XAPP547 (v1.0.1) November 28, 2006

PowerPC Processor with Floating Point Unit for Virtex-4 FX Devices

Authors: Gaurav Gupta, Ben Jones, and Glenn C. Steiner

Summary

This application note describes how to implement a Virtex™-4 FX PowerPC™ 405 system with the Xilinx floating point unit (FPU) coprocessor. An FPU connected to the PowerPC auxiliary processor unit (APU) interface can accelerate software applications from anywhere between three to twenty times. The reference design provided includes a completed design created using the Xilinx Embedded Development Kit (EDK). Source code for a finite impulse response (FIR) filter is provided along with a graphics utility for display output on a Windows-based PC.

Introduction

Floating-point intensive algorithms are frequently required for embedded systems and DSP applications such as image processing, digital pre-distortion, and audio. Software emulation of the floating point instructions is often too slow for many system requirements and dedicated, tightly-coupled floating-point circuitry can provide the needed performance.

The FPU for the Virtex-4 PowerPC 405 processor is a single precision IEEE-754-compliant (with minor and documented exceptions) peripheral that accelerates floating-point code execution by up to 20 times. The FPU is tightly coupled to the PowerPC processor through the auxiliary processor unit (APU) controller and fully supported by the Xilinx GNU compiler to ensure hardware abstraction and ease of use. For more details on the FPU, see [Ref 1].

As shown in Figure 1, the reference design demonstrates how an amplitude-modulated (AM) signal *buried in the noise* can be extracted via a FIR filter. This included reference design is built using the Xilinx EDK. A step-by-step tutorial for building the design under EDK is provided in an accompanying tutorial [Ref 2].

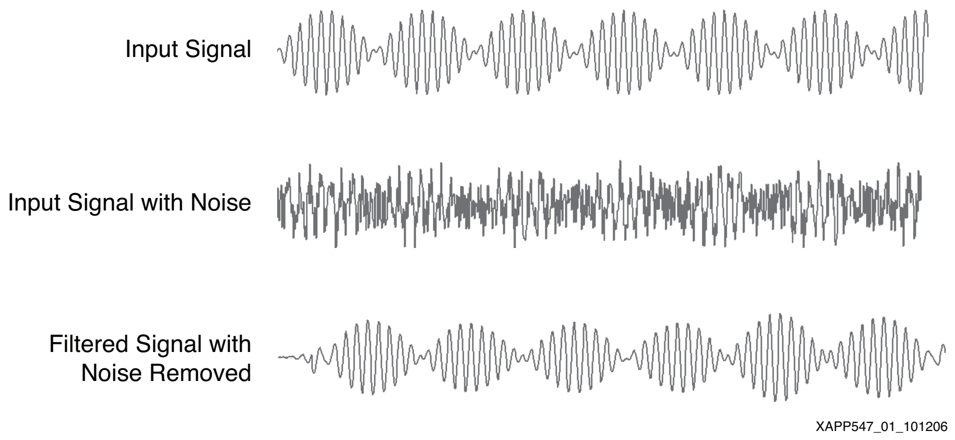

Figure 1: **Original AM Signal; AM Signal with Noise; and FIR Filtered Signal**

For this reference design (see Figure 2), the single-precision, lite version of the FPU is used. The lite version does not support floating-point division or square root. Since a FIR filter only utilizes floating-point multiplies and adds, the lite version of the FPU provides an optimal solution for performance with minimal logic utilization. When a floating-point function is not supported by a particular FPU, the operation is emulated in software. Thus, in this reference design, floating-point divide operations required to compute the performance metric are performed via software emulation.

