

## jSystemC & Behavior Coding

### Assignment 8, 2024-12-26

#### Abstract

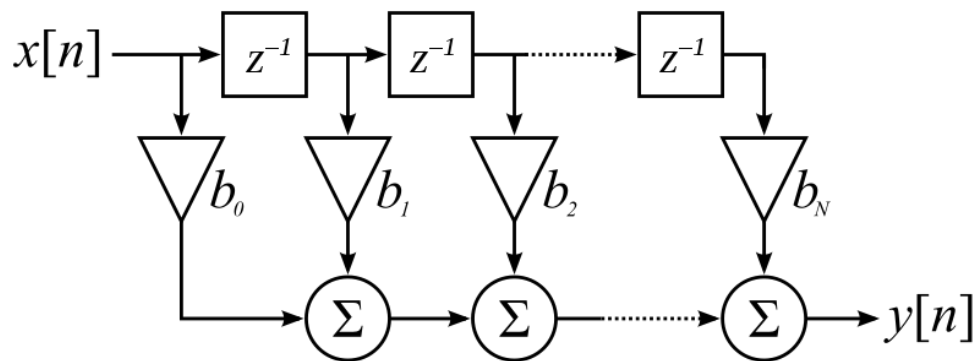
Implement an any-order discrete-time Finite Impulse Response (FIR) digital filter, in 200MHz clock.

Please read carefully. All outputs required are described in the text. Five (5) points will be taken for each bug, missing required output and behavior.

#### The any-order FIR module with a SC\_CTHREAD process

##### Description

1. A schematic of  $N^{\text{th}}$ -order discrete-time FIR filter is given below



2. Use the above schematic as the specification and implement a non-pipeline any-order FIR filter in SC\_MODULE with a SC\_CTHREAD process, for which the module must be named FIR. You also must name the SystemC files FIR.h and FIR.cpp. The default constructor defines a 16-tap FIR. The above restrictions are to make it easier to compile your code using my makefile. **Failure to do so will be penalized with 5 points.**
3. The input port is named `x` and its data type is `sc_uint<32>`.
4. The output port is named `y` and its data type is `sc_uint<32>`.
5. The positive triggering clock port is named `clk`. The synchronous active-low reset pin is named `rst`.
6. Let us do a *Moving Average Filter*, a.k.a. *boxcar filter*, where all  $b_i = 1/(N + 1)$ . For example, for  $N$  is 16,  $b_i = 1/17 \approx 0.05882353$ .
7. Let us use a fixed-point system of `wl=32` and `iwl=16` for all

computations. Then above  $b_i = 0x00000F0F$ . However, this is to explain how you are going to compute the  $b_i$  value in the module. Notice that in your code, you need to compute  $b_i$  based on the tap value using the above-mentioned fixed-point system. Then, you must transform the fixed-point  $b_i$  value computed into a `sc_uint<32>` variable. Below is the example code snippet that you **can use in the reset behavior** to generate the 32-bit fixed-point  $b_i$  value:

```
sc_fixed<32, 16> fb;
sc_uint<32> b;           // the  $b_i$ 
// tap value is taken from SC_CTOR
const unsigned int tap;
// Use sc_fixed to get the fixed-point value
fb = (float)1/(tap+1);
// Copy the computed fb value bit-by-bit to b
for (int i = 0; i < 32; i++) // copy bit-by-bit
    b[i] = fb[i];
```

8. The reset behavior is to reset all delays to 0 and  $b_i$  generation and setting.

## **sc\_main**

### Description

1. Create a test suite, i.e., `sc_main`, and you must name the file `main.cpp`, that
  - Instantiate an FIR module with 32 taps and name it `fir32`.
  - Instantiate another FIR module with 48 taps and name it `fir48`.
  - Give a 200MHz clock into both `fir32` and `fir48`.
  - Read 64 input data, one by one, and feed them into the `x` port of `fir32` and `fir48`, from a file named “`firData`.” Again, you **must** name the file exactly as specified above to avoid a 5-point penalty.
2. Create a trace file named `RESULT.vcd`. And trace ports are shown in the following order:
  - ▶ `clk`
  - ▶ `rst`
  - ▶ `x`
  - ▶ `y32`
  - ▶ `y48`

## **makefile**

### Description

A `makefile` must be provided, with proper modifications to your environment.

## **Using Generative AI**

It is encouraged to use Generative AI (GAI) to solve the problem as in earlier assignments. Having said that, using GAI is only a practice and not necessary for this assignment and will not be counted into the grading. If you use GAI to solve the problem, please compose a prompt to ask the GAI to create another 16 taps `FIR16` module and name it `FIR16.h` and `FIR16.cpp` and add it into the original `sc_main()` i.e., the test bench, to test the correctness of the `FIR16` module. This `FIR16` module is a `SC_CTHREAD` one that uses a negative-triggered `clock` with an asynchronous active-low `reset`. Turn in the prompt you composed and `FIR16.h` and `FIR16.cpp` generated. Note that the `makefile` must also include the compilation of this new `FIR16` module.

**Please** turn in all the `FIR` source codes and `main.cpp` described in the **sc\_main** and **Use Generative AI** sections only and the `makefile`. Do not turn in the executable and waveforms.

## **Due date**

3:00 PM, January 2<sup>nd</sup>, 2025

**Score weight** (towards the final grade) 10%