1. Introduction:

In this project a simple CPU and 3D graphic generator is designed. To demonstrate the capability of these design, a video game is coded with the CPU’s machine language. Furthermore, external storage unit is used to store data. As an external device parallel shift registers, and monitor with VGA protocol are used.

1. Design:

The main design consists of 4 modules: CPU, RAM, 3D graphic generator and controller unit.

1. Controller Unit:

Controller unit takes external data about buttons from BASYS pmod pins. Buttons data is loaded in parallel manner into parallel load shift register, when the parallel load pin of external shift register 74HC165 is enabled by BASYS. Then FPGA card sends clock signal to the register. At each clock signal new data arrival to the card. At the end of the cycle coming values are stored into FPGA inner register to use from CPU, and controller part send parallel load signal again to register to load next button values. To prevent bouncing in button values and staying in the operational range of 74HC165, clock divider is used to reduce clock frequency.

1. CPU (Central Processing Unit):

CPU control the coded video game according to its given instructions. It has four sub-modules. First module is ALU, it conducts arithmetic logic operations such as summation, subtraction, multiplication, division, and, or, xor. It has three inputs and one output. Two inputs take coming numbers, other input chooses operation. Output gives result. Second sub-module is instruction ram, it stores command of CPU. Third one is register. It stores the temporal data of running program. It compromises of 16 columns and 64 rows, so that its storage capacity is 128 bytes. Fourth part is control unit. It menages the CPU according to current instruction. It also determines the next instruction. If current instruction is not branching command, it increment address of instruction ram one in each code step. If instruction is branching command such as jump or if it arranges new address according to them. CPU also takes data from controllers, and it write them to register. Additionally, CPU can send or take data from RAM with asked address. These operations are called fetch.

CPU architecture is simple, and it is single cycle which all command is completed in only one cycle. Because design is not complex it can be said to have RISC architecture. CPU has 32 different instruction and each instruction consist of 32-bits. First five bits specify type of command, remaining bits determine register-ram address or constant according to type of instruction.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type of Instruction [31:27] | First Register Address [26:21] | Third Register Address [20:15] | Idle [14:8] | RAM Address [7:0] |
| Constant Value [14:0] | |
| Idle [14:6] | Second Register [5:0] |

