(Faculty Of Engineering And Technology
Program Of Computer Systems Engineering)

ADVANCED DIGITAL SYSTEMS DESIGN ENCS533

Project

Student Name:

Name : Ahmad Dar Khalil #:1120443

Name : Mohammad Modallal #:1120174

Name : Ehab Amriah #:1120871

Dr. Abdallatif Abuissa

1.Abstract

The aim of this project is to understanding , configuring and testing simple microprocessor which operate various process either arthimatic or logical process using VHDL language and simulation tools.

2 .Introduction

The simple microprocessor consist of three parts(fig2.1): Arithmetic logic unit (ALU) to find logical and arthmatic process,file register that implement physical RAM and register that contain the machine instruction.

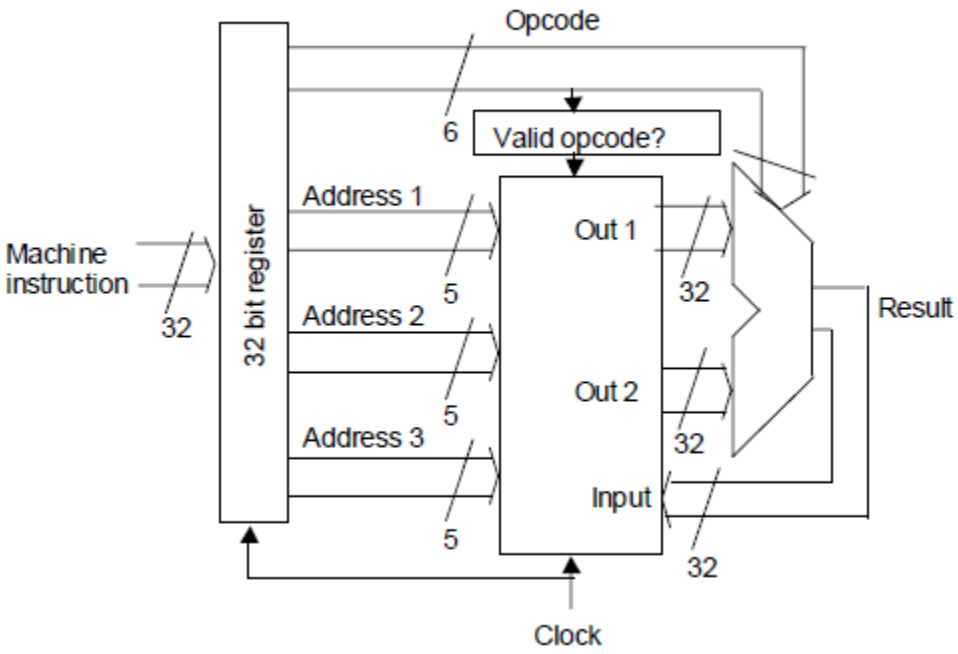


Figure 2.1 simple microprocessor

2.1 Arithmetic logic unit (ALU)

arithmetic logic unit (fig 2.2)is a digital circuit that performs arithmetic and bitwise logical operations on integer binary numbers. It is a fundamental building block of the central processing unit (CPU) found in many computers.

Our ALU operate many process depend on the opcode number as shown in table 2.1

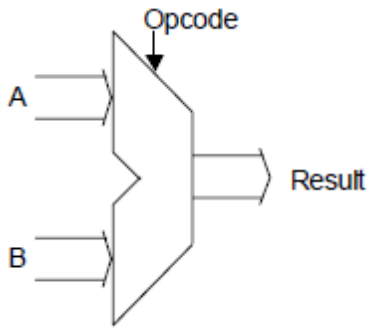


Figure 2.2 ALU

Digit of ID no.	8 (1+3+4)
a + b	1
a – b	6
|a|	13
-a	8
max(a,b)	7
min (a,b)	4
avg(a,b)	11
not a	15
a or b	3
a and b	5
a xor b	2

Table 2.1 ALU process

2.2 file register

This is small RAM contain 32 x 32-bit words, it have a 5-bit address to select out one of the 32-bit words. it can process three addresses at the same time, two of which are always read operations, and one of which is always written to(fig 2.3).

