ECE352 Computer Organization II
Department of Electrical and Computer Engineering
University of Toronto

Lab 6: Reaction Timer

In this lab, you will implement a system to measure a person's reaction time to a visual stimulus. The system, called a reaction timer, consists of an input and a number of outputs. The input is a switch on the protoboard which we will call *STOP*. The outputs are: an LED on the protoboard and the PC workstation screen. The reaction time is measured in the following way. The user starts an assembly program to begin the test. After a short period of time, the reaction timer turns on the LED. As soon as the user sees the LED light up, he/she must toggle the STOP switch as fast as possible. The reaction timer measures the time from when the LED is turned on to when the user toggles the STOP switch. The time is to be displayed on the monitor as a decimal number in thousandths of a second.

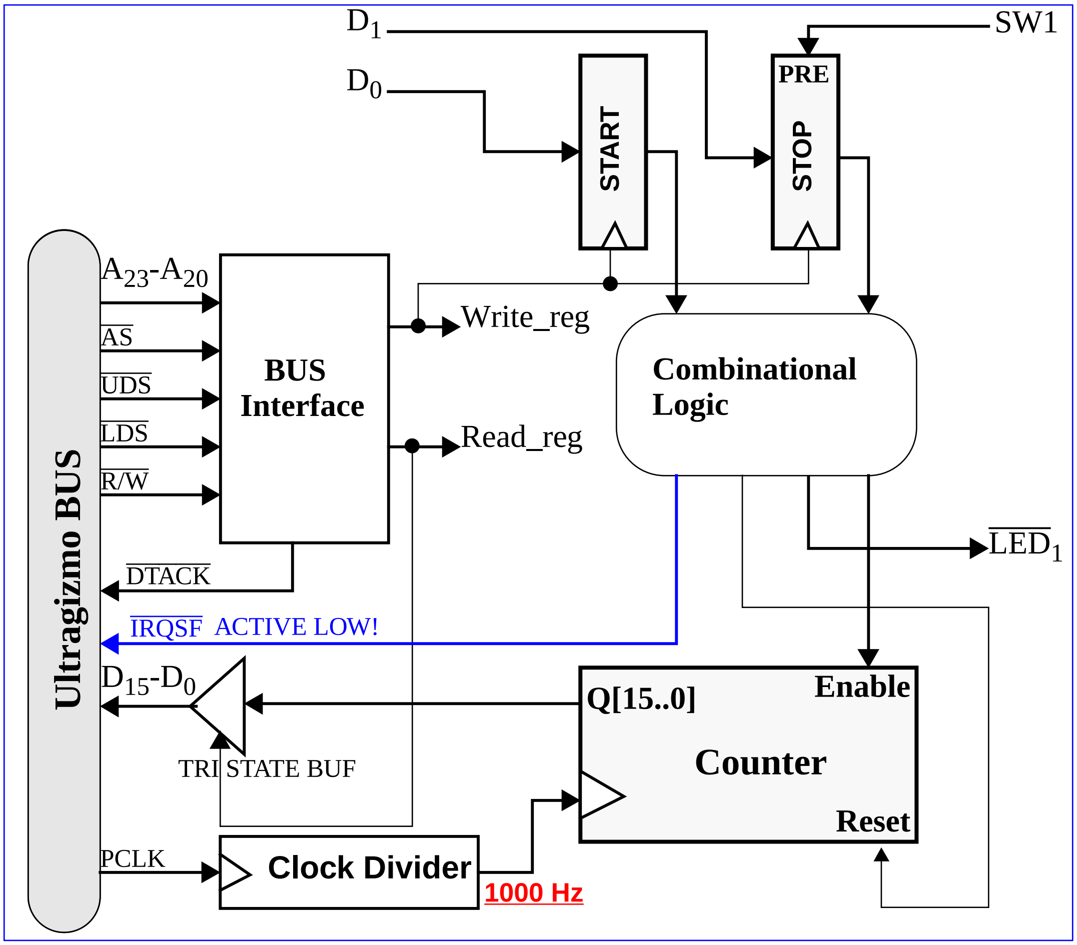
You are required to implement the reaction timer in hardware using the FPGA on the Ultragizmo board. A short 68000 program will be needed to start the reaction timer and display the result on the screen. Note that this means that you will also need to create interface circuitry to allow the reaction timer to communicate with the 68000 microprocessor. The 68000 should be able to initiate the reaction test by writing to a register and your reaction timer should be able to interrupt the 68000 when it has determined the reaction time. In this way, your hardware does all of the work and only interrupts the processor once the reaction time is known.

