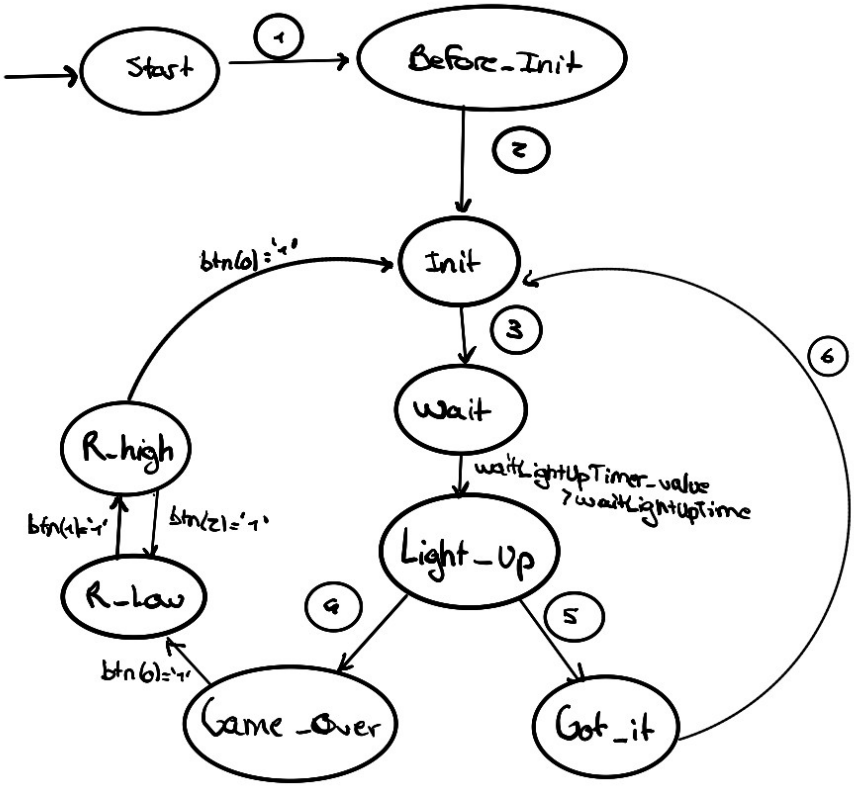
Laramas
Rémi

Project Documentation : Speden Spelit Game

Introduction

"Implementing a timer game. The user presses the button as fast as he can after a light has been turned on. The button pressed must be at the same position as the led. The game stops when the player missed either the position or the time taken."
The objective of this project was to implement a similar game to the Speden Spelit one using VHDL on a Zybo board. As of now the project contains 3 components this documentation will go through.

Theory : State-Machine



(1) After the player pushed the button 0, the component will go to the **Before_Init** state
(2) The component switches automatically to the **Init** state and initiate the random streams by pushing their reset signal to '1'
(3) The component switches automatically to the **Wait** state. It chose random values for the waiting time between the **Wait** and **Light_Up** states and the goal.
(4) If btn is triggered and the value is equal to the goal signal, then it switches to **Got_it** state. Otherwise, it switches to **Game over**. In every case, timers or stopped an reset.
(5) --

(6) First the gotItTimer is triggered. Then, once it reaches the celebrationTime value, it switches to the **Init** state and reset the goal value and gotItTimer.

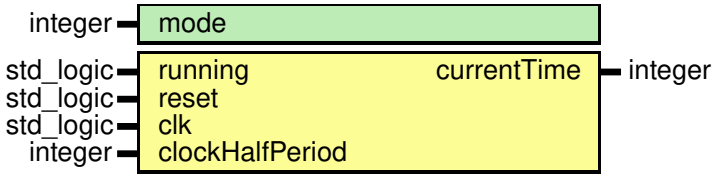
Just below you can find descriptions about the different states and their meaning :

Start	It is just a wrapper for the whole game.
Before_start	Allows to initialize the random number streams
Init	Set the goal and wait time before lightning up
Wait	Wait the right time before lightning up the led
Light_Up	Tracks the time and the buttons the user pushes.
Got_It	Just send a light signal that last a certain amount of time to inform the player he won the round. It also increases the score
Game_Over	Turn off the leds so the player understands he lost.
R_high	Print the second digit of the score in BSD format
R_low	Print the first digit of the score in BSD format

Component's Documentation

timer

Schema

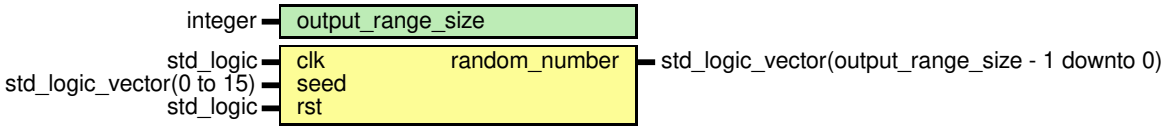


Notes

The running input behaves like a start/pause button. The mode represents the unit given in output : 0 stands for ns, 1 for us and 2 for ms (it theoretically can go higher). The timer updates every clock switch : in this model it means every 4ns. That's why some results might appear incoherent (the Test 4 from the Timer test bench for example).

randomNumberGenerator

Schema

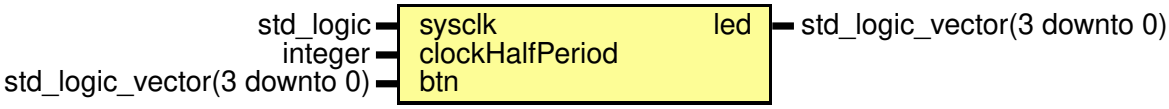


Notes

The algorithm behind is the Fibonnacci LFSR one (more details on the wikipedia page : https://en.wikipedia.org/wiki/Linear-feedback_shift_register)

gameStateMachine

Schema



Notes

It is the main component. It implements the state machine mentioned above.

Tests

Component / Test Bench	Simulation Time	Description	
gameStateMachine/ gameStateMachine_ tb	300 us *(see comment below)*	Test 1	**One win, and then one loose** : the test will simulate a player wining once and then loosing, verifying the score, and the states.
		Test 2	**Loosing straight** : the test will simulate a player loosing straight.
		Test 3	**Overtime after light up :** the test will simulate a player taking too much time to push the button after the led turned on
		Test 4	**Verifying the results** : the test will simulate a player wining 16 times and will verify if the score is coherent.
timer/timer_tb	2.01 ms	Test 1	**Start and stop running** : the test will start the timer, stop it just after and wait. It then verify the time timed.
		Test 2	**Restart** : Look at if the timer kept in memory the state of his internal counters.
		Test 3	**Reset** : Verify that the reset functionality is implemented as it should be .
		Test 4	**Count in ns** : Test if the timer is accurate for a period of 50 ns
		Test 5	**Count in us** : Test if the timer is accurate for a period of 10 us
		Test 6	**Count in ms** : Test if the timer is accurate for a period of 2 ms
randomNumberGen	50 ns	Test 1	**Reset** : Test if the reset

erator/ randomNumberGenerator_tb			functionality works
		Test 2	**Clock updates :** Test if the generator updates each clock switch

For the test to work properly, it is advised to edit the mode of each counter in the “gameStateMachine.vhd” file from 2 to 1 (line 143,158 and 173). It’ll lower the scale from ms to us, which will let the simulation run faster.