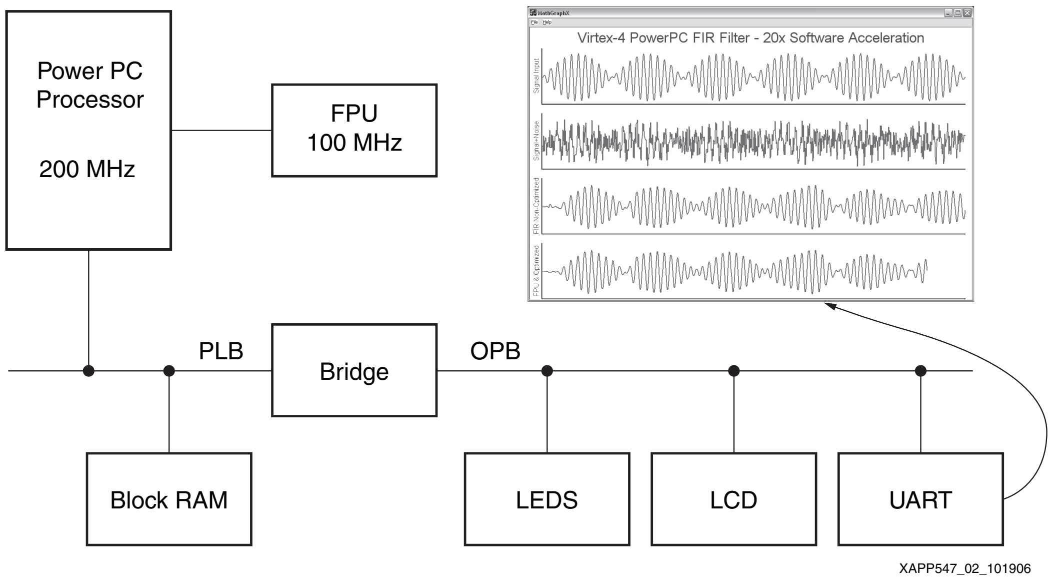

Figure 2: **PowerPC Processor with FPU Block Diagram**

Characteristics of the FIR Filter Reference Design

The FIR filter can be used to implement many forms of filtered frequency response and is typically implemented via a series of delays, multiplies, and adds. This filter is well suited for digital applications, and frequently a filter has tens to over a hundred filter stages. An overview of FIR filters can be found at [Ref 3]. Figure 3 shows the structure of the FIR filter.

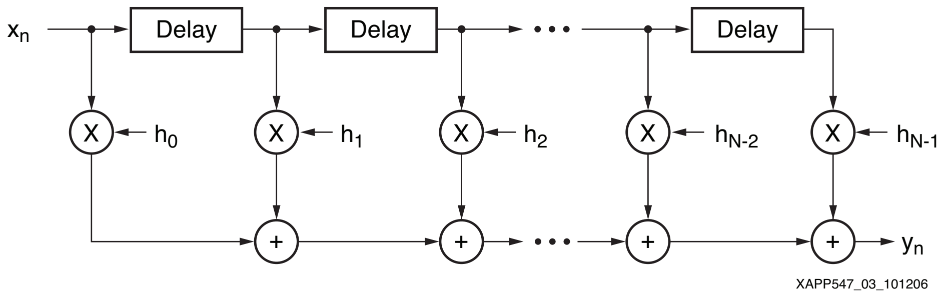

Figure 3: **FIR Filter Implementation**

In implementing the reference design, an amplitude-modulated (AM) signal is buried in the noise, and a bandpass filter is necessary to recover the original signal. A Windows-based application, ScopeFIR from Iowegian International, is used to aid in the design of the filter and in generation of the filter coefficients. ScopeFIR can be found at:

http://www.iowegian.com/

Two FIR filter algorithms were implemented: fir_basic and fir_8reg. Fir_basic is a *textbook* implementation of the FIR algorithm and is adapted from source obtained in [Ref 4]. The original source can be found at:

http://www.dspguru.com/info/faqs/firfaq.htm

The fir_basic algorithm follows:

```
FLOAT fir_basic(FLOAT *input, int ntaps, FLOAT h[], FLOAT z[])
{
    int ii;
    FLOAT accum;
    // store input at the beginning of the delay line
    z[0] = *input;
    // calc FIR
    accum = 0;
    for (ii = 0; ii < ntaps; ii++) accum += h[ii] * z[ii];
    // shift delay line
    for (ii = ntaps - 2; ii >= 0; ii--)  z[ii + 1] = z[ii];
    return accum;
}
```

Prior to algorithm execution, the z array, which is a shifted version of the input data, must be initialized to zero via the clear function:

```
void clear(int ntaps, FTYPE z[])
{
    int ii;
    for (ii = 0; ii < ntaps; ii++) z[ii] = 0.000;
}
```

The fir_basic algorithm is a direct implementation of the FIR filter shown in Figure 3, where h(ii) are the filter coefficients, z(ii) are the delayed (shifted) inputs, and accum is the accumulated output.

Using the FPU, the fir_basic filter achieves a performance of 9.9 MFLOPS — a performance improvement of 3.6 times when compared to the software floating-point emulation library. However, the PowerPC processor FPU has a peak throughput of 100 MFLOPS when operated at 100 MHz as in this reference design. By optimizing how data is handled and presented to the FPU, substantially higher performance can be achieved. The algorithm fir_8reg provides such an implementation, exploiting techniques to enable efficient utilization of the pipeline in the FPU:

- Loop unrolling
- Utilization of array pointers instead of indexes
- Reorganizing the code to eliminate FPU pipeline dependencies
- Holding of a partial set of coefficients in the FPU register file

Utilizing the above techniques, the fir_8reg algorithm obtains a performance of 56 MFLOPS, representing an improvement of 20 times over the performance of the software floating-point emulation library.

