Digital Systems

Prof. James Clark

ECSE 325

Lab #2: Fixed-point representation, VHDL testbench creation and simulation with ModelSim

Winter 2018

Introduction

In this lab, you will learn the basics of fixed-point representations, VHDL testbench creation and functional verification using ModelSim. You will implement a multiplication-accumulation (MAC) unit using a fixed-point representation in VHDL, write a testbench code in VHDL and validate your design in ModelSim through following a step-by-step tutorial.

Floating-Point vs. Fixed-Point

- Numbers are stored in binary format in digital hardware
- A binary number is a fixed-length sequence of bits
- Binary numbers are represented as either fixed-point or floating-point

Floating-point:

- More precision
- Large number of bits
- Variable decimal point (as name suggests)

Fixed-point:

- Less precision
- Less number of bits
- Fixed decimal point (makes computations easier)

Fixed-Point Representation

Fixed-point: $(\mathbf{W}, \mathbf{F})$

- **W**: word length
- **F**: fractional length

- Larger **W**, **F**: more precise computation
- Smaller **W**, **F**: less area, reduced power consumption

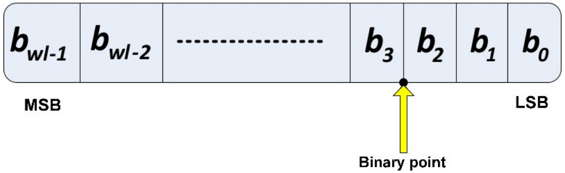


Figure 1: Fixed-point representation

Data Types and Conversion

Fixed-point representation can either be signed in following formats:

- Sign-magnitude
- 1's complement
- 2's complement (most common)

or can be an unsigned number.

Preliminary Examples

Given **W** $= 6$, **F** $= 2$; calculate following in unsigned binary format:

- $4.5 =$
- $1.25 =$
- $14.94 =$

Preliminary Examples

Given **W** = 6, **F** = 2; calculate following in unsigned binary format:

- 4.5 = 0100.10
- 1.75 = 0001.11
- 14.94 = 1111.00 (rounded!)

Arithmetic Conversion

Floating-point to fixed-point conversion:
Step 1: Calculate $b = a \times 2^F$

- a: Floating-point number
- F: Fractional length of the variable

Step 2: Round b to nearest integer:

- $round(4.22) = 4$
- $round(-1.79) = -2$

Arithmetic Conversion - cont'd

Floating-point to fixed-point conversion:
Step 3:

- Convert b from decimal to binary representation $c \rightarrow n$ bits long
- **W** should be equal to or larger than n
- If **W** $< n$, need truncation
- Else if **W** $\geq n$, add zero-bits to leftmost of c

Arithmetic Conversion - Examples

$a = 3.013$, $(\mathbf{W},\mathbf{F}) = (8,3)$

- Step 1: $b = a \times 2^F = 3.013 \times 2^3 = 24.1040$
- Step 2: $round(b) = 24$
- Step 3: $c = dec2bin(24) = 11000 \rightarrow 00011000 \rightarrow 00011.000$

$a = 9.51432$, $(\mathbf{W},\mathbf{F}) = (12,7)$

- Step 1: $b = a \times 2^F = 9.51432 \times 2^7 = 1217.8329$
- Step 2: $round(b) = 1218$
- Step 3:
  $c = dec2bin(1218) = 010011000010 \rightarrow 01001.1000010$

Arithmetic Conversion - Examples

$a = -9.0514$, $(\mathbf{W},\mathbf{F}) = (14, 9)$

- Step 1: $b = a \times 2^F = -9.0514 \times 2^9 = -4634.3$
- Step 2: $round(b) = -4634$
- Step 3:
  $c = dec2bin(-4634) = 10110111100110 \rightarrow 10110.111100110$

Fixed-Point Conversion in MATLAB

$a = fi(v, s, w, f)$

- a: Fixed-point value
- v: Floating-point value
- s: Signed(1)/Unsigned(0)
- w: Word length (**W**)
- f: Fractional length (**F**)

Examples:

- $fi(3.013, 1, 7, 4) = 3.625 \rightarrow 011.1010$
- $fi(9.51432, 1, 12, 7) = 9.5156 \rightarrow 01001.1000010$
- $fi(-9.0514, 1, 14, 9) = -9.0508 \rightarrow 10110.111100110$

Fixed-Point Arithmetics: Addition and Multiplication

Given the two operands a and b represented using $(\mathbf{W_1}, \mathbf{F_1})$ and $(\mathbf{W_2}, \mathbf{F_2})$ respectively, the output precision required to guarantee no overflow occurrence for $a + b$ is obtained by

