Report
on
COEN 316 LABORATORY EXPERIMENT #5

DATAPATH/CONTROL UNIT INTEGRATION AND SYSTEM TESTING

Submitted to

Hasiksishna Menon Thelakkal

Lab demonstrator's name

Date: December 5th, 2016
Due Date: December 6th, 2016

By

Michael Gravino	27088140
Name	student ID

Lab section: DI

Objectives

The objective of this laboratory is to design the CPU datapath and control unit and combine all previous labs to obtain a working CPU.

Theory

VHDL is used as a tool that allows the possibility to read the behavior of an electrical circuit that can be written by the user and it can also be converted to a physical component. It uses computers to design the digital circuits and it helps in the creation of more complex circuits as well

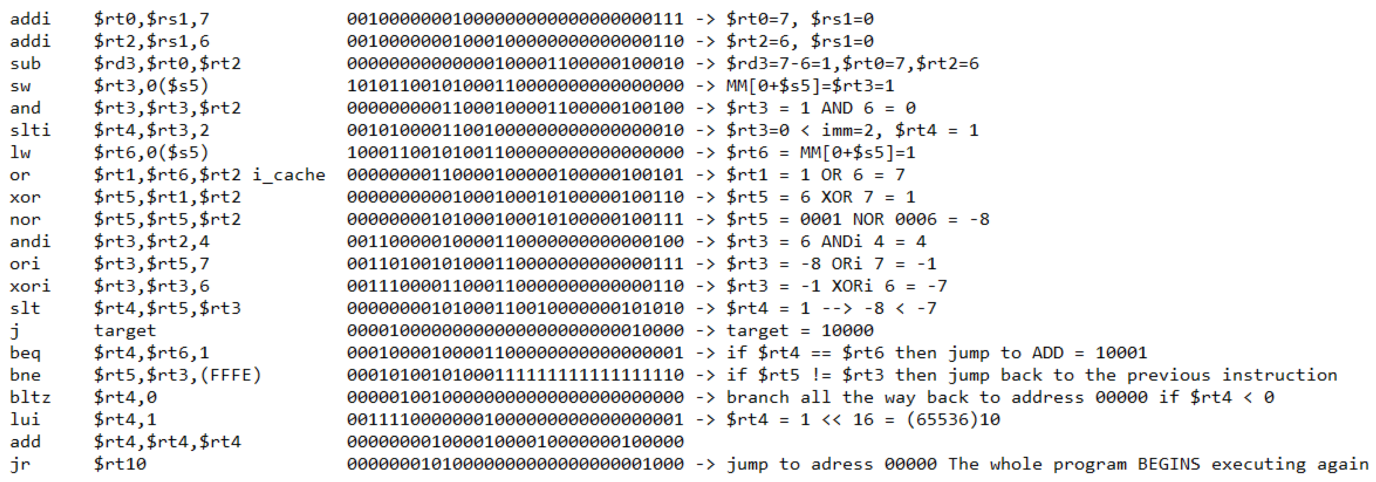
.

We also use Modelsim during this lab, it is a software that can simulate the HDL circuits you design with VHDL. It is used to test and debug the circuits mostly. We must also use the precision tool to get a schematic and the Xilinx software to obtain files needed for testing onto the hardware.

The control unit in this implementation is a combinational logic circuit whose inputs are the opcode and function fields of the instruction and the outputs are the 10 control signals. It will be used within the CPU's datapath when port mapping each component and ensure it correctly executes the code by pinpointing each function within his code.

