National Tsing Hua University
Department of Electrical Engineering
EE429200 IC Design Laboratory, Fall 2021

Lab 06-2: Coding for synthesis and optimization
Assigned on Oct 21, 2021
Due day in Oct 28, 2021

Objective

In this lab, you will learn:

1. Digital circuit implementation for digital signal processing (DSP).
2. Compare different design.

Demo checklist:

- ☐ Show simulation results and performance of the direct form design.
- ☐ Show simulation results and performance of the pipeline design.
- ☐ Answer the following questions
    1. Where is the critical path of direct form design?
    2. How to shorten the critical path of direct form design?
    3. Compare the performance of direct form and pipeline design.

Environment Setup

Copy lab file packages from ee4292. Decompress the package and enter it. You can check the file list in Appendix.

```
$ cp ~ee4292/iclab2021/lab06.zip .
$ unzip lab06.zip
$ cd lab06/lab06_part2
```

Note: please run simulation in the ***sim*** directory to maintain a fine data management.

Description

Measuring Electrocardiography (ECG) is important in medical applications, however the measurement usually suffers from noises, especially 60Hz noise in our power system as figure 1. Thus the signal processing for measured data is inevitable. In this lab, we will implement a FIR filter in digital circuit, explore different architectures and analyze their performance (ex. Power, Area, Latency, Frequency). The origin ECG data was sampled at 100Hz, so the 60Hz noise from electric power system will represent around 40~50Hz. After the signal analysis, a 16-tap finite impulse response (FIR) digital filter for ECG application has been designed to have behavior plotted below. (The ECG data is provided by MIT-BIH physionet database)

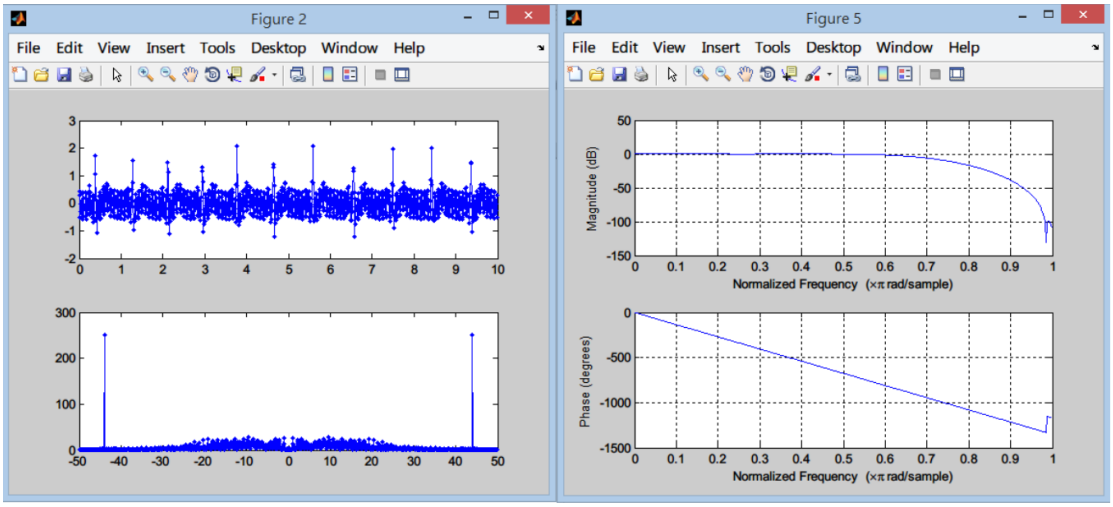

Figure 1. ECG signal.

The filtering process on the ECG signal can be expressed as following equation:

$$y[n] = \sum_{i=0}^{15} x[n-i] * a_i$$

a is the coefficient of the filter (refer to table 1).

The block diagram implementation of this filter is shown in figure 2, this is the most straightforward implementation, directly from the equation (**direct form design**).

