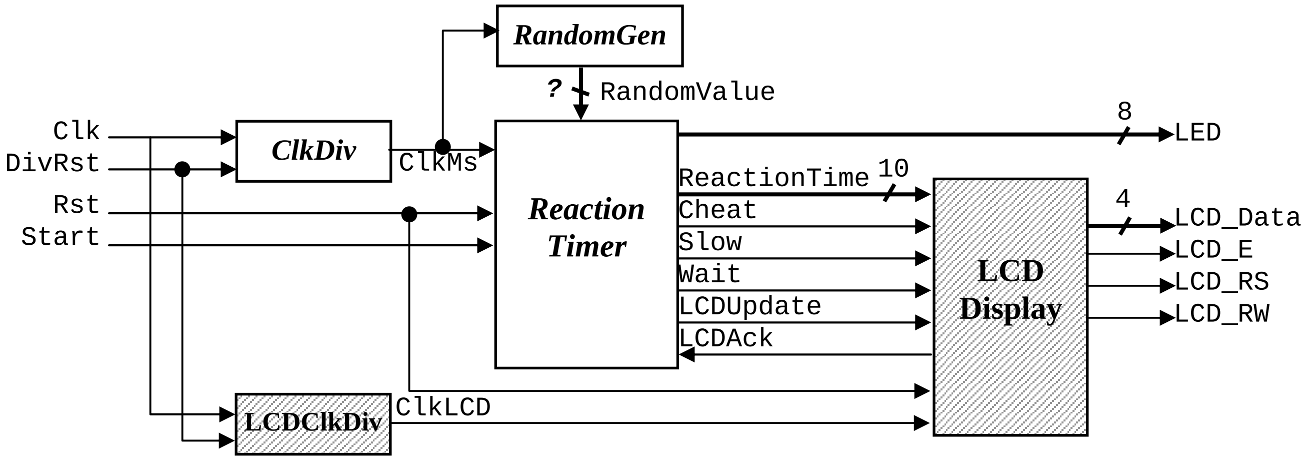
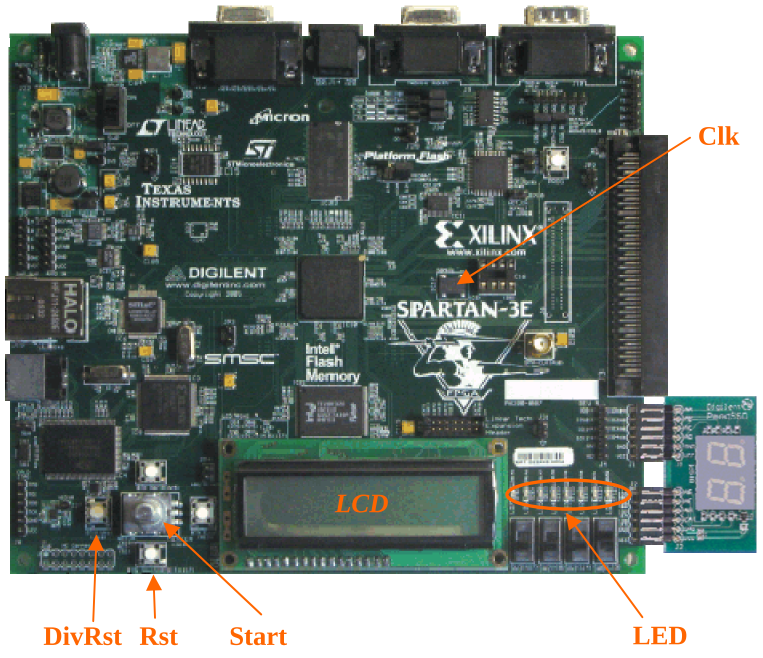
Lab 5: Reaction Timer

