Summary
Using the XC4000E dedicated carry logic, the performance of adders and counters can easily be predicted. This Application Note provides formulae for estimating the performance of such adders and counters.

Xilinx Family
XC4000E, XC4000L

Demonstrates
Dedicated Carry Logic

Introduction
In most LCA designs, performance cannot be estimated with any accuracy until after implementation. This is because the performance is affected by routing delays; and, prior to implementation, these are not known. However, in adders and counters using the XC4000E dedicated carry logic, delay estimation is possible.

The carry path in an adder uses dedicated interconnects between CLBs. These interconnects introduce a fixed delay, even when the carry passes from one CLB column to the next at the top or bottom of the array. This permits the routing delay to be incorporated into the CLB specifications published in the data book. Consequently, the propagation delay through an adder can be calculated directly from the data book specifications.

For a typical adder, this calculation can be reduced to a simple formula. In an XC4000E-3, the maximum propagation delay from the operand input to the sum output of an N-bit adder is approximately

\[
t_{pd} = 4.5 + 0.35N \text{ ns}
\]

This estimate does not include the delay from the operand source register to the adder or any additional delay reaching the destination register. However, it is still a useful benchmark.

This formula applies only to simple ripple-carry adders. However, such adders are adequate in most situations; conditional-sum and other adder-acceleration schemes are only appropriate for adders longer than 24 bits.

For an N-bit counter, the minimum clock period that permits the carry path time to settle is approximately

\[
t_{clk-\text{clk}} = 9.5 + 0.35N \text{ ns}
\]

The following discussion describes how these formulae were derived, under what conditions they apply, and the corrections that must be made when these conditions are not met.
If a carry output or overflow flag is generated, an additional CLB at the most significant end of the counter is required. Consequently, the delay to these outputs is one $T_{\text{BYP}}$ delay (0.7 ns) longer than to the MSB output.

An alternative organization for the adder (Figure 2) places the LSB and the MSB into individual CLBs, with each pair of intervening bits sharing a CLB. This organization results in one additional pair of intervening bits. Consequently, an additional $T_{\text{BYP}}$ delay (0.7 ns) is incurred in all paths using the carry chain.

In this organization, the carry chain can be initiated in the CLB used to implement the LSB of the adder. In this case, the delay from the carry input is faster that the delay from the operand LSB. The delay is reduced by $T_{\text{OPCY}}$ minus $T_{\text{INCY}}$ (0.9 ns).

**Figure 1: Basic Adder Organization**

If a carry output or overflow flag is generated, an additional CLB at the most significant end of the counter is required. Consequently, the delay to these outputs is one $T_{\text{BYP}}$ delay (0.7 ns) longer than to the MSB output.

An alternative organization for the adder (Figure 2) places the LSB and the MSB into individual CLBs, with each pair of intervening bits sharing a CLB. This organization results in one additional pair of intervening bits. Consequently, an additional $T_{\text{BYP}}$ delay (0.7 ns) is incurred in all paths using the carry chain.

In this organization, the carry chain can be initiated in the CLB used to implement the LSB of the adder. In this case, the delay from the carry input is faster that the delay from the operand LSB. The delay is reduced by $T_{\text{OPCY}}$ minus $T_{\text{INCY}}$ (0.9 ns).

**Figure 2: Alternative Adder Organization**

In the CLB implementing the MSB of the adder, it is possible to generate either a carry output or an overflow flag, but not both. The delay to this additional output is the same as to the MSB of the adder. If both carry and overflow are required, an additional CLB must be used for one of them, and the signal generated in this CLB incurs an additional $T_{\text{BYP}}$ delay (0.7 ns).

The organization of an adder with an odd number of bits is a hybrid of the two organizations discussed above. One end of the adder has two bits sharing a CLB, while the other end has a single bit in a CLB. Either end may have the shared CLB, and this end matches the first organization. The other end, with a single bit in a CLB, matches the second organization.

For delay calculations, the number of bits should be rounded up to an even number. The basic delay formula can then be applied without correction.

If the single bit is at the most significant end of the counter, the least significant end of the counter matches the first organization. If a carry input is provided, the delay from this input must use the adjustment for the first organization. The most significant end of the counter matches the second organization, and delays to carry-out or overflow must use
the corrections for that organization. If the single bit is at the least significant end of the counter, this situation must be reversed.

The set-up time from the carry chain to flip-flops in the same CLB matches the CLB output delay from the carry chain. Consequently, all the delays discussed above can also be considered as set-up times to the register contained in the same CLBs as the adder. Different delay formulae must be derived for adders not organized with two bits per CLB.

**Subtracters and Adder/Subtracters**

The performance analysis, described above, also applies to subtracters and adder/subtracters. In an adder/subtractor, however, there is an additional add/subtract control input that must be considered.

To estimate the add/subtract-to-carry delay, the operand-to-output delay, appropriate to the organization, must be modified. Its operand-to-carry delay ($T_{OPCY}$) must be replaced by an add/subtract-to-carry delay ($T_{ASCY}$). This causes an increase of 1.8 ns.

This increase also applies to delays from the add/subtract input to the carry output or overflow flag.

**Counters**

The performance of carry-logic-based counters implemented with two bits per CLB can be estimated in a similar way. These include loadable up counters and down counters, and non-loadable up/down counters.

As stated above, all of the delay estimates may also be considered set-up time estimates when using the register in the same CLB as the adder. This also applies to an incrementer or decrementer used to implement a counter.

To estimate the minimum clock period, the delay from the register to the incrementer/decrementer must be added to the incrementer/decrementer set-up time, as shown in Figure 3. This additional delay involves a clock-to-output delay ($T_{CKO} = 2.6$ ns) plus a typical routing delay of 1.2 ns. In addition, $T_{SUM}$ is replaced by $T_{CCK}$. Consequently, the minimum clock period for an N-bit counter is

$$t_{clk-clk} = 9.5 + 0.35N \text{ ns}$$

This assumes a counter with an even number of bits, organized in the same way as the first adder. If the alternative organization is used, the clock period must receive the same 0.7 ns adjustment that was applied to the adder delay. The carry input to the incrementer/decrementer may be used as a count enable, and the same set-up-time estimate applies. Also, the carry output may be used as terminal count. The delay from the clock to the terminal count output is the minimum clock period with any correction that might be necessary for estimating the carry-out delay with the equivalent organization.

In a non-loadable up/down counter, the add/subtract control becomes up/down. The estimate for add/subtract-to-output delay is equivalent to the set-up time for the up/down control. Loadable up/down counters cannot be organized such that these formulae can be applied.

**Other Speed Grades**

Similar estimation formulae can be derived for other speed grades. For an XC4000E-4, the basic operand-to-output delay for an N-bit adder is

$$t_{pd} = 5 + 0.5N \text{ ns}$$

The 0.7 ns correction factor, used above, increases to 1 ns, in all cases. The delay increase from the add/subtract input becomes 1.3 ns.

The minimum clock period for a counter is

$$t_{clk-clk} = 11.5 + 0.5N \text{ ns}$$

---

**Figure 3: Basic Counter Configuration**