Multirate Filtering: Implementation on TI TMS320C54x*

Douglas L. Jones
Swaroop Appadwedula
Matthew Berry
Mark Haun
Jake Janovetz
Michael Kramer
Dima Moussa
Daniel Sachs
Brian Wade

Abstract

You will implement a multirate system that includes three fininte impulse response filters. The sample-rate compression and expansion factors can be controlled in real time using a MATLAB graphical user interface.

1 Implementation

Before implementing the entire system shown in Multirate Processing: Introduction[1], we recommend you design a system that consists of a cascade of filters FIR 1 and FIR 2 without the sample-rate compressor or expander. After verifying that the response of your two-filter system is correct, proceed to implement the complete multirate system and verify its total response. At first, use fixed compression and expansion factors of $D = U = 4$. Later, you control this factor using a MATLAB interface; be sure to keep this in mind as you write your code.

1.1 Compressed-rate processing

In order to perform the processing at the lower sample rate, implement a counter in your code. Your counter will determine when the compressed-rate processing is to occur, and it can also be used to determine when to insert zeros into FIR 3 to implement the sample-rate expander.

*Version 2.9: Feb 25, 2004 12:51 pm +0000
†http://creativecommons.org/licenses/by/1.0
[1]"Multirate Filtering: Introduction", Figure 1 <http://cnx.org/content/m10024/latest/#fig1>

Some instructions that may be useful for implementing your multirate structure are the `addm` (add to memory) and `bc` (branch conditional) instructions. You may also find the `banz` (branch on auxiliary register not zero) and the `b` (branch) instruction useful.

1.2 Real-time rate change and MATLAB interface

A simple graphical user interface (GUI) is available (as mrategui.m[2] , which requires ser_snd.m[3]) that sends a number between 1 and 10 to the DSP via the serial port. This can be used to change the compression and expansion factor in real time.

Run the GUI by typing `mrategui` at the MATLAB prompt. A figure should automatically open up with a slider on it; adjusting the slider changes the compression and expansion factor sent to the DSP.

The assembly code that you have been given stores the last number that the DSP has received from the computer in the memory location labeled `hold`. Therefore, unless you have changed the serial portion of the given code, you can find the last compression and expansion factor set by the GUI in this location. You need to modify your code so that each time a new number is received on the serial port, the compression and expansion factor is changed. If a "1" is received on the serial port, the entire system should run at the full rate; if a "10" is received, the system should discard nine samples between each sample processed at the lower rate.

Note that the `READSER` and `WRITSER` macros, which are used to read data from and send data to the serial port, overwrite `AR0`, `AR1`, `AR2`, and `AR3` registers, as well as `BK` and the condition flag `TC`. You must therefore ensure that these registers are not used by your code, or that you save and restore their values in memory before you call the `READSER` and `WRITSER` macros. This can be done using the `mvdm` and `mvmd` instructions. The serial macros set up the `AR1` and `AR3` each time they are called, so there is no need to change these registers before the macros are called.

More detail about the `READSER` and `WRITSER` macros can be found in Core File: Serial Port Communication Between MATLAB and TI TMS320C54x[4].

[2] http://cnx.org/content/m10621/latest/mrategui.m
[3] http://cnx.org/content/m10621/latest/ser_snd.m
[4] "Core File: Serial Port Communication Between MATLAB and TI TMS320C54x" <http://cnx.org/content/m10821/latest/>