RECONFIGURABLE SYSTEMS PRINCIPLE

EECE8020

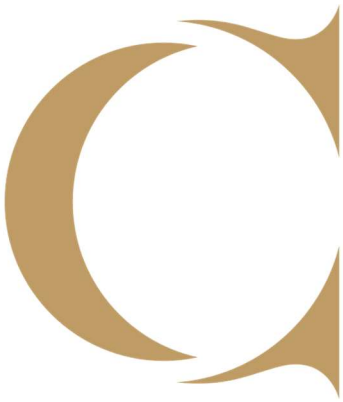
MICROCODE TECHNICAL DESIGN

EMBEDDED SYSTEM DEVELOPMENT

SANTHOSH NAGENDRAN

8272767

19-11-2019



CONESTOGA

Connect Life and Learning

INTRODUCTION

This document contains information about writing microcode for the processor using ModelSim software by Verilog programming. And this document tells more details about creation of the control unit from scratch with some inputs so that the reader will have a good understanding in creating the control unit, process involved to reach the outcomes and to test the control unit by writing microcode and with some other features it helps us to find the errors and will have idea to debug and make some improvements. After reading this document users could able to reproduce control unit design and will also have idea to upgrade the existing model.

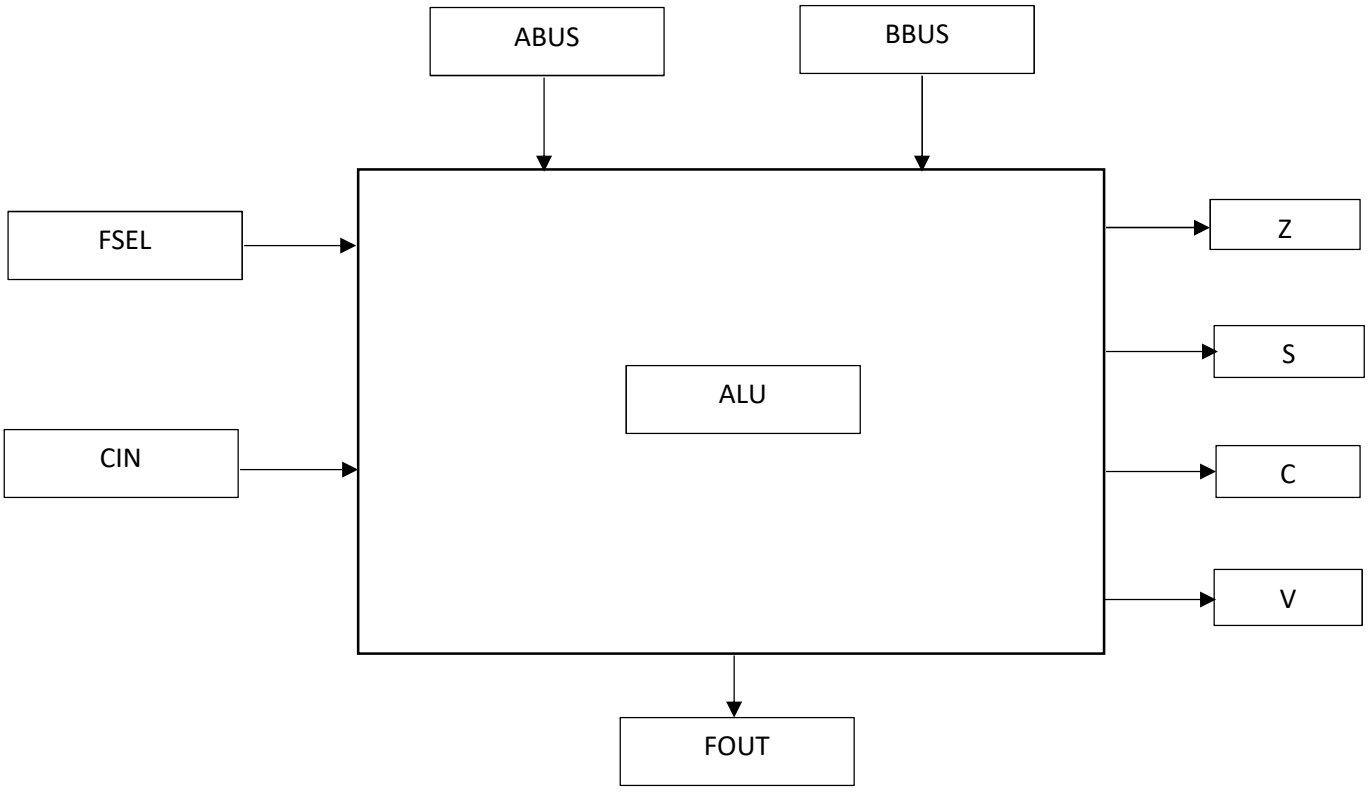
DESIGN OBJECTIVE

As the name itself says, the main objective of this design is to create a control unit and testing it by writing some microcode, which is also another important part of the processor or a system. This is the place where decision making and branching operations of the entire processor will take place. To talk something on big picture, when we are designing a processor or a system which will be served to the users in the real time, we will be facing situations to branch or decide what to do at the run time based on the data which is coming in. It could be either the data fed by the users to the processor or by any means of sensors and transducers. In those case to make the data as a sensible information we have to process the data based on the decisions made during runtime, for doing so we need to have a unit called control design. This part will be a best fit at those points.

