A Run-Time Reconfigurable Plug-In for the Winamp MP3 Player

Jonathan Scalera and Mark Jones
The Bradley Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg VA 24061
E-mail: jscalera, mtj@vt.edu

Abstract

This paper describes a plug-in for the Winamp MP3 player that uses a configurable computer to provide digital special effects for audio streams. The plug-in makes use of run-time reconfiguration, modification of FPGA configuration bitstreams at run time, and context switching. The plug-in effects processor provides filters for a graphic equalizer and an echo effect. These filters are context-switched during the runtime of an application to reduce the amount of hardware required. Further, the effects embodied in the filters are changed on-the-fly by modifying the configuration bitstreams at runtime using the JBits API. These techniques are described in the paper and results on the performance of the application are presented.

1. Introduction

Digital Signal Processing (DSP) tasks are typically computationally intensive. These stream-oriented operations are well suited for execution on configurable computers, with orders-of-magnitude speedups over standard microprocessors common. Digital Signal Processors, such as the Texas Instruments TMS320C30, have hardware support for many common digital signal processing operations and are competitive with configurable computers. Configurable computers have an advantage because their processing elements can be configured such that the entire chip is devoted to the current operation, allowing significant parallelism and pipelining to increase throughput. The DSP chips hold an advantage because it is relatively fast and simple to develop a new program; writing new programs or modifying an existing program for a configurable computer can require significant development time. This paper investigates practical Run-Time Reconfiguration (RTR) techniques that add flexibility to configurable computers.

This paper describes the design and implementation of a digital effects processor that demonstrates the use of JBits and JHDL to build a run-time reconfigurable, context-switching application. A digital effects processor receives a digital stream that describes one or audio channels and applies various special effects to those channels. A commonly implemented special effect, for example, is an echo filter. The effects processor described in this paper acts as a Winamp [2] plug-in, a popular MP3 audio player for personal computers. The effects processor plug-in takes a raw data stream from Winamp, performs special effects processing on that data, and then returns the processed stream to Winamp for audio playback.

The digital effects processor uses context-switching techniques to allow multiple effects to share the same FPGA and to operate on the same segment of the data stream, where one effect operates in hardware in a given time slice. JHDL [1] was used to quickly implement each digital effects filter and produce a Xilinx configuration bitstream for each filter. At runtime, users can modify aspects of the filters; changes to the filters are made by modifying the configuration bitstreams using JBits [3]. New bitstreams can be generated in seconds rather than the hours required by the Xilinx M1 tools.

2. System and Experiments

The implementation of this application is on an Annapolis Micro Systems WildForce board. This board has five Xilinx 4062XL FPGAs, where each FPGA is connected to a 1MB SRAM module. In this digital effects processor, each filter is a separate module. Each filter module reads the audio stream from a specified block of memory in SRAM, processes the data, writes it back to memory, and then raises an interrupt to signal that it is finished.

Because the audio data from the Winamp program is limited to only two audio channels, the effects processor is designed to process two 44.1 KHz channels. In this design, each channel is processed in a separate FPGA, but all filters for a single channel reside in the same FPGA. The current processor has two filter designs, a graphic equalizer based

on a 24th order FIR filter and a wet/dry mix echo filter.

The FIR filter processes 16 bit samples from Winamp, but experiments in Matlab led us to use 11 bits as the length of each filter coefficient. With symmetry, the filter design was implemented using twelve 11x16 multipliers, leading to a 95% utilization of the 4062XL. The wet/dry mix echo filter provides the echo effects that are commonly used in studios. The filter in this design has an adjustable delay as well as adjustable "wet" and "dry" gain. The echo filter design is substantially smaller than the graphic equalizer, largely because the delay blocks are implemented by using the SRAM memory rather than the CLBs. The echo effect used 12% of the 4062XL.

One of the goals of this project was to show that a design could be modified on-the-fly in a real application. In this case, the desire was to modify the filter designs at the user’s request during runtime. This modification required generating new configuration bitstreams for the FPGAs at runtime. Accomplishing this required the ability to bypass the Xilinx M1 tools with their map, place, and route steps that may take hours to run (clearly not a run-time response). To accomplish this objective, the straightforward design methodology in JHDL was used to create the basic filter designs, and then JBits was used to modify the resulting bitstreams.

Like JHDL, JBits is built within the Java language. It can be viewed as an API that provides low-level control over the configuration bitstream of a Xilinx FPGA [5]. The JBits programmer has access to all other aspects of Java, but only JBits API calls modify the FPGA configuration bitstream. The programmer can directly manipulate specific CLB configurations, including look-up table (LUT) settings, as well as specify low-level routing on the FPGA. Essentially, the API provides the programmer with access to all of the configuration bits on the FPGA.

In this project, the digital effects filters were designed to be modified at runtime. The modifications have been limited to changing the constant coefficients in the filters, for example, the coefficients corresponding the graphic equalizer sliders or the delay in the echo filter. The basic methodology used for this modification is to create an initial filter design in JHDL, create the configuration bitstream, and then modify the filter parameters in the configuration bitstream at runtime.

Before modifying the filter parameters, they had to be located in the configuration bitstream. This process can be done offline and needs to be done only once, after the configuration bitstream has been created by the M1 tools. Because the M1 tools perform several optimizations on the design, it is not possible to know where particular filter parameters are located. By using the JHDL map and place functions in the initial design, however, it was possible to determine the relative placement of the parameters with respect to one another. To further assist in locating the parameters, each parameter was assigned a specific bit-pattern. It was then simply a matter of using JBits to systematically search the CLBs for this bit-pattern. Once the first coefficient was located, their relative placement allowed the location of all of the coefficients to be determined.

A Java tool was created that would take as arguments the location of the constants and their desired values. This tool reads in a bitstream, modifies the CLBs to give the new constant values, and writes out the new bitstream. The only time-consuming aspect of this program is the file I/O time that could be eliminated by keeping the bitstreams in memory.

The overall application design is based on a plug-in written for Winamp. The plug-in accepts the data to the WildForce board, receives the processed data, and returns the audio stream to Winamp for playback. The plug-in is responsible for controlling which effects are downloaded on the board as well as changing the parameters in those effects at runtime. Finally, the audio stream must playback in real time, operating just as any other Winamp plug-in. The plug-in was instrumented for timing.

As expected, the download times for the configuration bitstreams dominate the runtime; the time to download one effect for each of the two channels is approximately 400 ms. The time for two effects on two channels rises to 800 ms. Reading (or writing) two 64K sample buffers requires approximately 8 ms.

The time for the digital effects processing depends on the number and type of effects as well as the clock rate. The clock rate is governed by the FIR filter in the graphic equalizer, which has a maximum clock rate of 12.8 Mhz. The graphic equalizer has a throughput of two clock cycles for every sample in the buffer. The total processing time for a single buffer of 64K is 10.24 ms. The delay filter requires 4 clock cycles per sample, leading to a total processing time of 20.48 ms for a single buffer ms. Thus, the total processing time for both effects on two channels simultaneously is 30.72 ms.

References