"FPGA-Programming" Exercise Sheet IX:

Debugging VHDL Hardware Designs

Developer of digital systems will inevitably be confronted with the task of debugging their imperfect initial designs. With rising system complexity this can be a huge task. The process of debugging requires the determination of all possible error sources in the described circuit. In this exercise we will examine some of the fundamental debugging concepts which should assist the programmer with this task.

As target hardware always choose FPGA chip Cyclone V SoC 5CSEMA5F31C6

1) *SignalTap II Logic Analyzer*

   The Quartus software comprises a system-level debugging tool named SignalTap II, which can be used to display signals of a FPGA design in real-time. In order to test the functionality a simple circuit should be implemented first. It simply links the four keys of the DE1-SoC board with the lower four red LEDs at every rising edge of the 50 MHz clock signal.

   Perform following steps:

   i) Create a new Quartus project for the circuit, named **e_my_keys**.

   ii) In order to use the SignalTap II Logic Analyzer with the free edition of Quartus, the TalkBack must be activated via "Tools –> Options –> Internet Connectivity –> TalkBack Options... –> Enable sending TalkBack data to Altera".

   iii) Create a SignalTap II file via "File –> New... –> SignalTap II Logic Analyzer File" named **e_my_keys.stp**. Add SignalTap file to project.

   iv) For exploring easy trigger options include the KEYs as Nodes, set the trigger condition to **basic AND**, KEY(0) to be **Low** and all remaining KEYs to be treated as **Don't Care**. Recompile the project and test design probing with SignalTap.

   v) Also experiment with more advanced trigger options. For instance by using four trigger conditions. Find out the difference of specifying only one trigger condition but using multiple signals for that condition and using multiple trigger conditions. Also specify your own **Advanced** trigger condition in which on either edge of any of the four KEYs the trigger should fire.

   vi) Also investigate settings like **Sample Depth** and **Buffer Acquisition Modes**. Try a Sample Depth of 128 segmented to four 32 sample segments by using a **post trigger position** for only one basic AND trigger condition which fires at the falling edge of KEY(0).

2) *Simple Reaction Tester Circuit*

This example circuit is a quite simple reaction tester which can be used to test the reaction time of a person to a visual stimuli. The user initiates the test by pressing and releasing KEY(1). After a delay (round about 2.68 seconds) the circuit will turn on a LED. As response the player must press KEY(2) as soon as possible to turn the LED off again. The elapsed time will be displayed on the 7-segment displays in hundreds of a second.

The circuit has been defined according to a hierarchical structure in which multiple sub-level circuit undertake different small tasks. This is a good VHDL programming style since the resulting circuit is much easier to understand and to debug.

The block diagram of the circuit is given in following figure:

