

Bilkent University

Computer Science

CS 223-Digital Design

**GAME OF CODES**

Final Project Report

Ayça Begüm Taşçıoğlu

ID: 21600907

Section: 01

Package: #4

Date: 18.12.2017

1. **MODULES**

module **keypad4X4**

(clk,

keyb\_row, // just connect them to FPGA pins, row scanner

keyb\_col, // just connect them to FPGA pins, column scanner

key\_value, //user's output code of detected pressed key: row[1:0]\_col[1:0]

key\_valid); // user's output valid: if the key is pressed long enough (more than 20~40 ms), key\_valid becomes '1' for just one clock cycle.

module **press\_control**(

input logic clk,

input logic key\_valid,

input logic reset,

output logic control )

* This module includes a simple Moore FSM, this FSM’ s states represents the statement of pressing of the button.

S0: statement of reset or not pressing a button so key\_valid is zero. if key\_valid increases from 0 to one, move on to S1, else stay in S0. Output is zero.

S1: button pressed yet, user has not been pressing the button so key valid is one and output is zero. If user keeps the button pressed, move on to the S2, else move on to S0.

S2: button is pressing for a while. so key valid is 1 yet, output is zero. If user keeps pressing, stay in S2, else move on to S0.

module **Pseudo\_random\_generator**(

input logic clk,

input logic reset,

output logic [3:0] random\_value)

* This module produces a random 16 bit number by connecting selected bits of the value to XOR gates (LFSR design). Outputs the last 4 bits of this 16 bit number.

module **randomRegister**( input logic clk,

input logic reset,

input [3:0] random\_value,

input key\_valid,

output logic [1:0] rotation\_duration,

output logic [1:0] direction,

output logic [3:0] out);

* Keeps the value which is produced by Pseudo\_random\_generator module. While it stores the random value, this module outputs the last 2 bit of the random number as direction, first 2 bit of the random number as rotation\_duration.

module **steppermotor\_wrapper**(

input clk, //100Mhz on Basys3

//user input for motor rotation direction.

// direction[0]: first movement

// direction[1]: second movement

// 1'b0: left 1'b1: right

input [1:0] direction,

//user input for motor rotation duration.

// rotation\_duration[0]: first movement

// rotation\_duration[1]: second movement

// 1'b0: short 1'b1: long

input [1:0] rotation\_duration,

// just connect them to FPGA (motor drivers)

output [3:0] phases,

//user input to initiate motor. a pulse (at least one clock cycle) starts 2 movements of motor.

// if you re-apply it before the motor finishes both movements, new start command is ignored.

input start

);

module **Comparator**(

input clk,

input logic [3:0] key\_value,

input logic [3:0] random\_toControl,

output logic point)

* This module takes clk signal, key\_value (the input value which is taken by the user) and random\_to\_control as a random value which is provided by pseudo\_random\_generator module.
* In this module, key\_value and the random\_to\_compare are compared in this module. According to this calculation, this module produces the output: point. The point turns 1 if these values are equal else, the point turns to zero.
* Moreover, Comparator module includes the map, according to this map game code is created.

MAP:

direction

rotation\_duration

|  | 0 0 | 0 1 | 1 0 | 1 1 |
| --- | --- | --- | --- | --- |
| 0 0 | 0 | 7 | D | # |
| 0 1 | 4 | 1 | 8 | \* |
| 1 0 | A | 5 | 2 | 9 |
| 1 1 | C | B | 6 | 3 |

module **Comparator\_cont**(

input logic point,

input logic clk,

input logic reset,

input logic key\_valid,

output logic [3:0] in0

)

* Comparator\_cont module updates the score regarding to the point input. If point input is 1, and key\_valid is 1 and the in0(output of this module in other words, in0 presents score in this game) is less than 9 in decimal numbers, it increases the in0.
* If point is 0 and if key\_valid is 1, it decreases the in0.

module **SevSeg\_4digit**

(clk,

in0, in1, in2, in3, //user inputs for each digit (hexadecimal value)

a, b, c, d, e, f, g, .dp(\<const0> ), // just connect them to FPGA pins (individual LEDs).

an // just connect them to FPGA pins (enable vector for 4 digits active low) )

**2. REFERENCES**

* SevSeg\_4digit module, steppermotor\_wrapper module, steppermotor module and keypad4x4 module is given in the project files, in ready modules.
* In FSM - press\_control module, I got helped from the Chapter 4 slides, in System Verilog.
* For Pseudo\_random\_generator module, I checked on the internet yet, I selected bits arbitrarily.

**3. CHANGES IN HIGH LEVEL DESIGN**

* FSM - press\_control module is added because pressing and keeping pressing differs and may cause false score increments.
* Comparator\_cont module is added to separate map and score comparator.