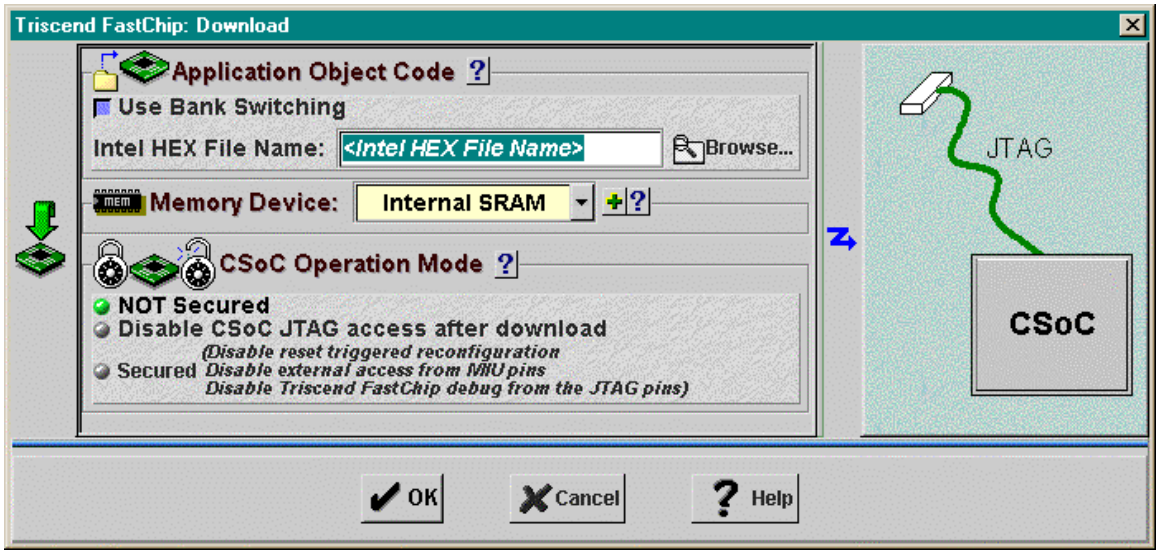
You can download your design once the bind process completes. *As always, make sure your CSoC Board is attached to the 9V DC power supply and it is connected to the parallel port of your PC with the downloading cable.* Click on the Download icon on the toolbar and then click on OK in the **Download** window that appears. This initiates the downloading of the configuration file into your CSoC Board.



With your design downloaded into the CSoC Board, you can try some test cases to see if various inputs are summed correctly. The first addend is driven by the first three DIP switches, the second addend is driven by the next set of three switches, and the carry input is set by the seventh switch. Figure 7 shows a few test examples.

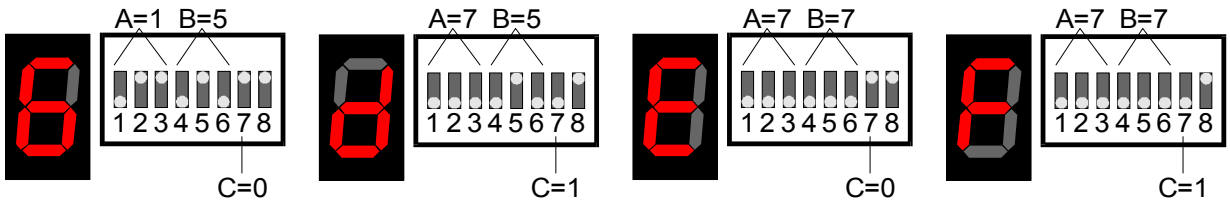


Figure 7: Testing the three-bit adder.

Once you are done testing the adder, finish by clicking on File⇒Save Project and File⇒Exit.

Design 1.3: A Simple Timer

Your previous designs were combinatorial in nature: the output was a function of only the current inputs. This design introduces a sequential component, a counter, whose output depends upon its previous state.

The counter is used to build a timer where the elapsed time is displayed on the seven-segment LED (Figure 8). A 25 MHz clock drives a 27-bit counter. The counter will roll-

over every $2^{27}/25x10^6$ seconds = 5.37 seconds. The upper four bits of the counter are connected to a seven-segment LED decoder. The LED digit will display all sixteen hexadecimal numerals in 5.37 seconds, so each numeral appears for 0.34 seconds. The count stops changing when the **enable** signal is pulled low.