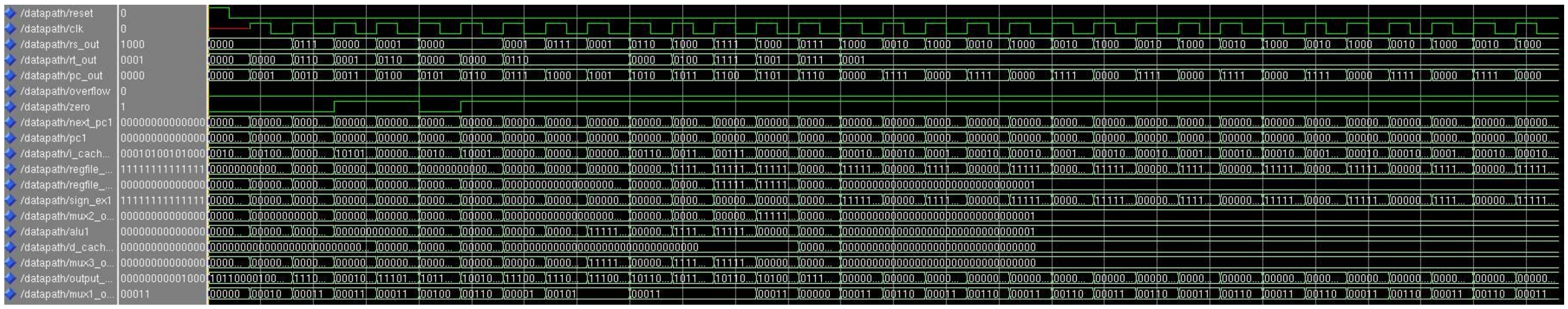
Observations

In designing the datapath of this CPU I had to make a lot of components that the other 3 labs did not cover. When looking at the diagram of the CPU within the given lab manual I observed 3 mux's needed to be created, 1 sign extend block, 1 data cache, 1 intruction cache and 1 pc register. Once all remaining parts are created within VHDL, we can then start writing the CPU datapath. Within another VHDL file, we add all components to the cpu within the architecture of the CPU entity, this will be the final CPU once we port map all signals to their respective positions and with the newly created control unit, everything will work harmoniously.
Here are the Instructions that I will be simulating within the modelsim results displayed shortly after:



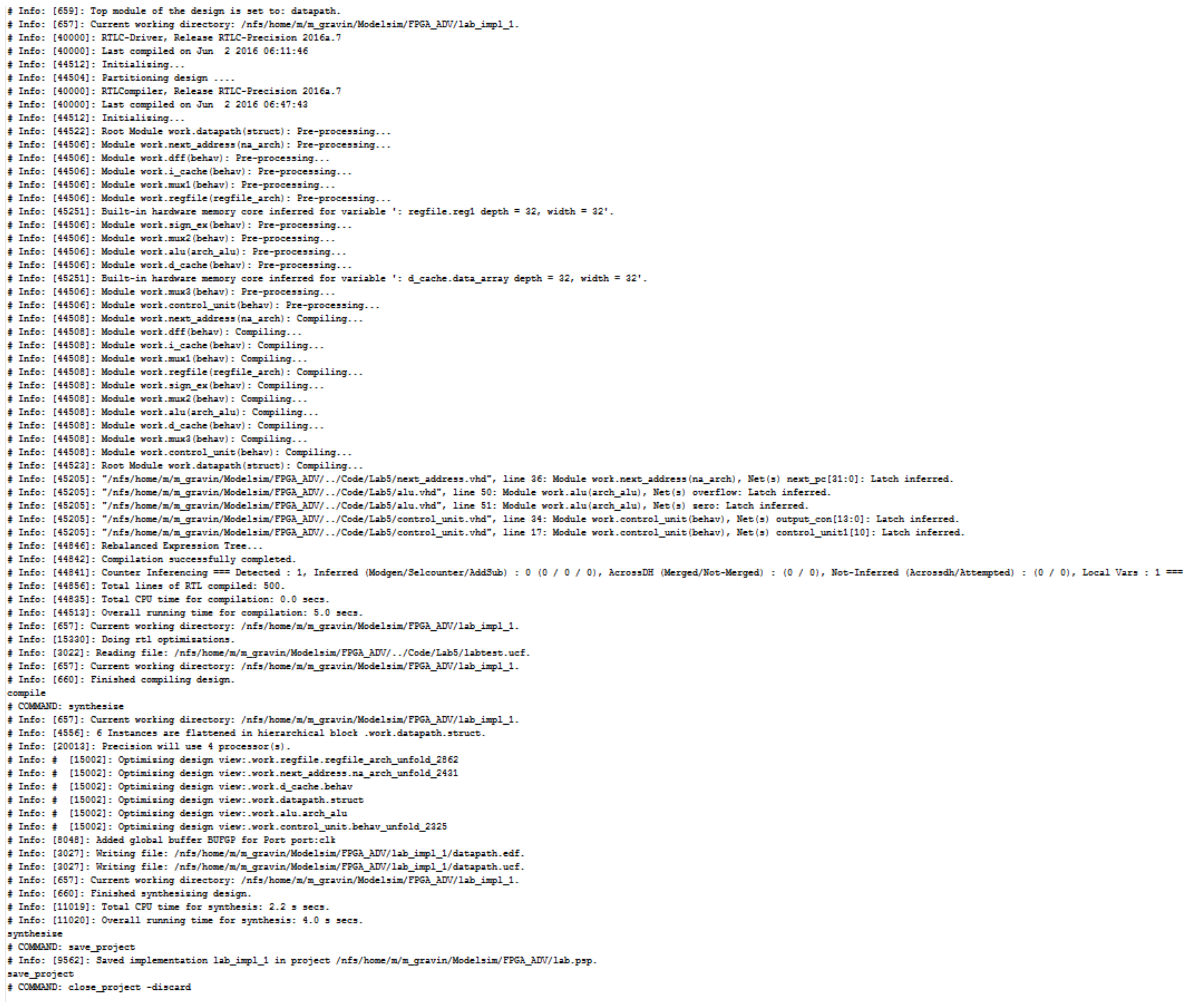
```
addi    $rt0,$rs1,7             00100000001000000000000000000111 -> $rt0=7, $rs1=0
addi    $rt2,$rs1,6             00100000001000100000000000000110 -> $rt2=6, $rs1=0
sub     $rd3,$rt0,$rt2          00000000000000010000110000100010 -> $rd3=7-6=1,$rt0=7,$rt2=6
sw      $rt3,0($s5)             10101100101000110000000000000000 -> MM[0+$s5]=$rt3=1
and     $rt3,$rt3,$rt2          00000000011000100001100000100100 -> $rt3 = 1 AND 6 = 0
slti    $rt4,$rt3,2             00101000011001000000000000000010 -> $rt3=0 < imm=2, $rt4 = 1
