ECE699 VHDL Design Project 2:
Project Specification:

"Multichannel Narrowband
FIR Filtering:
Hardware/Software Tradeoffs"

Group member: Brian Wagner
Dr. Gaj
Submitted: 4/1/02 (updated 4/9/02)

Introduction:

FIR filters are one of the most computationally complex parts of software radio algorithms. Modern software radio systems often have high-speed DSPs in addition to high-area FPGAs, both of which can perform FIR filtering operations. Therefore, I will compare a particular FIR filter between a DSP software (SW) implementation and an FPGA hardware (HW) implementation. This comparison cannot be extremely precise because of some unknown internals of the DSP and FPGA, but the best attempt to get an "apples to apples" comparison will be made.

In the specific application I'm targeting, the FPGA will be clocked at a rate significantly higher than the input signal's sampling rate, up to three orders of magnitude higher. As such, the HW FIR filter will not be optimized for throughput or latency. Its only optimization criteria will be for FPGA area, to allow a multichannel implementation to take minimal FPGA resources.

Additionally, in many other high-speed applications, throughput/latency are important optimization criteria and many people have studied this. Those applications can involve array or tree (full or partial) multiplication circuits. However, I think the application I'm studying (multichannel area minimization through, among other things, multiplexing) is less heavily studied than the case of throughput/latency minimization for modern FPGAs.

The FIR filter will be a multiplexed 8-channel implementation. That is to say, there will be 8 sets of registers (data store) for each filtering channel, but only 1 set of actual FIR filtering HW (the MAC) in the FPGA. Only filtering 1 channel will be active at any point in time. The circuit will multiplex between the 8 sets of data according to which channel the user requests filtering to occur on when inputing a new data sample.

Applications of FIR filters (not a complete list):

(1) SW: narrowband: Equalizers (adaptive FIR filters) for cell phones (or land-line modems)
(2) SW: narrowband: Filters in LPC voice codecs (typically on the encoder side)
(3) HW/SW: narrowband: RRC pulse-shaping matched filters (on transmit AND receive side)
(4) HW: Pipelined Frequency Transform (PFT)
(5) HW/SW: narrowband: Cross-correlation (for signal detection)
(6) HW/SW: wideband/narrowband: Polyphase filter banks
(7) HW: wideband: DDCs (Digital DownConverter)

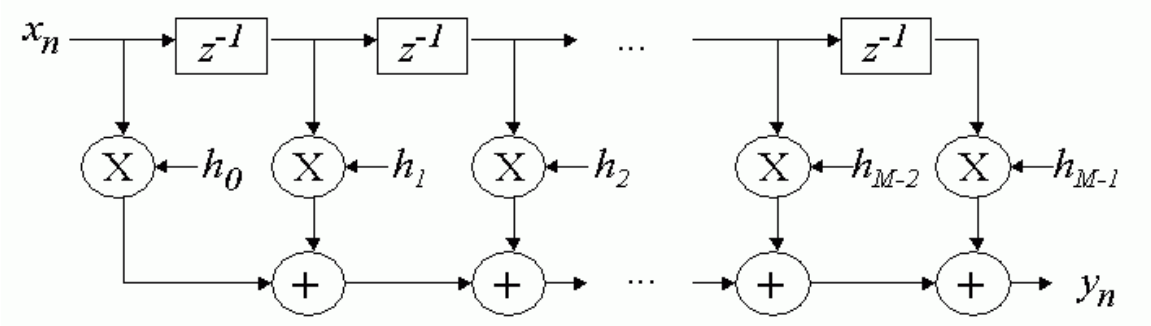
Specifics on target HW to be used:

(1) DSP: TI's TMS320C6201 at 200_MHz (VLIW-based processor); all PM and DM in the chip's internal memory space
(2) FPGA: Xilinx Virtex 1000-E FG1156 speed grade 6 (1 million equivalent gates)
-Target board: custom SenSyTech Gen-4 DSP receiver, Xilinx
F_{clk}=100_MHz; F_s<=100_kHz or F_s<=300_kHz (TBD)

Function to be implemented:

(1) $y(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$ where *y(n)* is the output of the filter at time *n*, *h* are the filter coefficients, *x* is the input array, and *M* is the length of the filter. [1]
<u>All numbers are 2's complement.</u>

(2) Filter block diagram:



(3) data sizes: 16-bit inputs => 32-bit result of multiplication => 40-bit accumulator (so 8 guard bits) => 32-bit final saturated result (This is the DSP's exact HW architecture for MACs, so it will be duplicated in the FPGA.)

(4) The data will be input 1 sample at a time to the FIR filter. Before each data input, the old data will be shifted to make one register location available for a new *x* input. Then filtering will start.

Details on the specific FIR filter to be tested:

(1) Low Pass Filter, fixed-point coefficients designed (and truncated) in MATLAB

(2) 17 coefficients

-These are constant, but reconfigurable. I will attempt to take advantage of reconfigurable constant coefficient multiplication capabilities provided by the Xilinx, but I don't know if this is easily possible.

(3) coefficients are symmetric

-The filter is, therefore, linear phase, constant group delay [1]

-Of the 17 coefficients, 9 are unique and 8 are redundant. I will attempt to take advantage of this coefficient redundancy, by adding *x* values with the same coefficients together, before multiplying them by their *h* coefficient [4].

Optimizations:

(1) HW: area ONLY (throughput isn't an issue and latency isn't very important)

-Compare with Xilinx Coregen results

(2) SW: latency ONLY (area is fixed by the chip);
optimal implementation without going to optimized/linear assembly language

Candidate algorithms:

(1) Sequential multiplication (additions and right-shifts)

(POSSIBLE: All 16 MACs going on in parellel (~16 clocks), followed by 16 final 40-bit adds (~16 clocks).)

(2) bit-serial (semisystolic, retimed semisystolic, systolic, or delay free)

(NOT POSSIBLE for systolic: All 16 MACs going on in parellel (~64 clocks), followed by 16 40-bit final adds (~2560 clocks).)

Proposed final analysis:

(1) Which takes longer (highest latency)? (HYPOTHESIS => SW)

(2) Which takes more overall silicon area? (HYPOTHESIS => HW)

(3) Which is harder to create? (HYPOTHESIS => HW)

Specifics on tools to be used (all at my workplace):

(1) DSP: TI's Code Composer Studio version 2.0 (2001)

(2) FPGA: Aldec's Active-HDL 4.2 (2001), Synopsis FPGA Express 3.6 (2001), and Xilinx ISE 4.1 (2001) tools.

HW Test Plan:

(1) Simulate filter impulse response in VHDL simulation tool

(2) Synthesize filter using VHDL synthesis tool

(3) Place-and-route using Xilinx place-and-route tools

-circuit clock speed is output by this tool

-circuit area is determined by the number of slices/CLBs used

SW Test Plan:

(1) Write, compile, and link in DSP SW development tool

(2) Test on software radio receiver hardware

-Function execution time will be measured by flipping a bit high when entering the function and flipping the bit low when exiting the function (This bit, which goes to a header on the pc-board will be read on an oscilloscope.)

-Amount of memory required by the function will be gotten from looking at the linker's output map (.map) file

---

References:

(1) Digital Signal Processing -- Principles, Algorithms, and Applications; Proakis, Manolakis; Prentice Hall

(2) Texas Instruments website (C6201 documentation); www.ti.com

(3) Xilinx website (Virtex-E documentation); www.xilinx.com

(4) "Modulation and Demodulation Techniques for FPGAs"; Andraka; Designcon 2000; http://www.andraka.com/papers.htm