Implementing the FPU Reference Design

Reference Design Environment

The PowerPC processor FPU reference design is built and tested using the following Xilinx software tools:

- ISE™ software – Version 8.2.02i (Service Pack 2)
- EDK – Version 8.2.01i (Service Pack 1)

The reference design is demonstrated utilizing the following hardware:

- Xilinx ML403 Virtex-4 evaluation platform
- Xilinx Parallel Cable IV or Xilinx Platform Cable USB for bitstream download and HW/SW debug
- Windows-based PC running the supplied graphics utility `MathGraphX.exe`

- Serial cable connected between the ML403 board and the PC

Refer to [Ref 2] for additional information on installing the design, building the reference design, using the above products, update information, and for board configuration data.

HDL Implementation and Software Implementation

Supplied Files

The reference design consists of a completed and tested EDK PowerPC processor design. All HDL in this design is either generated by EDK or is provided as a library element within EDK. The PowerPC software reference design example is written in C.

The reference design is provided as a ZIP file archive. [Ref 2] describes in detail how to unzip the design and gives the proper location for the design files. As shown in Table 1, the design is implemented in three forms and three associated directories, allowing the user to start at any one of three design points, bypassing hardware system builds. The completed design ready for download to the ML403 Evaluation Platform is located under the FPU_Lab_FIR_Filter_Part3 directory.

Table 1: **Design Directory Structure**

Directory	Description
`..\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter`	Initial FPU design project
`..\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter_Part2`	The built PowerPC design as created via the Base System Builder Wizard
`..\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter_Part3`	The completed and built PowerPC design with attached FPU

Table 2 lists the files contained within the PowerPC FPU archive in each of the directories listed in Table 1. Not listed in the table are files generated by XPS in the implementation of the design and located in the directories FPU_Lab_FIR_Filter_Part2, and FPU_Lab_FIR_Filter_Part3.

Table 2: **Source Files Included in the PowerPC FPU Archive**

File Name	Description
`..\code\fir_demo.c`	The top-level demonstration of the FIR filter reference design.
`..\code\fir_filters.c`	The FIR filter functions used by the design: fir_basic and fir_8reg.
`..\code\fir_filters.h`	`.h` file for `fir_filters.c`.
`..\code\gpio.c`	A collection of low-level I/O functions, including drivers for the LCD display located on the ML403 evaluation platform.
`..\code\io_fmt_utils.c`	A collection of functions for formatting data for display via the LCD or output via the UART.
`..\code\io_fmt_utils.h`	`.h` file for `io_fmt_utils.c`.
`..\code\plotlib.c`	A collection of functions to plot data via the program `MathGraphX.exe`.
`..\code\plotlib.h`	`.h` file for `plotlib.c`.

Table 2: **Source Files Included in the PowerPC FPU Archive *(Continued)***

File Name	Description
`..\code\stopwatch.c`	A collection of functions enabling timing and benchmarking of code segments.
`..\pcores\plb_tft_cntlr_ref_v1_00.c\..`	A set of files for adding a VGA-TFT controller to a design. These files are included for completeness as they are referenced by the ML403 board description file. These files are not used in this design.

Table 3 lists the custom Base System Builder board description file for the ML403 evaluation platform used by this design.

Table 3: **Source File for Base System Builder Board Description File**

File Name	Description
`..\EDK\board\Xilinx\boards\Xilinx_ML403_TFT\data\Xilinx_ML403_TFT_v2_2_0.xbd`	The ML403 evaluation platform board description data.

HDL Implementation

All HDL in this design is either generated by EDK or is provided as a library element within EDK. The PowerPC software reference design example is written in C.

External Port Connections

All port connections are generated by EDK as defined in the board description file as described in Table 3.

Software Implementation

The completed software reference design consists of the source files described in Table 2. Filter performance is shown on two separate display devices.The first, the LCD display on the ML403 evaluation platform shows MFLOPS performance data. The second display, MathGraphX, displays on a Windows-based PC the input signal, the input signal with noise, the filtered noisy signal, and the filtered noisy signal utilizing the optimized FIR filter. Figure 4 shows a typical MathGraphX data display.

