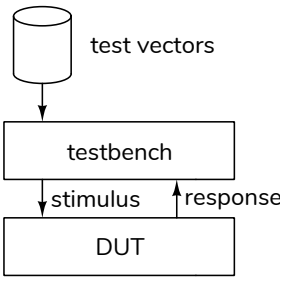
Simulation

Introduction

In this lab you will simulate the operation of a filter circuit that outputs a weighted average of the most recent inputs. This is called a Finite Impulse Response (FIR) filter and is useful for separating the different frequency components of signals.

Testbenches

A "testbench" is HDL code that applies inputs to the design being tested (often called the "device under test" or DUT) and checks that the outputs are correct:



The values of the inputs and expected outputs are called "test vectors." Test vectors are often read from a file generated by other software. In this lab you will be supplied with text files containing the test vectors.

Writing a testbench requires HDL language features and programming skills that are beyond the scope of this course so for this lab you will be supplied with a self-checking testbench and asked to use it to test your design.

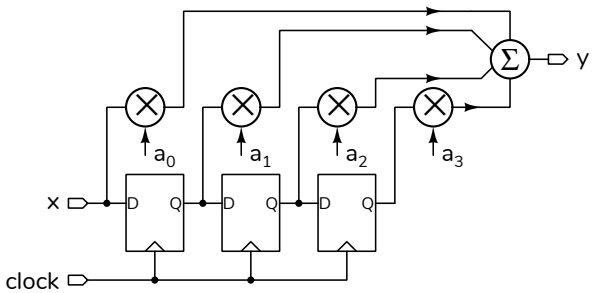
Procedure

You will use the version of the Modelsim simulator supplied by Intel along with their PLD synthesis software ("ModelSim-Intel® FPGA Starter Edition Software"). See the document and videos on the course web site for instructions on installing and using it.

Create a project directory (folder) for the simulation. Download the **`lab9_tb.sv`** testbench and the **`lab9testdata.zip`** file containing the test vector files from the course web site to this folder.

Design the FIR Filter

An FIR filter can be implemented as a shift register, multipliers and an adder:



You will write a Verilog module defined as:

```
// lab9.sv
// FIR filter
// your name & date here

module lab9
  ( input  logic [7:0] x,
    output logic [9:0] y,
    input  logic clk ) ;

   // your code here

endmodule ;
```

and save it to a file named **`lab9.sv`**. The module should compute one value of **y** as the sum of the current input **x** and the three previous inputs, weighted (multiplied by) four fixed values: $a_0 = a_3 = 1$ and $a_1 = a_2 = 2$.

Follow the course coding guidelines, including adding a comment at the beginning of the file with your name and the date.

As a hint, the following expression can implement the shift register:

```
always_ff @(posedge clk) {x1,x2,x3} <= {x,x1,x2} ;
```

Select the Correct Test Vector File

For this circuit the test vectors are pairs of values of the input, **x**, and the expected output, **y**. The **y** value is the weighted sum of **x** and the three previous values of **x** (labelled **x1**, **x2** and **x3** in the diagram above).

The supplied testbench reads the test vectors from a text file, and applies the **x** value before checking the

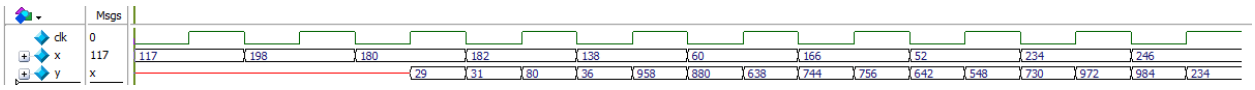


Figure 1: Sample Wave display.

y value and generating a rising clock edge. Download the **lab9testdata.zip** file from the course web site and extract the file **testdata***n***.txt**. where *n* is the last digit of *your* BCIT ID to your project directory. Change the last digit in the file name in **lab9_tb.sv** to match the last digit of your BCIT ID.

Run the Simulation

Follow the procedure in Software Installation and Use document and the video on the course web site to create a simulation project, add your **lab9.sv** and the edited **lab9_tb.sv** files to the project and compile them. After fixing any errors, run the simulation. The Transcript window should show the messages generated by the testbench and the Wave window should show the signal waveforms.

An example of the waveforms is shown in Figure 1. To show the values in decimal, right-click on a trace and set Radix to Ufixed.

Submit Results

Submit a PDF file to the appropriate Assignment folder that includes the following:

- a listing of your **lab9.sv** file
- a screen capture of the waveforms similar to that shown above
- a screen capture of the Transcript window showing the messages generated by running the simulation. These will resemble those shown below:

```
run -all
# FIR filter state= 182 180 198 117
# test passed: input x=182 and output y=31
# FIR filter state= 138 182 180 198
# test passed: input x=138 and output y=36
# FIR filter state=  60 138 182 180
# test passed: input x=60 and output y=880
# FIR filter state= 166  60 138 182
# test passed: input x=166 and output y=744
# FIR filter state=  52 166  60 138
# test passed: input x=52 and output y=642
# FIR filter state= 234  52 166  60
# test passed: input x=234 and output y=730
# FIR filter state= 246 234  52 166
# test passed: input x=246 and output y=984
# passed           7 and failed           0 test vectors
```