

**CS223 – Digital Design**

**Digital Design Project – Cellular Automata**

**Section – 4**

**Ümit Yiğit Başaran**

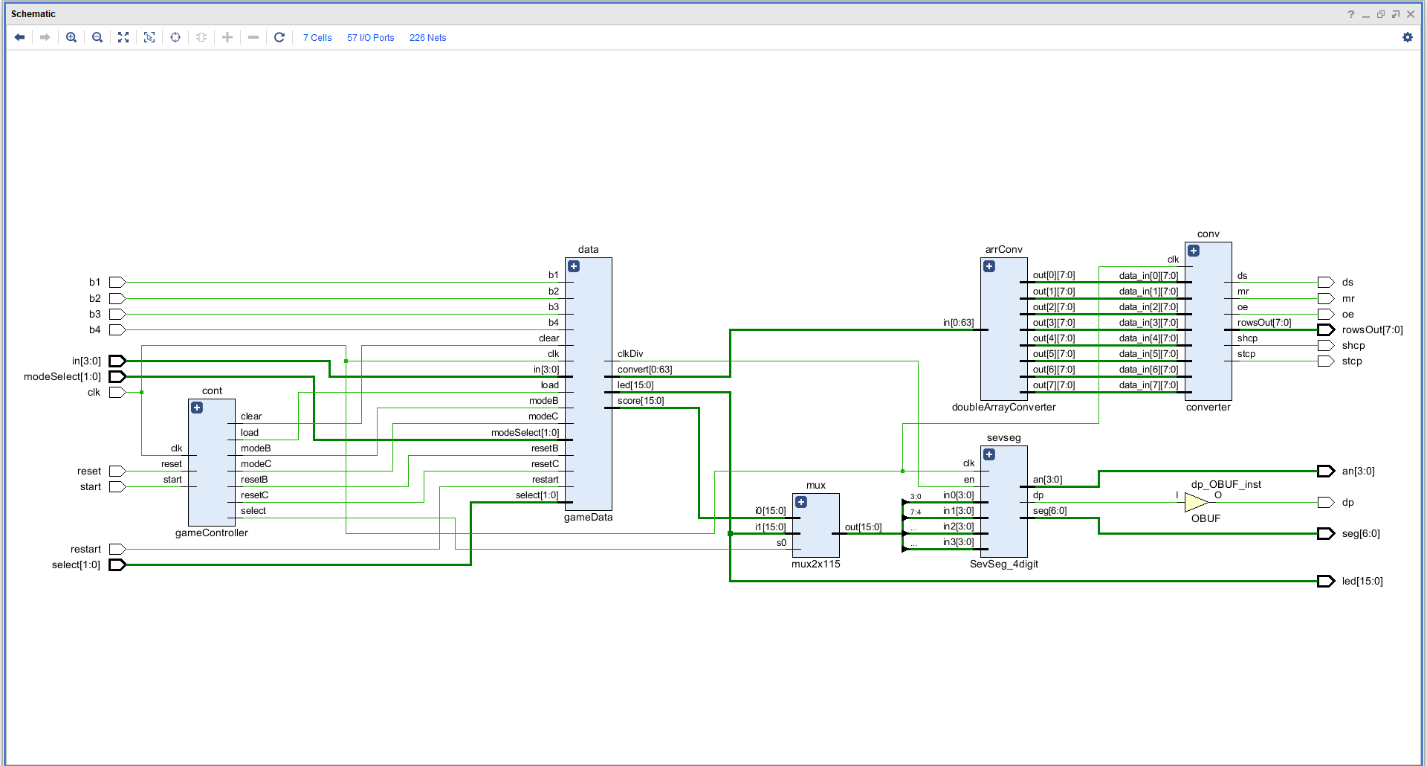
**21704103**

**29.12.2019**

1. **Introduction**

In this project, the main purpose is taking 64-bit number from the user and implementing these number to a game with its logic which is determined by my own student ID. In order to design this project, firstly I design a controller which is designed by using finite state machines and a data path for each game part. Then, I implement these designs to FPGA and BetiBoard (8x8 Matrix, Seven Segment, Led, Switches, Buttons) by using Vivado and System Verilog.

1. **Block Diagram**



**Explanation:**

Inputs:

* “b1” is the up button of the FPGA which plays the game for each “1” which is labeled in the 8x8 matrix.
* “b2” is the right button of the FPGA which plays the game for each “2” which is labeled in the 8x8 matrix.
* “b3” is the down button of the FPGA which plays the game for each “3” which is labeled in the 8x8 matrix.
* “b4” is the left button of the FPGA which plays the game for each “4” which is labeled in the 8x8 matrix.
* “in [3:0]” is the 4-bit input that is taken by the switches.
* “select [1:0]” is the 2-bit input that is taken by switches in order to select the digit wanted to modify.
* “modeSelect [1:0]” is the 2-bit input that is taken by switches in order to select, see and modify the 16-bit hexadecimal number in the Seven Segment.
* “clk” is the clock signal that is provided by the FPGA.
* “reset” is the middle button of the FPGA which resets all game and starts from the beginning of the game.
* “restart” is the 1-bit input that is taken by switch in order to restart the game depends on first given 64-bit input.

