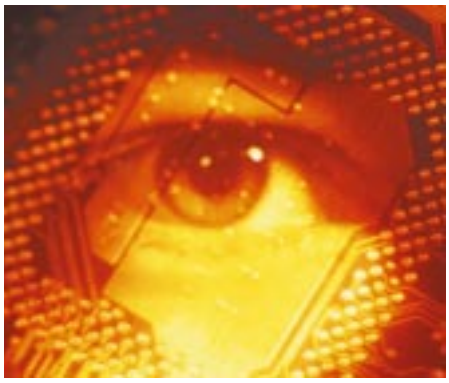
Using the Virtex



LOOK-UP TABLES

The Virtex Look-up Tables have some interesting capabilities that allow you to create very fast and efficient designs.

by Marc Defossez, FAE, Xilinx BeNeLux,
Marc.Defossez@xilinx.com

Xilinx FPGAs have always had combinations of Look-up Tables (LUTs) and flip-flops, combined into Configurable Logic Blocks (CLBs). With the introduction of the XC4000 family, the Xilinx LUTs also have RAM and ROM capability. Now, with introduction of the new Virtex architecture, LUTs can also be used as shift registers.

The Virtex LUT

The Virtex LUTs have four inputs and one output, and can be used as RAM, ROM, or Serial Shift Registers (SSRs):

- **Used as a look-up table** the LUT can contain (up to) any four-input function.
- **Used as RAM or ROM** the LUT can have a 1X16-address configuration.
- **Used as an SSR** the LUT can be used as a 16-bit shift register.

Creating Counters in LUTs

A counter, whether it's binary, Johnson ring, or LFSR, is a sequence of repeating patterns. If you program that sequence into a set of Look-up Tables, then you can easily create fast, simple, and large counters. Here are several examples.

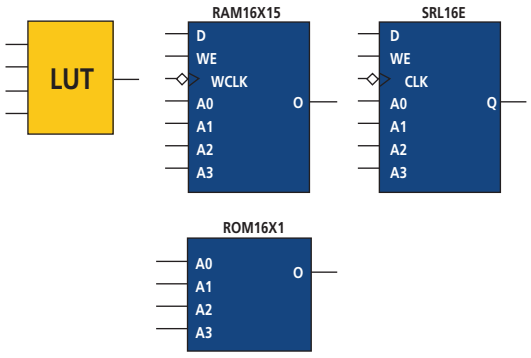


Figure 1 - LUTs used as RAM, ROM, or SSR.

A 4-Bit Binary Counter

A 4-bit binary counter has 16 possible states which can be stored in a Look-up Table. By cycling through the addresses you can generate this binary counter pattern as shown in Figure 2.

When we initialize an SRL16 with these values and cycle the shift register, the output will behave as a counter. However, the shift register will be empty in 16 clock cycles so you need to connect the input to the output to make the counter repeat the cycle. It's also possible to give

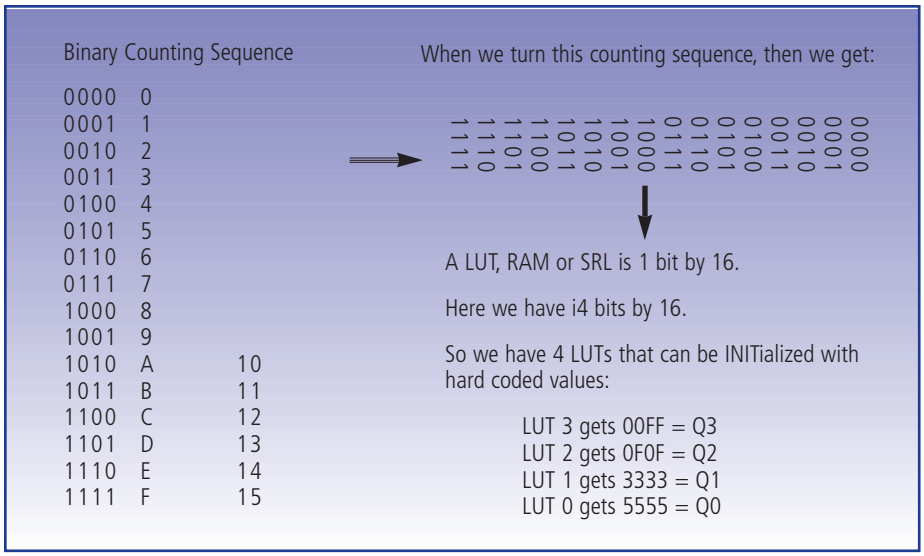


Figure 2 - 4-bit binary count sequence.

this counter a terminal count. To do this, use the carry chain as a wide AND gate and combine all the outputs to create the desired terminal count, as shown in Figure 3.