lw      $rt6,0($s5)             10001100101001100000000000000000 -> $rt6 = MM[0+$s5]=1
or      $rt1,$rt6,$rt2 i_cache  00000000110001000001000000100101 -> $rt1 = 1 OR 6 = 7
xor     $rt5,$rt1,$rt2          00000000001000100010100000100110 -> $rt5 = 6 XOR 7 = 1
nor     $rt5,$rt5,$rt2          00000000101000100010100000100111 -> $rt5 = 0001 NOR 0006 = -8
andi    $rt3,$rt2,4             00110000010000110000000000000100 -> $rt3 = 6 ANDi 4 = 4
ori     $rt3,$rt5,7             00110100101000110000000000000111 -> $rt3 = -8 ORi 7 = -1
xori    $rt3,$rt3,6             00111000011000110000000000000110 -> $rt3 = -1 XORi 6 = -7
slt     $rt4,$rt5,$rt3          00000000101000110010000000101010 -> $rt4 = 1 --> -8 < -7
j       target                  00001000000000000000000000010000 -> target = 10000
beq     $rt4,$rt6,1             00010000100001100000000000000001 -> if $rt4 == $rt6 then jump to ADD = 10001
bne     $rt5,$rt3,(FFFE)        00010100101000111111111111111110 -> if $rt5 != $rt3 then jump back to the previous instruction
bltz    $rt4,0                  00000100100000000000000000000000 -> branch all the way back to address 00000 if $rt4 < 0
lui     $rt4,1                  00111100000001000000000000000001 -> $rt4 = 1 << 16 = (65536)10
add     $rt4,$rt4,$rt4          00000000100001000010000000100000
jr      $rt10                   00000001010000000000000000001000 -> jump to adress 00000 The whole program BEGINS executing again
```