Outputs:

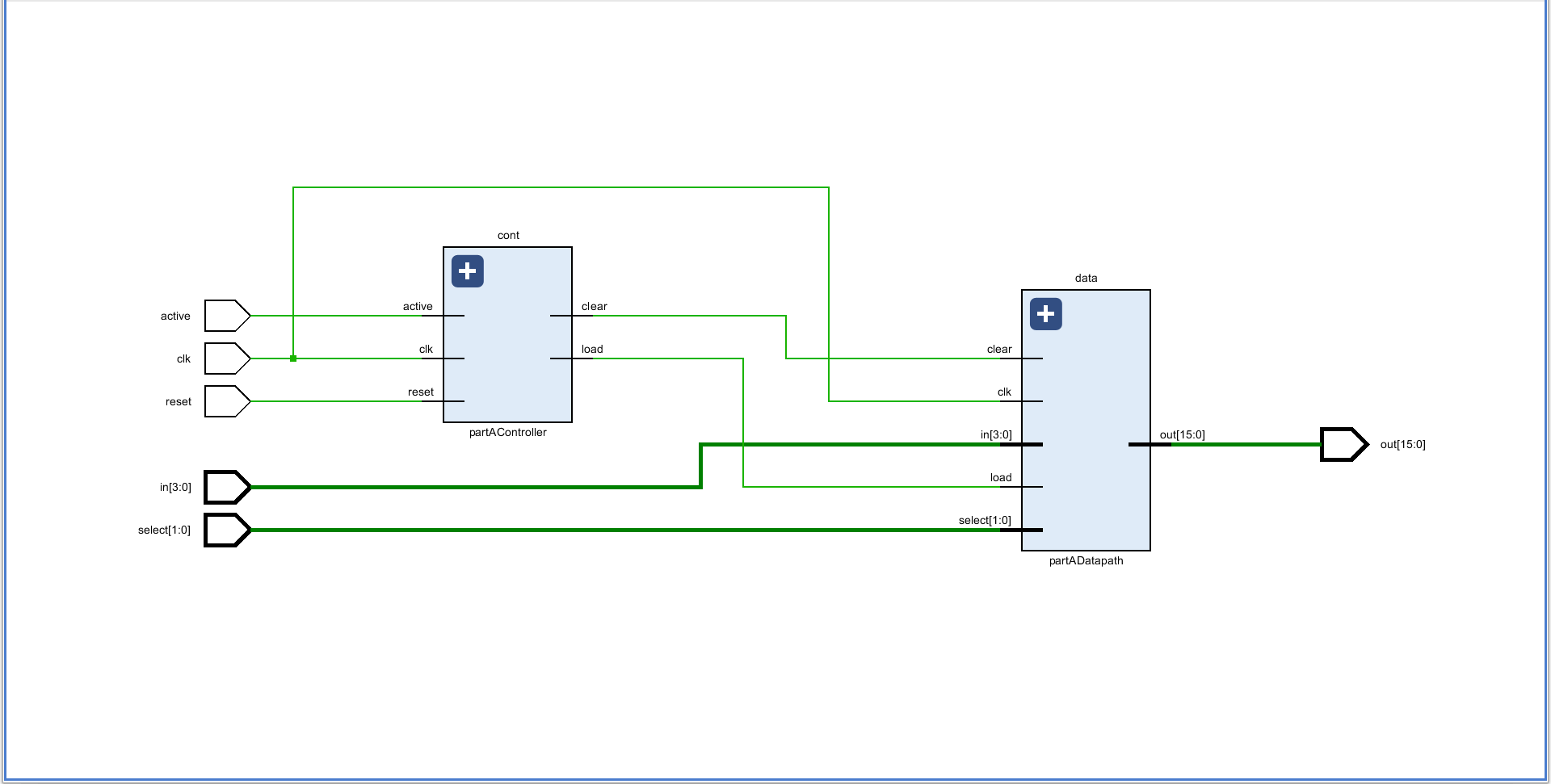
* “led [15:0]” is the 16-bit output that is represented in the leds of the FPGA in order to show every 16-bit of the 64-bit.
* “seg [6:0], dp, an [3:0]” are the outputs of the “SevSeg\_4digit” module in order to show outputs (every 16-bit of the 64-bit and score) on the Seven Segment.
* “ds, mr, oe, rowsOut [7:0], shcp, stcp” are the outputs of the “converter” module in order to show the game in the 8x8 matrix.

Modules:

* “cont” is the controller of the whole game. It controls the part B and part C connections.
* “data” is the datapath of the whole game. It contains part B and part C and combine them depends on signals that are come from controller.
* “mux” is a 2x1 multiplexer that decides whether Seven Segment shows the 16-bit of the 64-bit output or score depends on select signal that is come from controller.
* “arrCon” converts a 64-bit string to 8x8 double array and throw it to “converter”.
* “converter” throw doble array to 8x8 matrix to display.
* “sevseg” shows the decided output on the Seven Segment.