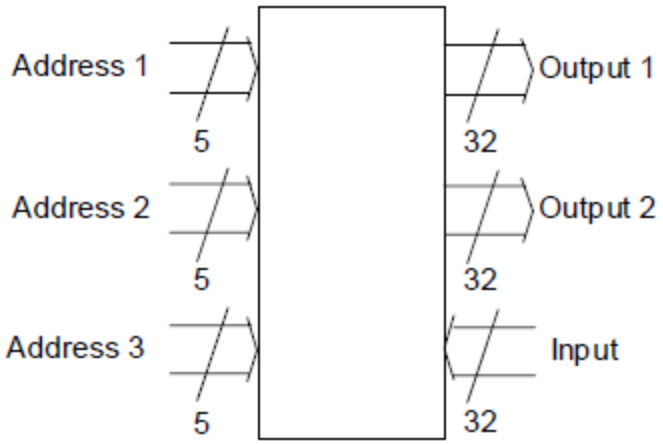


Figure 2.3 file register

2.3 register

Register consist of 32-bit.it use to hold the machine instruction.

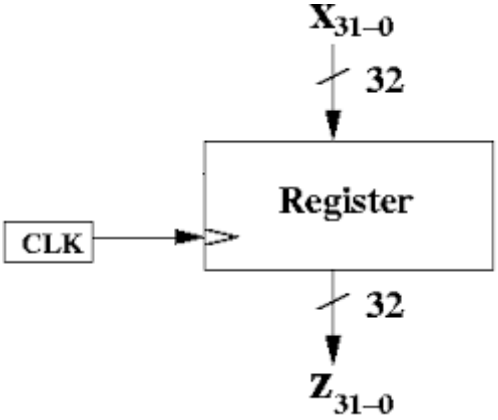


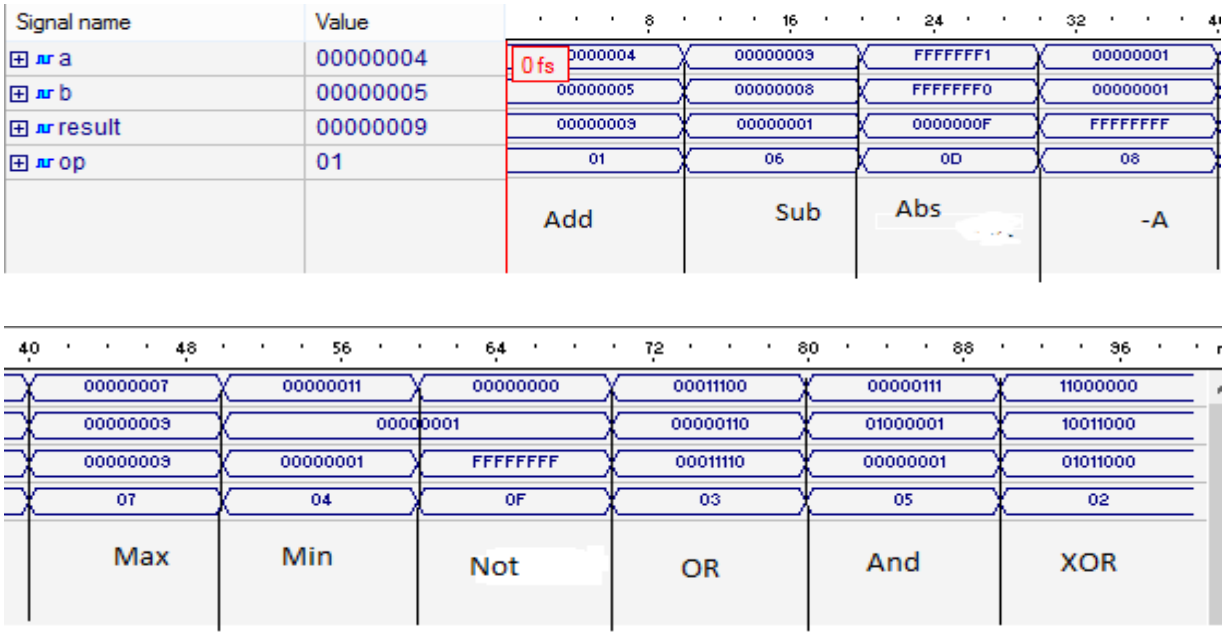
Figure 2.4 register

3.Procedure

3.1 build ALU：

The ALU is combinational circuit , as we see in figure 2.2 . it receive opcode that determine which operation will be operate (Table 2.1 show different type of operations that ALU can do).Moreover, it also have two operands and one output which is the result that computing with respect to these parameters .