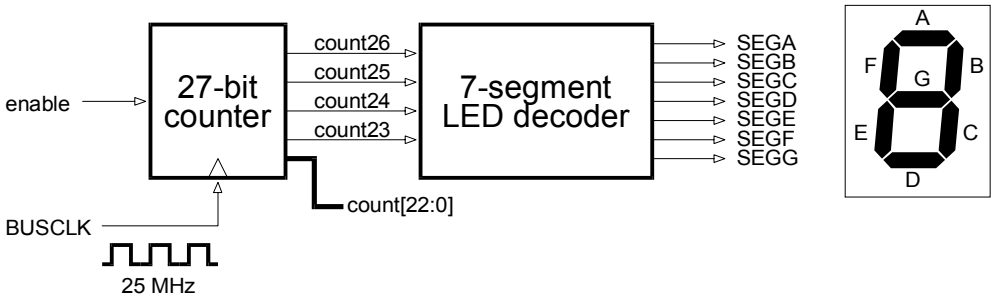
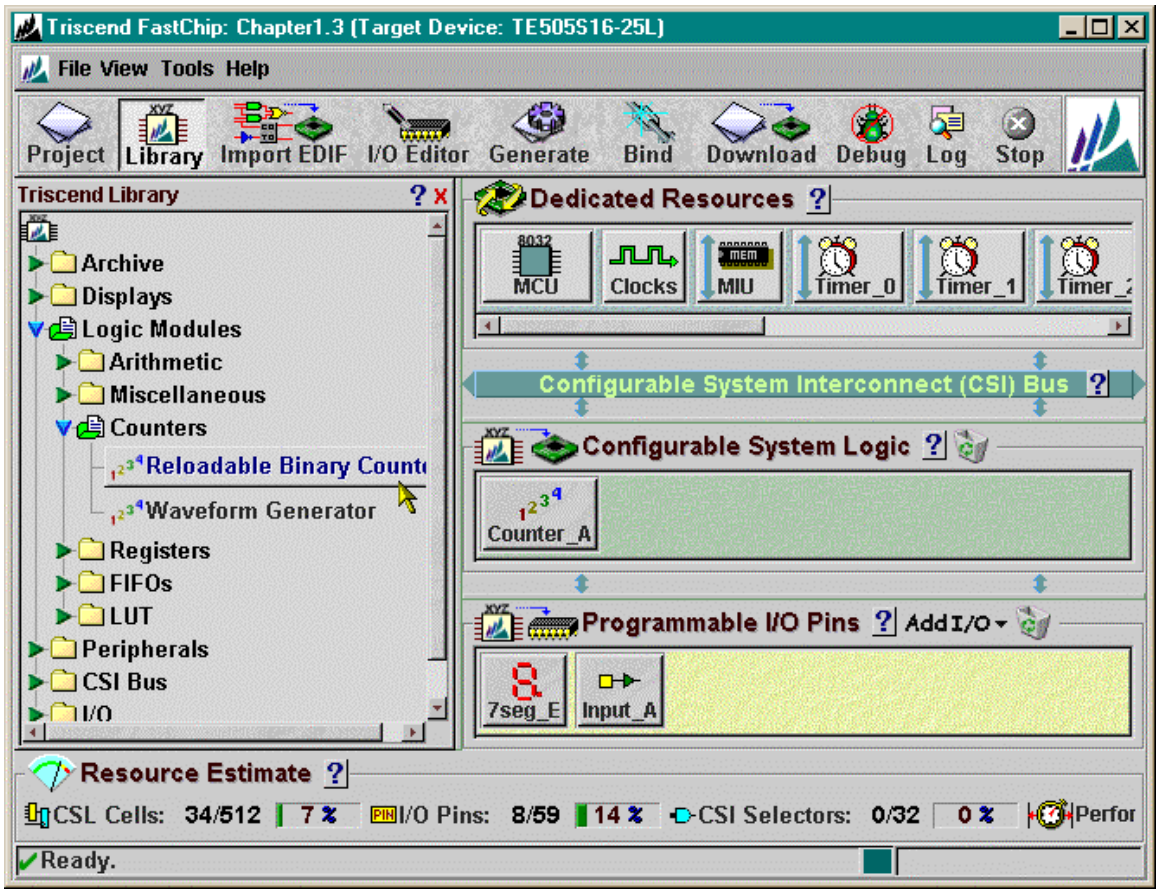


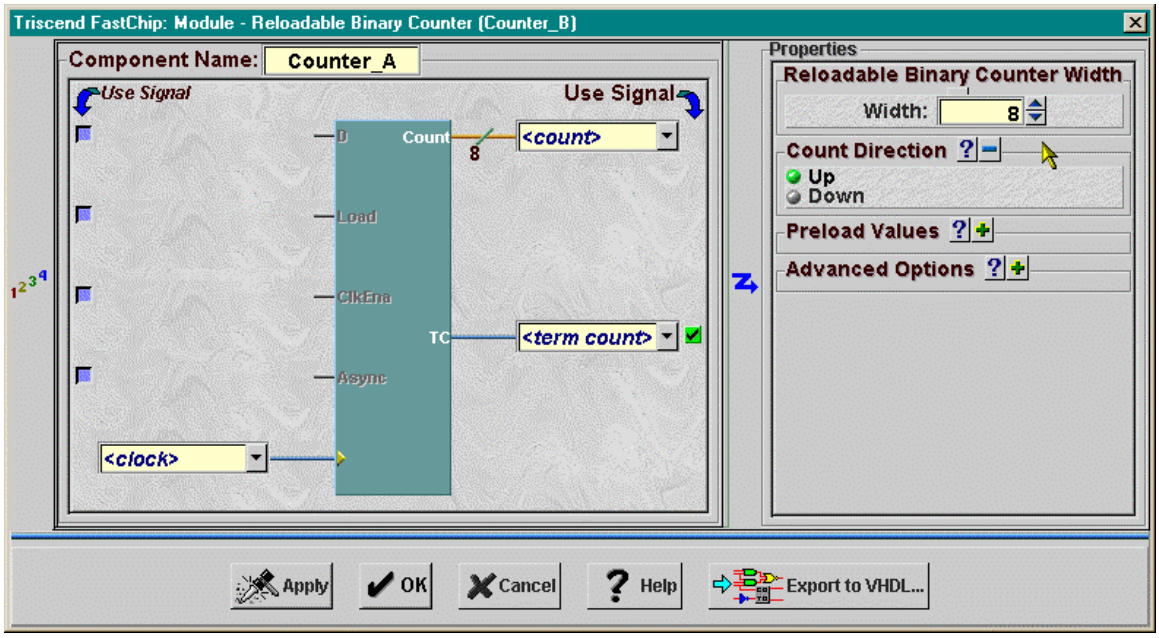
Figure 8: Block diagram of a simple timer that displays elapsed time on a seven-segment LED.