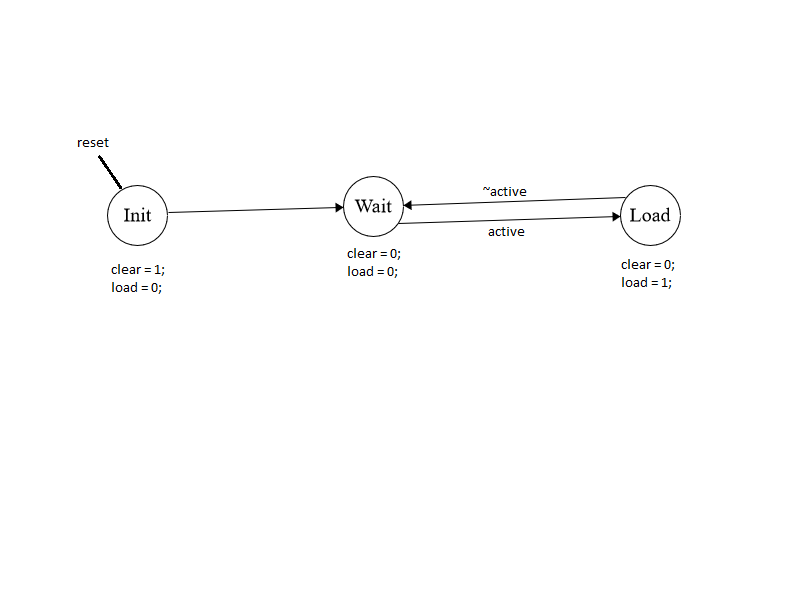
1. **Detailed Explanation of the Work**

**Part 1:**

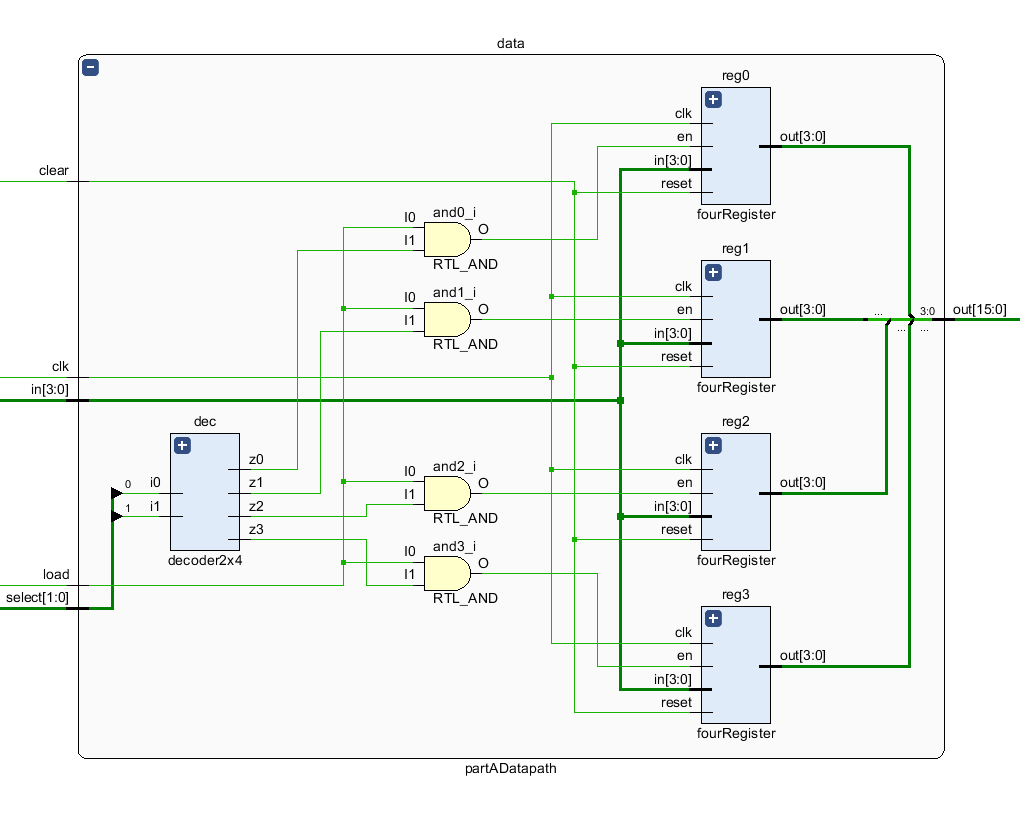


**Explanation:**

Controller:

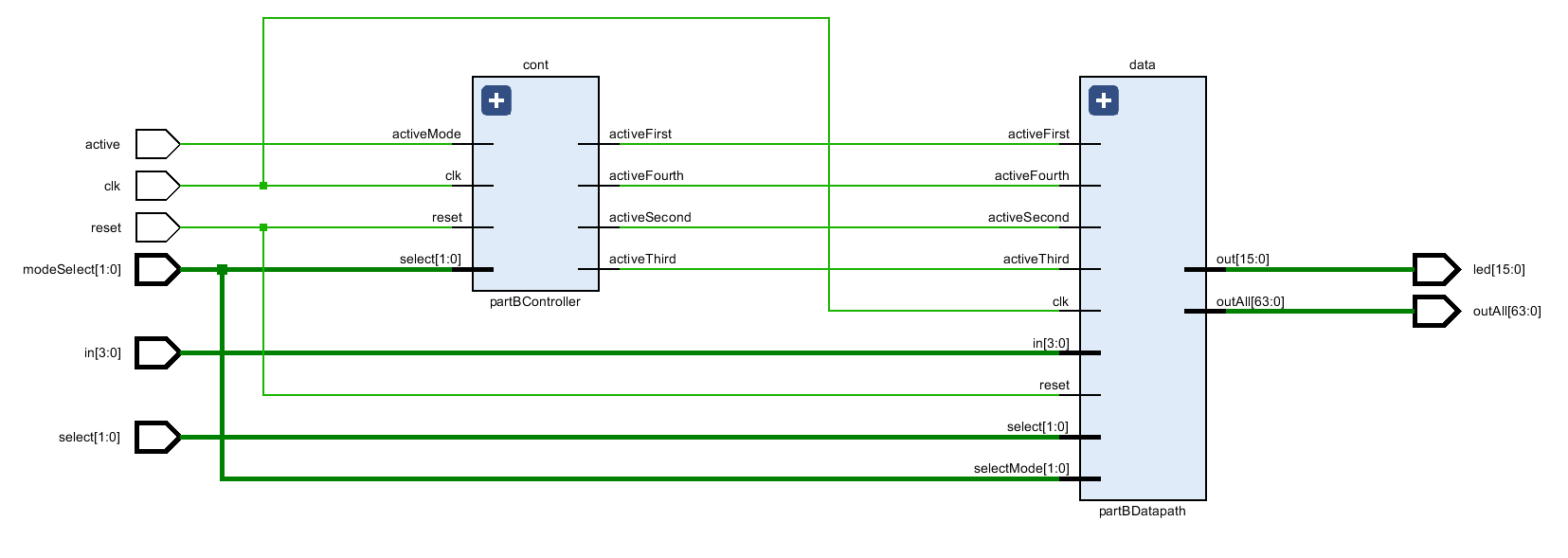


Datapath:



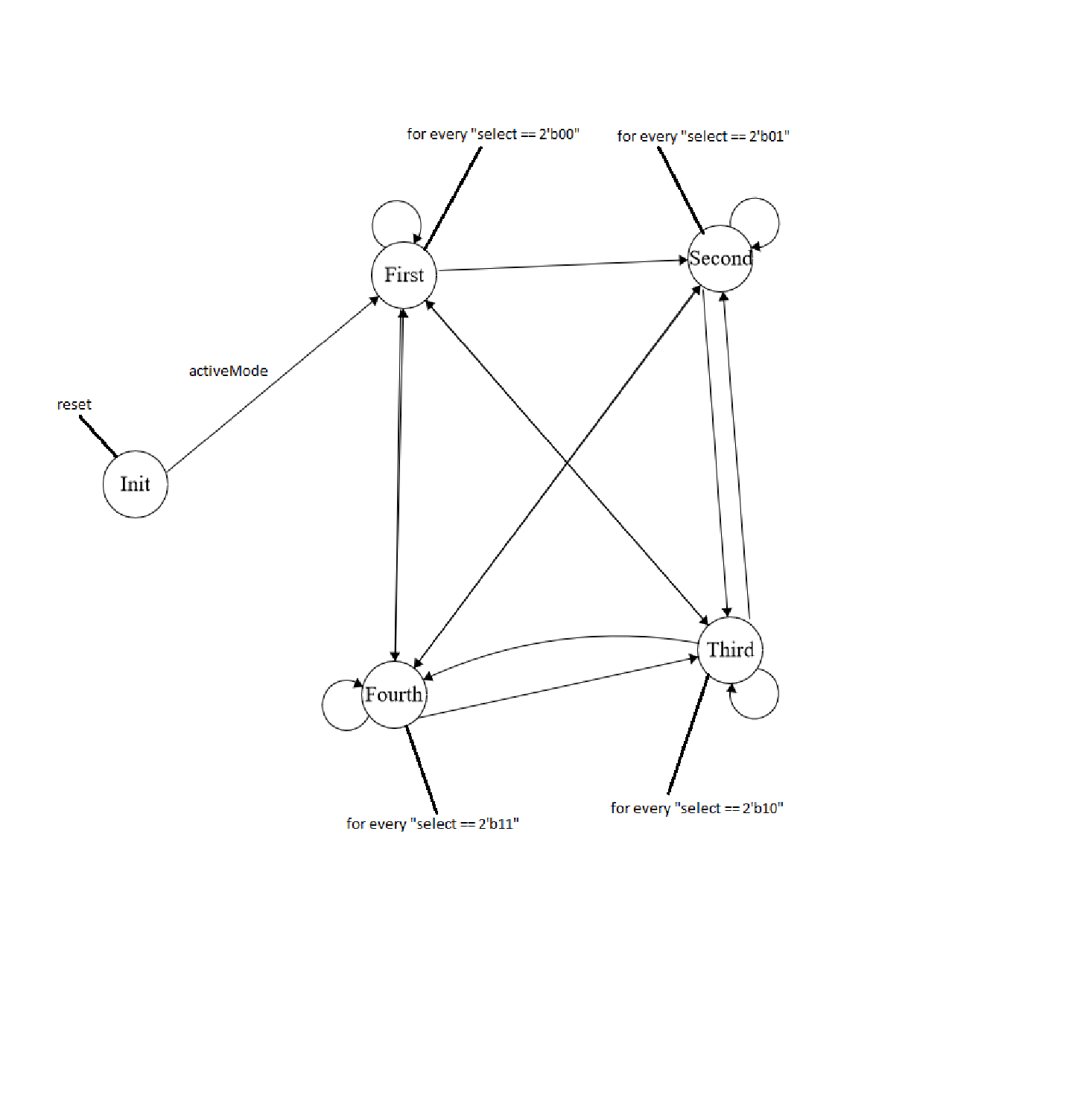
Here, I design a datapath that is controlled by a controller which provides inputs “clear” and “load”. These inputs control the registers depends on the “select [1:0]” input.