To test that this entity is work correctly , we make a test bench that got all the values of opcode(table 2.1) and checked the result which was corrected . Figure 3.1 below show some of the test cases that help us to make sure that the circuit is work correctly .



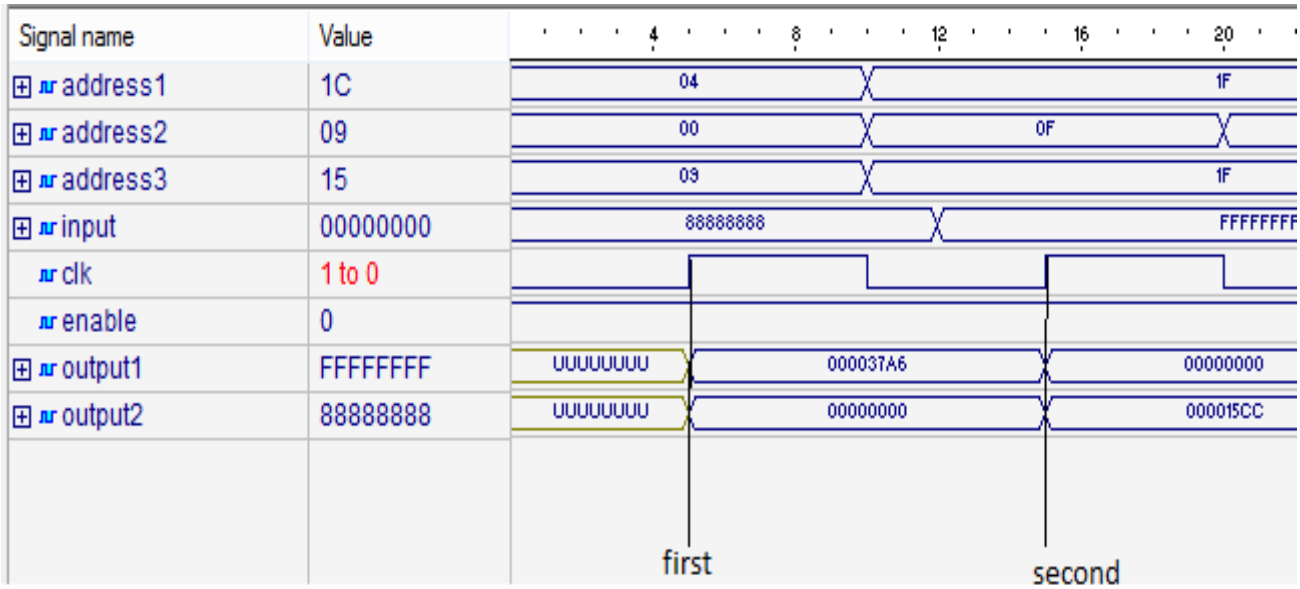
This is the output of the test bench that showing the result for different opcode to, it is easy to see that every thing works well .

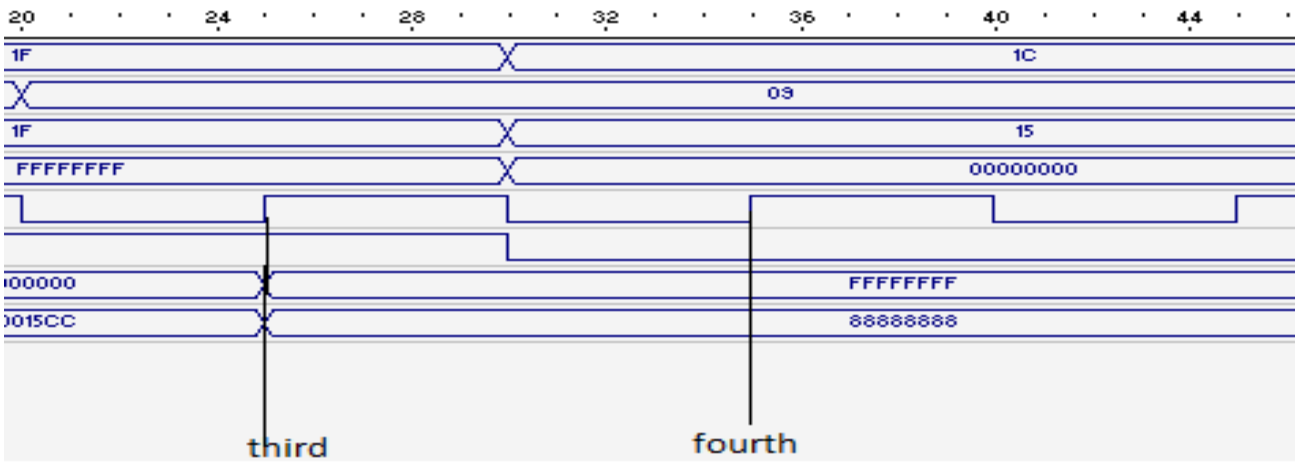
- Note that we separate the output into two figure to make the results clear .