- $(\mathbf{W_1}, \mathbf{F_1}) + (\mathbf{W_2}, \mathbf{F_2}) \rightarrow (\max(\mathbf{W_1}, \mathbf{W_2}) + 1, \max(\mathbf{F_1}, \mathbf{F_2}))$

and for $a \times b$ is computed by

- $(\mathbf{W_1}, \mathbf{F_1}) + (\mathbf{W_2}, \mathbf{F_2}) \rightarrow (\mathbf{W_1} + \mathbf{W_2}, \mathbf{F_1} + \mathbf{F_2})$

Fixed-Point Arithmetics - Addition Examples

- E.g. 4-bit numbers: a and b are both represented using $(4, 3)$
  - $0.100(0.5) + 0.011(0.375) = 00.111(0.875)$
  - $0.101(0.625) + 0.011(0.375) = 01.000(1)$
- E.g. 6-bit numbers: a and b are represented using $(6, 2)$ and $(6, 4)$ respectively
  - $0111.10(7.5) + 11.0110(-0.625) = 00110.1110(6.875)$
  - $0111.11(7.75) + 00.0100(0.25) = 01000.0000(8)$

Fixed-Point Arithmetics - Multiplication Examples

- E.g. 4-bit numbers: a and b are both represented using $(4, 1)$
  - $010.1(2.5) + 100.0(-4) = 110110.00(-10)$
  - $011.1(3.5) + 001.1(1.5) = 000101.01(5.25)$
- E.g. 5-bit numbers: a and b are represented using $(5, 2)$ and $(5, 3)$ respectively
  - $100.01(-3.75) \times 01.011(1.375) = 11010.11011(-5.15625)$
  - $100.00(-4) \times 10.000(-2) = 01000.00000(8)$

VHDL Description of MAC Unit

In order to practice the fixed-point representations of numbers, you will describe a MAC unit in VHDL. The MAC unit consists of three main sub-units: multiplier, adder and register (see Fig. 2). The MAC unit takes two inputs at each clock cycle, multiplies and accumulates them over a given time period *N*. The following pseudo code summarizes the functionality of the MAC unit:

```
mac = 0;
ready = 0;
for i = 1:N
    mac = mac + x(i) * y(i);
end
ready = 1;
```

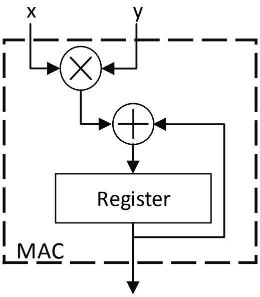


Figure 2: The high-level architecture of MAC unit

VHDL Description of MAC Unit

Before describing the MAC unit in VHDL, you need to find the required precision of each sub-block. You are provided with input data in files *lab1-x.txt* and *lab1-y.txt*: each file contains 1000 real values represented in floating-point format. First, use any programming language of your own choice and the given data to find the precision required for each sub-unit to guarantee no overflow occurrence (i.e. no quantization error). Then, convert the integer values into 2's complement format and store them into the files named as *lab1-x-fixed-point.txt* and *lab1-y-fixed-point.txt*. Finally, describe the MAC unit in VHDL using the obtained precision for each sub-unit and the following entity declaration:

```vhdl
entity gNN_MAC is
port ( x      : in std_logic_vector (?? downto 0); -- first  input
       y      : in std_logic_vector (?? downto 0); -- second input
       N      : in std_logic_vector (9  downto 0); -- total number of inputs
       clk    : in std_logic; -- clock
       rst    : in std_logic; -- asynchronous active-high reset
       mac    : in std_logic_vector (?? downto 0); -- output of MAC unit
       ready  : in std_logic); -- denotes the validity of the mac signal
end gNN_MAC;
```

Testbench Creation in VHDL

So far, you have implemented the MAC unit in VHDL. Now, you need to test the correctness of your implementation. To this end, a non-synthesizable VHDL code known as *testbench* is generally used. In fact, testbench code iteratively applies a sequence of controlled inputs to a circuit and compares its output against the expected output (see Fig. 3). If a mismatch is detected, an error is displayed in the VHDL simulators log which can then be consulted to help direct a designer search for the problem in the circuits RTL description. Note that the expected outputs are usually generated by the programming language that was used for floating-point to fixed-point conversion (why?).