You can begin this design from scratch. Start the FastChip software and type `Chapter1.3` in the Project Name field. Then use the Target Device area to select the E5 device in a 128-pin LQFP with a maximum clock frequency of 25 MHz. Click on OK to get to the project window.

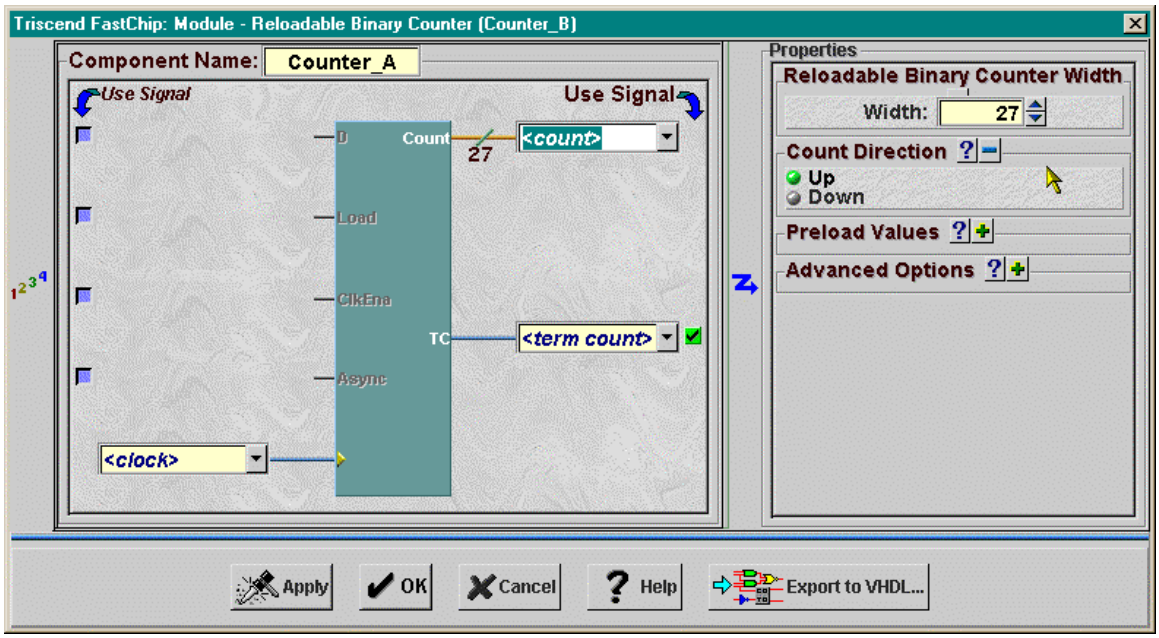
Once the project window appears, display the Triscend Library and drag-and-drop a seven-segment LED driver and an input module into the Programmable I/O Pins area. Then drag-and-drop a Reloadable Binary Counter icon to the Configurable System Logic area.



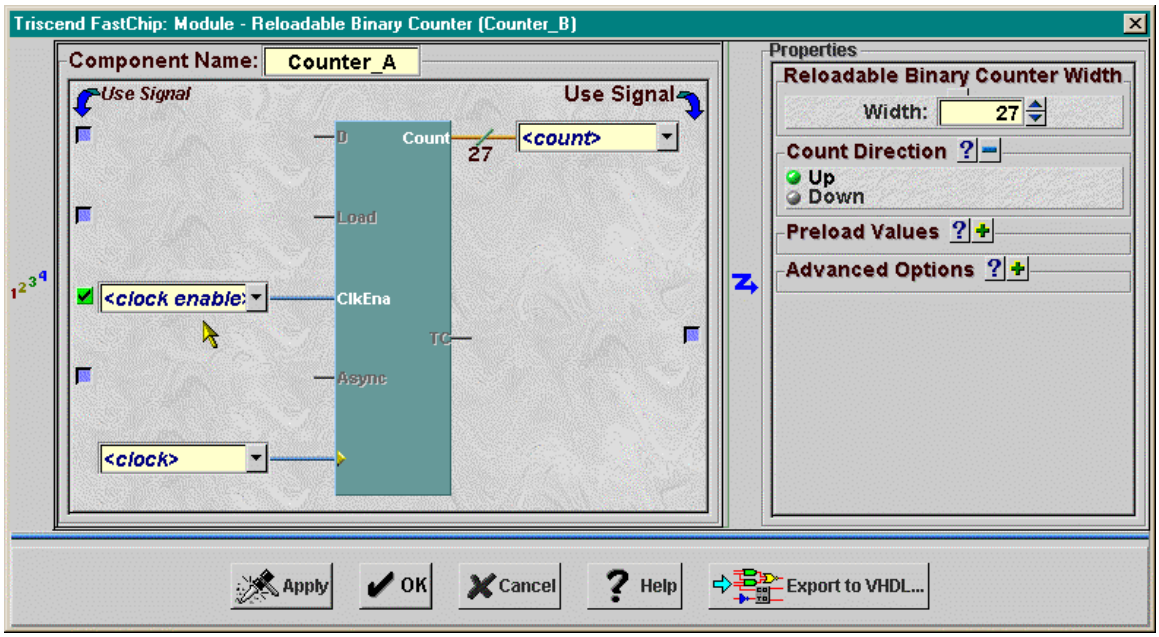
Now, click on the Counter_A icon so you can setup the counter. The **Reloadable Binary Counter** window shows the module is initially configured as an eight-bit up-counter.



