UNIVERSITY OF CALIFORNIA
College of Engineering
Department of Electrical Engineering
and Computer Sciences

Jan Rabaey

EECS 141
Spring 2008

FIR Filter Project – Part 1

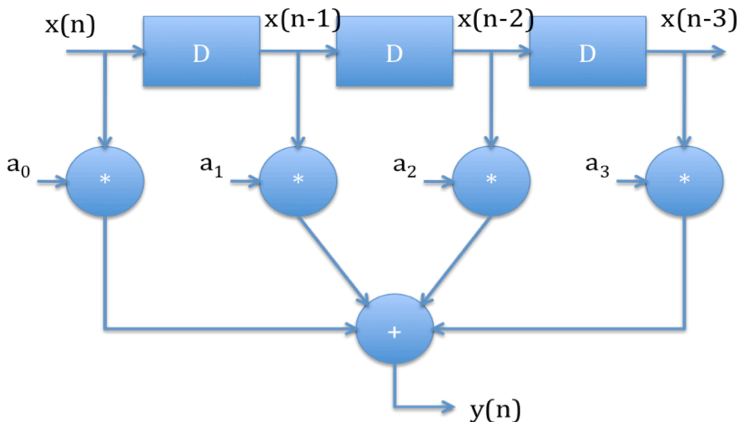
Due We, April 2nd, 5pm @ 240 Cory

Introduction

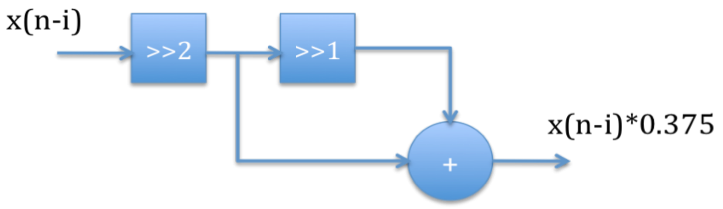
Filters play an important role in many communications and signal processing applications. Very often, they are implemented in software on DSP's or application-specific processors. In this project, we explore area and energy efficient implementations of a Finite Impulse Response (FIR) filter. An N-tap Finite Impulse Response Filter implements the following function:

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i)$$

A block diagram picturing how such a function could be implemented is shown in the Figure below.



The D-elements represent delay elements and are implemented as registers. From the diagram, it seems like the implementation of an FIR filter requires the availability of adders, multipliers and registers. However, when the coefficients a_i are fixed and known in advance, the multipliers can be eliminated, and replaced by a number of add-shift units. For instance, x(n-i)* 0.375 can be realized as x(n-i)*2^{-2}+x(n-i)*2^{-3}. Multiplying by a power of two is nothing more than a shift. Add-shift functions are easy to implement, and a lot cheaper than complete multipliers. One possible realization of x(n-i)* 0.375 is shown in the diagram below.

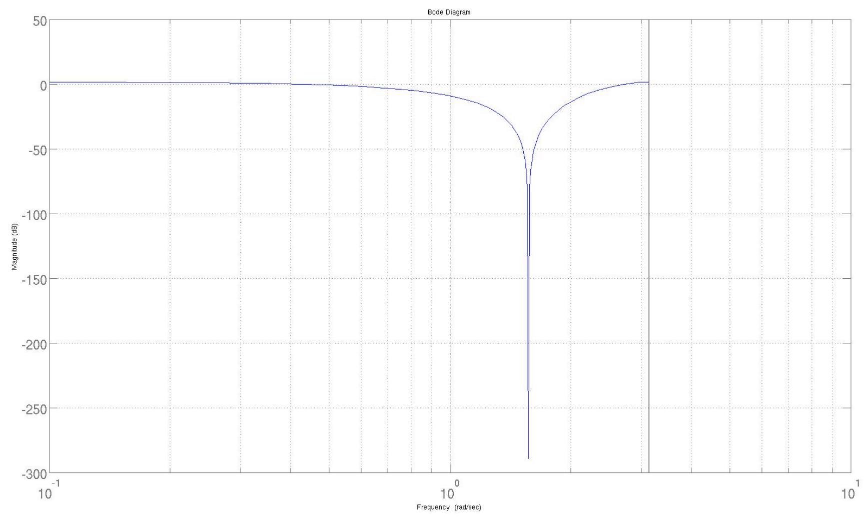


This is getting even better when one realizes that shifting over a fixed number can be implemented as simple wiring.