OVERVIEW

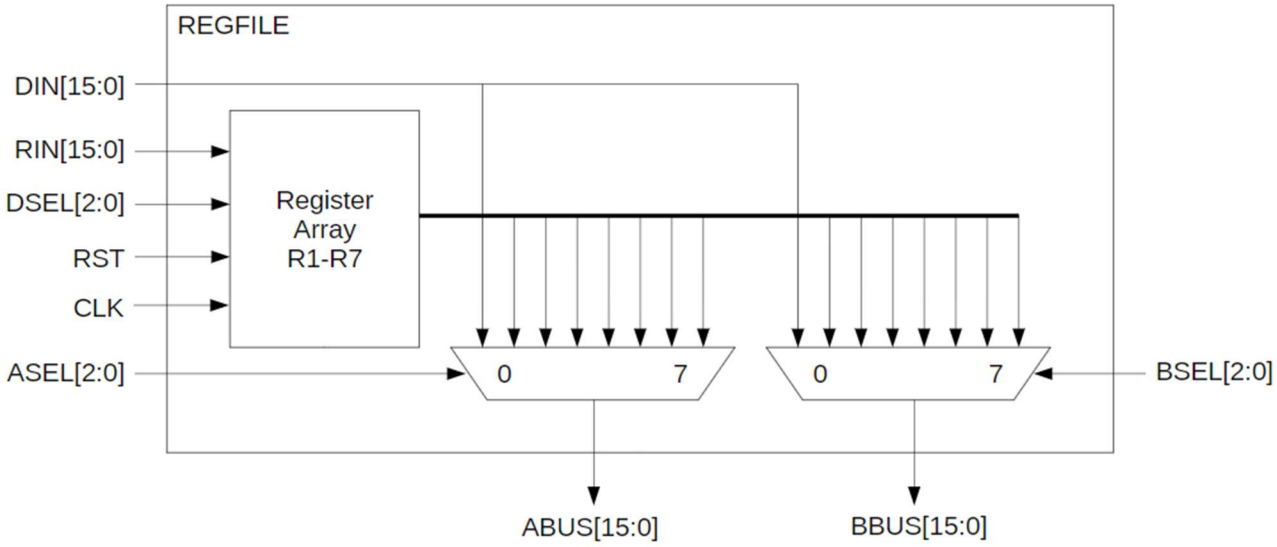
BLOCK DIAGRAM FOR ALU

Ref.1.1 Mike's Slides



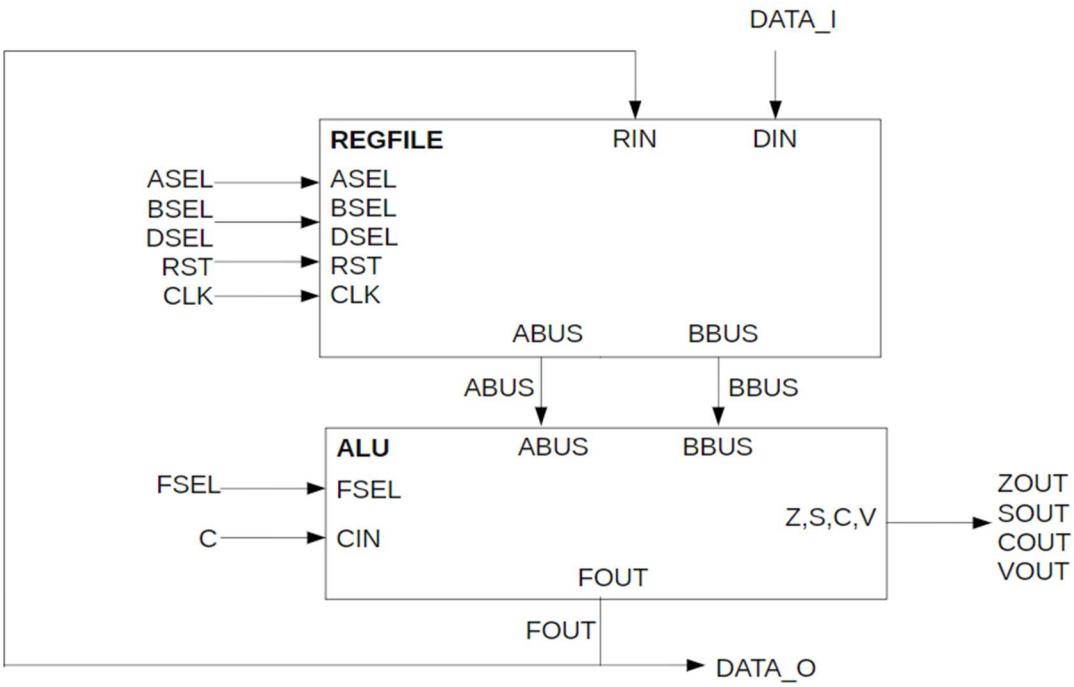
BLOCK DIAGRAM FOR REGISTER FILE

Ref.1.2 Mike's Slides



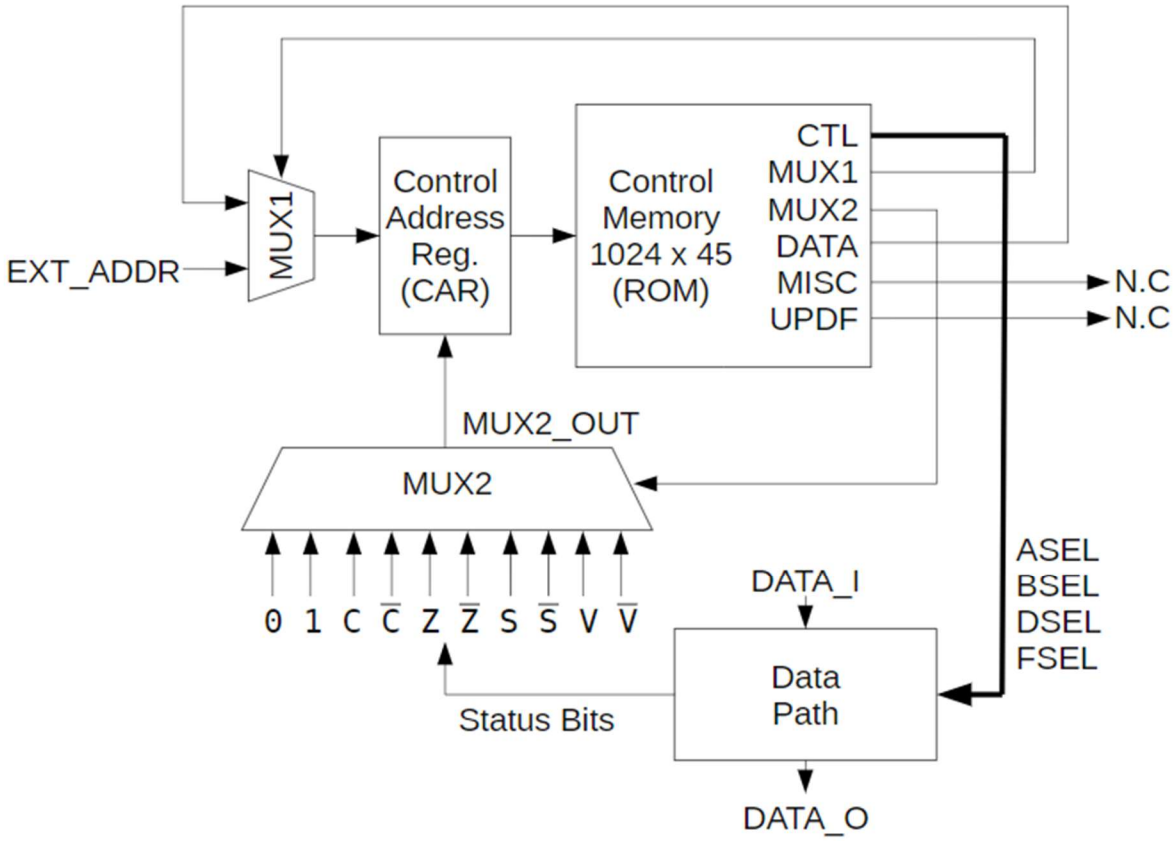
COMBINING BOTH ALU AND REGISTER FILE (CPU)

Ref.1.3 Mike's Slides



BLOCK DIAGRAM OF CONTROL UNIT DESIGN

Ref.1.4 Mike's Slides



REQUIREMENTS AND SPECIFICATIONS

As a simple design and most common in practice and as every other system constrains, we need some inputs and outputs. Since we need to get the inputs from the users, and then process and to store the values from the variable and to select the task to perform inside the processor we need some more variables in addition to that, so the variables we need to create this design are as follows,