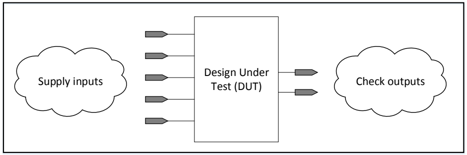


Figure 3: Testbench architecture

Testbench Creation in VHDL - ModelSim Setup

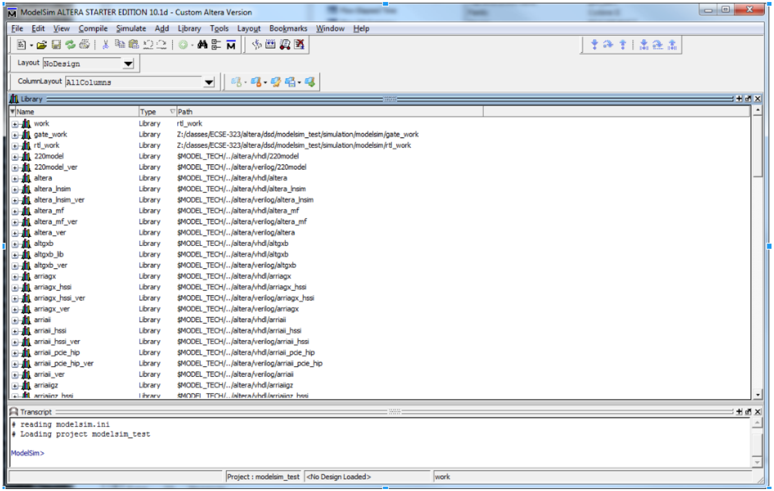
You will now write a testbench code from scratch for your implementation of MAC unit in ModelSim. Use the following steps to setup ModelSim:

- Launch ModelSim and Create a new project with File > New > Project . . .
- Name the project and choose its path of your own choice
- Click on the Add Existing File button and the VHDL code of your MAC circuit
- Create a new VHDL file with File > New > Source > VHDL and name it as "gNN_MAC_tb.vhd"
- The testbench entity is unique in that it has NO inputs or outputs (why?). Define an empty entity in the created VHDL file.
- Compile the imported and created VHDL files.

ModelSim should now successfully compile all the files.

Launching ModelSim

Double-click on the ModelSim desktop icon to startup the ModelSim program. A window similar to the one shown below will appear.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Transcript

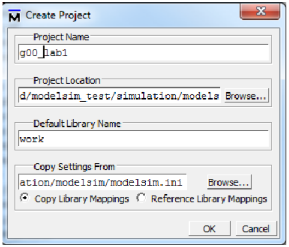
Name	Type	Path
work	Library	rtl_work
gate_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/gate_work
rtl_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/rtl_work
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220model
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_lnsim
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera_lnsim
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera_mf
altera_ver	Library	$MODEL_TECH/../altera/verilog/altera
altgxb	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_lib	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_ver	Library	$MODEL_TECH/../altera/verilog/altgxb
arriagx	Library	$MODEL_TECH/../altera/vhdl/arriagx
arriagx_hssi	Library	$MODEL_TECH/../altera/vhdl/arriagx_hssi
arriagx_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriagx_hssi
arriagx_ver	Library	$MODEL_TECH/../altera/verilog/arriagx
arriaii	Library	$MODEL_TECH/../altera/vhdl/arriaii
arriaii_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaii_hssi
arriaii_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_hssi
arriaii_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriaii_pcie_hip
arriaii_pcie_hip_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_pcie_hip
arriaii_ver	Library	$MODEL_TECH/../altera/verilog/arriaii
arriaiigz	Library	$MODEL_TECH/../altera/vhdl/arriaiigz
arriaiigz_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaiigz_hssi

```
# reading modelsim.ini
# Loading project modelsim_test

ModelSim>
```

Project : modelsim_test | <No Design Loaded> | work

Launching ModelSim

Select File → New → Project and, in the window that pops up, give the project the name "*gNN_lab2*". Click OK. Another dialog box will appear, allowing you to add files to the project. Click on "Add Existing File" and select VHDL file that was generated earlier (*gNN_MAC.vhd*). You can also add files later.


Create Project

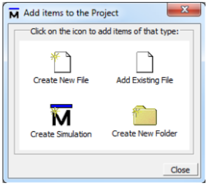
Project Name: g00_lab1

Project Location: d/modelsim_test/simulation/models

Default Library Name: work

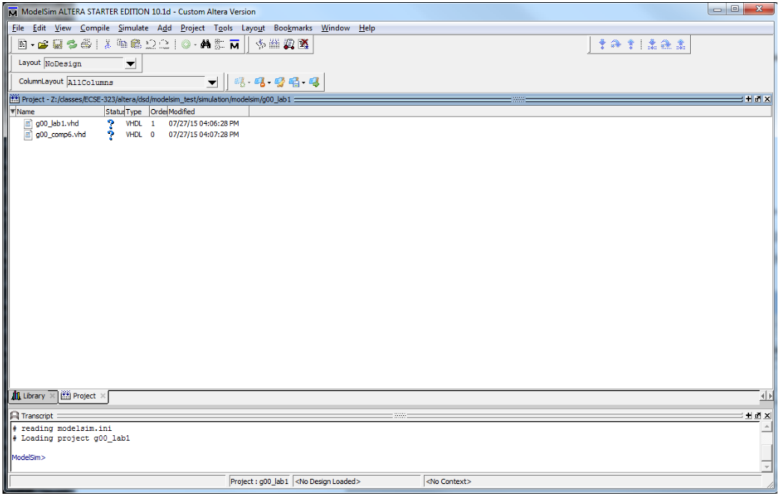
Copy Settings From: ation/modelsim/modelsim.ini

