Introduction to Digital Logic
EECS/CSE 31L

Assignment 4

EECS Department
Henry Samueli School of Engineering
University of California, Irvine

November, 1, 2014

Due on Saturday 11/8/2014 11:00pm. Note: this is a one-week assignment

1 Design and implementation of sequential logic blocks [100 points + 5 bonus points]

The goal of this assignment is to practice sequential logic blocks in VHDL.

1.1 Assignment Description

The objective of this project is to assess your understanding of how to implement a sequential hardware blocks. In this assignment you are supposed to implement a register, register file, counter, and FIFO in VHDL and verify to see it is working as expected. The blocks are:

Register
The register design should be parameterized (default 32-bit), supporting both asynchronous, synchronous reset and parallel-input parallel-output data. It should also have incrementing feature.

Register file
The register file design should be fully parameterized in both number of registers (default 8) and register width (default 32-bit). This is a single-port register file, supporting two read and one write at each cycle.

Counter
Counter width should be parameterized (default 32-bit), supporting preloading, synchronous reset, and variable counting step.

1.1.1 Register

Code 1: Sample entity of a register in VHDL

```
ENTITY reg IS
 GENERIC (NBIT: INTEGER :=32);
 PORT(
  clk     : IN std_logic;
  rst_a   : IN std_logic;  -- asynchronous reset
  rst_s   : IN std_logic;  -- synchronous reset
  inc    : IN std_logic;  -- increment
  we      : IN std_logic;  -- write enable
  din      : IN std_logic_vector(NBIT-1 downto 0);       -- input data
  dout    : OUT std_logic_vector(NBIT-1 downto 0)    -- output data
 );
END reg;
```

1.1.2 Register file

You may use your already developed Register block for implementing a Register file. Note that this block only has synchronous reset.

Code 2: Sample entity of a register file in VHDL

```
ENTITY regfile IS
 GENERIC (NBIT: INTEGER := 32;
                  NSEL: INTEGER := 3);
 PORT(
      clk      : IN std_logic;
      rst_s    : IN std_logic;    -- synchronous reset
      we    : IN std_logic;      -- write enable
      raddr_1 : IN      std_logic_vector(NSEL-1 DOWNTO 0);  -- read address 1
      raddr_2 : IN      std_logic_vector(NSEL-1 DOWNTO 0);  -- read address 2
      waddr    : IN      std_logic_vector(NSEL-1 DOWNTO 0);  -- write address
      rdata_1 : OUT std_logic_vector(NBIT-1 DOWNTO 0);    -- read data 1
      rdata_2 : OUT std_logic_vector(NBIT-1 DOWNTO 0);    -- read data 2
      wdata    : IN      std_logic_vector(NBIT-1 DOWNTO 0)     -- write data 1
 );
END regfile;
```

1.1.3 Counter

Your counter block should be able to count ascending (asc = 1) or descending (asc = 0). Also in case the preload = 1, it should load din input to its internal counter register. The counting step should be also a parameter.

Code 3: Sample entity of a counter file in VHDL

```
ENTITY counter IS
 GENERIC (NBIT: INTEGER := 32;
                  STEP: INTEGER := 1);
 PORT(
              clk          : IN         std_logic;
              rst_s      : IN  std_logic;     -- synchronous reset
              asc        : IN  std_logic;     -- ascending (if it is '0', count descending)
              preload    : IN  std_logic; -- read din as the counting seed
              din          : IN         std_logic_vector(NBIT-1 downto 0);
              dout      : OUT std_logic_vector(NBIT-1 downto 0)
        );
END counter;
```

1.2 Assignment Deliverables

Your submission should include the following:

- Design report for each block
- Waveform snapshot for each block
- VHDL code file of each block
- Testbench VHDL code file of each block

Note: Remember to compress all VHDL files (with rar or zip extension) and name your compressed file as **assignment4_STUDENT-ID_codes.rar** and your report as **assignment4_STUDENT-ID_BLOCK-NAME.pdf**.