**Part 2:**

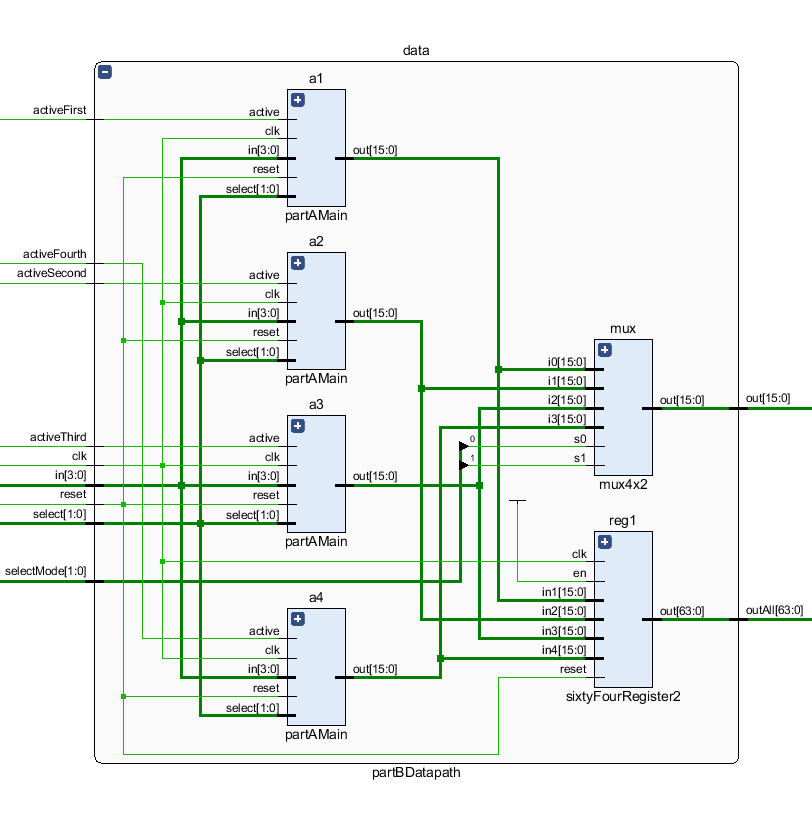


**Explanation:**

Controller:

