

# Computer Systems and -architecture

## Project 4

1 Ba INF 2012-2013

Bart Meyers  
bart.meyers@ua.ac.be

*Don't hesitate to contact the teaching assistants of this course. You can reach them in room M.G.3.17 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, 25 2012, 23u55**
- Evaluation and feedback: **November, 27 2012**

## Project

Read sections C.7, C.8 and C.10 of Appendix C. You can use all Logisim libraries for this assignment.

1. Build a **12-bit register** using 12 D flip-flops that are updated on the **falling edge** (beware: in Logisim D flip-flops are by default on the rising edge). Inputs are:
  - 12-bit D, which denotes the input data
  - 1-bit Reset, that sets the contents of the register to 000000000000
  - 1-bit write, write enabled?
  - 1-bit C, the clock signal

The only output is an 12-bit Q that contains the contents of the register.

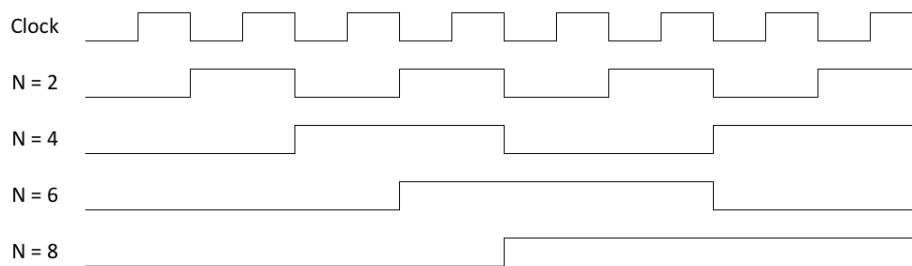
2. Build a **register file** made of eight of your own 12-bit registers. The register file must be able to read from and write to specified registers. In this case, the register file reads from two registers, and can possibly write to a register at the same time. Register 0 is a special

case: it always contains zero, and writing to it doesn't modify its contents. The register file has the following in- and outputs:

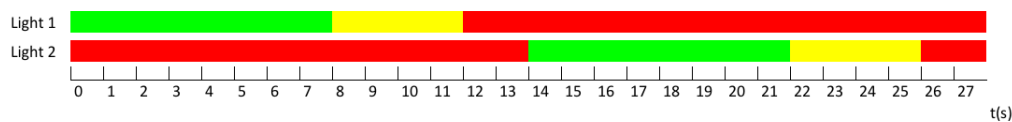
name	in/out	width	meaning
rs	I	3 bits	register <b>\$rs</b> index number
rt	I	3 bits	register <b>\$rt</b> index number
rd	I	3 bits	register <b>\$rd</b> index number
Data	I	12 bits	used as input for the write operation
write	I	1 bit	write to <b>\$rd</b> enabled?
C	I	1 bit	clock input
reset	I	1 bit	reset all registers?
S	O	12 bits	register <b>\$rs</b> content
T	O	12 bits	register <b>\$rt</b> content

- Build a **counter** using your own 12-bit carry lookahead adder and 12-bit register. Inputs are C (the clock) and D (an 12-bit number up to which the counter counts), the output is the current 12-bit value of the register. At every clock tick, the counter adds 1 to the number in the register. When the register value is equal to or greater than D, the value is reset to zero. A counter with its D-input equal to 3 counts from 0 to 2. You can use the Logisim built-in *Comparator*.
- Build a **clock divider**. A clock divider is used to create a slow "daughter" clock from a faster "parent" clock. Inputs are C (the clock) and N (12-bit number). You can assume that N is even. The clock divider generates an output clock signal with a frequency that is N times lower than the input clock signal. Use components of your own as much as possible.

*Optional: can you make the divider work with odd numbers as well?*



- Build a **finite-state machine** that implements a traffic light system on a cross section. Finite-state machines use memory and a clock. Since finite-state machines are *synchronous*, a new state is computed every clock cycle. A 2 Hz clock has a full clock cycle of 1 second. Use your counter and clock divider to advance through the states and make sure your state transitions happen at the right time. The two traffic lights behave like the following figure:



- To prepare for the next lab session, read sections 4.1, 4.2, 4.3 and 4.4 of Chapter 4.