This is the ModelSim depicting every case with regards to the instructions in the previous figure:



```
/datapath/reset        0
/datapath/clk          0
/datapath/rs_out       1000
/datapath/rt_out       0001
/datapath/pc_out       0000
/datapath/overflow     0
/datapath/zero         1
/datapath/next_pc1     00000000000000
/datapath/pc1          00000000000000
/datapath/i_cach...    00010100101000
/datapath/regfile_...  11111111111111
/datapath/regfile_...  00000000000000
/datapath/sign_ex1     11111111111111
/datapath/mux2_o...    00000000000000
/datapath/alu1         00000000000000
/datapath/d_cach...    00000000000000
/datapath/mux3_o...    00000000000000
/datapath/output_...   00000000001000
/datapath/mux1_o...    00011
```

Here are the last 70 lines or so of the precision.log file:

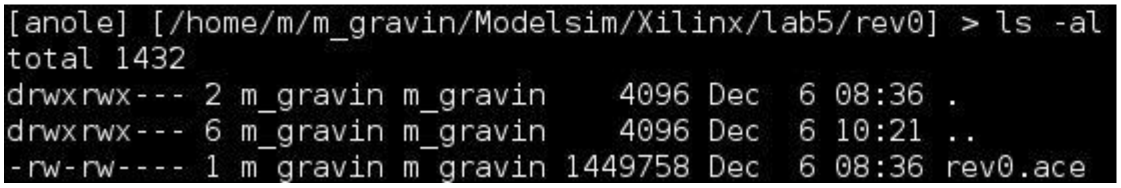


```
# Info: [659]: Top module of the design is set to: datapath.
# Info: [657]: Current working directory: /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/lab_impl_1.
# Info: [40000]: RTLC-Driver, Release RTLC-Precision 2016a.7
# Info: [40000]: Last compiled on Jun  2 2016 06:11:46
# Info: [44512]: Initializing...
# Info: [44504]: Partitioning design ....
# Info: [40000]: RTLCompiler, Release RTLC-Precision 2016a.7
# Info: [40000]: Last compiled on Jun  2 2016 06:47:43
# Info: [44512]: Initializing...
# Info: [44522]: Root Module work.datapath(struct): Pre-processing...
# Info: [44506]: Module work.next_address(na_arch): Pre-processing...
# Info: [44506]: Module work.dff(behav): Pre-processing...
# Info: [44506]: Module work.i_cache(behav): Pre-processing...
# Info: [44506]: Module work.mux1(behav): Pre-processing...
# Info: [44506]: Module work.regfile(regfile_arch): Pre-processing...
# Info: [45251]: Built-in hardware memory core inferred for variable ': regfile.reg1 depth = 32, width = 32'.
# Info: [44506]: Module work.sign_ex(behav): Pre-processing...
# Info: [44506]: Module work.mux2(behav): Pre-processing...
# Info: [44506]: Module work.alu(arch_alu): Pre-processing...
# Info: [44506]: Module work.d_cache(behav): Pre-processing...
# Info: [45251]: Built-in hardware memory core inferred for variable ': d_cache.data_array depth = 32, width = 32'.
# Info: [44506]: Module work.mux3(behav): Pre-processing...
# Info: [44506]: Module work.control_unit(behav): Pre-processing...
# Info: [44508]: Module work.next_address(na_arch): Compiling...
# Info: [44508]: Module work.dff(behav): Compiling...
# Info: [44508]: Module work.i_cache(behav): Compiling...
# Info: [44508]: Module work.mux1(behav): Compiling...
# Info: [44508]: Module work.regfile(regfile_arch): Compiling...
# Info: [44508]: Module work.sign_ex(behav): Compiling...
# Info: [44508]: Module work.mux2(behav): Compiling...
# Info: [44508]: Module work.alu(arch_alu): Compiling...
# Info: [44508]: Module work.d_cache(behav): Compiling...
# Info: [44508]: Module work.mux3(behav): Compiling...
# Info: [44508]: Module work.control_unit(behav): Compiling...
# Info: [44523]: Root Module work.datapath(struct): Compiling...
# Info: [45205]: "/nfs/home/m/m_gravin/Modelsim/FPGA_ADV/../Code/Lab5/next_address.vhd", line 36: Module work.next_address(na_arch), Net(s) next_pc[31:0]: Latch inferred.
# Info: [45205]: "/nfs/home/m/m_gravin/Modelsim/FPGA_ADV/../Code/Lab5/alu.vhd", line 50: Module work.alu(arch_alu), Net(s) overflow: Latch inferred.
# Info: [45205]: "/nfs/home/m/m_gravin/Modelsim/FPGA_ADV/../Code/Lab5/alu.vhd", line 51: Module work.alu(arch_alu), Net(s) zero: Latch inferred.
# Info: [45205]: "/nfs/home/m/m_gravin/Modelsim/FPGA_ADV/../Code/Lab5/control_unit.vhd", line 34: Module work.control_unit(behav), Net(s) output_con[13:0]: Latch inferred.
# Info: [45205]: "/nfs/home/m/m_gravin/Modelsim/FPGA_ADV/../Code/Lab5/control_unit.vhd", line 17: Module work.control_unit(behav), Net(s) control_unit1[10]: Latch inferred.
# Info: [44846]: Rebalanced Expression Tree...
# Info: [44842]: Compilation successfully completed.
# Info: [44841]: Counter Inferencing === Detected : 1, Inferred (Modgen/Selcounter/AddSub) : 0 (0 / 0 / 0), AcrossDH (Merged/Not-Merged) : (0 / 0), Not-Inferred (Acrossdh/Attempted) : (0 / 0), Local Vars : 1 ===
# Info: [44856]: Total lines of RTL compiled: 500.
# Info: [44835]: Total CPU time for compilation: 0.0 secs.
# Info: [44513]: Overall running time for compilation: 5.0 secs.
# Info: [657]: Current working directory: /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/lab_impl_1.
# Info: [15330]: Doing rtl optimizations.
# Info: [3022]: Reading file: /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/../Code/Lab5/labtest.ucf.
# Info: [657]: Current working directory: /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/lab_impl_1.
# Info: [660]: Finished compiling design.
compile
# COMMAND: synthesize
# Info: [657]: Current working directory: /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/lab_impl_1.
# Info: [4556]: 6 Instances are flattened in hierarchical block .work.datapath.struct.
# Info: [20013]: Precision will use 4 processor(s).
# Info: #  [15002]: Optimizing design view:.work.regfile.regfile_arch_unfold_2862
# Info: #  [15002]: Optimizing design view:.work.next_address.na_arch_unfold_2431
# Info: #  [15002]: Optimizing design view:.work.d_cache.behav
# Info: #  [15002]: Optimizing design view:.work.datapath.struct
# Info: #  [15002]: Optimizing design view:.work.alu.arch_alu
# Info: #  [15002]: Optimizing design view:.work.control_unit.behav_unfold_2325
# Info: [8048]: Added global buffer BUFGP for Port port:clk
# Info: [3027]: Writing file: /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/lab_impl_1/datapath.edf.
# Info: [3027]: Writing file: /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/lab_impl_1/datapath.ucf.
# Info: [657]: Current working directory: /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/lab_impl_1.
# Info: [660]: Finished synthesizing design.
# Info: [11019]: Total CPU time for synthesis: 2.2 s secs.
# Info: [11020]: Overall running time for synthesis: 4.0 s secs.
synthesize
# COMMAND: save_project
# Info: [9562]: Saved implementation lab_impl_1 in project /nfs/home/m/m_gravin/Modelsim/FPGA_ADV/lab.psp.
save_project
# COMMAND: close_project -discard
```