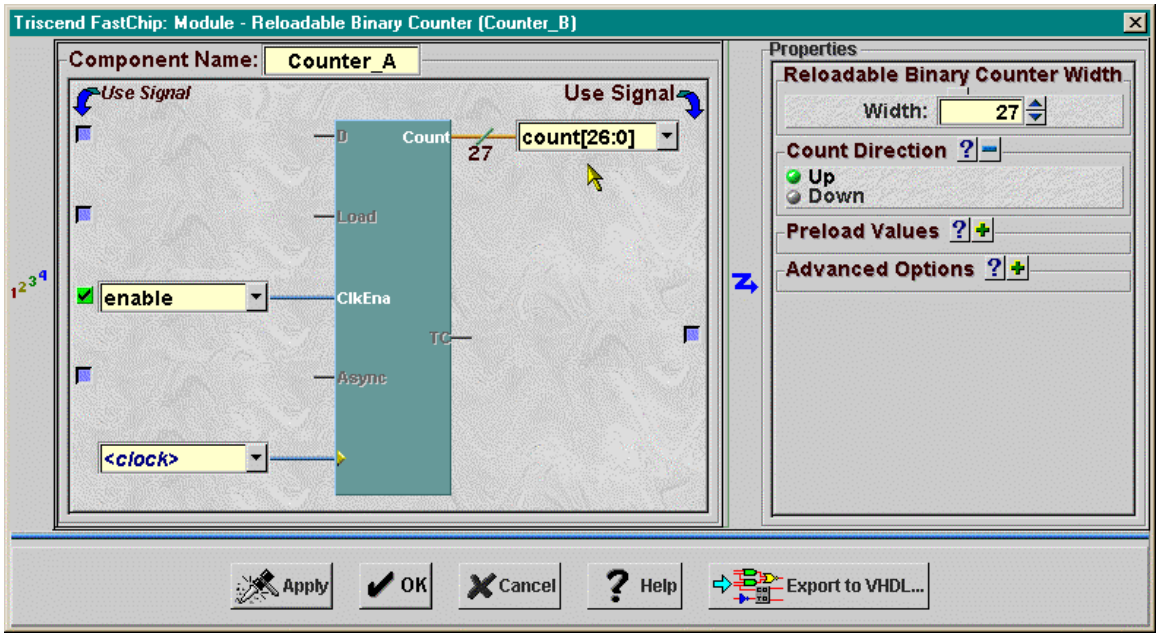
First, increase the counter width to 27 bits.



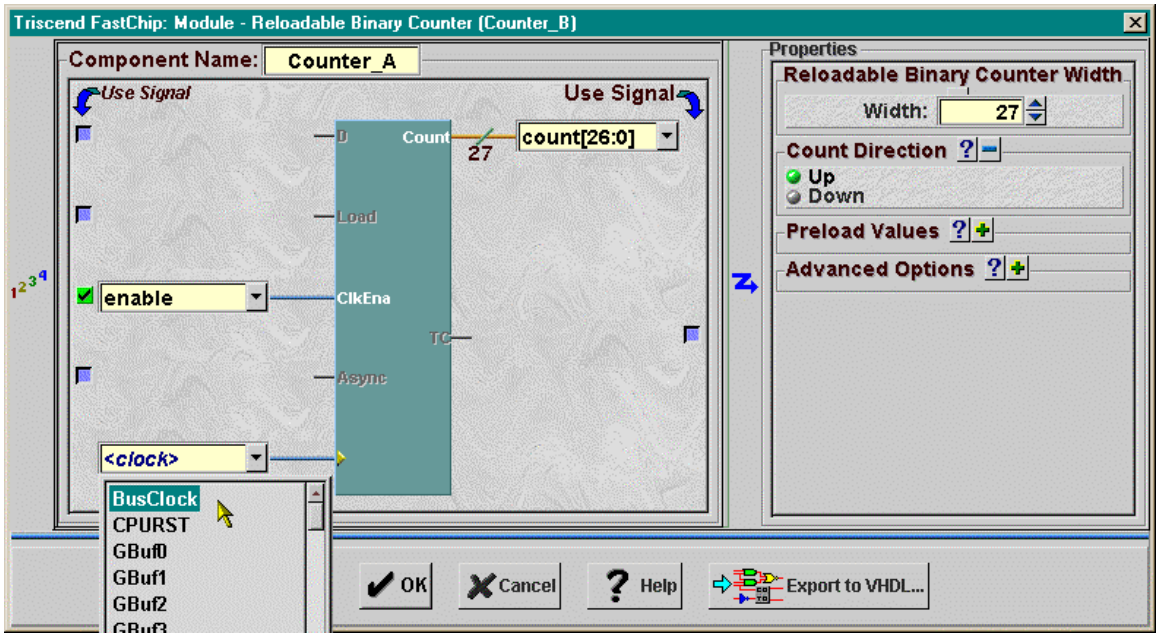
Then, click on the checkbox at the right of the <term count> field to disable this output. (You only need this output when you are cascading one or more counters.) Then click on the unchecked box at the left of the counter enable input (ClkEna).