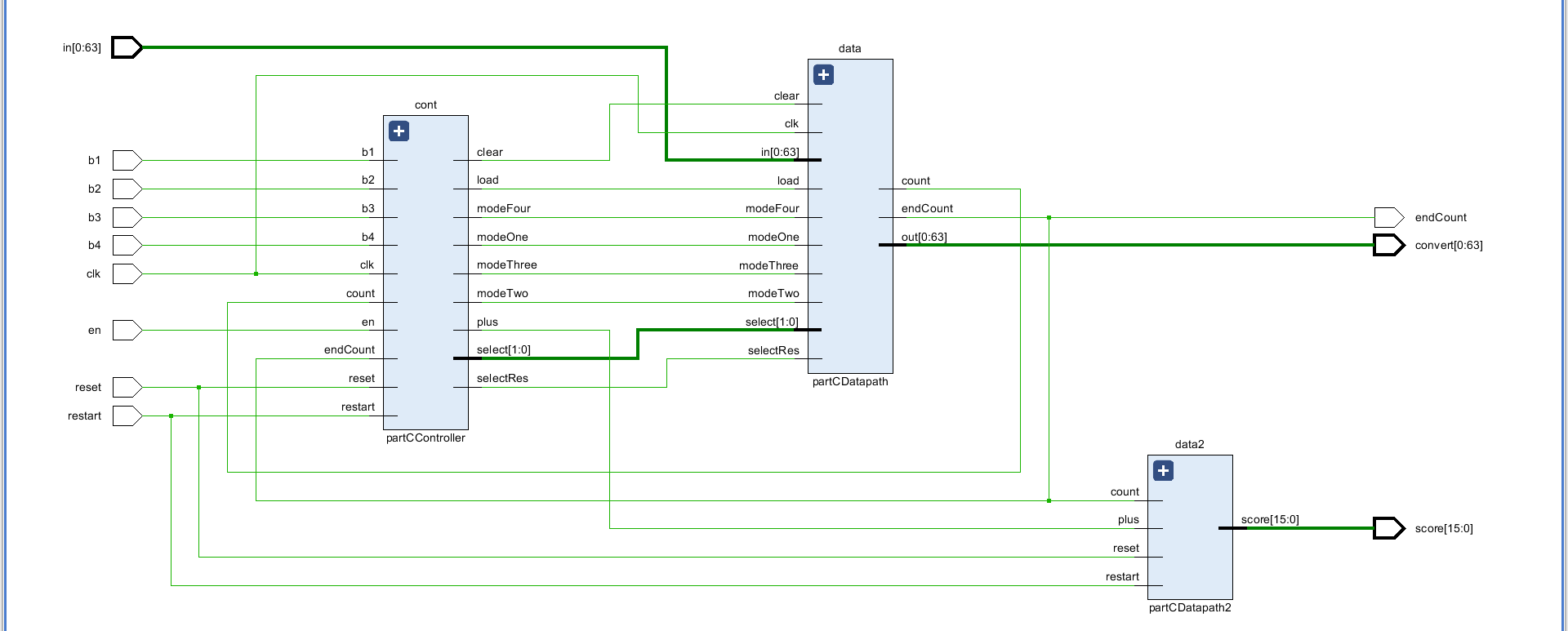


Datapath:



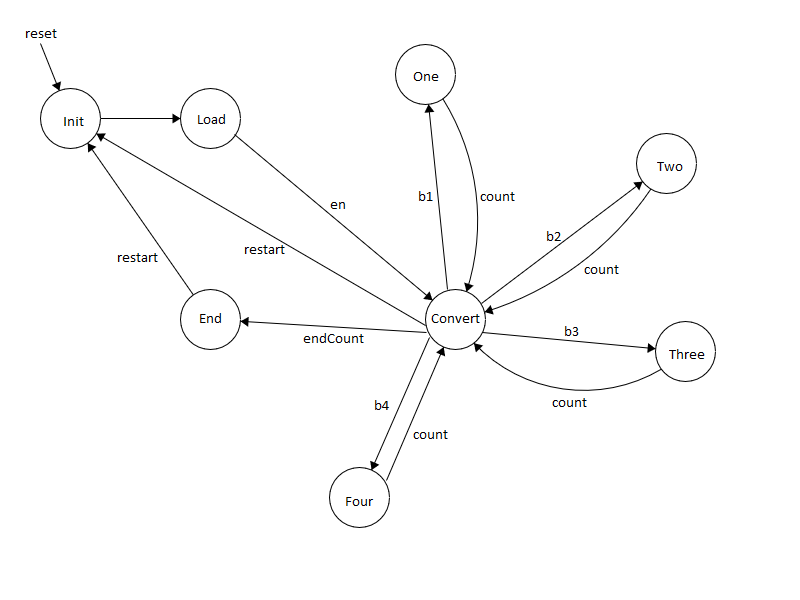
Here, I design a datapath that is controlled by a controller which provides inputs “activeFirst”, “activeSecond”, “activeThird” and “activeFourth”. These inputs control each of the “partAMain” modules depends on the “select [1:0]” and “selectMode [1:0]” inputs. “mux” module provides a 16-bit output depends on “selectMode [1:0]” inputs. “reg1” module stores and combine all the 16-bit outputs that are provided by “partAMain” modules and provides a 64-bit output.

