**g48\_note\_timer:**

**Description:**

The note timer circuit was designed to produce a tone based on various inputs. Each tone is defined as a 16-bit string of information, and is divided up as follows:

* Bits 0-3: Note Number. This indicates the specific note that is to be played, in accordance with a 12 note musical scale (because this string can go up to 15, values 12-15 correspond to the same note).
* Bits 4-6: Octave Number: This value controls the frequency at which the note is to be played. An increase of 1 in this value corresponds to doubling the frequency.
* Bits 7-9: Note Duration: How long the note is to play, based on the beat. 0 is one eighth of a beat, 1 is one quarter of a beat, etc.
* Bit 10: Triplet: This is a simple yes/no, indicating whether or not the note should be given a duration that’s 2/3 that of a regular note.
* Bits 11-14: Loudness: Determines the amplitude of the note’s wave.
* Bit 15: End of Song Marker: Marks the end of the song.

Our circuit took in as input a clock signal (clk), an asynchronous reset (reset), note\_duration (see above), triplet (see above), and a tempo\_enable signal that set the basic tempo of the circuit (creating a pulse at every beat). The circuit gave output signals of count\_out, which was essentially a count up to the duration of the note (effectively signaling when it’s okay to play a note), and TRIGGER, which. The circuit used a lookup table process block to select the note to be played, based on the note\_duration and triplet signals.

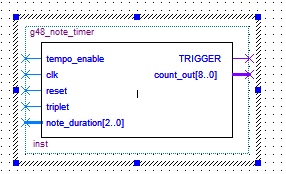
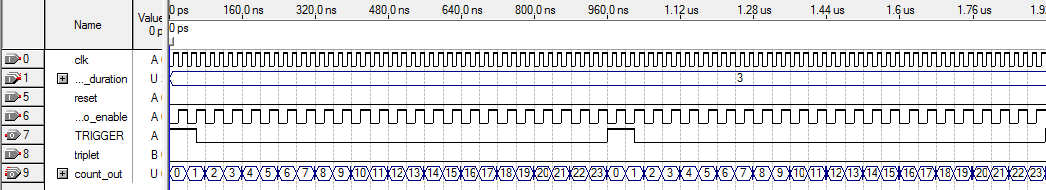
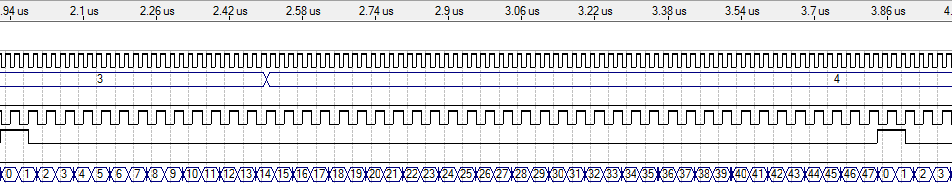
The circuit also implemented a counter process block based on the reset and clk signals. When reset is pressed, the count\_out output is set to 0, and TRIGGER is set to 1. When the clock is high (and on a rising edge), tempo enable is then checked; if that signal is also high, then the count is incremented. If the count is already 1, then TRIGGER is set back to a value of 0. Finally, the count is checked in this process block. As it has been incremented at each clock cycle, if the count ever reaches a value equal to target-1 while tempo\_enable is high, then the count is reset back to 0 and TRIGGER is set to 1.

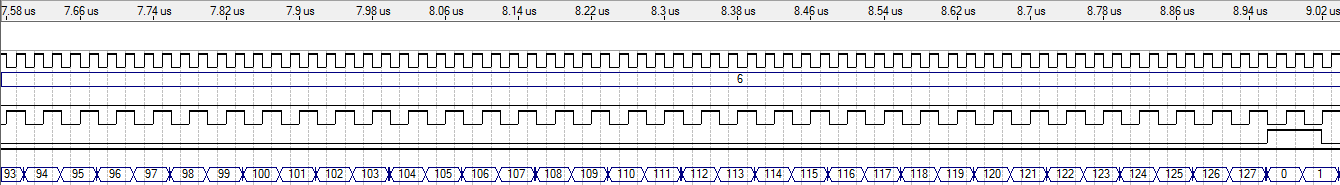
Figure : Pinout diagram of the note\_timer circuit.

**Testing:**

First we preformed simulation tests on the note\_timer circuit. We tested the circuit over several different inputs. The tempo\_enable pulses were entered by hand as requested by the testing specifications. The testing performed was successful and confirmed by the TA on duty at the time. The results are displayed below.