Table 1: Instruction’s Bits Duties

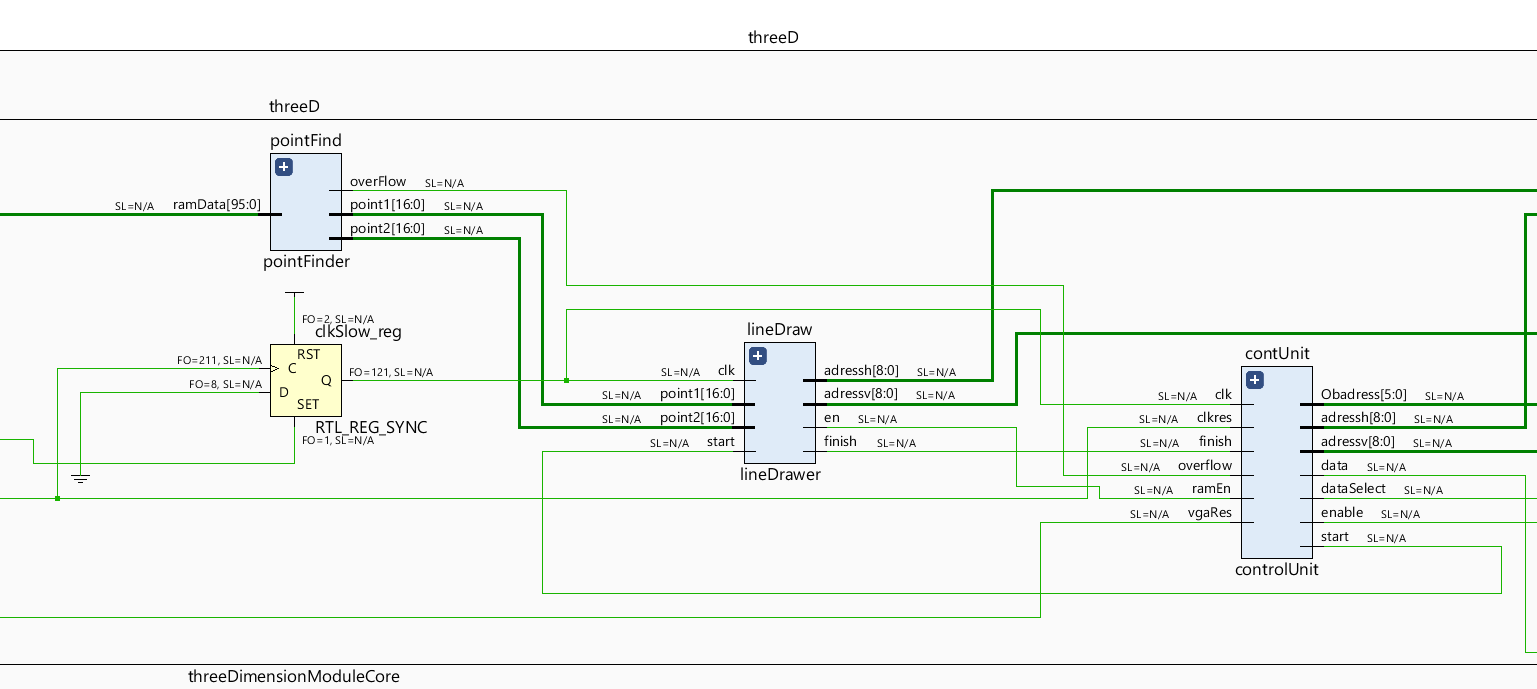
|  |  |  |
| --- | --- | --- |
| Order of Instruction | Task Of Instruction | Location in Storage Elements |
| 0 | Summation | r3<=r1+r2 |
| 1 | Subtraction | r3<=r1-r2 |
| 2 | Multiplication | r3<=r1\*r2 |
| 3 | Division | r3<=r1/r2 |
| 4 | Modulus | r3<=r1%r2 |
| 5 | OR | r3<=r1 or r2 |
| 6 | AND | r3<=r1 and r2 |
| 7 | XOR | r3<=r1 xor r2 |
| 8 | Shift Left | r3<=r1<<cons |
| 9 | Shift Right | r3<=r1>>cons |
| 10 | Summation 2 | r3<=r1+cons |
| 11 | Subtraction 2 | r3<=r1-cons |
| 12 | Multiplication 2 | r3<=r1\*cons |
| 13 | Division 2 | r3<=r1/cons |
| 14 | Modulus 2 | r3<=r1%cons |
| 15 | OR 2 | r3<=r1 or cons |
| 16 | AND 2 | r3<=r1 and cons |
| 17 | XOR 2 | r3<=r1 xor cons |
| 18 | Subtraction 3 | r3<=cons-r1 |
| 19 | Division 3 | r3<=cons/r1 |
| 20 | Negation | r3<=-r1 |
| 21 | Take from RAM | r3<=RAM |
| 22 | Send to RAM | RAM<=r1 |
| 23 | Copy to register | r3<=r1 |
| 24 | Write to register | r3<=cons |
| 25 | Jump | Program counter<=cons |
| 26 | If greater 1 | Program counter<=++2 |
| 27 | If smaller | Program counter<=++2 |
| 28 | If greater 2 | Program counter<=++2 |
| 29 | Idle | None |
| 30 | Idle | None |
| 31 | Idle | None |

Table 2: Instructions Set. Red ones are arithmetic-logic operations, green ones are fetch operations, blue ones are branching operations, yellow ones are idle

1. RAM:

The RAM module also stores data, but it is bigger than CPU register, and 3D unit takes position data from here. RAM module is connected to both CPU and GPU. Nevertheless, they use different data and address busses. Additionally, GPU does not have permission to write any data into RAM. CPU has WE cable connecting to RAM. When, this line is logic one, CPU writes values into RAM with specified address. When, this line is logic zero, RAM sends data in specified address to CPU.

1. 3D Graphic Unit:

This module conducts calculation of 3D graphic and communicates with the monitor with VGA port. 3D graphic is generated with lines which compose bigger geometric shape. The lines’ values are stored at the RAM. It consists of point-finder, line-drawer, screen RAM, VGA communication unit, and control unit. The point-finder takes position data from RAM. This data is word with 96-bits length. First 48 bits include Cartesian coordinate values of point which constitutes one end of line. Second 48 bits include second point’s value. According to this point location, the point-finder finds projection of tips of line onto screen plane. If points are not in the screen range, it gives overflow error. The line-drawer unit draw line on the screen between points founded by point-finder, and delivers its result to the screen RAM. The screen RAM stores the pixels’ values. It stores these values as bit; thus, in the screen there are only white and black regions. The VGA communication module, transfer the screen RAM values to monitor via VGA protocol. The control unit manages the 3D unit. It chooses which line data is brought to the point-finder, clears the screen ram and has lines drawn before VGA monitor passes to new frame.