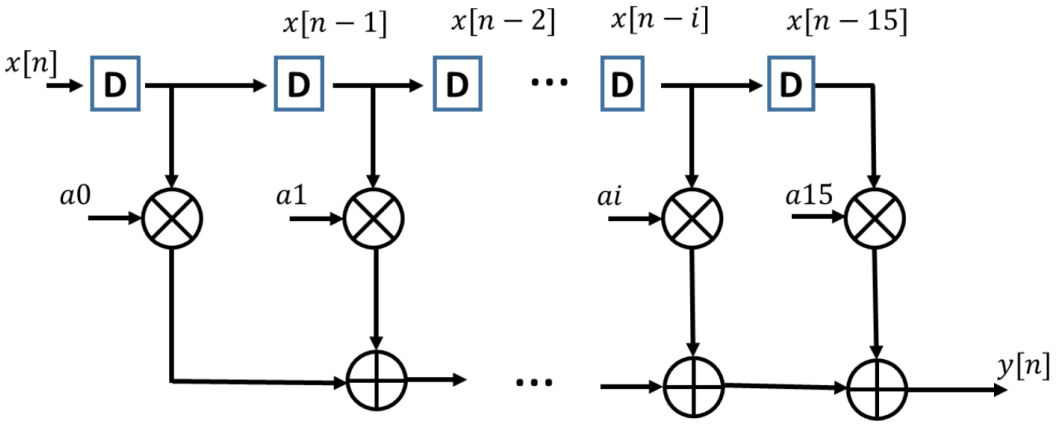

Figure 2. Direct form design.

Term	Value	Fraction	Term	Value	Fraction
a_0	-0.0024	-157/65536	a_8	0.05624	36857/65536
a_1	0.058	380/65536	a_9	-0.0303	-1987/65536

a_2	-0.0061	-399/65536	a_{10}	-0.0694	-4548/65536
a_3	-0.0128	-838/65536	a_{11}	0.0529	3466/65536
a_4	0.0529	3466/65536	a_{12}	-0.0128	-838/65536
a_5	-0.0694	-4548/65536	a_{13}	-0.0061	-399/65536
a_6	-0.0303	-1987/65536	a_{14}	0.058	380/65536
a_7	0.05624	36857/65536	a_{15}	-0.0024	-157/65536

Action Items

I. Hardware Implementation.

Implement the filter in RTL. The circuit should have I/O as in Figure 3. To simplify the design, the finite state machine is provided to decide whether the output is valid. You only need to finish the computation part.

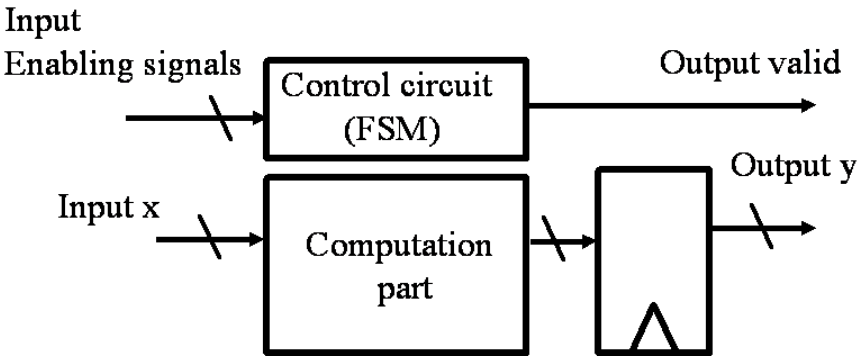

Figure 3. Function architecture.

All of the registers in the circuit will first be reset by ***rst_n***. Then it will start to receive input x after the enable signal is set to high. Since this is a 16-tap filter, the first output will delay 15 cycles from the first input data received (you need to collect 16 data for computation). The timing diagram is shown in Figure 4. Again, since the control part is provided, you only need to finish the computation part, and ensure that the output arrives with valid signal.