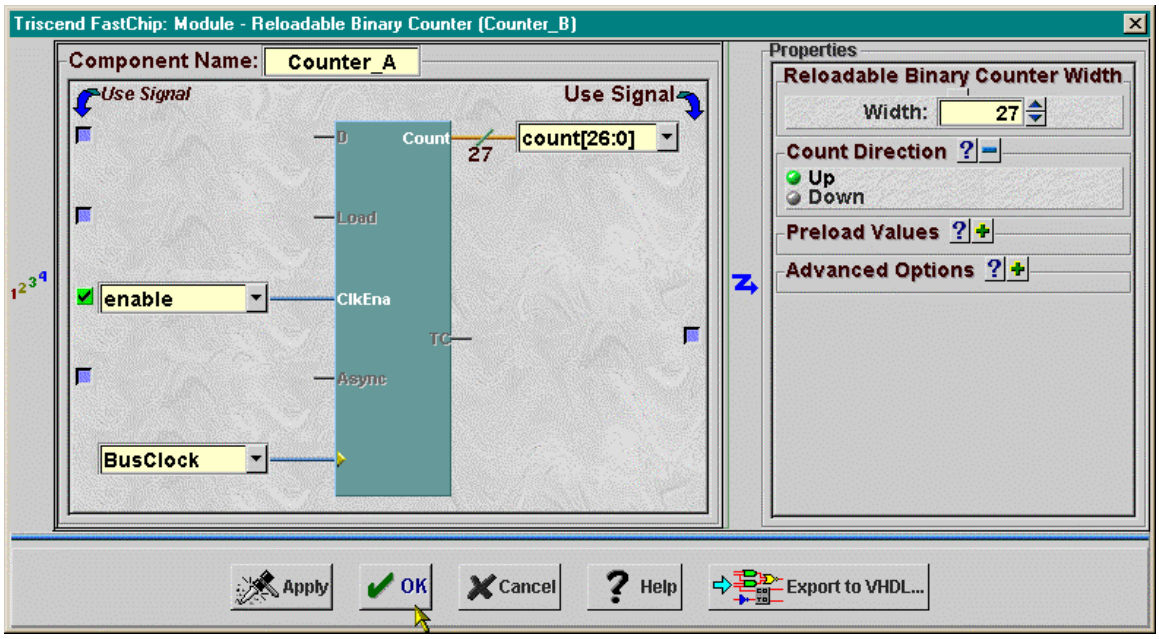
At this point you can begin connecting the counter. Type `enable` in the ClkEna field. This will be the signal driven by the **Input_A** module which turns the timer on and off. Then type `count` into the <count> field. Click outside of the field and the width ([26:0]) is automatically appended to the signal name.



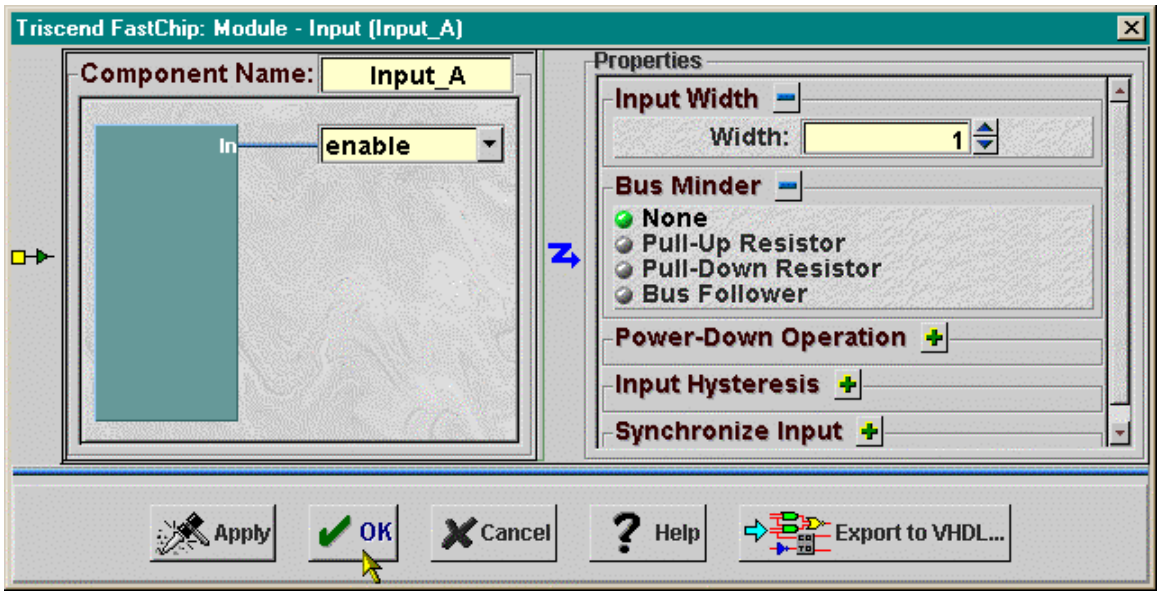
To connect a clock to the counter, click on the button to the right of the <clock> field and select BusClock from the drop-down list. **BusClock** is one of the global clock signals that is available to the cells in the CSL. Later you will see how to select the clock that is distributed over the **BusClock** net.



