Exercise Sheet 4
(10 Points)

Lab exercises for the period from Wednesday, 15th November 2017

The lab exercises take place at room OH16/U08. The exercise sheets will be solved during the exercise sessions.

For this exercise sheet, we use the virtual machine ES-FPGA.

4.1 Reaction Game with State Machine (5 Points)

In this exercise, you will implement a reaction game for two players according to the state machine illustrated in Figure 1. Since Visual State does not support VHDL as target language, you will have to do the implementation manually.

Figure 1: State machine of the game
Rules:

- There exist two players to each of which a button is assigned.
- A round of the game is started, when both players keep their buttons pressed.
- After a random time interval (3 – 10 seconds), “GO” is displayed and the players must release their buttons as quickly as possible.
- The first player who releases his or her button wins.
- If a player releases his or her button before “GO” is displayed, he or she is disqualified. This does not imply that the other player automatically wins, since he or she could also be disqualified.
- Another round can be started only after both players released their buttons.

Assignment: Implement the state machine describing the game in the provided file GameLogic. The transition from GAME_READY to GAME_COUNTDOWN has already been implemented by means of illustration. To implement the other state transitions, you have to request the triggers activating the respective state transitions in the branches of the case instruction using if ... then ... end if; and to then execute the respective actions and state transitions. If only a signal name as, e.g., player1_release is given in the diagram, this is a boolean signal which can be directly formulated as a condition of an if-instruction.

A VHDL example can be found on the last page of this exercise sheet.

4.2 Game Logic Improvement (5 Points)

Having implemented the given state chart, two problems become evident:

- The players can cheat - if one player releases his or her button during the waiting period and subsequently presses it again, he or she is disqualified but is still able to win the game. Make sure that a disqualified player is not accepted as the game’s winner.

- If both players are disqualified, the game still waits until the countdown is finished and, furthermore, the game can be restarted no earlier than after one player presses his or her key again (if you fixed the previous issue, it can be started never again). If both players are disqualified, the game should be aborted.

Implement a game abortion for the case that both players are disqualified.

General information: An overview about the exercise sessions as well as further information can be found on https://ls12-www.cs.tu-dortmund.de/daes/lehre/lehrveranstaltungen/wintersemester-20172018/es-1718.html. The exercise sheets will usually be published on the course website on Mondays and will be solved during the respective exercise sessions. The exercises are divided into two parts, in each of which at least 50% of the points must be achieved in order to receive the exam admission.
VHDL overview

-- the entity should have at least one input called "clock" and one output called "output"

library IEEE;
use IEEE.std_logic_1164.all;

architecture Behavioral of Example2 is

-- enumeration types are useful for state machines

type demo_enum is (ENUM_A, ENUM_B, ENUM_C);

signal state: demo_enum := ENUM_A;

signal bool_sig: boolean; -- can be only "true" or "false"

signal sl_sig : std_logic := '0'; -- can be '0' or '1' (or 'U'ndefined or...)

signal toggle : boolean := false; -- some more example signals

signal delay : integer;

signal another_one : std_logic;

begin

-- processes are blocks of instructions that are always executed

process(Clock) begin

if rising_edge(Clock) then

-- do something only if the clock changes from 0->1

-- everything here is executed once at each rising clock edge

-- more precisely, in the order in which it is listed here

-- BUT: The "visible" signal values change no earlier after the process is finished

-- "toggle" changes between true and false at each clock pulse

-- boolean values directly appear in if-statements

-- nested if-statements can also be used:

-- commands that are executed when state = ENUM_A

-- commands that are executed when state = ENUM_B

-- commands that are executed when state = ENUM_C

-- the complete case block must have an "end"

-- the processes are finished with "end"

end if;

end process;

end Behavioral;