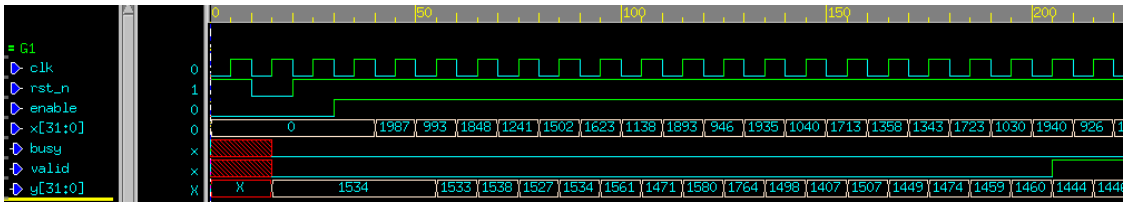

Figure 4. Waveform of function.

II. Function Verification.

The testbench is provided in ***sim/***. Verify your circuit with the testbench. Besides, for your convenience a Makefile is provided in ***sim/***. The Makefile can simplify the process of simulation and file managements. You can use Makefile with following commands in ***sim/***. (or you can do this lab with commands used in previous labs.)

III. Hardware Synthesis.

1. After Function verification, you need to synthesize the circuit, and record the timing, area, and power. The circuit will be evaluated with the performance index defined as below.

$$\text{Performance} = \frac{1}{A * T * C}$$

A: Area

T: Test cycles

C: Average cycle counts for one output

2. Record your synthesis result

Area (um^2)	Timing (ns)	Cycles (#)	Performance

IV. Implemented with pipeline design.

1. Finish three stage pipeline architecture as figure 5.
2. Pipeline structure can cut the critical path, but suffers from additional registers and latency. To reach better performance, the path between each pipe, should be as balance as possible.
3. Since the latency is larger (first valid output delay more cycles), you need to modify the state machine to ensure that behavior of the valid signal is correct.
4. Record your synthesis result

Area (um^2)	Timing (ns)	Cycles (#)	Performance

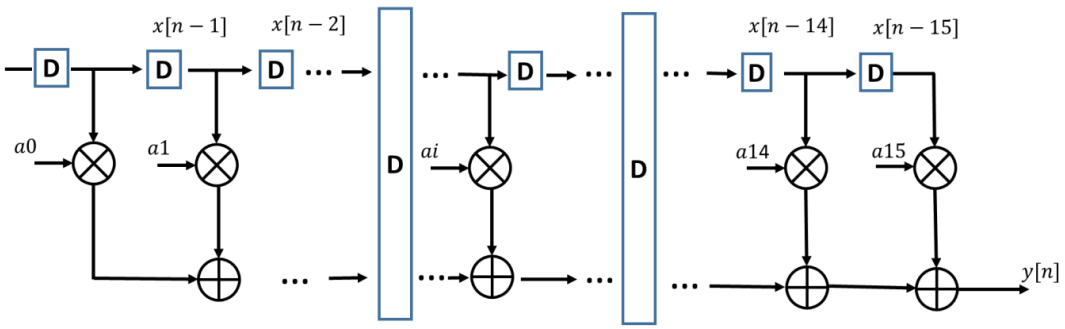

Figure 5. Pipeline architecture.

Appendix

Directory	Filename	Description
hdl	fir1.v	Your design here
hdl	fir1.f	Filelist
sim	fir1_test.	Testbench
sim	ecg.csv	Test patter
sim	data_truth.csv	Golden response
syn	.synopsys_dc.setup	Synthesis setup file
syn	0_readfile.tcl	Synthesis scripts
syn	1_setting.tcl	Synthesis scripts
syn	2_compile.tcl	Synthesis scripts
syn	3_report.tcl	Synthesis scripts
syn	synthesis.tcl	Synthesis scripts
syn	run_dc.bat	Synthesis command