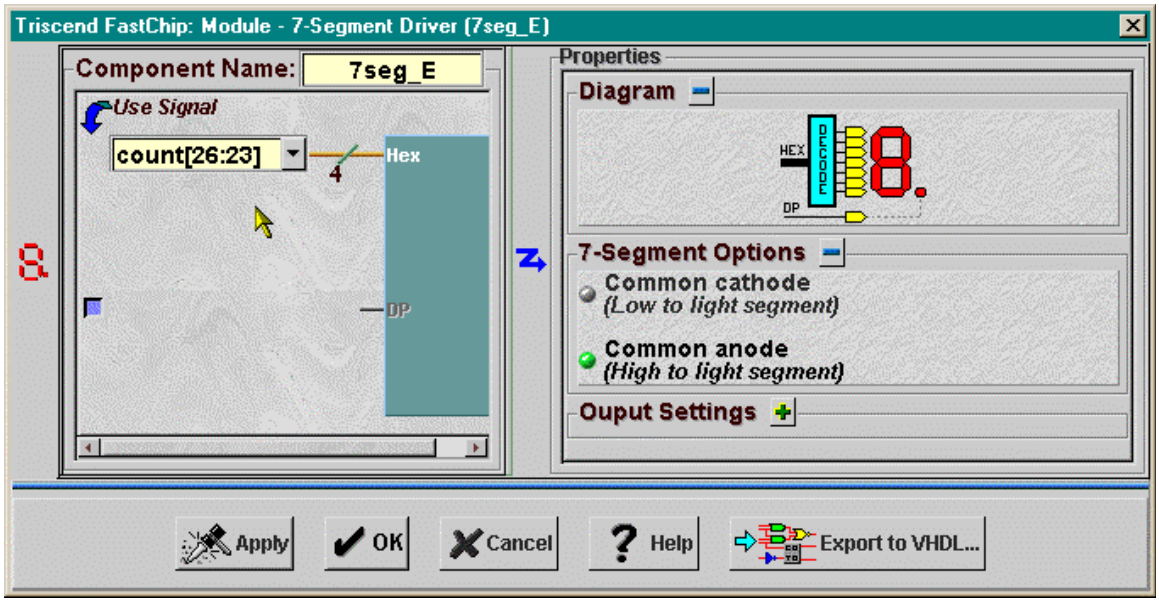
You have finished configuring the Counter_A module, so click on OK.



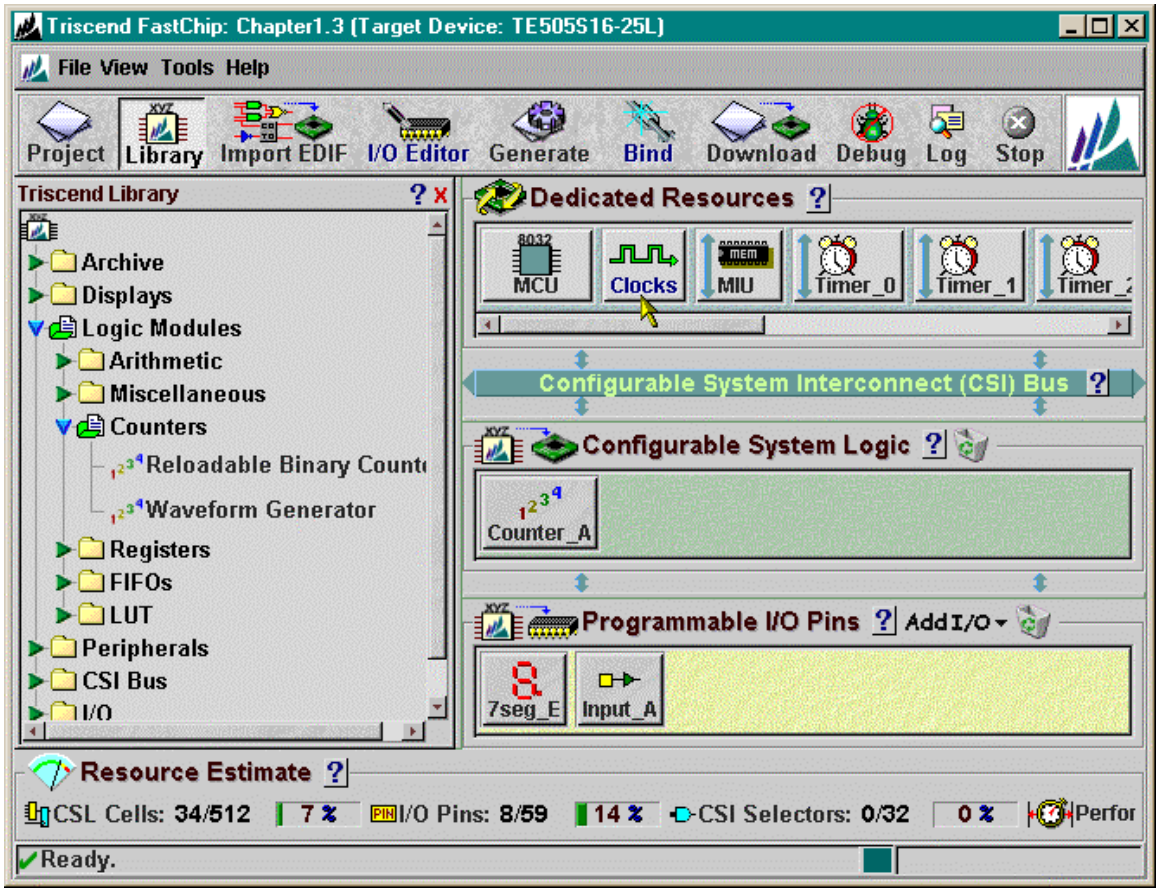
Once you return to the project window, click on the Input_A icon and set-up that module. The only thing you needs to modify is to connect the output of the module to the enable signal. Click on OK to leave the window.