Here is the impact.log (last 30 lines or so):

```
------------------ GUI: Wizard Data Report  -------------------------
      File Mode : ACECF
      Collection Name : labtest
      Collection Locatnion : /nfs/home/m/m_gravin/Modelsim/Xilinx/
      Collection Size : 0 Mbits
      Reserved Space : 0 Mbits
      Number of Design : 1
      Config Address and Design Name list :
Version: 0   DesignName: rev0
------------------------ END of Report ------------------------------
V2Pro Part: xc2vp30 w/2 ppc
// *** BATCH CMD : setMode -acecf
1. Initializing V2Pro File...
// *** BATCH CMD : addDevice -p 1 -file"/nfs/home/m/m_gravin/Modelsim/Xilinx/labtest/datapath.bit"
'1': Loading file '/nfs/home/m/m_gravin/Modelsim/Xilinx/labtest/datapath.bit'...
done.
INFO:iMPACT:1777 -
   Reading   /nfs/sw_cmc/x86_64.EL7/tools/xilinx_10.1/ISE/virtex2p/data/xc2vp30.bsd...
INFO:iMPACT:501 - '1': Added Device xc2vp30 successfully.
----------------------------------------------------------------------
----------------------------------------------------------------------
Add one device.// *** BATCH CMD : setAttribute -configdevice -attr path -value"/nfs/home/m/m_gravin/Modelsim/Xilinx"
// *** BATCH CMD : generate -active labtest
INFO:iMPACT:794 - Creating SVF File   /nfs/home/m/m_gravin/Modelsim/Xilinx/labtest/rev0/rev0.svf...


'1': Programming device...
PROGRESS_START - Starting Operation.
 Match_cycle = NoWait.
Match cycle: NoWait
done.
INFO:iMPACT:579 - '1': Completed downloading bit file to device.
 Match_cycle = NoWait.
Match cycle: NoWait
INFO:iMPACT - '1': Checking done pin....done.
'1': Programmed successfully.
PROGRESS_END - End Operation.
Elapsed time =      1 sec.
Creating Ace File /nfs/home/m/m_gravin/Modelsim/Xilinx/labtest/rev0/rev0.ace...
Copy active xilinx.sys /nfs/home/m/m_gravin/Modelsim/Xilinx/labtest/xilinx.systo root.
// *** BATCH CMD : saveProjectFile -file"/nfs/home/m/m_gravin/Modelsim/Xilinx/default.ipf"
```

And finally the listing of the working directory with the rev0.ace file:



```
[anole] [/home/m/m_gravin/Modelsim/Xilinx/lab5/rev0] > ls -al
total 1432
drwxrwx--- 2 m_gravin m_gravin    4096 Dec  6 08:36 .
drwxrwx--- 6 m_gravin m_gravin    4096 Dec  6 10:21 ..
-rw-rw---- 1 m_gravin m_gravin 1449758 Dec  6 08:36 rev0.ace
```