In this project, we will realize a simple bandstop filter, represented by the following expression:

$$y(n) = x(n) * a_0 + x(n-1) * a_1 + x(n-2) * a_2 + x(n-3) * a_3 + x(n-4) * a_4$$

with: $a_1 = a_3 = 0$; $a_0 = a_4 =$ `0.296875` ; $a_2 =$ `0.59375`;
For your interest, the frequency response of the filter is plotted below.



PROJECT DESCRIPTION

The first task is to find a partner – the project will be executed in **groups of 2**. Smaller and larger groups (each of which is discouraged) need special permission from Prof. Rabaey.

The project will be executed in three phases.

- Phase 1: Analysis of predefined structure and topology (March 14 – April 2)
- Phase 2: High level optimization (April 2- April 16)
- Phase 3: Your ultimate design (April 16 – May 7)

For phase 1 and 2, the outcome will be a simple report, while Phase 3 culminates in the creation of a poster and a short interview, where you will have the opportunity to describe how brilliant your design is.

PHASE 1:

This phase will get you familiarized with the design of large arithmetic functions. A circuit implementation of the full adder and register cells are provided. It is your task to put these together in a straightforward implementation of the FIR filter. No optimization is required at this time – focus is on analysis. **We assume that the x and y signals (and all intermediate results) are represented in 8-bit format.**

More specifically, you are asked to perform the following tasks:

a. Determine the timing characteristics of the full adder and registers cells; that is, the delay between inputs and outputs as a function of load.
b. Derive a block diagram of the FIR filter using only the provided cells. Prove that the obtained module indeed implements the FIR filter functionality by simulating the impulse response of the filter. (Note: the impulse response of a system is obtained by applying an impulse at time 0, followed by subsequent zeroes.). Determine the value of the input so that no overflow in any of the adders occurs.
c. Determine the critical timing path of your design, and identify the value of the input vectors that cause this critical path to be triggered. Using the results of part a., estimate the delay of that critical path.
d. Simulate the delay of the critical path, using SPECTRE. Explain any difference you observe with respect to the manual analysis.
e. Estimate the power dissipation of the filter. To do so, a prescribed input sequence must be applied. The exact sequence will be provided on the website.

LIBRARY

A full adder cell and a master-slave D-type flip-flop are provided. Both located in the /home/ff/ee141/project/s08 directory. In order to access these two cells, you must edit your *cds.lib* file. This file is found in the directory where you run '*icfb2*'. Before starting Cadence, add the following line to your *cds.lib* file using any text editor:

```
DEFINE ee141s08_project
/home/ff/ee141/project/s08/ee141s08_project
```

Save your *cds.lib* file and then run Cadence. You should be able to see two read-only cells (*MSFF* and *full_adder*) that you can instantiate and simulate.

If you wish to edit these cells (later in the project – NOT NOW), you can copy them to any of your libraries or you can also create a new library and copy these cells there. Use the *Library Manager* to create libraries or to copy cells from one library to another.

Report

Please use the report template provided at the website. Be sure to justify important design decisions and emphasize all the vital information. Organization, conciseness, and completeness are of paramount importance. Good reports are short and to the point. Make sure to include and annotate important plots to illustrate your effort. Do not repeat the information we already know. Make sure to fill out the cover-page and use the correct units.

Grading

The quality of the report is a major part of the grade.

For phase 1, the grade will be equally divided over the correctness and completeness of the results, and the quality of the report.

Have fun, and good luck!