Computer Systems and -architecture

Project 3

1 Ba INF 2011-2012

Ruben Van den Bossche
ruben.vandenbossche@ua.ac.be

Sam Verboven
sam.verboven@ua.ac.be

Don't hesitate to contact the teaching assistants of this course. You can reach them in room M.G.2.12 or by e-mail.

Time Schedule

Projects are solved in pairs of two students. Projects build on each other, to converge into a unified whole at the end of the semester. During the semester, you will be evaluated three times. At these evaluation moments, you will present your solution of the past projects by giving a demo and answering some questions. You will immediately receive feedback, which you can use to improve your solution for the following evaluations.

For every project, you submit a small report of the project you made by filling in `verslag.html` completely. A report typically consists of 500 words and a number of drawings/screenshots. Put all your files in a tgz archive, as explained on the course's website, and submit your report to the exercises on Blackboard.

- Report deadline: **November, 6 2011, 23u55**
- Evaluation and feedback: **November, 8 2011**

Project

Read section C.5 of Appendix C. Build a arithmetic logic unit (ALU) for 16-bit two's complement data words. To do this, create a circuit that implements a 1-bit ALU. Combine them to obtain a 16-bit ALU. Implement the operations below, giving each operation a 4-bit binary code. Your ALU will execute the right operation according to a 4-bit operation input. Next to this, your ALU should have two 16-bit words as input, one 16-bit word as output, and one "error" bit as output, denoting an error.

Use the Logisim `ALU_GroupXX.circ` file provided on the course page.

Your ALU should be able to perform the operations listed below. Make sure to test everything, including the different possible overflow cases!

1. **generate 0** (`0000`).
   Example:

result	0000000000000000

2. **AND** (`0001`).
   Example:

a	0010010010101010
b	1010100101010010
result	0010000000000010

3. **OR** (`0010`).
   Example:

a	0010010010101010
b	1010100101010010
result	1010110111111010

4. **NOT** (`0011`).
   Example:

a	0010010010101010
result	1101101101010101

5. **numeric inverse (two's complement)** (`0100`).
   Example:

a	0010010010101010
result	1101101101010110

   Mind overflow!

6. **numeric addition (two's complement)** (`0101`). Ripple carry addition suffices.
   Example:

a	0010010010101010
b	1010100101010010
result	1100110111111100

   Mind overflow!

7. **numeric subtraction (two's complement)** (`0110`).
   Example:

a	0010010010101010
b	1010100101010010
result	0111101101011000

   Mind overflow!

8. **shift left** (`0111`).
   Example:

a	0010010010101010
result	0100100101010100

9. **shift right** (`1000`).
   Example:

a	0010010010101010
result	0001001001010101

10. **signed shift left (two's complement)** (`1001`). This implements "times two".
    Example:

a	`0010010010101010`
result	`0100100101010100`

    Mind overflow!

11. **signed shift right (two's complement)** (`1010`). This implements "divide by two".
    Example:

a	`0010010010101010`
result	`0001001001010101`

    Mind overflow!

12. **less than** (`1011`). Results in 1 if $a < b$, 0 if a $\geq$ b.
    Example:

a	`0010010010101010`
b	`1010100101010010`
result	`0000000000000000`

13. **greater than** (`1100`). Results in 1 if $a > b$, 0 if a $\leq$ b.
    Example:

a	`0010010010101010`
b	`1010100101010010`
result	`0000000000000001`

14. **equals** (`1101`). Results in 1 if $a = b$, 0 if a $\neq$ b.
    Example:

a	`0010010010101010`
b	`1010100101010010`
result	`0000000000000000`