In this lab, you will be designing a *Reaction Timer* that measures a user's reaction time by measuring the time elapsed between illuminating a set of LEDs and the pressing of a button by a user. On reset, the *Reaction Timer* will initially display an introduction message "`Reaction Timer`" on the LCD of the Spartan-3E FPGA board. When the `Start` button is pressed, the reaction timer will wait for a random length of time between 1 and 6 seconds while displaying "`Wait for LEDs...`" on the LCD. The reaction timer will then illuminate all eight of the individual LEDs of the Spartan-3E FPGA board and measure the length of time in milliseconds before the user presses the `Start` button again. The measured reaction time will be displayed on the LCD display in the following format: "`0.345 s`". If the user did not press the `Start` button within 1 second (1000 milliseconds) of illuminating the LEDs, the reaction timer will display "`Too Slow!`" on the LCD. Alternatively, if the user pressed the `Start` button before illuminating the LEDs, the reaction timer will display "`No Cheating!`" on the LCD. The *Reaction Timer* will continue to display the last recorder reaction time, cheat message, or slow message, until the user pushes the `Start` button to perform a new measurement. The following figures provide a top level view of the various components of the *Reaction Timer* you will be designing and provide an overview of the connections that will be used to implement your *Reaction Timer* on the Spartan-3E FPGA board. The shaded components are predesigned components provided to you that you must interface with to implement the *Reaction Timer* design. The top level component *(Top.v)* and User Constraint File *(Top.ucf)* are available on the course webpage.





Reaction Timer

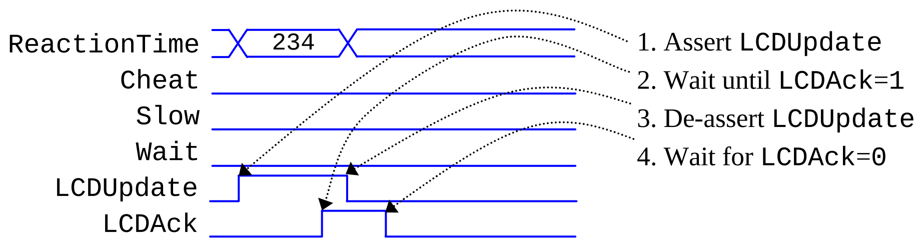
The *Reaction Timer* component provides the main functionality of the overall design and will interface with the clock divider *(ClkDiv)*, the random number generator *(RandomGen)*, and the LCD display *(LCDDisplay)*. In this lab, you will create an RTL design of the above specified reaction timer as high-level state machine (HLSM). The *Reaction Timer* component has four inputs, `ClkMS`, `Rst`, `Start`, and `LCDAck`, and six outputs, `LED`, `ReactionTime`, `Cheat`, `Slow`, `Wait`, and `LCDUpdate`.

The `ClkMS` input is a 1kHz clock signal generated by the *ClkDiv* component. While the *Reaction Timer's* HLSM could directly operate using the 50MHz clock provided by Spartan-3E FPGA board, a 1Khz clock signal provide a simplified method for counting the elapsed time in milliseconds – 1 clock cycle of a 1 kHz clock is equal to 1 millisecond. You will need to design the *ClkDiv* component that will generate a 1 kHz clock output, `ClkMS`, from the 50 MHz clock input provided by Spartan-3E FPGA board

The `RandomValue` input is an ***N***-bit number output from the *RandomGen* component that provides a random value between 1 and 6 seconds used to delay the illumination of the LEDs for a random length of time. As there are many ways to design the *RandomGen* component, the number of bits used to represent the `RandomValue` signal is not defined. Instead, in designing the *RandomGen* component, you will need to determine the number of bits needed for this value. The *RandomGen* component need not implement a true random number generator, or pseudo-random number generator. Instead, the `RandomValue` should appear to be random when read by the *Reaction Timer* at the appropriate time. Please note that the built-in Verilog function *$random* is not synthesizable and should not be used. *Hint: this circuit may be simpler than might first anticipate.*

The `LED` output is an 8-bit output that will connect to the eight individual LEDs available on the Spartan-3E FPGA board. The *Reaction Timer* should illuminate all eight LEDs to indicate that the user should press the `Start` button as soon as possible. In all other circumstances, the LEDs should not be illuminated.