**Part 3:**

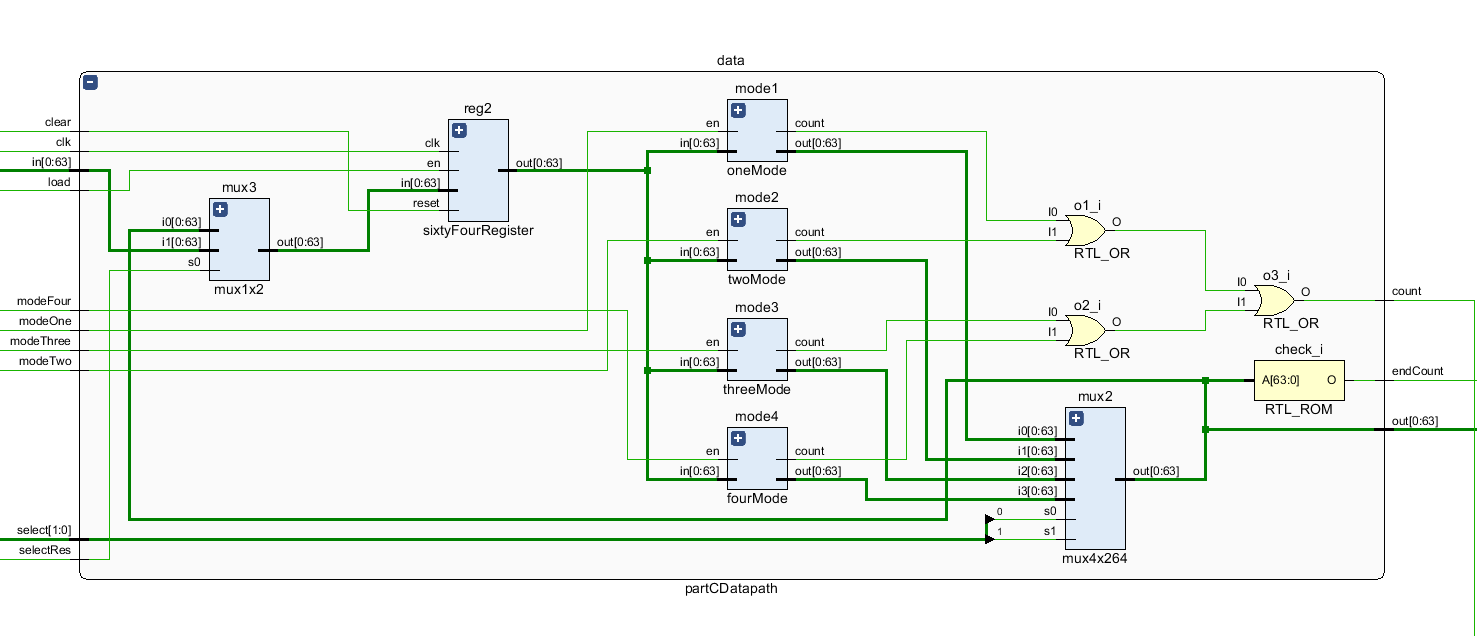


**Explanation:**

Controller:

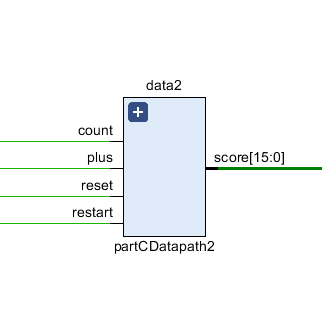


Datapath - 1:



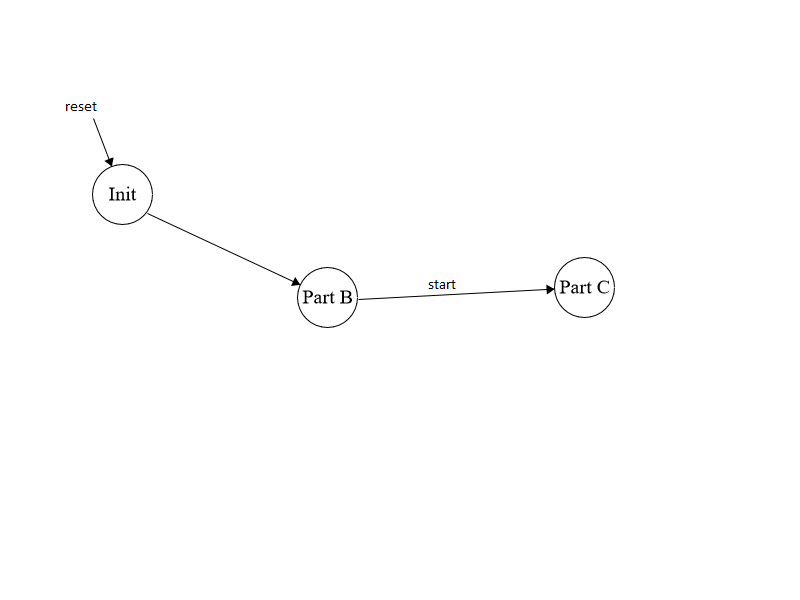
Here, I design a datapath that is controlled by a controller which provides inputs “clear”, “load”, “modeOne”, “modeTwo”, “modeThree”, “modeFour”, “select [1:0]” and “selectRes”. These inputs control “mux3”, “reg2”, “mode1”, “mode2”, “mode3”, “mode4” and “mux2” modules depends on the states in the finite state machine. “mux3” module decide whether using first given input or modified input by using “selectRes” signal. “reg2” module stores the 64-bit input that are whether modified or not modified depends on “mux3”. “mode1” plays the game for every cell which is labeled with one and “modeOne” signal activate this mode. “mode2” plays the game for every cell which is labeled with two and “modeTwo” signal activate this mode. “mode3” plays the game for every cell which is labeled with three and “modeThree” signal activate this mode. “mode4” plays the game for every cell which is labeled with four and “modeFour” signal activate this mode. “check\_i” function checks the output and returns one if the output equals to zero or returns zero if the output is not equal to zero. “count” output that is provided by mode modules set modules in order to play the game one time for each button press and combined count signal goes to the controller and causes to change of state.