[✓ Copy Library Mappings] Reference Library Mappings



Setting up ModelSim

The ModelSim window will now show your VHDL file in the Project pane. An example is depicted below.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Project - Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/g00_lab1
Library
Project
Transcript

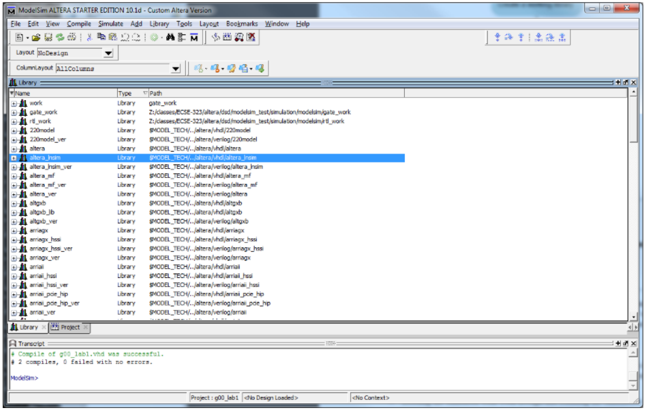
Name	Status	Type	Order	Modified
g00_lab1.vhd	?	VHDL	1	07/27/15 04:06:28 PM
g00_comp6.vhd	?	VHDL	0	07/27/15 04:07:28 PM

```
# reading modelsim.ini
# Loading project g00_lab1

ModelSim>
```

Project : g00_lab1 <No Design Loaded> <No Context>

Setting up ModelSim

In order to simulate the design, ModelSim must analyze the VHDL files, a process known as compilation. The compiled files are stored in a library. By default, this is named "work". You can see this library in the "library" pane of the ModelSim window.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Library
Project
Transcript

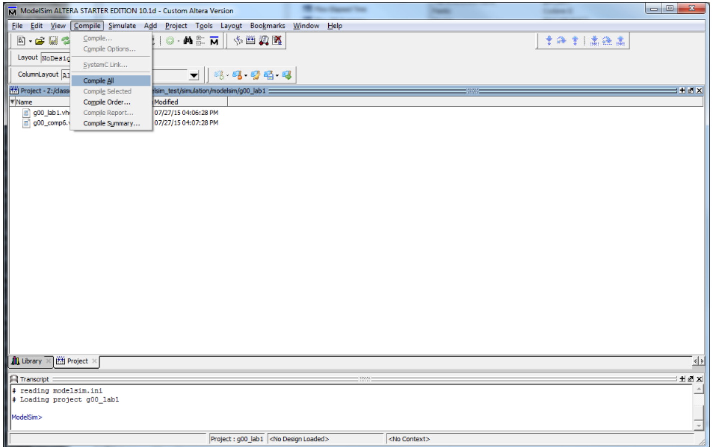
Name	Type	Path
work	Library	gate_work
gate_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/gate_work
rtl_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/rtl_work
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220model
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_lnsim
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera_lnsim
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera_mf
altera_ver	Library	$MODEL_TECH/../altera/verilog/altera
altgxb	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_lib	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_ver	Library	$MODEL_TECH/../altera/verilog/altgxb
arriagx	Library	$MODEL_TECH/../altera/vhdl/arriagx
arriagx_hssi	Library	$MODEL_TECH/../altera/vhdl/arriagx_hssi
arriagx_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriagx_hssi
arriagx_ver	Library	$MODEL_TECH/../altera/verilog/arriagx
arriaii	Library	$MODEL_TECH/../altera/vhdl/arriaii
arriaii_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaii_hssi
arriaii_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_hssi
arriaii_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriaii_pcie_hip
arriaii_pcie_hip_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_pcie_hip
arriaii_ver	Library	$MODEL_TECH/../altera/verilog/arriaii

```
# Compile of g00_lab1.vhd was successful.
# 2 compiles, 0 failed with no errors.

ModelSim>
```

Project : g00_lab1 <No Design Loaded> <No Context>

Setting up ModelSim

The question marks in the Status column in the Project tab indicate that either the files havent been compiled into the project or the source file has changed since the last compilation. To compile the files, select Compile → Compile All or right click in the Project window and select Compile → Compile All.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help
Compile...
Compile Options...
SystemC Link...
Compile All
Compile Selected
Compile Order...
Compile Report...
Compile Summary...
Layout
ColumnLayout
Project - Z:/class...sim_test/simulation/modelsim/g00_lab1
Name
Modified
Library
Project
Transcript