The input `LCDAck` and outputs `ReactionTime`, `Cheat`, `Slow`, `Wait`, and `LCDUpdate` are used to interface with the provided *LCDDisplay* component. The `ReactionTime` output is a 10-bit output that corresponds to the measured time in milliseconds of a successful reaction time measurement. The `Cheat` output is a one bit output indicating the user pressed the `Start` button before the LEDs were illuminated and the "`No Cheating!`" message should be displayed on the LCD. The `Slow` output is a one bit output indicting the user did not press the `Start` button within 1 second after illuminating the LEDs and the "`Too Slow!`" message should be displayed on the LCD. The `Wait` output is a one bit output indicting that the "`Wait for LEDs...`" message should be displayed on the LCD. Finally, the `LCDUpdate` output and `LCDAck` input are used for communicating with the *LCDDisplay* component to specify when and how the LCD display should be updated. Because the *LCDDisplay* and *Reaction Timer* components operate using two different clock signals, in order to communicate between the two components, your design will have to implement a simple handshaking scheme. As illustrated in the following timing diagram, to update the LCD, the *Reaction Timer* must first assert the `LCDUpdate` output and continue to assert this output until the `LCDAck` input is asserted by the *LCDDisplay* component. Upon detecting that the `LCDAck` input is high, the *Reaction Timer* must de-assert the `LCDUpdate` signal and wait for the `LCDAck` input to be de-asserted as well. This handshaking sequence is required for all updates to the *LCDDisplay*. For example, to display a measured reaction time on the LCD, the *Reaction Timer* should assign the measured reaction time in milliseconds to the `ReactionTime` output, assign `0` to `Cheat`, `Slow`, and `Wait`, assert the `LCDUpdate` output, and complete the handshaking as described above. Note that the values assigned to `ReactionTime`, `Cheat`, `Slow`, and `Wait` must remain constant as long as `LCDUpdate` is asserted. As another example, to display the wait message on the LCD, the *Reaction Timer* should assign `1` to the `Wait`, assign `0` to `Cheat` and `Slow`, assert the `LCDUpdate` output, and complete the handshaking as described above.



The provided top-level component *(Top.v)* integrates the various components together to implement the overall reaction timer design. In designing the *Reaction Timer*, *RandomGen*, and *ClkDiv* components, be sure to use consistent module names to ensure your components will integrate effortlessly.

Lab Procedure

1. Behaviorally design the *ReactionTimer* as a high-level state machine. Behaviorally design the *RandomGen* component. Note that you will need to submit a diagram for your *Reaction Timer's* HLSM as part of your lab report.

2. Create a testbench that connects the *Reaction Timer* and *RandomGen* components to test your *Reaction Timer* design for correct functionality. Your testbench should check for correct reaction time measurement, correct cheating detection, and correct slow reaction detection. *Note that as you are not yet interfacing with the LCDDisplay component, you will need to control the `LCDAck` signal from within your testbench.*

3. Design the clock divider, *ClkDiv*, that outputs a 1kHz clock from the 50 MHz clock input provided on the Spartan-3E FPGA board. You may design the clock divider behaviorally (similar to the *ClkDiv* provided in Lab 3) or structurally (similar to your *Refresher* circuit of Lab 4). Create a testbench to test your *ClkDiv* component for one full clock period (1 millisecond).

4. Using the provided top level component *(Top.v)* and User Constraint File *(Top.ucf)*, synthesize, download, and test your overall reaction timer design on the Spartan-3E FPGA board.

Demo *(you must demo the following aspects to the TA)*

1. Verilog code for *Reaction Timer* and *RandomGen* components. Testbench and simulation waveforms demonstrating correct functionality of the connected *Reaction Timer* and *RandomGen* components.
2. Behavioral or structural code for *ClkDiv* circuit generating 1 kHz clock output. Testbench and simulation waveforms demonstrating correct functionality of the *ClkDiv* component for one clock period (1 millisecond).
3. Synthesis and implementation of your overall design to the Spartan-3E FPGA Board demonstrating correct reaction timer functionality.

Lab Report Requirements *(In addition to the standard lab report format)*

1. All Verilog code for *Reaction Timer* implementation.
2. High-level state machine diagram for *Reaction Timer*.
3. Simulation waveforms demonstrating correct functionality of the connected *Reaction Timer* and *RandomGen* components. Simulation waveforms demonstrating correct functionality of the *ClkDiv* component for one clock period (1 millisecond).
4. Each group should submit their Verilog code to the appropriate dropbox on the D2L course website by Nov 30, 2007. Your online code submission will count for 5 points of your lab report score. Only one submission is required per group, but you must indentify yourself and your lab partner within comments at the top of ***all*** your Verilog files.