

Assignment 03 – program counter, registers and memories

EDB HDL

ANM, v05

The material in this assignment is taken from <https://www.nand2tetris.org/> under [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#).

Introduction

In a microcontroller CPU there is the need for sequential logic:

- **Registers** need to store the inputs and the output of the ALU.
- The **program counter** needs to control the address of the next command in the program memory
- **Memories** are required for an efficient storage of large amounts of information

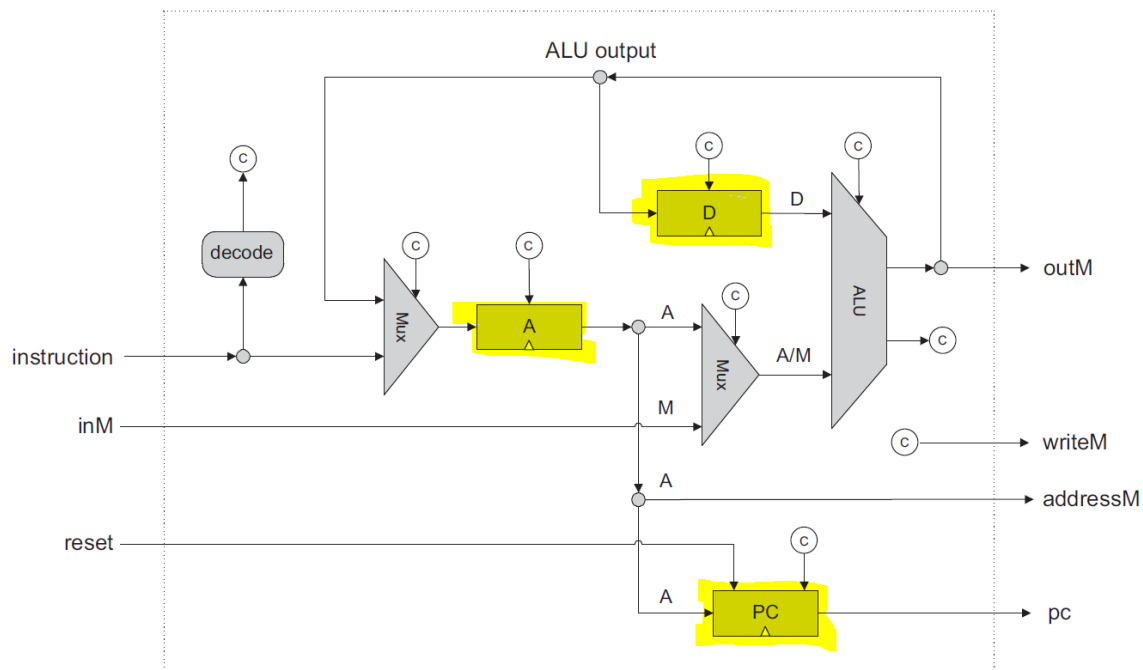


Figure 1: HACK CPU. The yellow blocks show the registers. Source: <https://www.nand2tetris.org/>

Tasks

- Create the folder structure
 - ./seq/sim
 - ./seq/src

Task 01 – Implementation of a D register

- [0.5cp] Create a SystemVerilog file “dreg.sv” in the ./seq/src folder that implements the specified D register “dreg”.



Figure 2: Block diagram of the D register.

- Requirements
 - Name of the module
 - dreg
 - Parameters
 - W ... width of the register, default value shall be 16
 - Inputs
 - rst_n ... active low reset; async resets the register to all zeros
 - clk50m ... clock; active edge is the positive edge
 - en ... enable signal
 - load ... enables the input
 - d[W-1] ... register input of width W
 - Outputs
 - q[W-1] ... register output of width W
 - Functionality
 - If $\text{rst_n}(t-1) == 0$ then $q(t) = 16'h0000$
 - else if $(\text{load} \ \& \ \text{en})$ then $q(t)=d(t-1)$
 - else $q(t)=q(t-1)$
- [0.5cp] Create a SystemVerilog test bench “tb_dreg.sv” in the ./seq/sim folder
 - Create a testbench that checks that the register
 - Is reset to all zeros when $\text{rst_n} == 0$
 - Check that the register can be set to the following values: 16'h0000, 16'hffff, 16'haa55, 16'h55aa.
 - The test bench shall allow to check the correctness of the implemented logic in the wave window of the simulator.
 - All results (i.e. the output **q**) are self-checked by the testbench. The number of errors is recorded. At the end of the test bench a Pass/Fail message is shown.
 - Create a TCL script file “sim_tb_dreg.tcl” that controls the compile and simulation process. A wave window is opened and all relevant signals are displayed.