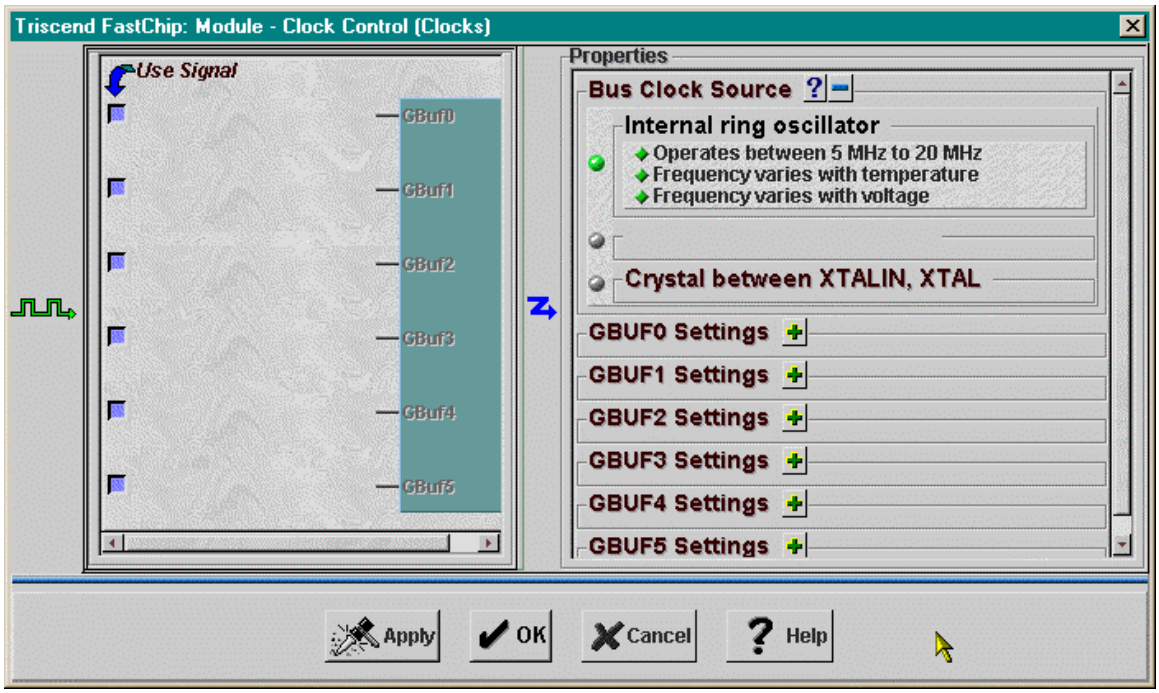
Next, click on the 7seg_E icon to configure the LED driver module. Click on the button to the left of Common anode to match with the driver with the polarity of the LED digit on your CSoC Board. Then type `counter[26:23]` into the <hex value> field. This will cause the LED to display the hexadecimal numeral that corresponds to the upper four-bits of the counter. The counter rolls-over every 5.37 seconds, so each of the sixteen numerals will appear for only 0.34 seconds. That completes the changes you need to make to the **7seg_E** module, so click the OK button.



The timer circuit is almost complete except you haven't set-up the **BusClock** yet. To do this, click on the Clocks icon in the Dedicated Resources area of the project window.

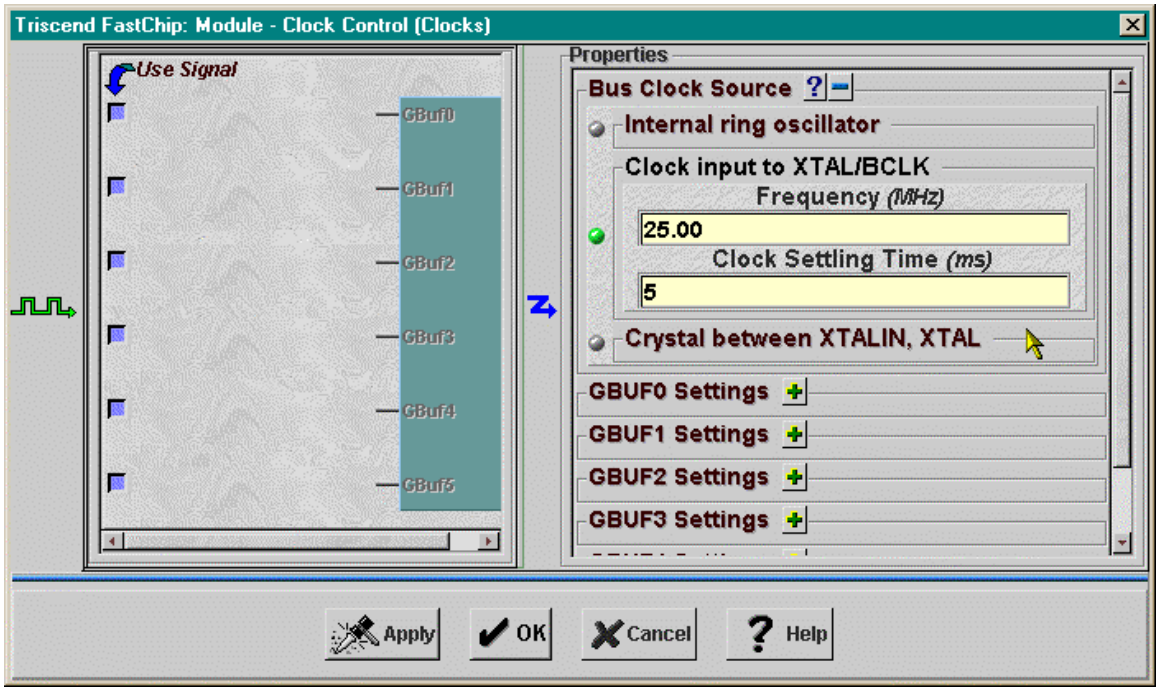


The **Clock Control** window that appears lets you select one of several sources for the **BusClock** signal. In the Bus Clock Source area, the internal ring oscillator is selected as the default clock source. This oscillator uses the delay through a loop of logic gates to set the clock period. Because the delay through a logic gate is dependent on factors such as temperature and voltage variations, the ring oscillator frequency can vary from 5 to 20 MHz.