3.2 build register file：

The register file is sequential circuit working on the positive edge of the clock that input on it . This entity is actually works as 32x32 read/write memory ,it has two addresses to read certain location on it and the values appear using two output. Moreover , it has also one address to write at any location , note that the read and write is acts only in rising edge of the clock . finally , this entity contains additional input 'Enable' when it is '0' the file register will ignore the inputs, and will not update its outputs.

The test of this entity needs very tricky choices , so that any problem will appear by using efficient values for test , we do that and the test bench is in the figure below ：





we choose this test vectors to test every thig about file register . And we will explain why:

- At first positive edge (shown in the graph) , we will read the address 0 and 4 and the output will show as output1 ,output2 respectively
  ➔(output1 => 0000000,output2 => 000037A6) . Other thing to do is to write the input (88888888)at address 9 .

- At second positive edge (shown in the graph) , we will read the address 1F and 0F and the output will show as output1 ,output2 respectively
  ➔(output1 => 0000000,output2 => 000015CC) . Other thing to do is to write the input (FFFFFFFF)at address 1F .

- At third positive edge (shown in the graph) , we will read the address 1F and 09 and the output will show as output1 ,output2 respectively
  ➔(output1 => FFFFFFFF,output2 => 88888888 .
  we see the data that we wrote before in first two cyclyes ,this means that our read write are work correctly (e.g : we write to address 9 then we read it to make sure that it contain the new data) .

- At forth positive edge,we change the enable value from 0 to 1.As a result, we note actually that the **inputs are ignored** (output1,2 remains the same) , and **we were not update the output** (you can see that we put new input "00000000" but the output stay the same) . This cases lead to say **Every thing is working correctly .**

We think that this set of test data is the best one to make general test for file register and we give you all the tricks from four points above .