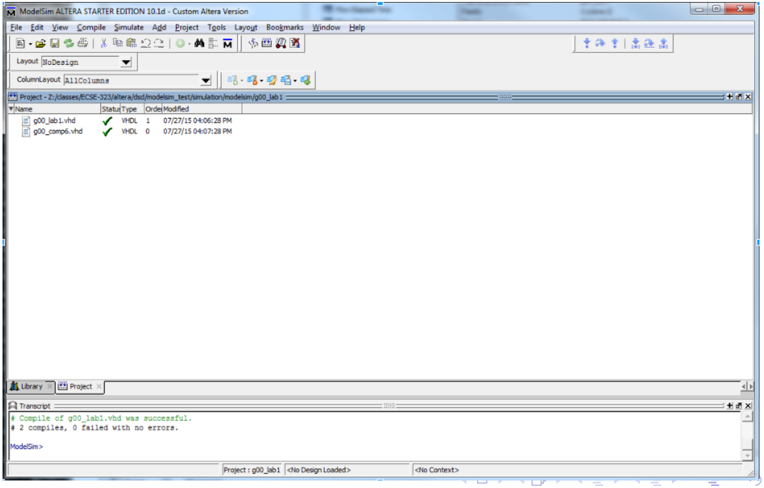
```
g00_lab1.vh   07/27/15 04:06:28 PM
g00_comp6.v   07/27/15 04:07:28 PM
```

```
# reading modelsim.ini
# Loading project g00_lab1

ModelSim>
```

Project : g00_lab1 <No Design Loaded> <No Context>

Setting up ModelSim

If the compilation is successful, the question marks in the Status column will turn to check marks, and a success message will appear in the Transcript pane.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Project - Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/g00_lab1
Library
Project
Transcript

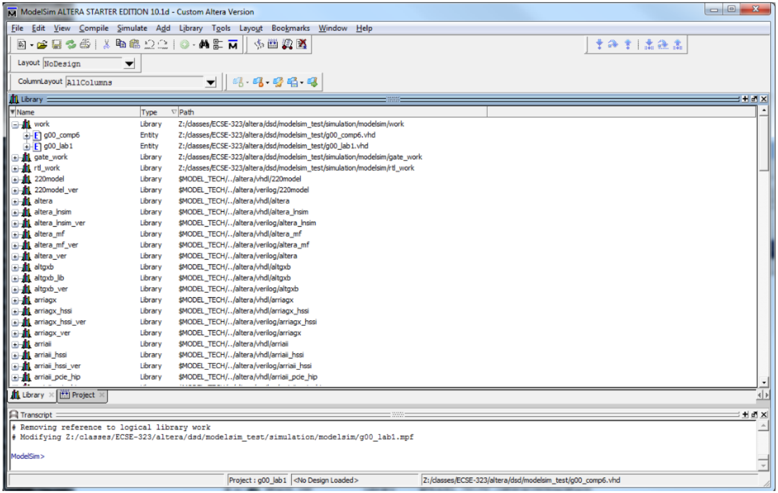
Name	Status	Type	Order	Modified
g00_lab1.vhd	✓	VHDL	1	07/27/15 04:06:28 PM
g00_comp6.vhd	✓	VHDL	0	07/27/15 04:07:28 PM

```
# Compile of g00_lab1.vhd was successful.
# 2 compiles, 0 failed with no errors.

ModelSim>
```

Project : g00_lab1 <No Design Loaded> <No Context>

Setting up ModelSim

The compiled vhdl files will now appear in the library "work".



ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version

File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library

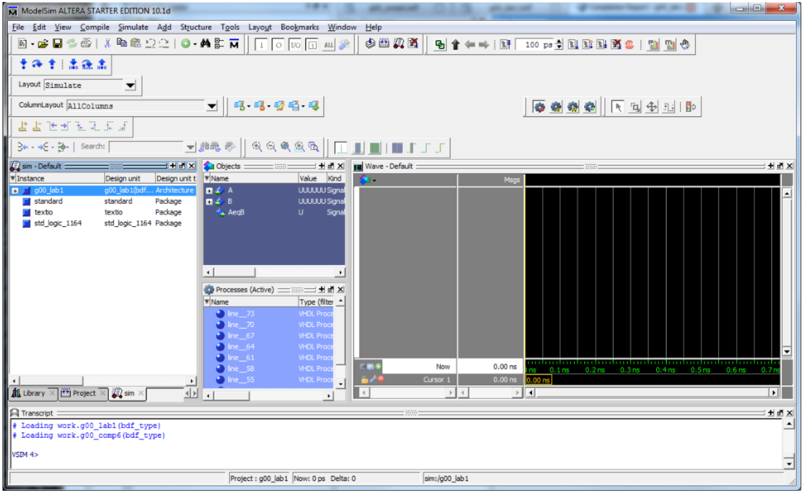
Name	Type	Path
work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/work
g00_comp6	Entity	Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_comp6.vhd
g00_lab1	Entity	Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_lab1.vhd
gate_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/gate_work
rtl_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/rtl_work
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220model
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_lnsim
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera_lnsim
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera_mf
altera_ver	Library	$MODEL_TECH/../altera/verilog/altera
altgxb	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_lib	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_ver	Library	$MODEL_TECH/../altera/verilog/altgxb
arriagx	Library	$MODEL_TECH/../altera/vhdl/arriagx
arriagx_hssi	Library	$MODEL_TECH/../altera/vhdl/arriagx_hssi
arriagx_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriagx_hssi
arriagx_ver	Library	$MODEL_TECH/../altera/verilog/arriagx
arriaii	Library	$MODEL_TECH/../altera/vhdl/arriaii
arriaii_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaii_hssi
arriaii_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_hssi
arriaii_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriaii_pcie_hip

```
# Removing reference to logical library work
# Modifying Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/g00_lab1.mpf

ModelSim >
```

Project : g00_lab1 <No Design Loaded> Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_comp6.vhd

Starting Simulation

In the library window, double-click on *gNN_lab2*. This will open up a bunch of windows which will be used in doing the simulation of the *gNN_lab2* module.


