Computer System Design Lab
Design Experiment 3:
Sequential Components and Memory Design

Indian Institute of Technology Tirupati

Sept 3, 2019

1 The Problem Statement:

Memory system is one of the essential components of computer system. In this experiment different components of memory system will be designed and tested. The experiment give a hands-on experience of writing a HDL code to design the components such as DFF, Registers, Counters, and Memories. A set of test benches will be written to test the HDL design. All the designs will be written in BHDL of Nand2Tetris and tested using the corresponding TSL.

Following components are to be designed and tested.

Chip name	Functionality
myDFF	D Flip-flop
Reg1Bit	One bit register with Load control signal
Reg8Bit	b-bit register with Load control signal
Reg16bit	16-bit register with Load control signal
Reg32bit	32-bit register with Load control signal
LSR8bit	8-bit left shift register
RSR8bit	8-bit right shift register
Counter8	8-bit up-down counter
PC8bit	8-bit program counter
PC16bit	16-bit program counter

Design Specification:

- **D Flip-flop:** A multiplexor-inverter based D flip-flop with following I/O pins.

  Input/Output:

  Input 1: Data-in (D), Input 2: Clock input (Clk); Output 1: Data-out (Q)

  Functionality:

  ```
  if (Clock)
  then Data-out(t) <= Data-in(t - 1)
  esle Data-out(t) <= Data-out(t - 1)
  ```

- **1-bit Register:** A single bit register built from DFF with the following functional specification.

  Input/Output:

Input 1: Load, Input 2: Clock, Input 3: Data-in; Output 1: Data-out.

Functionality:

```
if (Load = 1 and Clock)
then Data-out(t) <= Data-in(t - 1)
else Data-out(t) <= Data-out(t - 1)
```

- **Reg8bit, Reg16bit, and Reg32bit:** The functional specification of *n-bit* register is equivalent to 1-bit register, where for *n-bit* register the width of Data-in is n.
- **Shift register:** Shift register can be of two possible type: left shift or right shift register. The functional specification of shift register could be as follow.

  Input/Output:

  Input 1: n-bit wide Data-in, Input 2: single bit load signal (Load), Input 3: single bit shift signal (Shift), Input 4: Clock; Output 1: n-bit wide Data-out.

  Functionality:

```
    if(Load = 1 and Shift = 0 and Clock)
    then Data-out(t) <= Data-in(t - 1);

    if(Load = 0 and Shift = 1 and Clock)
    then Data-out(t) <= shift(Data-out (t - 1)) // one bit shift

    if(Load = 1 and Shift = 1 and Clock)
    then Data-out(t) <= Data-in(t - 1);

    if(Load = 0 and Shift = 0 and Clock)
    then Data-out(t) <= Data-out(t - 1);
```

- **Counter:** An up-down counter with the following functional specification.

  Input/Output:

  Input 1: *n-bit* Data-in, Input 2: Up/Down control input, Input 3: Clock; Output 1: *n-bit* Data-out.

  Functionality:

```
    if(Load = 1 and Up/Down = x and Clock)
    then Data-out(t) <= Data-in(t - 1)

    if(Load = 0 and Up/Down = 0 and Clock)
    then Data-out(t) <= Data-out(t - 1) + 1 // Up count
```

```
if(Load = 0 and Up/Down = 1 and Clock)
then Data-out(t) <= Data-ou(t - 1) - 1 // Down count
```

- **Program counter:** An special counter used for fetching instructions from the main memory. The functional specification to be coded is following.

  Input/Output:

  Input 1: *n-bit* Data-in, Input 2: Load, Input 3: Reset, Input 4: two bit increment control signal (Incr1 and Incr2); Output 1 = *n-bit* Data-out.

  Functionality:

```
if(Reset)
then Data-out(t) <= 0

if(Load and Clock)
then Data-out(t) <= Data-in(t - 1)

if(Incr0 = 0 and Incr1 = 0)
then Data-out(t) <= Data-out(t -1) + 0

if(Incr0 = 0 and Incr1 = 1)
then Data-out(t) <= Data-out(t -1) + 1

if(Incr0 = 1 and Incr1 = 0)
then Data-out(t) <= Data-out(t -1) + 2

if(Incr0 = 1 and Incr1 = 1)
then Data-out(t) <= Data-out(t -1) + 4
```

  In the above codes the = means comparison and the <= means write the right side value to left.

2 The Experimental Flow:

1. Design specification: to decide on what are the valid inputs and correct outputs for the given design.

   At first, you need to decide on the design specification. Which can be done, for the small design, in a truth table. For every design asked as part of this exercise, you need to specify the truth table.

2. Design description: to describe (code) the design using HDL.

   As part of this design exercise, we will be using the HDL language to describe the given designs. The HDL program could be edited in any of the editor of choice (gvim, emacs, atom etc) and then be read in to the *nand2tetris* synthesis tool for compilation. Refer to the tool user manual for "How to use".

3. Synthesis: to compile the design which would check for syntax error.

   In this stage you will be performing compilation of the design which you have described in HDL and make it error free to go to the next stage for simulation.

4. Design Verification: to simulate the design by writing a test bench in TSL (test script language).

   Once the design compilation is completed, the next task is to perform simulation to verify the correctness of the design. For any design to verify by simulation a test bench need to be written. The test bench can be written in TSL script, a manual for TSL script is provided in Appendix B of the text book. Along with the TSL script, a comparison file in .cmp format for each design needs to be prepared as per the format of .cmp.

5. Library Preparation: to keep the verified design in an appropriate location where the design could be reused in other larger design.

   Once the design verification is done the design could be placed in a specific directory such that it can be reused in further design.

Design rules:

- Naming: The Chip name, Input and Output pins, proper commenting for statement are essential part of good coding. Each chip name (and file name) should be prefixed with your registration number such as cs16b01Reg32.
- Error Logging: It is good to maintain a log file each of the error that you encounter while compilation or simulation. It is a good practice to maintain proper log file with description of error and its fix.
- Report: Prepare a document on various design decision that you take related to design optimization, simulation method etc.

3 Tools:

- Language: The Nand2Tetris HDL and TSL (test scripting language)
  Refer: Appendix A and B of text book.
- Tools: Hardware Simulator of Nand2Tetris.
  `https://www.nand2tetris.org/software`
- Machine and OS: x86_64 machines with any distribution of Linux (Ubuntu or CentOS).

4 Reporting and Evaluation

All designs will be evaluated as per the criteria of evaluation which is primarily based on correctness and elegance of the design. Elegance here means how well the concept of design reuse, naming of the chip and interconnects, and commenting on the function of each line has been done. And, most importantly how well your design is optimized in terms of length of critical path [1] and total number of basic gates. The design which follows the Design Rule will be evaluated accordingly.

A report need to be prepared for documentation of the work. A final report, at the end of all experiments which describe the design of computer system from basic to system level, will be evaluated during your final test.

[1] Critical path is defined as a longest combinational path between any two input and output either from primary side or from flip-flop.