Ripple-Carry Gray Code Counters in FLEX 8000 Devices

May 1994, ver. 1 Application Brief 135

Summary

Files using the techniques described in this application brief are available from the Altera BBS at (408) 954-0104 in the following self-extracting file:

ab_135.exe

Ripple-carry Gray code counters are counters in which only one bit switches at a time. Ripple-carry Gray code counters take advantage of the dedicated carry chain feature available in FLEX 8000 devices. A carry chain is a fast (less than 1 ns) carry-forward function path between contiguous logic elements (LEs) within a Logic Array Block (LAB) and between adjacent LABs. Ripple-carry Gray code counters provide high-speed carry generation and use LE resources efficiently. Design techniques described in this application brief can be used to create design files optimized for the following characteristics:

Design Goals:		*Design Results:*		
Architecture	**Optimization**	**Width**	**Logic Cells**	**Speed (MHz)**
✓ Look-Up Table	Routability	8 Bits	9	103
Product Term	✓ Speed	16 Bits	17	63
	✓ Area	32 Bits	33	35

Ripple-Carry Gray Code Counters vs. Binary Counters

Ripple-carry Gray code counters have less system-level switching noise and, consequently, less ground plane noise than binary counters have, because multiple bits may switch simultaneously in binary counters. In addition, the output of a ripple-carry Gray code counter can be asynchronously sampled at any time, with a maximum sampling error of one. In contrast, if a binary counter is sampled at or near the Clock edge, some of the flipflops may have transitioned before others, which can cause errors with the asynchronous sample.

Ripple-Carry Gray Code Counters

You can implement a ripple-carry Gray code counter with T flipflops (TFF primitives), and include an additional TFF, called the "dummy" bit, with its T input tied high. The following equation calculates Q_n.T, where Q is the counter inputs and T is the input to the register:

$$(Q_{n-1}..0, \text{ dummy}) = B"100...001"$$

Table 1 shows the detailed counter bit pattern for a 4-bit ripple-carry Gray code counter. For example, Q2 switches when Q1=1, Q0=0, and dummy=1. Q3 switches when Q3 or Q2 is high, Q1 and Q0 are low, and dummy is high.

Table 1. 4-Bit Ripple-Carry Gray Code Counter Bit Pattern

Count Value	Bit Pattern for Q[3..0]	Dummy Bit
0	0000	0
1	0001	1
2	0011	0
3	0010	1
4	0110	0
5	0111	1
6	0101	0
7	0100	1
8	1100	0
9	1101	1
10	1111	0
11	1110	1
12	1010	0
13	1011	1
14	1001	0
15	1000	1

As shown in Table 1, the least significant bit (LSB) toggles when the `dummy` register is low. The highest possible value for a ripple-carry Gray code counter occurs when the most significant bit (MSB) is `1` and all others are `0`. The MSB must toggle when (`Q`$_n$ `$ Q`$_{n-1}$`)` and (`(Q`$_{n-1}$`..0, dummy) = B"000...001"`), which causes the MSB to switch to zero, rolling the counter over from the maximum value to zero.

Application Note 40 has been incorporated into the FLEX 8000 Programmable Logic Device Family Data Sheet.

You can adapt the Up/Down Counter mode available in FLEX 8000 LEs for use with ripple-carry Gray code counters. For more information about the Up/Down Counter mode, see *Application Note 40 (FLEX 8000 Architecture Details)* in this handbook. In a binary counter, the carry chain is used to propagate the `AND` of all preceding bits. However, in a Gray code counter, `Q`$_n$ toggles when `Q[(`*n*`-2)..0] = 0` and the `dummy` bit is high. The carry chain propagates the signal.

Instead of propagating the signal when all the LSBs are high, the carry signal goes high when the `dummy` bit is high and all other bits are low. For example, the carry-out of the fifth logic cell is as follows: `!Q4 & !Q3 & !Q2 & !Q1 & !Q0 & dummy`.

In the Up/Down Counter mode, the output of the register is fed directly into the look-up table (LUT), emulating a `TFF`. The Up/Down Counter mode supports a synchronous load, a synchronous Clear, or an Output

Enable signal. To use a synchronous Clear signal, you must connect the load inputs to GND. To use an Output Enable signal, you must feed the counter outputs back to the load inputs. You can implement an asynchronous load without increasing LE usage. See *Application Note 36 (Designing with FLEX 8000 Devices)* in this handbook for more information.

Figure 1 shows the general Altera Hardware Description Language (AHDL) implementation of a ripple-carry Gray code counter.

Figure 1. AHDL Implementation of a Ripple-Carry Gray Code Counter

```
count0.d       = ldn &(!dummy $ count0) # (!ldn & data0);
countcarry0    = dummy;

count1.d       = ldn & ((count0 & countcarry0) $ count1)# (!ldn & data1);
countcarry1    = countcarry0 & !count0;

count2.d       = ldn & ((count1 & countcarry1) $ count2)# (!ldn & data2);
countcarry2    = countcarry1 & !count1;
```

The counter bits are implemented as D flipflops (DFF primitives), which use the D flipflop structure of the registers in the FLEX 8000 device. When the carry signals that feed the register and the previous bit are both high, each register is toggled (XORed with itself). The carry chain for each LE is the carry signal feeding that LE, ANDed with the inverse of the previous bit. Hence, the carry chain computes the dummy bit ANDed with the inverse of all counter bits.

Designs that use the FLEX 8000 carry chain run relatively fast and use the fewest number of LEs ($n + 1$). However, since carry chains must be placed in contiguous LEs and LABs, longer carry chains may reduce the routing resources available for implementing other logic.

For ripple-carry counters that are not speed-critical, it is not necessary to use the carry chain. However, ignoring the carry chain increases the design's LE usage and decreases its speed.