Task 02 – Implementation of the program counter

- [1cp] Create a SystemVerilog file “pcount.sv” in the ./seq/src folder that implements the specified program counter “pcount”.

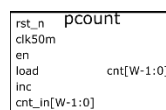


Figure 3: Block diagram of the HACK program counter.

- Requirements
 - Name of the module
 - pcount
 - Parameters
 - W ... width of the counter, default value shall be 15
 - Inputs
 - rst_n ... active low reset; async resets the register to all zeros
 - clk50m ... clock; active edge is the positive edge
 - en ... enable
 - load ... loads the counter to cnt_in (parallel load)
 - inc ... increments the counter at an active clock edge
 - cnt_in[W-1:0] ... input for a parallel load of the counter
 - Outputs
 - cnt[W-1:0] ... counter output of width W
 - Functionality
 - If $\sim\text{rst_n}(t-1)$ then $\text{cnt}(t)=0$
 - else if (load & en) then $\text{cnt}(t)=\text{cnt_in}(t-1)$
 - else if (inc & en) then $\text{cnt}(t)=\text{cnt}(t-1)+1$
 - else $\text{out}(t)=\text{out}(t-1)$
- [1cp] Create a SystemVerilog test bench “tb_pcount.sv” in the ./seq/sim folder
 - Create a testbench that checks that the counter
 - Is correctly reset
 - Can be loaded
 - Increments correctly
 - The test bench shall allow to check the correctness of the implemented logic in the wave window of the simulator.
 - All results (i.e. the output **cnt**) are self-checked by the testbench. The number of errors is recorded. At the end of the test bench a Pass/Fail message is shown.
 - Create a TCL script file “sim_tb_pcount.tcl” that controls the compile and simulation process. A wave window is opened and all relevant signals are displayed.

Task 03 – 16k data memory verification

The HACK computer needs different memory types. As a typical example a 16kbyte RAM module shall be implemented in SystemVerilog (see below):

```

/*-----
Project:    Hack Computer
Module:     RAM16k in verilog
Purpose:    Data RAM in SystemVerilog as it has a lower delay than the generated
IP
Author:     Andre Mitterbacher
Version:    v0, 13.05.2019
-----*/

module ram16k_verilog(
    // same interface as the generated RAM IP!

```

```

    input  logic      aclr,
    input  logic      [13:0] address,
    input  logic      clock,
    input  logic      [15:0] data,
    input  logic      wren,
    output logic      [15:0] q
);

logic [15:0] mem [0:(1<<14)-1];

always_ff @ (posedge clock) begin
    // The reset branch is deleted as it is not supported by Intel FPGA RAM memory.
    /*if (aclr) begin
        mem <= '{default:'0};
    end
    else */if (wren) begin
        mem[address] <= data;
    end
end

assign q = mem[address];

endmodule

```

Figure 4: SystemVerilog description of a 16k RAM memory.

As can be seen in the code, the functionality of the input **aclr** was removed to allow for a more efficient synthesis.

- Copy the SystemVerilog code from Figure 4 and paste it into the ./seq/src/ram16k_verilog.sv file.
- [1cp] Create a SystemVerilog test bench “tb_ram16k_verilog.sv” in the ./seq/sim folder
 - Set the RAM to different values:
 - All zeros
 - All ones
 - Each byte to value of its address.

Task 04 – Documentation

- [1cp] Create a short summary report “doc_seq.pdf” that shows the simulation result.
 - Show that your design fulfils the specification (verification).
 - Discuss why you chose your implementation method (advantages, disadvantages).

Appendix