Workshop 03 - Sequential Hardware

Workshop 03 - Sequential Hardware

The purpose of this workshop is to show you how memory could be built in hardware. This is based on **project 03 (https://www.nand2tetris.org/project03)** from the Nand2Tetris course. This workshop uses builtin (working) versions of the chips you have already seen.

We have created a **Nand2Tetris Project 03 (https://myuni.adelaide.edu.au/courses/54311/pages/nand2tetris-project-03)** assignment in the **Web Submission System (https://cs.adelaide.edu.au/services/websubmission)** so that you can attempt this project independently from this workshop and be awarded up to 5 additional participation marks.

Participation Marks

Up to 5 participation marks are available for every workshop, 2 for preparation, 1 for attendance and 2 for completing an activity, subject to the conditions described on the **Participation Marks (https://myuni.adelaide.edu.au/courses/54311/pages/participation-marks)** page.

Attendance

One participation mark will be awarded if your workshop attendance is recorded and you make a submission to the Workshop 03 assignment in the Web Submission System by **Friday 11.55pm** of week 3. **Do not** leave a workshop until you have checked your attendance mark.

If you are using a CAT suite computer running Linux in a timetabled workshop, your presence should be automatically recorded when you visit our **practical marker (https://cs.adelaide.edu.au/services/pracmarker/)**. If your attendance mark is not displayed when you visit our **practical marker (https://cs.adelaide.edu.au/services/pracmarker/)**, your presence will need to be manually recorded. **Do not click** on the "flag me for marking" button until a supervisor is standing next to you and is ready to record your presence. The flag is only to speed up the data entry and is only visible for 30 seconds.

Workshop 03 Background Reading

Read the third chapter of the text book and the project 03 description from the Nand2Tetris course. Then think about how you would answer these two questions:

Question 1

Purely combinatorial circuits don't need a clock. However, a clock becomes important once we have a memory to store data. Why is a clock is important for memory to function properly? (Hint: this relates to the definition of memory itself).

Question 2

Read chapter 3.1 of the textbook - under the paragraph heading "Time Matters". In the Hack machine only memory chips and registers have sequential logic. This means that the other (combinatorial) logic is time insensitive. This means that, depending on the length of the wires, operations such as those in the ALU will, for some moments of time, be producing garbage as they wait for all signals coming in to be in sync. Why does this period of garbage not affect the correct working of the chip?

Pencil and Paper

You will find it much easier to understand what is happening if you bring some paper and something to write with. Drawing pictures of how gates fit together and truth tables of what inputs produce what outputs can be very helpful.

Workshop 03 Preparation Activity

Two participation marks will be awarded for completion of this preparation activity if it is completed at least 10 minutes before the first timetabled workshop of the week, see **Participation Marks (https://myuni.adelaide.edu.au/courses/54311/pages/participation-marks)** for details.

Note: in example commands % is the shell's prompt, it is not part of the command.

Note: this workshop assumes that you have already created directories for every assignment, workshop project and exam in your svn repository, as described on the **Startup Files for Workshops and Assignments (https://myuni.adelaide.edu.au/courses/54311/pages/startup-files-for-workshops-and-assignments)** page.

Note: the web submission system will record 0 marks for completing this activity, the 2 marks are awarded later if and only if your attendance is recorded.

1. If required, checkout a working copy of the workshop03 directory from your svn repository.
2. Change directory to the working copy of the workshop03 directory.
3. Copy the newest zip file attached below into the updates sub-directory and add it to svn. Do not unzip the file.
4. Run the following command to place the workshop's startup files in the correct locations:

   ```
   % make install
   ```

5. Add the .hdl files to your svn repository:

   ```
   % svn add --depth=empty a b
   % svn add [ab]/*.hdl
   % svn commit -m "Workshop 03 Startup Files"
   ```

6. Goto the Web Submission System and make a submission to the Workshop 03 assignment. A successful submission that passes the preparation tests will complete the preparation activity.

Workshop 03 Activity

Two participation marks will be awarded for completing the following activity. This need not be completed during the workshop but no participation marks will be awarded if the activity is not completed by **Friday 11.55pm** of week 3.

After completing each of the following steps, commit your changes to svn:

```
% svn commit -m workshop03-activity
```

After each commit, go to the Web Submission System and make a submission to the Workshop 03 assignment. A successful submission that passes the Bit, Register and RAM8 tests will complete the workshop activity.

1. Build and test a one-bit register from project 03 of the Nand2Tetris course. This requires you to change to the *a* sub-directory of your workshop 03 startup files and edit the HDL file **Bit.hdl**.

   If you draw yourself a picture of the information flow you will see that you only need two chips to implement this, a Mux chip and a DFF chip.

2. Build and test using HDL a 16 bit register from project 03 of the Nand2Tetris course. This requires you to edit the HDL file **Register.hdl**.

If you draw yourself a picture of the information flow you will see that you only need one chip, the Bit chip, but you need 16 copies of it. To separate out the 16 bit inputs and outputs, you can name a specific wire using array notation. So to refer to input wire 3, you can write in[3] and to refer to output wire 3, you can write out[3]. The individual wires are numbered form 0 to 15.

3. Build and test using HDL a memory of 8 registers. This requires you to edit the HDL file **RAM8.hdl**.

   If you draw yourself a picture of the information flow you will see that you need several chips: Register chips, Mux8Way16 chips and DMux8Way chips. The **key observation** to make is that you cannot choose which register to send the input to, it must go to **all** registers. That is why we never implemented any DMux8Way16 chips. Instead you need to use a DMux8Way chip to choose which register receives the load signal.

4. Build and test using HDL a memory of 64 registers. This requires you to edit the file **RAM64.hdl**.

   If you draw yourself a picture of the information flow you will see that you could replace the Register chips from the previous step with RAM8 chips. The key observation to make is that you need to carefully choose which part of the address is used by your DMux8Way and Mux8Way16 chips and which part is passed through to the RAM8 chips. You can refer to a subset of a set of wires using a combination of array notation and the **..** operator. For example, to refer to the least significant 3 bits of an address, you can write address[0..2].

5. If time permits you could change to the *b* directory and attempt to build bigger memories. The construction of each larger memory can reuse the existing chips in exactly the way as the RAM64 chip reuses the RAM8 chip. The only differences are the selection of writes from the address.

6. After completing the RAM64 chip you may instead prefer to attempt an implementation of the Program Counter. This is more challenging since it requires you to implement a reset signal, use an adder, or choose a jump based on the two output signals from the ALU, **zr** and **ng** in combination with select bits from the current instruction.

Startup Files

The newest of the following zip file(s) must be placed in the updates sub-directory and added to svn. When make is run, the newest zip file in the updates directory is used to update the startup files. Any files you are required to edit will not be updated but, a copy of the latest version of those files will be placed in the sub-directory originals.

- **workshop03-20200806-105046.zip (https://myuni.adelaide.edu.au/courses/54311/files/7198789/download?wrap=1)**
- workshop03-20200724-193245.zip