ModelSim ALTERA STARTER EDITION 10.1d
File Edit View Compile Simulate Add Structure Tools Layout Bookmarks Window Help
sim - Default
Objects
Processes (Active)
Wave - Default
Msgs
Now
Cursor 1
Library
Project
sim
Transcript
Search:

Layout Simulate

ColumnLayout AllColumns

Instance	Design unit	Design unit t
g00_lab1	g00_lab1(bdf...	Architecture
standard	standard	Package
textio	textio	Package
std_logic_1164	std_logic_1164	Package

Name	Value	Kind
A	UUUUUU	Signal
B	UUUUUU	Signal
AeqB	U	Signal

Name	Type (filter
line__73	VHDL Proce
line__70	VHDL Proce
line__67	VHDL Proce
line__64	VHDL Proce
line__61	VHDL Proce
line__58	VHDL Proce
line__55	VHDL Proce

Now 0.00 ns
Cursor 1 0.00 ns

```
# Loading work.g00_lab1(bdf_type)
# Loading work.g00_comp6(bdf_type)

VSIM 4>
```

Project : g00_lab1 Now: 0 ps Delta: 0 sim:/g00_lab1

Starting Simulation

You are not quite ready to start the simulation yet!

Notice that, in the "Objects" window, the signals have the value "UUUUUU". This means that all of the inputs are undefined. If you ran the simulation now, the outputs would also be undefined.

So you need to have a means of setting the inputs to certain patterns, and of observing the outputs' responses to these inputs.

In Modelsim, this is done by using a special VHDL entity called a Testbench.

A testbench is some VHDL code that generates different inputs that will be applied to your circuit so that you can automate the simulation of your circuit and see how its outputs respond to different inputs.

Testbench Creation in VHDL

In order to validate the under test design (i.e., MAC), we need to add an architecture, as well as include the libraries containing various data types we are interested in manipulating (std_logic, std_logic_vector, integer, ...). Since the test vectors and expected outputs are usually generated in other programming languages such as C/C++, we also need **textio** package that allows the reading and writing of text files from VHDL.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use STD.textio.all;
use ieee.std_logic_textio.all;

entity gNN_MAC_tb is
end gNN_MAC_tb;

architecture test of gNN_MAC_tb is
begin
end architecture test;
```

Testbench Creation in VHDL - Declaration

In order to verify the under test component (i.e., the MAC unit), we must have access to interact with it. Therefore, the next step is to declare the under test design, instantiate from it, and wire it into the testbench. Wiring the MAC into the testbench requires information about its entity. More precisely, you need to create a signal for every port in the MAC entity as well as every I/O file. Note that it is necessary to guarantee that the testbench is in sync with what the MAC expects as inputs/outputs.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_textio.all;

entity gNN_MAC_tb is
end gNN_MAC_tb;

architecture test of gNN_MAC_tb is
```

Testbench Creation in VHDL - Declaration

```
--------------------------------------------------------------------------------
-- Declare the Component Under Test
--------------------------------------------------------------------------------

component gNN_MAC is
  port (x      : in   std_logic_vector(?? downto 0);
        y      : in   std_logic_vector(?? downto 0);
        N      : out  std_logic_vector(9  downto 0);
        clk    : in  std_logic;
        rst    : in  std_logic;
        mac    : in  std_logic_vector (?? downto 0);
        ready  : in  std_logic);
end component gNN_MAC;

--------------------------------------------------------------------------------
-- Testbench Internal Signals
--------------------------------------------------------------------------------
file file_VECTORS_X : text;
file file_VECTORS_Y : text;
file file_RESULTS : text;

constant clk_PERIOD : time := 100 ns;

signal x_in      : in   std_logic_vector(?? downto 0);
signal y_in      : in   std_logic_vector(?? downto 0);
signal N_in      : out  std_logic_vector(9  downto 0);
signal clk_in    : in  std_logic;
signal rst_in    : in  std_logic;
signal mac_out   : in  std_logic_vector (?? downto 0);
signal ready_out : in  std_logic;
```