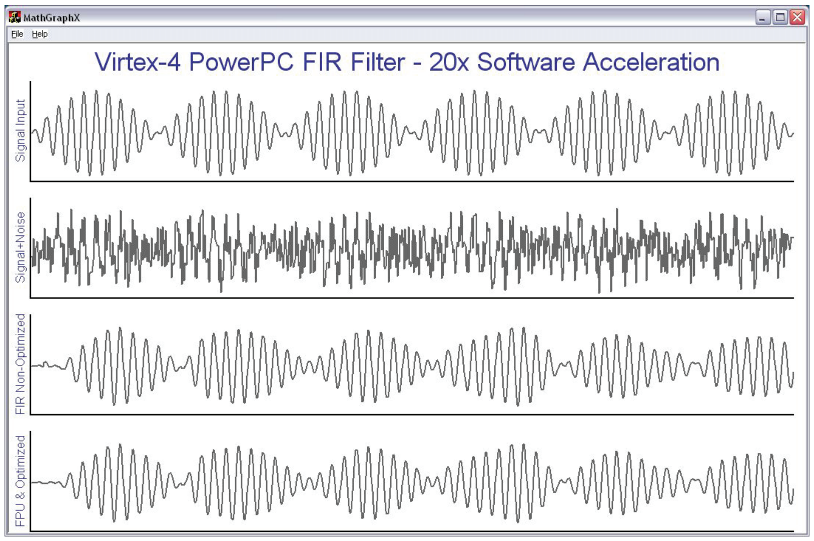

Figure 4: **MathGraphX Windows-Based PC Display**

The reference design also demonstrates the increased performance obtained by the software PowerPC Floating Point Performance Library. This library is available with EDK version 8.2 and is enabled with the compiler option:

```
-mppcperflib
```

When a floating point unit is added to a design the following also occur:

- The **`-mfpu=<fpu_switch>`** compiler switch is automatically generated, telling the compiler to create code targeting the floating point unit. In this reference design, the compiler option **`-mfpu=sp_lite`** is generated.
- **`#define HAVE_XFPU`** and **`#define HAVE_XFPU_<fpu_switch>`** are generated by the compiler in response to the **`-mfpu`** compiler switch. In this reference design, the defines **`#define HAVE_XFPU`** and **`#define HAVE_XFPU_SP_LITE`** are created. This define HAVE_XFPU is used by code in `fir_demo.c` to enable the additional test case demonstrating optimized floating point code execution by the FPU.

Key Software Module Descriptions

The top-most file in the software design is `fir_demo.c`, containing the function main and benchmark routines. The following functions are executed in `fir_demo.c`:

main()

- Enables caches
- Initializes and writes a message to the ML403 LCD display
- Blinks four LEDs on the ML403
- Calls the benchmark routine benchmark_fir() to benchmark the FIR filters and compute MFLOPS
- Calls the filter demonstration and plotting routines via fir_filter_demo()

benchmark_fir()

- Generates and modulates the carrier sin wave, and adds random noise
- Runs and times the non-optimized FIR filter: fir_basic
- Runs and times the optimized FIR filter: fir_8reg
- Displays the performance results on the ML403 LCD display
- Sends the performance results to a serial-port-connected PC

fir_filter_demo()

- Generates and modulates the carrier sin wave, and adds random noise
- Formats and plots the original signal along with the signal with noise via MathGraphX on a serial-port-connected PC
- Generates a filtered signal via fir_basic
- Generates a filtered signal via fir_8reg
- Formats and plots the fir_basic and fir_8reg filtered signal results via MathGraphX on a serial-port-connected PC

Resource Utilization and System Timing

The reference design in a Virtex-4 FX-12 part consumes less than 30% of the available logic resources, enabling significant additional logic to be added to the reference design. See Table 4.

Table 4: **Reference Design Resource Utilization**

Resource	Number	Percentage Used
Slice Flip-Flops	1,909 out of 10,944	17%
4-input LUTs	3,161 out of 10,944	28%
Bonded IOBs	15 out of 320	4%
BUFG/BUFGCTRLs	2 out of 32	6%
FIFO16/RAMB16s	34 out of 36	94%
DSP48 Slices	4 out of 32	12%

The 200 MHz processor with 100 MHz processor local bus (PLB) and 100 MHz FPU design meet all timing requirements. Enabling the APU interface for FPU operation limits the maximum PowerPC operating frequency. Check [Ref 5] for limitations.

Conclusion

Xilinx provides a capable software floating-point emulation library; however, many applications have more demanding requirements that are possibly not met by this library. The Xilinx floating FPU enables acceleration of software floating-point operations. Acceleration is highly dependent on code implementation and actual floating point utilization. For floating-point-intensive code, the Xilinx FPU can accelerate floating-point software from three to twenty times. The FPU reference design supplied with this application note demonstrates this range of performance improvement.

References

1. DS535, Xilinx, Inc., *APU Floating-Point Unit v2.1 Data Sheet*.
2. UG243, Xilinx, Inc., *Implementing a Virtex-4 FX PowerPC System with Floating Point Coprocessor using Platform Studio.* A step-by-step tutorial for building the design under EDK.
3. Barr, Michael and Wanger, Brian. *Introduction to Digital Filters*, Embedded Systems Design, December 2002, pp. 47-48. Also availabe on line:

   http://www.embedded.com/columns/showArticle.jhtml?articleID=9900828
4. Griffin, Grant. *Finite Impulse Response FAQ*, dspGuru, http://www.dspguru.com
5. DS302: the *Virtex-4 Data Sheet: DC and Switching Characteristics.*

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/16/06	1.0	Initial Xilinx release.
11/28/06	1.0.1	Fixed link to UG243 in “References” section.