The Figure on the next page shows the basic structure of the circuit that you are to implement. The following is a more detailed description of the operation and the main parts of the design:

- Design a **Bus-Interface** circuit. This module should assert a signal named *Write_reg* when the 68000 requests a write to location $B00000 in memory. Similarly it should assert *Read_reg* when a read from $B00000 is requested. This is a simple combinational circuit that looks for when a "B" is on the upper nibble of the address lines, and a logic '0' on $\overline{AS}$, $\overline{LDS}$, and $\overline{UDS}$. Reads and writes are differentiated by the $R/\overline{W}$ signal. You must also generate an active-low acknowledge signal $\overline{DTACK}$, as soon as either *Read_reg* or *Write_reg* is asserted. This signal must be connected in an **open-drain** configuration.

- The **START** and **STOP** flip-flops are triggered by the *Write_reg* signal and latch the values of D_0 and D_1 from the bus. The 68000 can control the state of these flip-flops by writing to bit-0 and bit-1 of address $B00000. The reaction timer is initialized by writing a logic '1' to the **START** flip-flop and a logic '0' to the **STOP** flip-flop. Your combinational logic should respond to this condition by lighting up an LED on the Ultragizmo board and enabling a counter to keep track of the reaction time.

  Notice that the **STOP** flip-flop has a preset input hooked to a switch on the digital board. When this switch is changed to a logic '1' the **STOP** flip-flop will immediately go high. With **START** a logic '1' and **STOP** a logic '1', your combinational logic circuitry should disable the counter enable, turn off the LED, and request an interrupt. You circuit can generate an interrupt by asserting the $\overline{IRQSF}$ line in the wrapper file. You must derive logic equations to



produce the counter enable, interrupt request and LED drive signals. Details on how to configure the Ultragizmo board so that the FPGA can generate an interrupt can be found in Section 8.2.2. and Section 8.2.3 of the Ultragizmo board manual. Note that the $\overline{IRQSF}$ signal in the wrapper corresponds to “pin AV20 ($\overline{IRQ}$) of SFPGA” in Table 10 in Section 8.2.2.

Hint: Be sure to add code to the *interrupt* routine that will de-assert the interrupt request after the counter has been read. If you don’t de-assert the request then the processor will continuously get interrupted. You can do this by writing a ‘0’ to the **START** flip-flop and a ‘1’ to the **STOP** flip-flop, at which point your combinational logic circuit can de-assert the $\overline{IRQSF}$ signal. Note that the **START** and **STOP** flip-flops are being used to keep the state of the system. Another state will be used to reset the counter.

- The Ultragizmo board is equipped with a programmable clock that outputs a signal named *PCLK*, which is available in the wrapper file. This clock can be programmed to any frequency between 392kHz and 90MHz. Read **Section 8.7** of the Ultragizmo board manual to learn how to configure the clock frequency. Your reaction timer will use a 16-bit counter clocked by a 1000Hz signal (we will measure milliseconds). Use an appropriate **clock-divider** circuit along with the programmable clock to generate the 1000Hz signal. What frequency would you set the programmable clock to? How many bits should the clock-divider use?

- The **16-bit counter** will keep track of the reaction time. Since it is clocked by a 1000Hz signal, it will be able to keep track of reaction times up to 65.535 seconds. The counter should have an *enable* input as shown in the block diagram so it does not start counting until the appropriate time. Notice the **reset** signal connected to the counter. You will need to reset the counter before every reaction test. Devise your combinational logic circuit such that the 68000 can reset the counter by writing a ‘0’ to the **START** flip-flop and a ‘0’ to the **STOP** flip-flop.

Implement each of the components discussed above in VHDL. In the top-level file *wrapper.vhd*, “wire” these components together so that they model the block diagram for the reaction timer. **Compile** and **simulate** each component, as well as the entire design.