Advanced Carry-Logic Techniques

All XC4000 series FPGAs provide dedicated carry generation logic within configurable logic blocks (CLBs), and dedicated routing to propagate carry signals between CLBs. Most basic carry-logic applications, like adders and counters, are supported by library macros. However, the carry logic can be used in other ways for more specialized circuits.

MULTIPLICATION

Perhaps the simplest adaptation of the carry logic is to provide a multiplier function. In a multiplier, one of the input words, X, is ANDed separately with each bit of the other input word, Y, and the resulting words are summed with appropriate binary weighting. Consequently, there is a need for gated adders, such as shown in **Figure 1(a).**

However, it is inefficient to implement this function as drawn. The carry logic connects directly to the input pins of the CLB, and there is no provision to dynamically gate the carry logic inputs. Instead of using an additional CLB for the gating, the function can be modified as shown in **Figure 1(b).** This circuit is functionally equivalent to the one of Figure 1(a), provided that all inputs are gated with the same signal, and the carry signal is not used directly.

Figure 1(b) can be implemented easily, as shown in **Figure 2**. Each bit of a conventional adder is modified such that the unconditional sum is created as an internal node in the function generator. This sum is multiplexed with the partial sum to the adder, which is already available within the function generator. The gating signal, Y_i, controls the multiplexer, and is brought in on a spare pin.
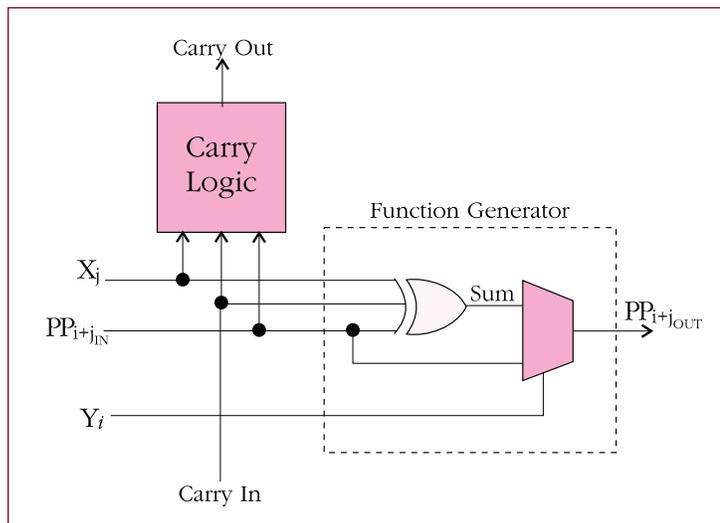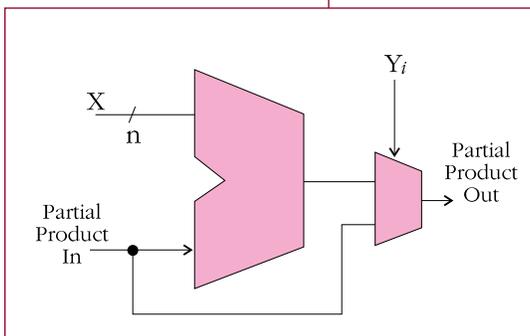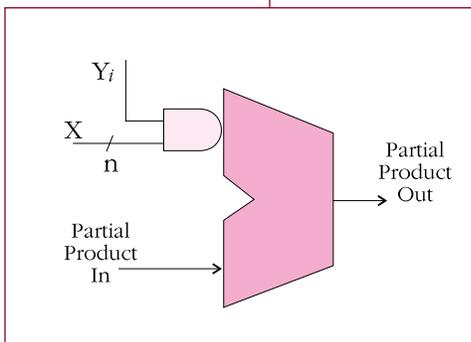

Figures 1(a), top left, & **1(b)**, bottom left, *Two Versions of a Gated Adder*

Figure 2
Implementation of a Gated Adder

42

2'S COMPLEMENT AND ABSOLUTE VALUE

The strategy for generating an absolute value is similar to that used in multiplication. The 2's complement of the input is generated unconditionally as an internal signal in the function generator. The sign of the input value is then used to select between the input directly or the 2's complement. Thus, the output is always positive.

The 2's complement function is best implemented by decrementing the input and inverting the result. This alternative to the traditional invert-and-add-one technique gives exactly the same result, but avoids having to modify data at the input to the carry logic.

Figure 3 shows how one bit of a standard decrementer is modified to provide the absolute value function. The inversion in front of the XOR gate is a part of the decrement, while the inversion at the output completes the generation of the 2's complement.

**Figure 3**
Absolute Value Generator
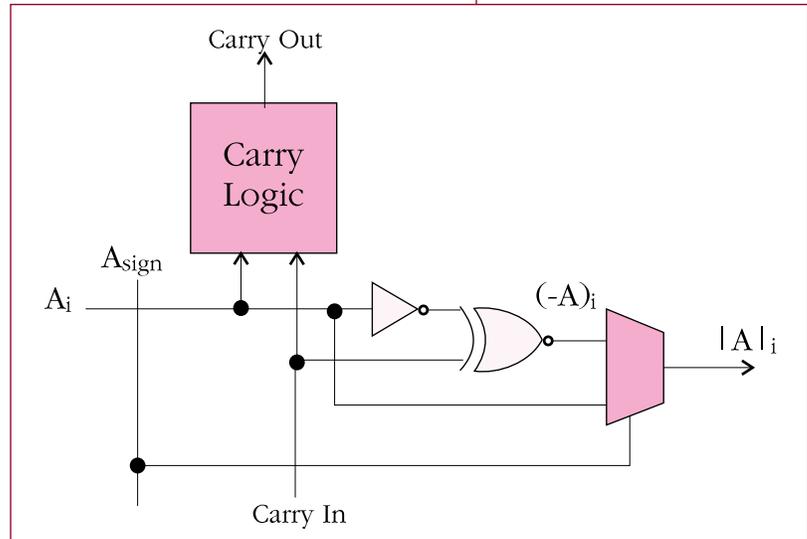
ABSOLUTE DIFFERENCE

Where time allows, the absolute value of a difference, |A–B|, can be calculated using a single carry chain. This is essentially a two-stage operation, and can be completed in two successive clock periods or in the two phases of a single clock period.