Using four LUTs, configured as an SRL16 shift register, can make a small 4-bit counter with a full and repeatable sequence.

You can also make a counter or sequencer that generates any count sequence you like, as shown in Figure 4. By initializing the SRL16 with a sequence, so that the counter starts counting at a specific value and stops at another value, then you only need to set the address of the SRL16 to the correct number of stages.

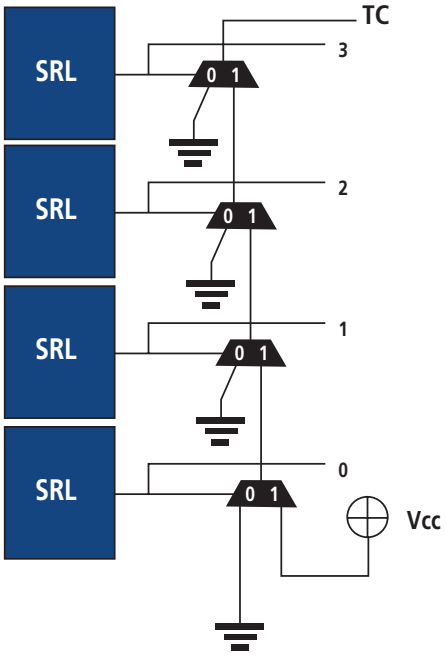


Figure 3 - Creating a terminal count.

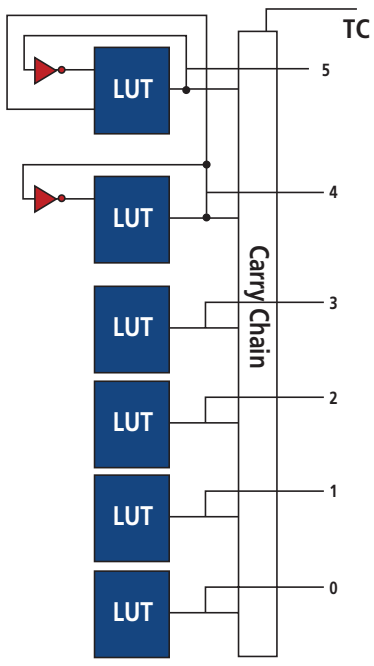


Figure 4 - A 6-bit counter.

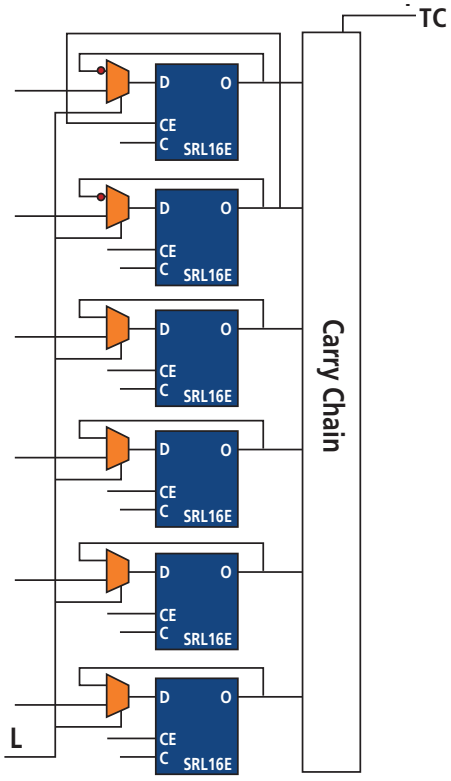


Figure 5 - A loadable counter.

A 5-Bit Binary Counter

To create a 5-bit counter, initialize the extra LUT (SRL16) to "0" and feed it's output back to it's input via an inverter. Then for the first 16 cycles the LUT/SRL will give a zero at the output, while loading a "1" into the SRL/LUT. For the next 16 cycles the counter will produce a "1" as the high order bit.

A 6-Bit Binary Counter

To create a 6-bit counter, first create a 5-bit counter as previously described. Then, to create the sixth bit, initialize the LUT to all zeros. Connect the output of the SRL to it's input via an inverter and then connect the enable of the SRL to the output of the previous bit, as shown in Figure 4.

