**CPU Design Project – Part 5**

-Rujun Bai, Chengzhu Piao and Yuanze Li

1. **What did you learn from this project?**

We have learned a lot from this project including how to design a basic CPU with pipeline structure. Each week we modified our original design since we got more deeper understanding about how each component work with others. We practiced a lot on how to use the Verilog language and got familiar how to use Modelsim software to check our design result. And at the last step, we learned how to implement our program into the FPGA board which is the most excited moment while still the moment challenging our patient because each tiny mistake will ruin our final result. Then we patiently check each control signal and datapath then follow the tutorial strictly. At last we got the result which we expected.

1. **What would you do differently next time?**

The component which have been design to realize beq and bne are not work very well which may caused by our Jump signal is a little bit complex. Next time we may add one more mux make each component focus on simple function then may get expected result. Additionally, we should consider more about the hazards like add some forwarding component make it work more efficiency. However, this time, we just use bubble to avoid the three kinds of hazards in pipeline data path also work well.

1. **What is your advice to someone who is going to work on a similar project?**

If you use the Verilog as your design language, you should pay attention to the generation file when you generate the memory component using Quartus II. You should also generate the .cmp file in your working folder. When you choose the clock pin you should choose the clock 50MHz instead of the 27MHz, I am not sure the season but for our case only clock 50MHz could work. So trying different pin location is needed.

Tab.1 Assembly code and binary code of the final test program

|  |
| --- |
| Test programs |
| a=7; % a is saved in $5b=1; %b is saved in $6c=0+1d=a & b; %d is saved in $12e=a | b; %e is saved in $13f=b-a; %f is saved in $14 |
| Assembly code | Binary code |
| sw $4, 0($4)lw $5, 0($4) add $6, $0, $1and $12,$5,$6 %12 change to 0or $13,$5,$6 %13 change to 5sub $14,$6,$5 %14 change to -3 | 1110 0100 0100 00001101 0100 0101 00000100 0000 0001 01100010 0101 0110 11000011 0101 0110 11010101 0110 0101 1110 |