A better clock source for a timer is selected by clicking the button for the second option: Clock input to XTAL/BCLK. This selects the dedicated clock input to the Triscend CSoC as the driver for **BusClock**. An external programmable oscillator is connected to this input on your CSoC Board. If you have configured your CSoC Board as described in Appendix A2, then the external oscillator will output a stable 25 MHz clock frequency. Type `25.00` into the Frequency field to let the FastChip software know this is the frequency of the external clock source. Then type `5` into the Clock Settling Time field. (This value is used by the Triscend CSoC to keep the chip in a reset state until the main clock is stabilized after power-up. This parameter isn't very important in your current environment since the CSoC Board is already powered before the chip is ever configured with your timer circuit.)

This completes the set-up for the timer clock source, so click on OK.

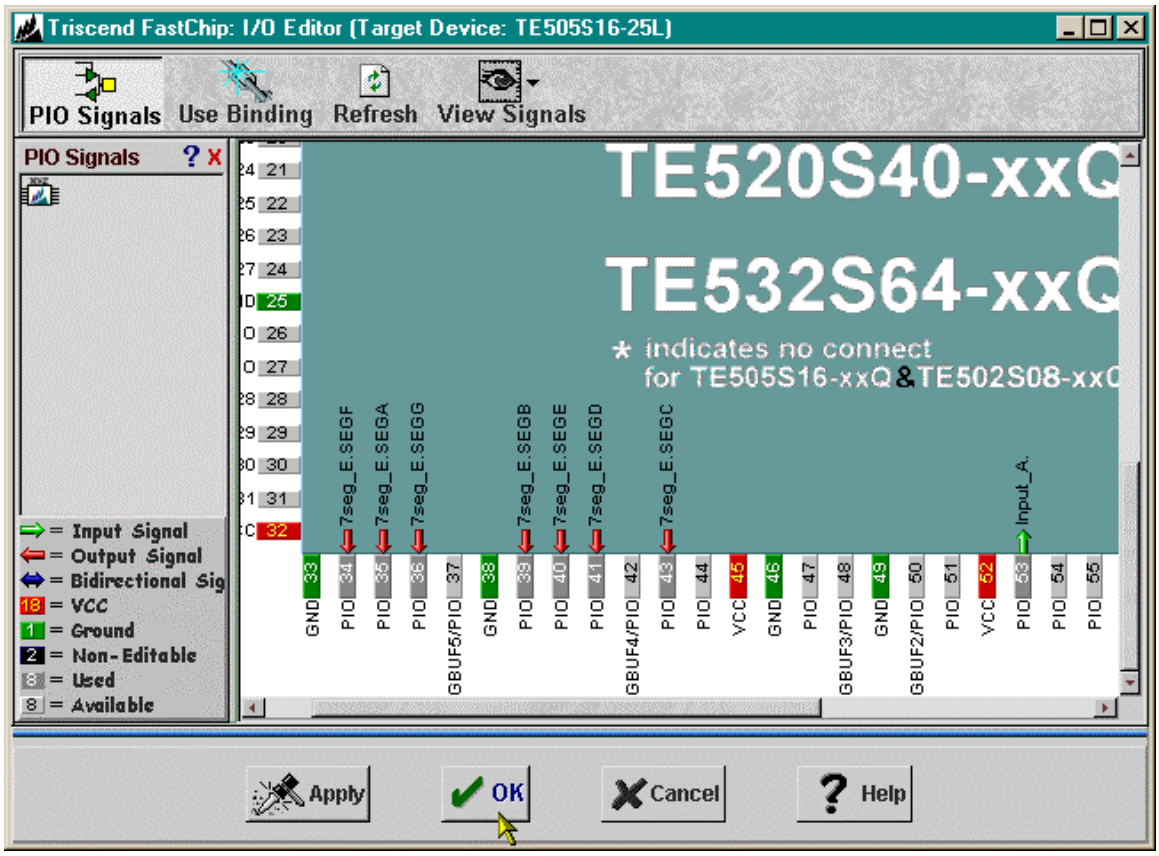


Now that you have the internal circuitry defined, bring up the **I/O Editor** window and assign the inputs and outputs as follows:

Table 3: Pin assignments and functions for the timer design.

Signal	Pin	CSoC Board Resource
Input_A.0	53	DIP switch position #1
7seg_E.SEGA	35	LED digit segment A
7seg_E.SEGB	39	LED digit segment B
7seg_E.SEGC	43	LED digit segment C
7seg_E.SEGD	41	LED digit segment D
7seg_E.SEGE	40	LED digit segment E
7seg_E.SEGF	34	LED digit segment F
7seg_E.SEGG	36	LED digit segment G

After you make the pin assignments, the **I/O Editor** window will appear as follows:



Now that the pin assignments are complete, click on OK to return to the project window. Then bind and download the timer circuit to your CSoC Board. Once the circuit is in the CSoC Board, you can make the timer run by placing DIP switch #1 into its lower position. You should see all sixteen hexadecimal numerals displayed repeatedly on the CSoC LED in a 5 second loop. Raising DIP switch #1 will halt the incrementing of the LED digit.

Design 1.4 - PS/2 Keyboard Scanner

This example creates a circuit that accepts scan codes from a keyboard attached to the PS/2 interface of the CSoC Board. If a scan code for one of the keys "0"–"9" arrives, then the numeral will be displayed on the LED digit of the CSoC Board.

The format of the scan code transmissions from the keyboard are shown in Figure 9. The keyboard drives the clock and data lines. The start of a scan code transmission is indicated by a low level on the data line at the falling edge of the clock. The eight bits of the scan code follow on successive falling clock edges (starting with the least-significant bit). These are followed by an odd-parity bit and then a high-level stop bit.