Conclusion

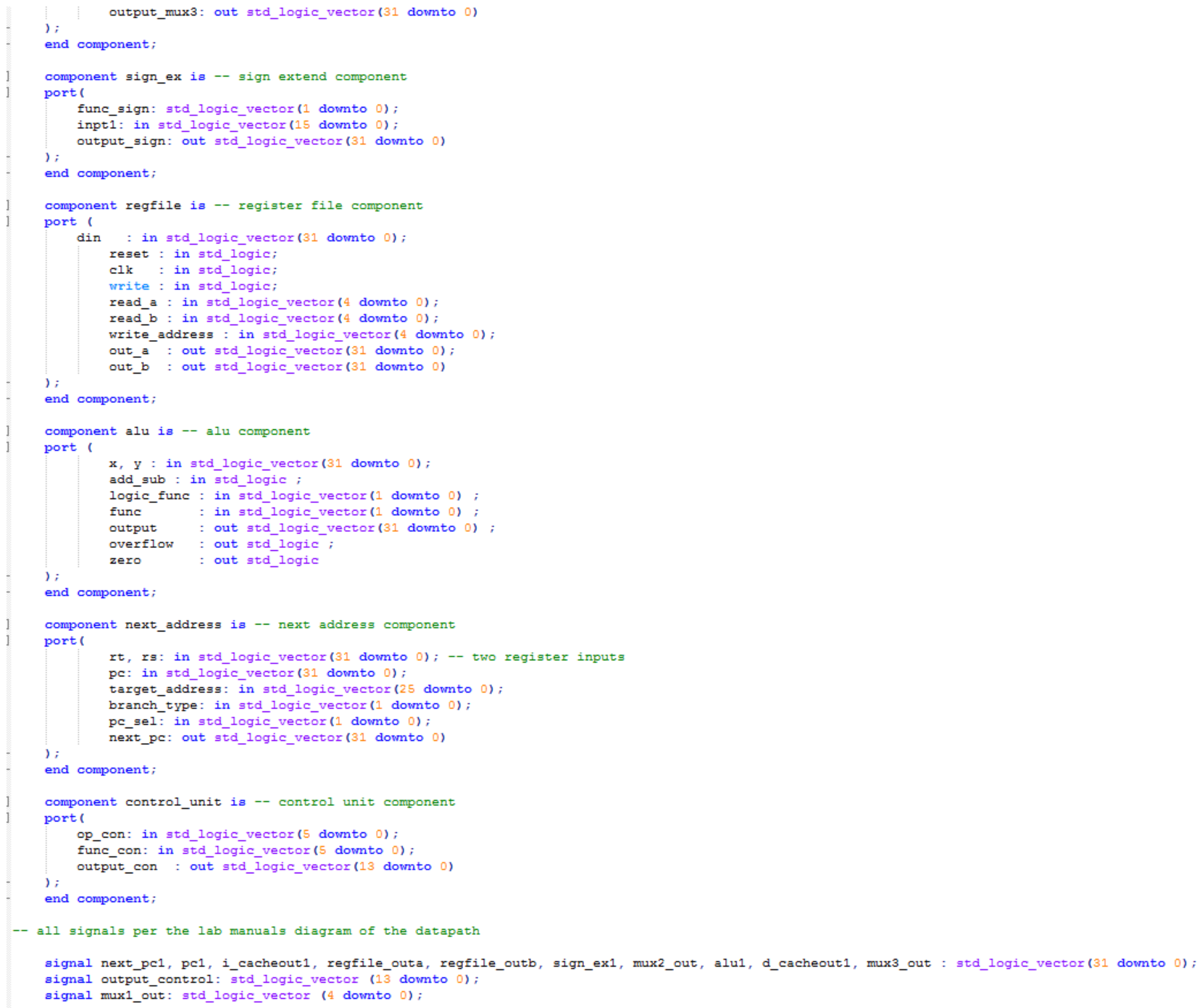
All in all this lab accomplished what it needed, I achieve the objectives that it set us out to complete. Because of this experiment I am more familiar using VHDL to design a CPU datapath as well as been reacquainted with the simulation software ModelSim and other related software to properly use it in the future. The hardest part of this lab was probably understanding what the lab manual wanted the CPU datapath to be and how exactly to implement it in an optimized and clear way without doing much debugging. This lab has helped me immensely with the material of this course and has bettered my understanding of CPUs and microprocessors and will definitely be of great help in my future endeavors.