The above mentioned are the general inputs and outputs of the design but to describe more about each variable, there are different specifications for each of them. The design developed in this part is a 16bit ALU which means the inputs and the outputs of the processor will be 16 bits wide. Only 16-bit data can be processed in this design. So, the main inputs and the outputs of the system are 16 bits wide. And the type of the data to be processed should also be looked carefully. Apart from this the status of the processor after certain process is done should be taken care,

the selection of the task to be performed which will be 16 different types in our case. The types of the variables and the size used will be shown clearly below as follows,

ABUS [15:0] – A input bus (unsigned)
BBUS [15:0] – B input bus (unsigned)
FOUT [15:0] – Function Output (unsigned)
FSEL [3:0] – Function select Input (std_logic_vector)
Z, S, C, V – Status flags output (std_logic)
CIN – Carry In (std_logic)
DSEL [2:0] – Destination Select Input (unsigned)
ASEL [2:0] – bus Selection Input (unsigned)
BSEL [2:0] – bus Selection Input (unsigned)
DIN [15:0] – Input Data Input (unsigned)
RIN [15:0] – Destination Register data In Input (unsigned)
CLK – Clock signal Input (std_logic)
RST – Reset Signal Input (std_logic)
DATA_I [15:0] – Data Input (unsigned)
DATA_O [15:0]– Data Output (unsigned)

While designing the processor these are all the important task which are to be taken care of. And another important point to be remembered is the name of the entity or the module to be created, it should be named according to the module or entity with which the users are working on. This will not create any confusions in the future developments. The name of the design should be straight forward and easy to identify that way it will not create any problems in the future improvements and it will be easy to understand.

IMPLEMENTATION METHOD

This part here is the most important part of all the above discussed parts. After considering all the above discussed topics, the implementation process should be done. The tool or the software that is done to implement this was “ModelSim” software with which either a VHDL or Verilog code can be written to make this work. The special thing about the software is, it contains the simulator with it which becomes handy to the testers and the developers to do something form the scratch. We can write the code and the compiler in it says whether the code written is right or wrong with some error messages. Once the code compiles good it can be taken further to make it work. When all the things work well it could be given to the manufacturers to make it as a real working chip and can be implemented in any of the processor or a real time system where these operations are needed. Since the ROM is created already the control unit should be connected to it and should try to work on the fields which can operate it and perform the lines written in the micro code.

VERIFICATION AND VALIDATION

After creating the actual ALU itself, we will be creating a test bench to test the work. Verifying the work will take time and most important thing to do. Once when we develop the processor, when it is given to the manufacturer it will be converted into a chip where we cannot make any changes if there are any problem in the process. So, before hand we have to create a test bench to test every single logic and possibility. Make sure to add some print statements to show where, what and when the test is happening, which will be helpful for us to debug in case.
So, in the test bench file, we will be creating a task where it has some assignment statements to send the data from the testers to the real busses. This task should pass the parameters as a function call in C programming where the parameters sent should be FSEL, CIN, ABUS and BBUS along with the results (FOUT, Z, C, S, V).
Similarly, after creating all the other fields shown in the top part of the document, should try to connect it and make it work with rest of the fields as well. And the testing part should be written with the micro code so that the assembler should take the following by itself using the ROM file and the assembler developed already which was really helpful in the development of the process.

ALGORITHM

- Copy your ALU and REGFILE code into the project, along
- with their respective test benches.
- Inside your CPU instantiate your sub blocks:
    - ALU
    - REGFILE
- Connect them up to the broken-out signals from the
- microcode ROM from the template to the control inputs of
- your ALU and regfile.
- Create signals or wires as shown in the block diagram
- Using a combination of sequential and combinational
- blocks build the control logic for the CAR, and the muxes.
- Create signals/regs for:
- C
- Z
- S
- V
- MUX1_OUT
- MUX2_OUT
- EXT_ADDR
- The block sensitive to the rising edge of the CLK should:
    - Reset CAR to 0 when RST is asserted low.
    - Increment CAR when MUX2 output is '0'
- Load the value of the output of MUX1 when MUX2 output is '1'
- The combinational block(s) should implement:

- MUX1 – The output should be the DATA field from the microcode ROM when the MUX1 signal is '0', or the EXT_ADDR input when the MUX1 signal is '1'.
- MUX2 – The output should be a selection from the list of status bits and constant values based on the MUX2 output from the microcode ROM.

Ref.1.5 – Mike's Slides

REFERENCE SECTION AND RESOURCES

The resources and the reference are made in the Mike's Slides. The slides of Mike clearly explained the objective, block diagram and the description along with the variables used in the processor, size and type.

Ref.1.1 – Mike's Slides Block Diagram
Ref.1.2 – Mike's Slides Variable name, size and type
Ref.1.3 – Mike's Slides Define functions with the data
Ref.1.4 – Mike's Slides Status Flag explanations
Ref.1.5 – Mike's Slides Control unit design