Testbench Creation in VHDL - Instantiation

In order to test a component, we must have access to it. Therefore, the next step is to instantiate the MAC unit, and wire it into the testbench. After this step, we would be able to interact with the MAC and test its functionality.

```
begin
--------------------------------------------------------------------------------
-- Instantiate MAC
--------------------------------------------------------------------------------
  gNN_MAC_INST : gNN_MAC
    port map (
      x => x_in,
      y => y_in,
      N => N_in,
      clk => clk,
      rst => rst,
      mac => mac_out,
      ready => ready_out
      );
```

Testbench Creation in VHDL - Clock Generation

In almost any testbench, a clock signal is usually required in order to synchronize stimulus signals within the testbench. One way to generate the clock signal is provided below.

```
--------------------------------------------------------------------------------
-- Clock Generation
--------------------------------------------------------------------------------
clk_generation : process
begin
    clk <= '1';
    wait for clk_PERIOD / 2;
    clk <= '0';
    wait for clk_PERIOD / 2;
end process clk_generation;
```

Testbench Creation in VHDL - Feeding Inputs

Now that the MAC unit is instantiated and wired into the testbench, we can start feeding the test vectors into the circuit through the following steps:

- Read the input test vectors from the input data files (i.e., *lab1-x-fixed-point.txt* and *lab1-y-fixed-point.txt*)
- Supply the input signals of the MAC unit with the data obtained from the files
- Write the final output values into a file to be checked with the expected values
- Reset the under test circuit and repeat the above steps for new test vectors

Testbench Creation in VHDL - Feeding Inputs

The following codes read each element of test vectors at each clock cycles and pass it to the circuit. Once the *ready* signal goes high, the final output is written into an output file.

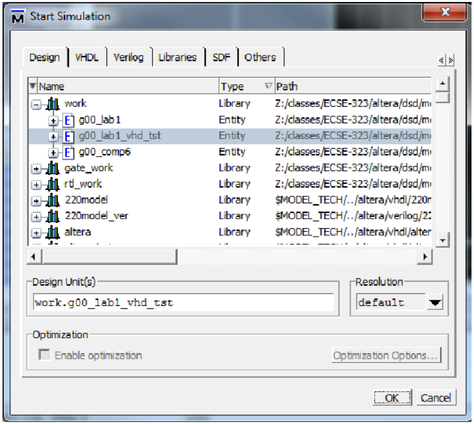
```
------------------------------------------------------------------------
-- Feeding Inputs
------------------------------------------------------------------------
feeding_instr : process is
    variable v_Iline1 : line;
    variable v_Iline2 : line;
    variable v_Oline : line;
    variable v_x_in       : in  std_logic_vector(?? downto 0);
    variable v_y_in       : in  std_logic_vector(?? downto 0);
  begin
    --reset the circuit
    N_in <= "1111101000"; -- N = 1000
    rst  <= '1';
    wait until rising_edge(clk);
    wait until rising_edge(clk);
    rst  <= '0';
```

Testbench Creation in VHDL - Feeding Inputs

```
    file_open(file_VECTORS_X, "lab1-x-fixed-point.txt", read_mode);
    file_open(file_VECTORS_Y, "lab1-y-fixed-point.txt", read_mode);
    file_open(file_RESULTS, "lab1-out.txt", write_mode);

    while not endfile(file_VECTORS_X) loop
      readline(file_VECTORS_X, v_Iline1);
      read(v_Iline1, v_x_in);
      readline(file_VECTORS_Y, v_Iline2);
      read(v_Iline2, v_y_in);

      x_in <= v_x_in;
      y_in <= v_y_in;

      wait until rising_edge(clk);
    end loop;

    if ready_out = '1' then
      write(v_Oline, mac_out);
      writeline(file_RESULTS, v_Oline);
      wait;
    end if;
  end process;
```

Starting Simulation

Now everything is ready for you to actually run a simulation! Select "Start Simulation" from the Simulate toolbar item in the ModelSim program. The following window will popup:



Start Simulation

Design | VHDL | Verilog | Libraries | SDF | Others

Name	Type	Path
work	Library	Z:/classes/ECSE-323/altera/dsd/m
g00_lab1	Entity	Z:/classes/ECSE-323/altera/dsd/m
g00_lab1_vhd_tst	Entity	Z:/classes/ECSE-323/altera/dsd/m
g00_comp6	Entity	Z:/classes/ECSE-323/altera/dsd/m
gate_work	Library	Z:/classes/ECSE-323/altera/dsd/m
rtl_work	Library	Z:/classes/ECSE-323/altera/dsd/m
220model	Library	$MODEL_TECH/../altera/vhdl/220r
220model_ver	Library	$MODEL_TECH/../altera/verilog/22
altera	Library	$MODEL_TECH/../altera/vhdl/alter