Source Code

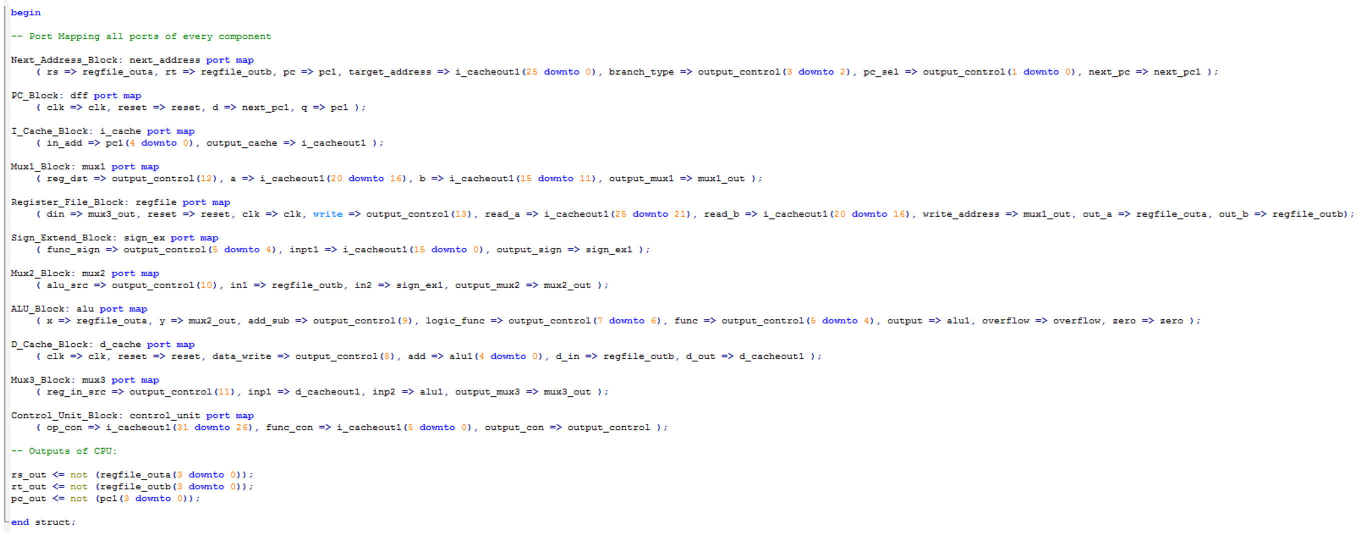
Control_Unit.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_signed.all;

entity control_unit is
port(
    op_con: in std_logic_vector(5 downto 0);
    func_con: in std_logic_vector(5 downto 0);
    output_con  : out std_logic_vector(13 downto 0)
);

end control_unit;

architecture behav of control_unit is
signal control_unit1 : std_logic_vector(13 downto 0);
begin
   control_unit1 <=
      "11100000100000" when func_con = "100000"
    else
      "11101000100000" when func_con = "100010"
    else
      "10100000010000" when func_con = "101010"
    else
          "11101000110000" when func_con = "100100"
    else
      "11100001110000" when func_con = "100101"
    else
      "11100010110000" when func_con = "100110"
    else
      "11100011110000" when func_con = "100111"
    else
      "01111111000010" when func_con = "001000";

   output_con <=
         "10110000000000" when op_con = "001111"
    else
         "10110000100000" when op_con = "001000"
    else
         "10110000100000" when op_con = "001010"
    else
             "10110000110000" when op_con = "001100"
    else
         "10110001110000" when op_con = "001101"
    else
         "10110010110000" when op_con = "001110"
    else
         "10010010100000" when op_con = "100011"
    else
         "00010111100000" when op_con = "101011"
    else
         "01111111000001" when op_con = "000010"
    else
         "01111111001100" when op_con = "000001"
    else
         "00000000000100" when op_con = "000100"
    else
         "00000000001000" when op_con = "000101"
    else
         control_unit1    when op_con = "000000";
end behav;
```

Datapath.vhd
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity  datapath  is
port(
    reset : in std_logic;
    clk   : in std_logic;
        rs_out, rt_out : out std_logic_vector(3 downto 0); -- output ports from reg.file
        pc_out : out std_logic_vector(3 downto 0);
        overflow, zero : out std_logic
); -- will not be constrained in Xilinx since not enough LEDs
end datapath;

architecture struct of datapath is -- adding all components to the datapath

    component dff is -- component for the pc flip flop
    port (
        clk, reset: in std_logic;
        d : in std_logic_vector(31 downto 0);
        q: out std_logic_vector(31 downto 0)
    );
    end component;

    component i_cache is -- i-cache component
    port(
        in_add : in std_logic_vector(4 downto 0);
        output_cache: out std_logic_vector(31 downto 0)
    );
    end component;

    component d_cache is -- d-cache component
    port(
        clk, reset: std_logic;
        data_write: std_logic;
        add: std_logic_vector(4 downto 0);
        d_in: in std_logic_vector(31 downto 0);
        d_out: out std_logic_vector(31 downto 0)
    );
    end component;

    component mux1 is -- reg_dst mux component
    port (
        reg_dst: in  std_logic;
            a: in  std_logic_vector(4 downto 0);
            b: in  std_logic_vector(4 downto 0);
            output_mux1: out std_logic_vector(4 downto 0)
    );
    end component;

    component mux2 is -- alu_src mux component
    port (
        alu_src: in  std_logic;
            in1: in  std_logic_vector(31 downto 0);
            in2: in  std_logic_vector(31 downto 0);
            output_mux2: out std_logic_vector(31 downto 0)
    );
    end component;

    component mux3 is -- reg_in_src mux component
    port (
        reg_in_src: in  std_logic;
            inp1: in  std_logic_vector(31 downto 0);
            inp2: in  std_logic_vector(31 downto 0);
```



```vhdl
            output_mux3: out std_logic_vector(31 downto 0)
    );
    end component;

    component sign_ex is -- sign extend component
    port(
        func_sign: std_logic_vector(1 downto 0);
        inpt1: in std_logic_vector(15 downto 0);
        output_sign: out std_logic_vector(31 downto 0)
    );
    end component;

    component regfile is -- register file component
    port (
        din   : in std_logic_vector(31 downto 0);
            reset : in std_logic;
            clk   : in std_logic;
            write : in std_logic;
            read_a : in std_logic_vector(4 downto 0);
            read_b : in std_logic_vector(4 downto 0);
            write_address : in std_logic_vector(4 downto 0);
            out_a  : out std_logic_vector(31 downto 0);
            out_b  : out std_logic_vector(31 downto 0)
    );
    end component;

    component alu is -- alu component
    port (
            x, y : in std_logic_vector(31 downto 0);
            add_sub : in std_logic ;
            logic_func : in std_logic_vector(1 downto 0) ;
            func       : in std_logic_vector(1 downto 0) ;
            output     : out std_logic_vector(31 downto 0) ;
            overflow   : out std_logic ;
            zero       : out std_logic
    );
    end component;

    component next_address is -- next address component
    port(
            rt, rs: in std_logic_vector(31 downto 0); -- two register inputs
            pc: in std_logic_vector(31 downto 0);
            target_address: in std_logic_vector(25 downto 0);
            branch_type: in std_logic_vector(1 downto 0);
            pc_sel: in std_logic_vector(1 downto 0);
            next_pc: out std_logic_vector(31 downto 0)
    );
    end component;

    component control_unit is -- control unit component
    port(
        op_con: in std_logic_vector(5 downto 0);
        func_con: in std_logic_vector(5 downto 0);
        output_con  : out std_logic_vector(13 downto 0)
    );
    end component;

-- all signals per the lab manuals diagram of the datapath

    signal next_pc1, pc1, i_cacheout1, regfile_outa, regfile_outb, sign_ex1, mux2_out, alu1, d_cacheout1, mux3_out : std_logic_vector(31 downto 0);
    signal output_control: std_logic_vector (13 downto 0);
    signal mux1_out: std_logic_vector (4 downto 0);
```



```vhdl
begin

-- Port Mapping all ports of every component

Next_Address_Block: next_address port map
    ( rs => regfile_outa, rt => regfile_outb, pc => pc1, target_address => i_cacheout1(25 downto 0), branch_type => output_control(3 downto 2), pc_sel => output_control(1 downto 0), next_pc => next_pc1 );

PC_Block: dff port map
    ( clk => clk, reset => reset, d => next_pc1, q => pc1 );

I_Cache_Block: i_cache port map
    ( in_add => pc1(4 downto 0), output_cache => i_cacheout1 );

Mux1_Block: mux1 port map
    ( reg_dst => output_control(12), a => i_cacheout1(20 downto 16), b => i_cacheout1(15 downto 11), output_mux1 => mux1_out );

Register_File_Block: regfile port map
    ( din => mux3_out, reset => reset, clk => clk, write => output_control(13), read_a => i_cacheout1(25 downto 21), read_b => i_cacheout1(20 downto 16), write_address => mux1_out, out_a => regfile_outa, out_b => regfile_outb);

Sign_Extend_Block: sign_ex port map
    ( func_sign => output_control(5 downto 4), inpt1 => i_cacheout1(15 downto 0), output_sign => sign_ex1 );

Mux2_Block: mux2 port map
    ( alu_src => output_control(10), in1 => regfile_outb, in2 => sign_ex1, output_mux2 => mux2_out );

ALU_Block: alu port map
    ( x => regfile_outa, y => mux2_out, add_sub => output_control(9), logic_func => output_control(7 downto 6), func => output_control(5 downto 4), output => alu1, overflow => overflow, zero => zero );

D_Cache_Block: d_cache port map
    ( clk => clk, reset => reset, data_write => output_control(8), add => alu1(4 downto 0), d_in => regfile_outb, d_out => d_cacheout1 );

Mux3_Block: mux3 port map
    ( reg_in_src => output_control(11), inp1 => d_cacheout1, inp2 => alu1, output_mux3 => mux3_out );

Control_Unit_Block: control_unit port map
    ( op_con => i_cacheout1(31 downto 26), func_con => i_cacheout1(5 downto 0), output_con => output_control );

-- Outputs of CPU:

rs_out <= not (regfile_outa(3 downto 0));
rt_out <= not (regfile_outb(3 downto 0));
pc_out <= not (pc1(3 downto 0));

end struct;
```