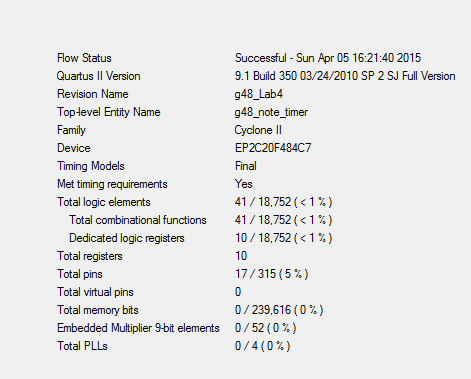


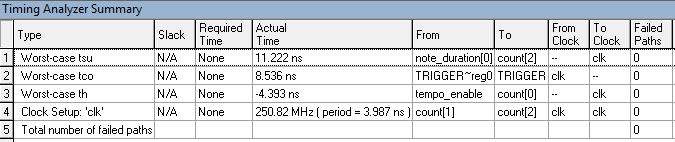


After completing the simulation tests we created a test bed circuit. This entailed setting the bpm input to correspond to the switches on the board, allowing us to dynamically test different input values on the board. The TRIGGER output was set to a single LED, which would blink if the signal was high, allowing us to further gauge whether our circuit was functioning properly. The count\_out output was left open, as it was only used for the timing analysis on the computer. The note\_duration and triplet inputs were also attached to the switches on the FPGA, for the same reason as the bpm input, allowing us to do a full test of the circuit. Internally, we included two components: the Tempo and note\_timer circuits. Using bpm as an input to the Tempo component we were able to use the tempo\_enable output and connect it to the tempo\_enable signal of the note\_timer circuit. This allowed for the final desired output of TRIGGER to be set to a LED, location on the Altera DE1 board. By adjusting the three inputs bpm, triplet and note\_duration by using the switches we were able to test many different inputs that yielded correct results as confirmed by the TA.

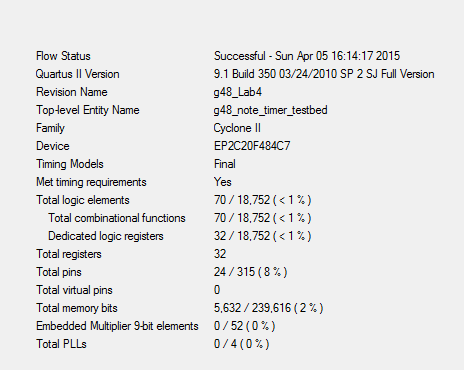
**Timing Performance and Resource Utilization**

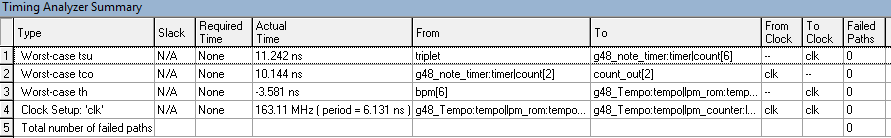
For the note\_timer circuit our timing performance and Resource Utilization were as follows:





For the note\_timer\_testbed circuit our timing performance and Resource Utilization were as follows:



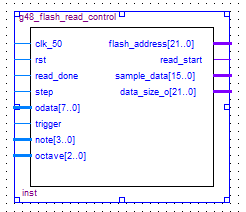


**g48\_flash\_read\_control:**

**Description:**

The flash\_read\_control provided in the lab allows us to interact with the Flash memory chip on our Altera board. The g48\_flash\_read\_control circuit is designed to read a 16-bit wave data from the flash memory, and play it back to the user as a mono audio output via the AUDIO\_OUT on the board.

Initially, the circuit modification we implemented was to help read the wave data provided to the circuit. The first modification we made was to wait until the step signal went low. Once this happened we loaded the appropriate address and waited until the step went high and the previous read was complete. Once this happened we signalled for the read to begin, once this was completed and the read was done, the data was loaded into sample\_data and the address incremented. After this step the second read was performed, once complete the data was loaded and the address was incremented. Finally we return to our first modification (state = 6) to start the next data read sequence until the wavefile reaches its end.



**Testing, Timing Performance and Resource Utilization**

Testing for the g48\_flash\_read\_control circuit was performed on the Altera DE1 board using a testbed circuit called g48\_Sound\_Testbed. The note and octave inputs of this testbed were connected to switches to allow for testing of several different pitches and notes. The flash\_read\_control was used as a component for reading the data of the provided wavefile, which was a guitar strum. The testing was done by changing the input values through the use of switches, our tests were successful and our outputs were verified by the TA.

The timing performance and resource utilization of the g48\_Sound\_Testbed are as follows:

