving both logic resources and critical-path delay. If, instead, the AND gates had been merged into the

24

Figure 5:
Adder-Tree Multiplier

inputs, the XOR gates would have required function generators. In this case, the re-source utilization would have been the same, but the delay longer. Only in the final adder is a separate function generator needed for the XOR gate.

The speed of the multiplier can be increased by rearranging the cascade of adders into a tree, as in **Figure 5**. After the first set of adders, the input of each adder can absorb the output XOR gate of one of the two preceding adders, as described in the cascade multiplier. The other preceding adder, however, must use a separate function generator for its XOR gate.

In the first set of adders, the objective is to multiply the X input by bit pairs of Y, creating products that are 0X, 1X, 2X or 3X. This is usually achieved by gating X and 2X into an adder, as is shown in Figure 5.

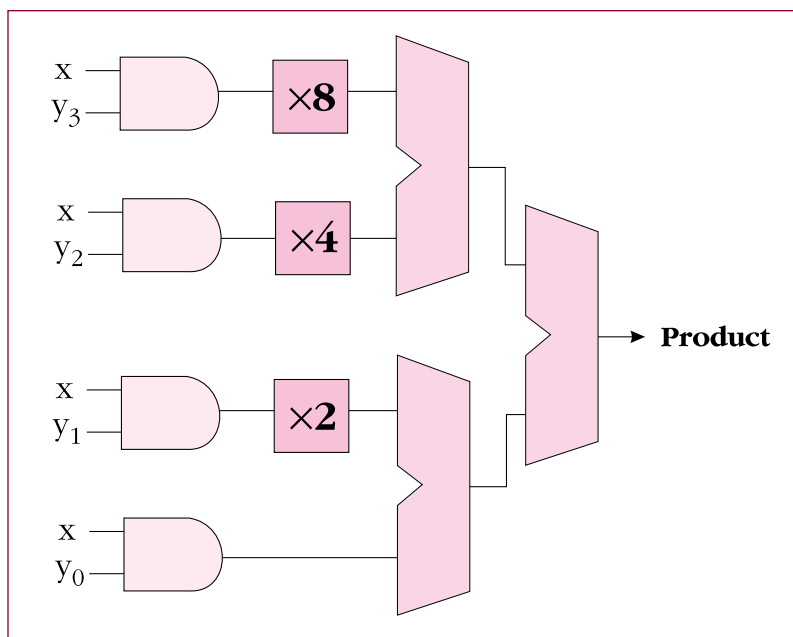However, a different, more direct approach may be used in XC5200 FPGA.

A bit slice of this more efficient approach is shown in **Figure 6**. The carry chain is only used to add X and 2X when the 3X output is required. Otherwise, 0, X or 2X is se-lected in the first function generator and routed to the second function generator where it passes through unaltered. While the carry chain continues to operate in a meaningless way, it cannot damage itself, and its outputs are not required. When 3X is required, the first function generator XORs X and 2X to control the carry chain in a normal addition. The sum is completed in the XOR gate that precedes the multi-plexer in the second function generator. ◆

25

Figure 6:
First Stage of a Tree-Adder Multiplier