Design Unit(s): work.g00_lab1_vhd_tst

Resolution: default

Optimization

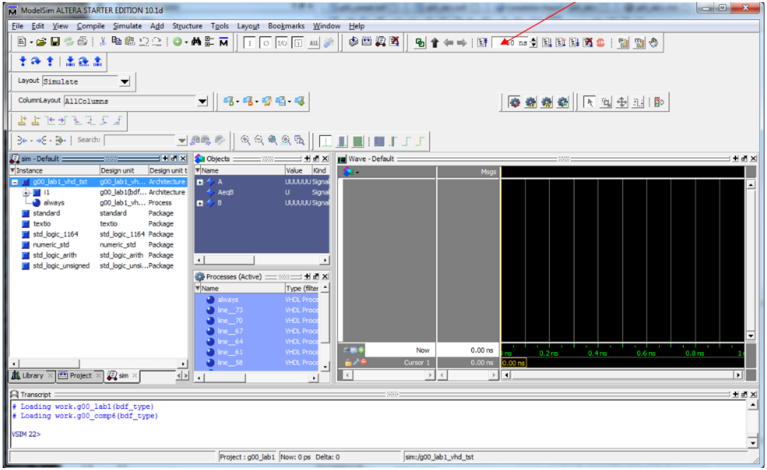
Enable optimization

Optimization Options...

OK | Cancel

Starting Simulation

The ModelSim window should now look like this. Enter a value of 60ns into the simulation length window.



Instance	Design unit	Design unit t
g00_lab1_vhd_tst	g00_lab1_vh...	Architecture
i1	g00_lab1(bdf...	Architecture
always	g00_lab1_vh...	Process
standard	standard	Package
textio	textio	Package
std_logic_1164	std_logic_1164	Package
numeric_std	numeric_std	Package
std_logic_arith	std_logic_arith	Package
std_logic_unsigned	std_logic_unsi...	Package

Name	Value	Kind
A	UUUUUU	Signa
AeqB	U	Signa
B	UUUUUU	Signa

Name	Type (filter
always	VHDL Proce
line__73	VHDL Proce
line__70	VHDL Proce
line__67	VHDL Proce
line__64	VHDL Proce
line__61	VHDL Proce
line__58	VHDL Proce

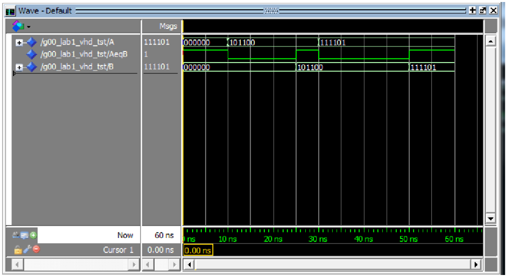
```
Now     0.00 ns
Cursor 1     0.00 ns
ns   0.2 ns   0.4 ns   0.6 ns   0.8 ns   1
0.00 ns
```

```
# Loading work.g00_lab1(bdf_type)
# Loading work.g00_comp6(bdf_type)

VSIM 22>
```

```
Project : g00_lab1   Now: 0 ps   Delta: 0   sim:/g00_lab1_vhd_tst
```

Simulation

At first, the "Wave" window will not have any signals in it. You can drag signals from the "Objects" window by click on a signal, holding down the mouse button, and dragging the signal over to the Wave window. Do this for all three signals. Now, to actually run the simulation, click on the "Run" icon in the toolbar (or press the F9 key). Below is an example output (you can right-click in the right-hand pane and select "Zoom Full" to see the entire time range).



Exercise

So far, we have optimized the MAC unit and tested it for 1000 input instances. Now, we want to optimize the circuit for each 200-input set of the given test vector by spiting it into 5 batches (each set contains 200 instances). Follow the steps below to customize the circuit for the new test vectors:

- Split the 1000 input data into 5 batches, each containing 200 values
- Convert the values of each batch into the fixed-point representation with the minimum number of bits
- Create new test bench files for each batch (e.g. lab1-x-fixed-point-batch1.txt,lab1-y-fixed-point-batch1.txt)
- Customize the MAC unit for the new test vectors if it is necessary
- Write a testbench and verify the MAC unit with the new test vectors

Writeup the Lab Report

You are required to submit a written report and your code to my-Courses on the first Friday after the Lab2 period at midnight.

- The report in the electronic file should be in PDF format.
- The report should be written in the standard technical report format.
- Document every design choice clearly.
- Everything should be organized for the grader to easily reproduce your results by running your code through the tools.
- The code should be well-documented and easy to read.
- The grader should not have to struggle to understand your design.

Writeup of the Lab Report - cont'd

Please refer to the example lab report from myCourses content for an example lab report template.

Your report must include:

- An introduction in which the objective of the lab is discussed
- The VHDL code you wrote for the MAC unit, with explanation and comments
- Testbench VHDL code for the MAC unit, with explanation and comments
- Discussion on the resource utilization of your design (i.e. how many registers are used and why? What changes would you respect in the resource utilization if you increased the bit size of the counter?)
- Using provided vector set, screenshots of the testbench for verification of the MAC unit

Grading Sheet

Group Number:
Name 1:
Name 2:

Task	Grade	/Total	Comments
VHDL for MAC		/30	
Testbench VHDL		/25	
Resource Utilization		/10	
Design Verification		/35	