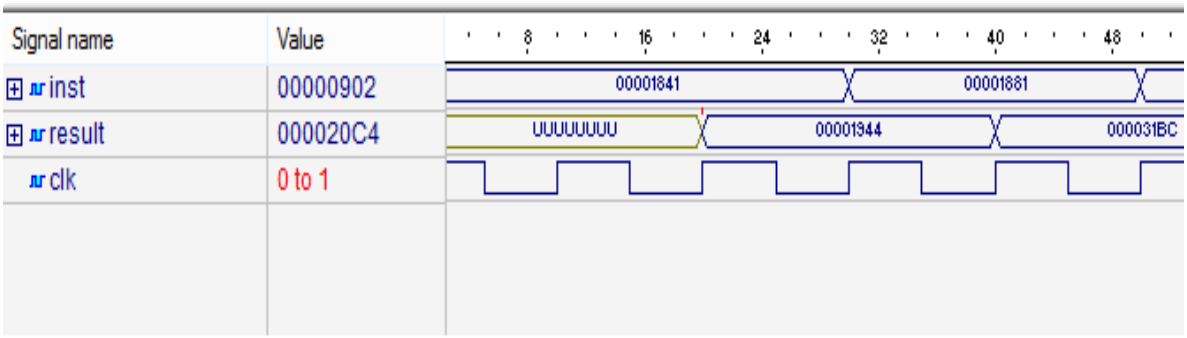
3.3 build simple microprocessor

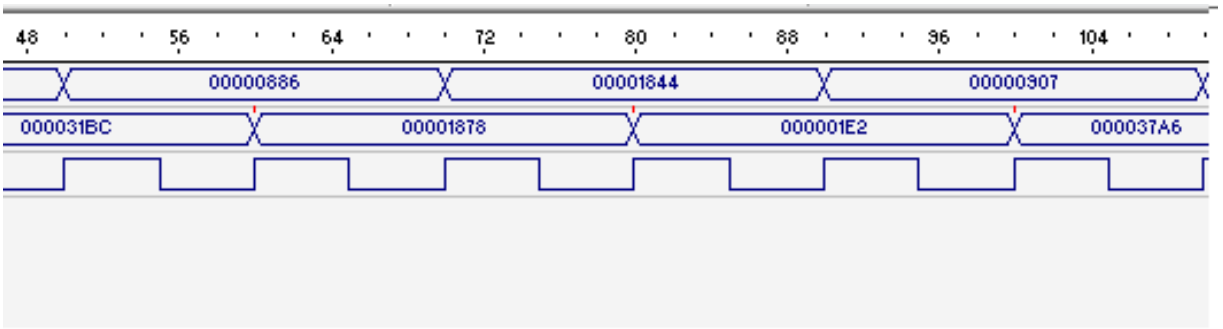
Now , after we trust that ALU and File register are work well . we can use them to build simple microprocessor system , as we know the microprocessor take machine instruction as input and it is work at the positive edge of the global clock of hole system .

From the given design , we notice that the instruction will not enter to file register until the positive edge , and we have to make sure that the opcode working well . after that, we have to take the result of any instruction after 2 clock cycles because the first one will pass the instruction to the file register (6 to 10 Address1 , 11 to 15 address2 , 16 to 20 address3) and the second one will take the output from the file register and make the operation according the opcode (instruction 0 to 5) .and store the results in the given location again (instruction 16 to 20) .

- Each instruction must stay 2 clock cycles to get the result .
- we make the opcode so that we will pass to ALU .at the same time, the data of the addresses of the same instruction reached to keep the results convenient (will appear In the output of the test bench).

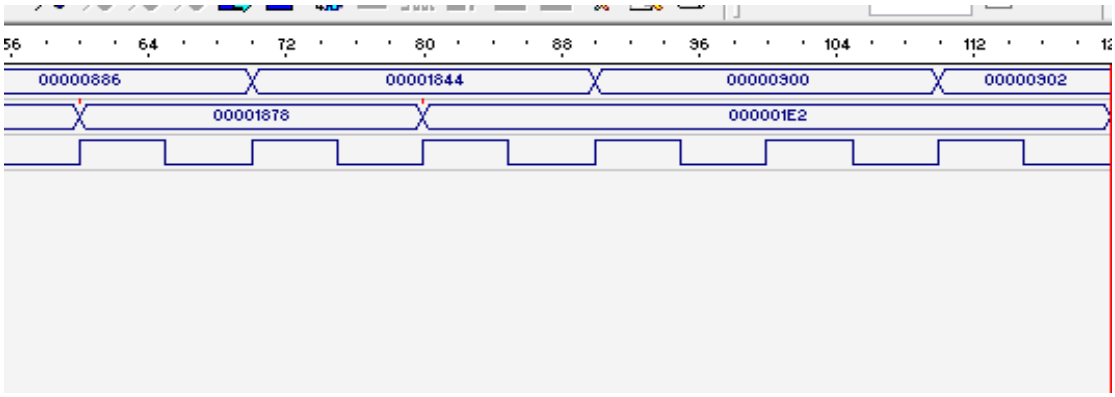
- We checked the value of opcode , if it is invalid opcode then we made the enable signal = 1 , so that the processor will do nothing .





- The output of the test cases in previous figures was just to show that each instruction needs two cycles to get the result . and I made some examples
  - ⇨ (address1 + address3) = 1944
  - ⇨ (address2 + address3) = 31BC
  - ⇨ (address2 - address1) = 1878
  - ⇨ (min(address1 , address3)) = 1E2
  - ⇨ (max(address1 , address4)) = 3A76

- Every things are work well .(note that if we enter invalid opcode(**900 in this case**) the enable signal will be 0), so we notice that the result remain the same (ignore the inputs and no update for output).