Datapath – 2:

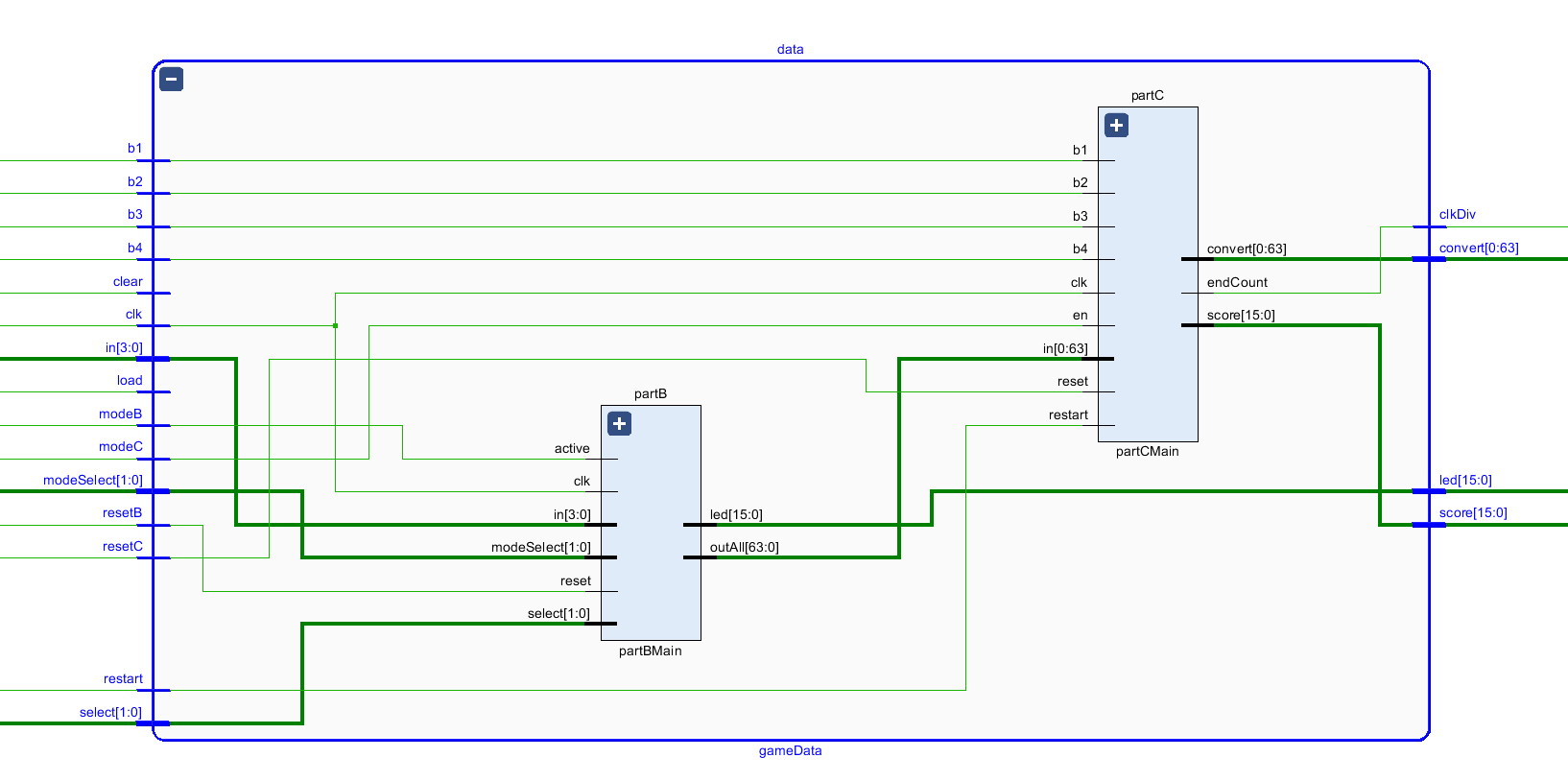


Here, I design a datapath that is controlled by “count (endCount from datapath - 1)” and “plus (from controller)”. “plus” signal adds one to the current score. “count” signal stops the summation when the game is over. “reset” and “restart” signals clear the score and syncs it to zero. I combine “reset” and “restart” signals with using an or gate.

**Game Controller:**



**Game Datapath:**



Here, I design a datapath which is controlled by a controller and combines part B and part C of the game.

1. Conclusion

As a conclusion, in this project we made a game that starts with taking 64-bit values from the switches and then, we start to play it with using different buttons. In the design part, I use controllers and datapaths for every part of the game. I think the most challenging part of the project is the combining all the controllers and datapaths in one top module.

1. References & Appendix

* Digital Desgn and Computer Architecture, David M. Harris, Sarah L. Harris, 2013 Second Edition, Morgan Kaufmann
* Digital Design, with RTL Design ,VHDL and Verilog, Farank Vahid, 2011 2nd Edition, John Wiley