The sequence for both upper bits will then be as follows:

1. Both upper bits will be "0."
2. After 16 clock cycles, bit 5 will become a "1" and bit 6 stays "0."

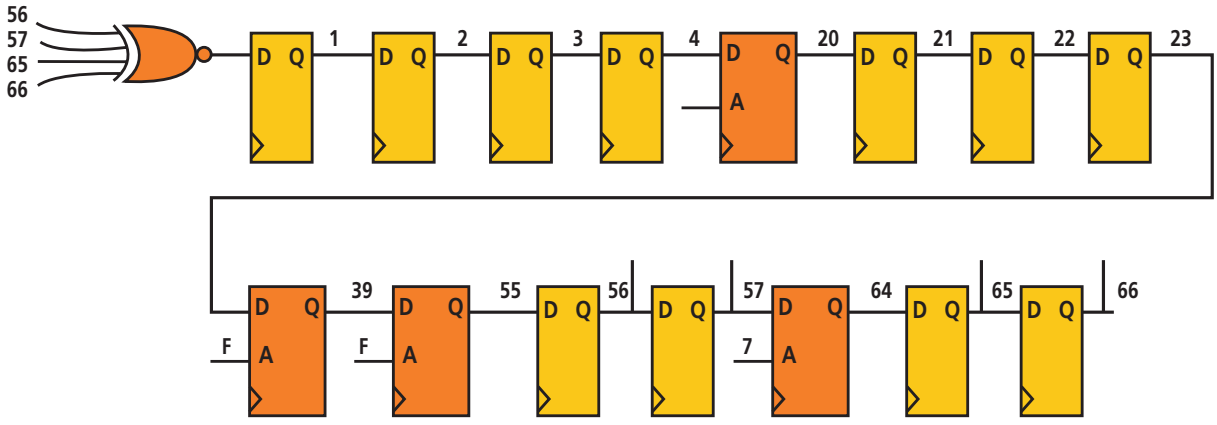


Figure 6 - Example of an LFSR counter.

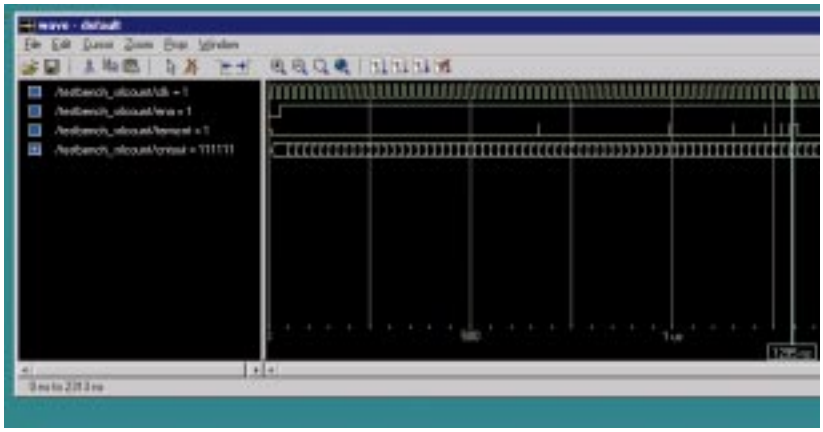


Figure 7 - Simulated counter waveforms.

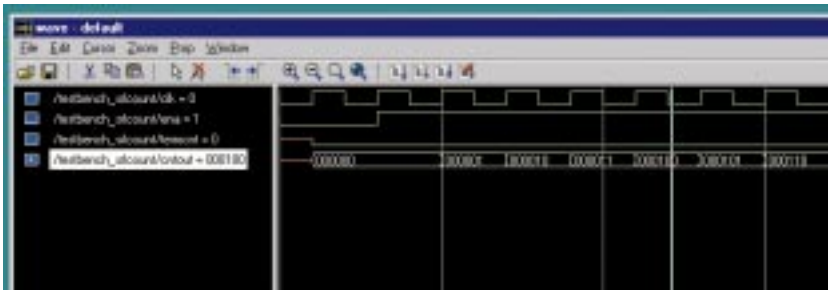


Figure 8 - Simulated counter waveforms, zoomed in.

3. The LUT of bit 6 will be enabled for the next 16 clock cycles.
4. The LUT of bit 6 will output 16 zeros and then load a "1." Bit 5 will load "0" while the output is "1."
5. After 16 clock cycles, bit 5 will become "0" and bit 6 will be a "1."

The simulated counter is shown in Figure 7 and Figure 8.

A Loadable Counter

Figure 5 shows how to create a loadable counter. A load operation will take 16 clock cycles, and you can load the counter while it is counting.

A Large LFSR Counter

An LFSR counter is a shift register with it's input fed back (XORed) from the bits of the different stuck states that can appear in the sequence. In the Virtex architecture we have the SRL16 elements that represent 16 flip-flops, thus large LFSR counters with only certain outputs of interest can be made efficiently. Σ

For more information see Application note (Xapp052) on "Efficient Shift Registers, LFSR counters, or see Xcell 35 article on "Pseudo Random Noise Generators."