The technique depends upon two alternative methods of subtracting. Traditionally, B is inverted at the input of an adder, and the carry is asserted to give the result A–B, as in **Figure 4(a)**. However, if both B and the output of the adder are inverted, and the incoming carry is not asserted, the result is B–A, as in **Figure 4(b)**.

The standard subtractor used with XC4000 carry logic is of the traditional variety. A configuration bit causes the B input to the carry logic to be inverted, and this cannot be changed dynamically. However, this inversion is common to both methods of subtraction described above. The carry input to the adder can easily be made a dynamic input, and there is space in the function generator to add an XOR gate that inverts the output

Continued on the next page

Figures 4(a), left, & **4(b)**
Two Ways to Subtract
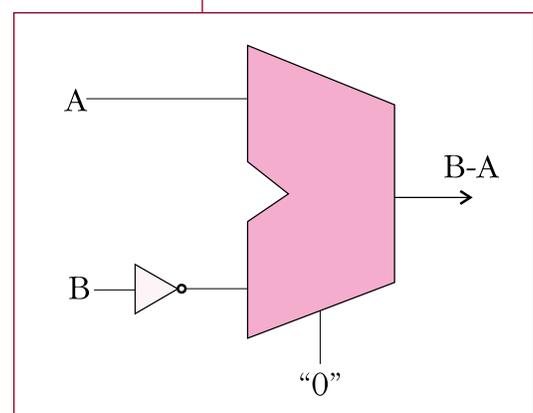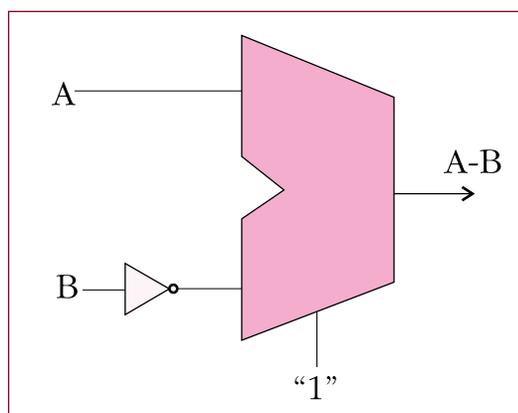
Carry-Logic

Continued from the previous page

when needed. Thus, it is possible to generate either A–B or B–A on demand.

Given this capability, the absolute difference is obtained by simply choosing the function that yields a positive result. However, directly feeding back the sign of the output to select the function will result in instability, and a flip-flop must be added to eliminate this possibility (**Figure 5**).

In the first of two operations on the same inputs, either subtraction can be performed. If the first result is negative, the flip-flop is toggled to select the other function to achieve a positive result in the second operation. If the first result is positive, the flip-flop is not toggled, and the first operation is repeated. In either case, the result of the second operation is positive.
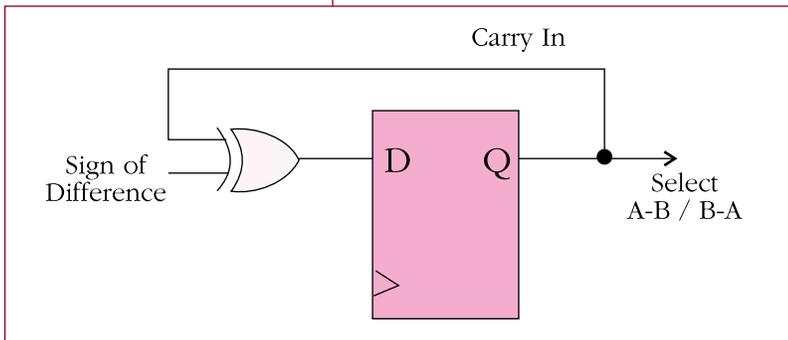
Figure 5
Function Select Flip-Flop

PEAKDETECTOR

In a peak detector, the current peak value is stored in a register, and is subtracted from every new input value. If the difference is positive, the new input is larger, and it replaces the value in the register as a new peak. Otherwise, the register is unchanged.

Only the sign of the difference is of interest in determining whether the register is updated or not. The other difference bits need not be generated, and the corresponding function generators are free to be used in controlling the register.

Figure 6(a) shows a typical bit. The sign of the difference is routed to all bits to select the value loaded into the register. This operation includes the sign bit of the register. Consequently, the subtraction must be sign-extended by one bit so that the sign of the difference is also available.

The peak detector can be reset by forcing the sign output to a one. This causes the current input value to be loaded as a new peak, regardless of the value of the previous peak.

The circuit described above is equivalent to using the sign of the difference to enable the peak register, and the CLB Enable Clock pin could be used equally well. However, the two techniques impose different routing constraints, and either may be more effective than the other in different situations.

If enable clock is used, the function generators can be used for other purposes. For example, to initialize the peak detector with a predetermined value that is only changed by a peak that exceeds it, as in **Figure 6(b)**. ◆

Figures 6(a), left, & **6(b)**
Two Peak Detectors