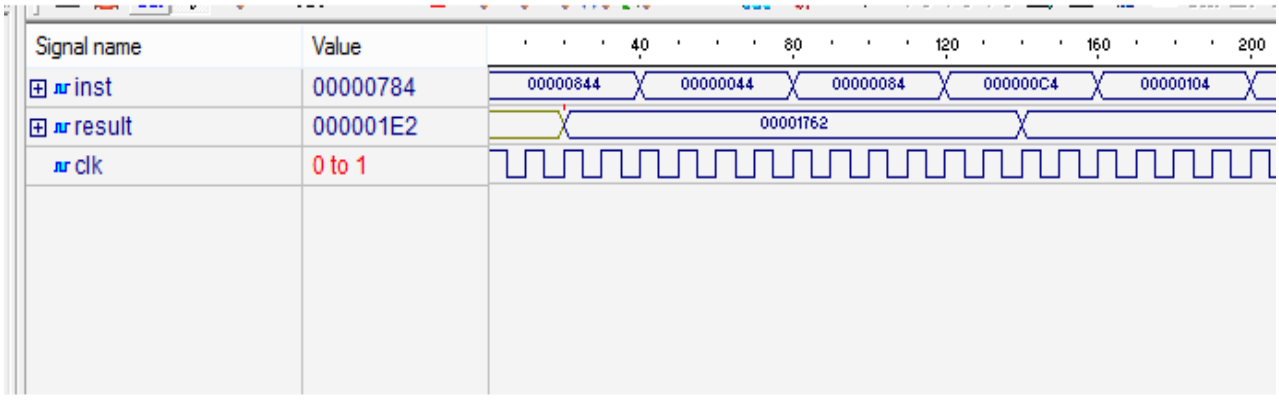
3.4 Find the minimum number :

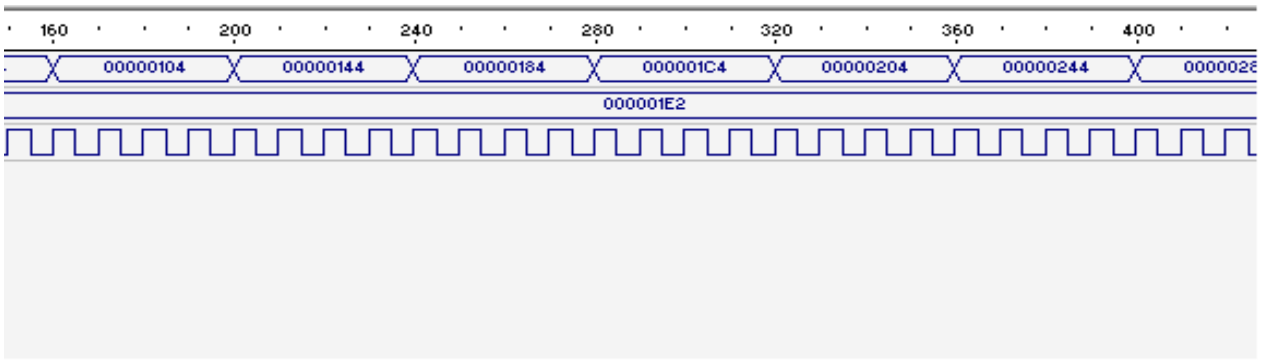
To find the minimum number of memory , if I use the processor that I made , just make a loop around all over the memory except index 0 and 31 , and compare using instruction ‘min’ then store the result in first location in memory .

I put additional output called result just to show you that every thing is ok and the to read the answers easily .

Moreover , I use process to generate all instructions that we need , otherwise is I has huge memory then I can not write all the instructions by hand! .and other benefit why I use this way is to make sure that if the memory elements are refreshed then we find the minimum of refreshed version of the memory which is more practical (given that it not straightforward like writing them one by one) .

Below the output of the minimum of our data :





- **We notice that the minimum number in the memory Is 1E2 .**

4.Conclusion :

in this project ,we learn how to design simple microprocessor using AHDL programming language and we test it using simulation tool.We learn how to build each component of microprocessor(ALU and file register).we learn how ALU work and we did many operation logical and we understand that the operations need usually arithmetic operation. more than one cycle to execute and we know actually the reason .

5.Reference :

1) **http://esd.cs.ucr.edu/labs/tutorial**

2) **http://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html**

3) **slides of the course .**

4) **http://www.freerangefactory.org/dl/free_range_vhdl.pdf**

5) **http://cegt201.bradley.edu/projects/proj2006/vhdlmcpr**