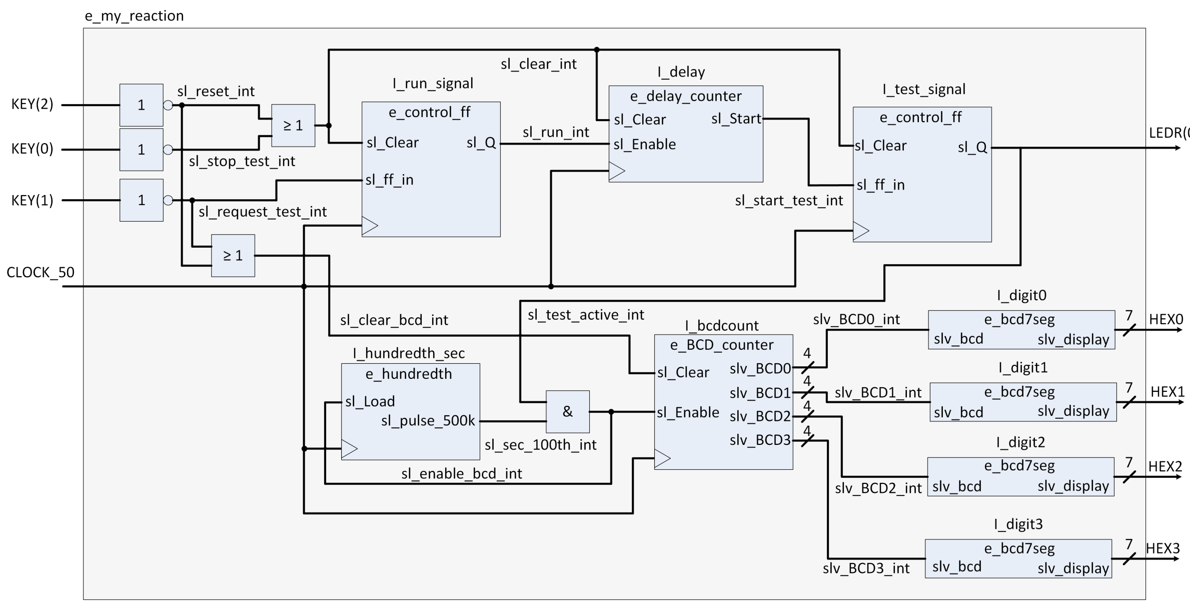

Figure 2.1: block diagram of reaction tester circuit

Perform following steps:

i) Create a new Quartus project for the circuit, named **e_my_reaction**.

ii) Describe the 7-segment decoder named **e_bcd7seg**, which decodes a 4-bit BCD number into 7-bit for driving a 7-segment display. Use a *combinatorial process* together with a **case** statement.

iii) Describe the control flip-flop component **e_control_ff**. The circuit creates an output signal which will be set to high if the data input sl_ff_in is high and stores this value even if sl_ff_in will change to low again until it will be reset by the Clear input.

iv) Describe the delay counter component **e_delay_counter**. This is basically a 28-bit upwards counter. The most-significant bit u_delay_count(27) will be set after round about 2.68 seconds. This signal will be used as sl_start_test signal.

v) Describe the hundredth second counter component **e_hundredth**. This is basically a downwards counter which creates an output impulse each time the counter reaches the value zero. In order to realize a hundredth seconds interval the counter each time will be preset to 0x7A120 (500.000). The output of the circuit sl_sec_100th allows for incrementing the BCD counter 100 times per second, as long as the red LED is turned on.

vi) Describe a stage of the BCD counter **e_BCD_stage**. Use following port signals: **sl_Clock**, **sl_Clear**, **sl_Ecount**, **slv_BCDq** and **sl_Value9**. During a synchronous clear, the BCD value should be reset to zero. Otherwise, only if the synchronous enable signal is set, the counter should be incremented. If the counting value is already nine, the counting value should be reset to zero. The output **sl_Value9** will only be set when the counting value is nine.

vii) Describe the BCD counter component **e_BCD_counter**. The circuit will instantiate four versions of the sub-level circuit **e_BCD_stage**.

viii) Describe the top-level entity **e_my_reaction**. Since the push-buttons of the DE1-SoC boards will yield a logic zero, if pressed, some signals (sl_reset, sl_request_-test and sl_stop_test) have been inverted. If the sl_request_test signal will change to high the run signal will be set, which activates the counter circuit. It will set the sl_start_test signal after a delay of approximately 2.6 seconds. The sl_start_test signal activates the sl_test_active signal, which will turn on LEDR(0). From that time on the four digit BCD counter starts counting in a hundredth second interval. It will be activated by an impulse (one clock cycle) which will be generated by the circuit e_hundredth. The four 7-segment digits BCD0 till BCD3 will be decoded by a 7-segment decoder circuit and displayed on the 7-segment displays HEX0 till HEX3. A clear signal, which will be set either by the reset signal or the sl_stop_test signal, will reset the control signals sl_run and sl_test_active to low, which freezes the displayed value on the 7-segment displays.

ix) Import the necessary pin assignments, compile, download the circuit and test its functionality in hardware.