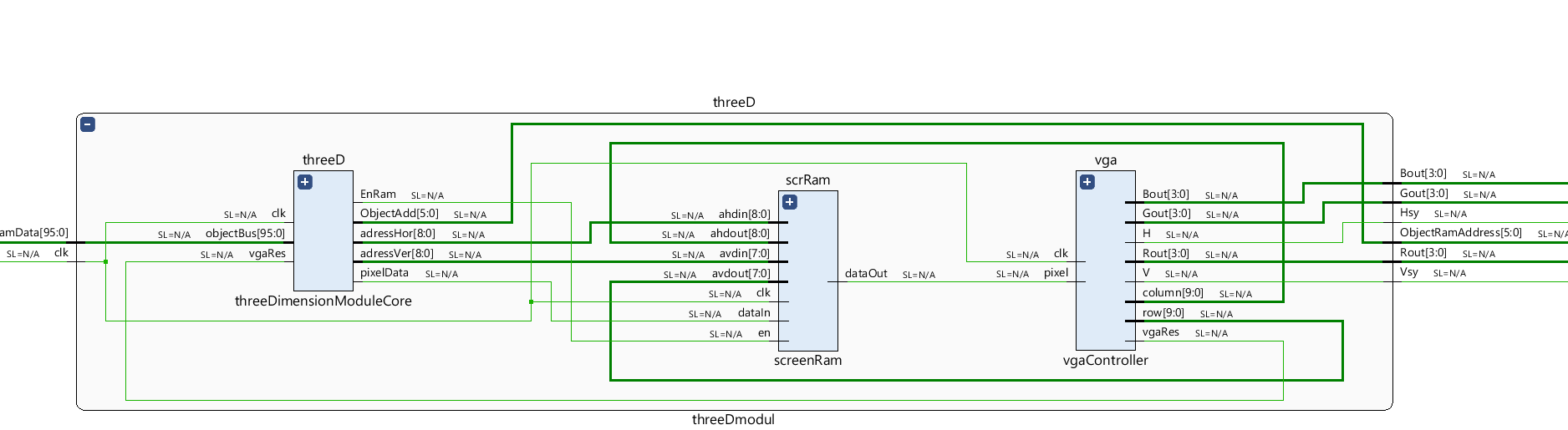
Figure 1: 3D Graphic Generator Module

Figure 2: 3D Graphic Generator Core Module

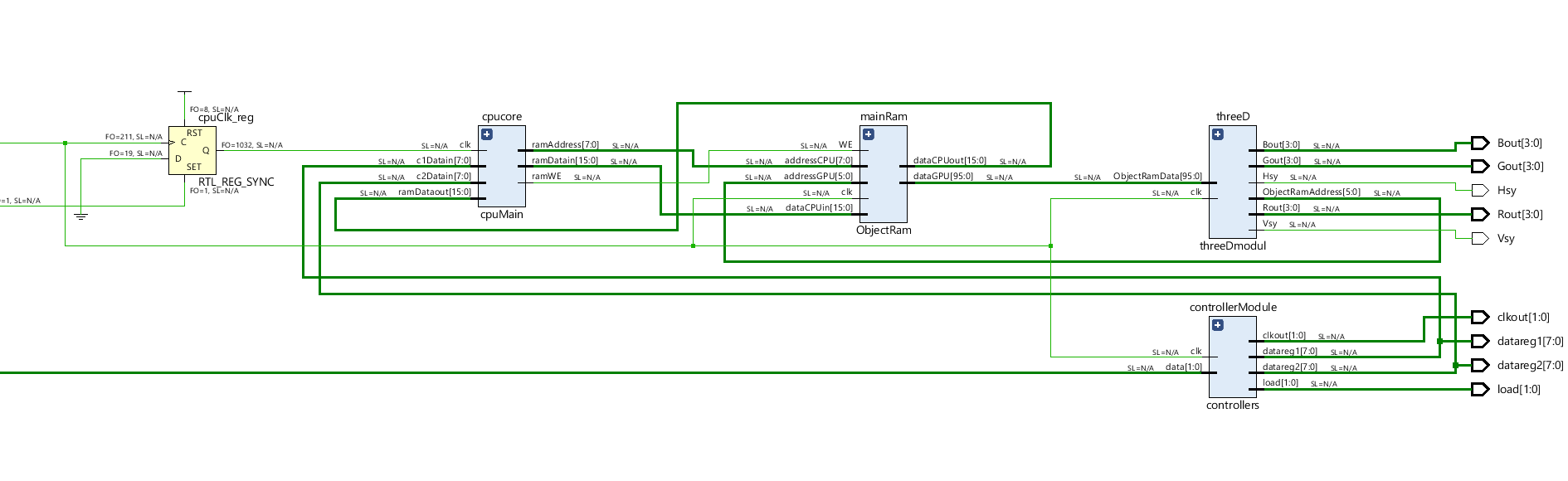


Figure 3: Main Design (except CPU clock division unit)

1. Video Game:

In the video there are two spaceship whose shapes are cube. One of these cubes can fire to right side while the other try to escape this fire. Cubes can only move in x and y axis. These movement and fire orders are directed by controllers. If escaping cube is shot, the game stops. To reset the game, the FPGA is reset.