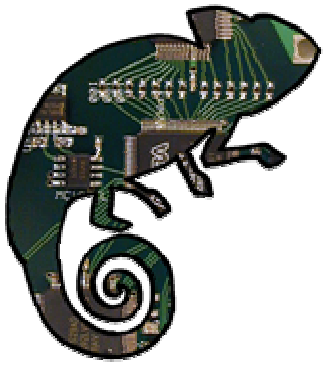

Chameleonic Radio

Technical Memo No. 2

An Investigation of the Blackfin/uClinux Combination as a Candidate Software Radio Processor

S.M. Shajedul Hasan

S.W. Ellingson

January 24, 2006

Bradley Dept. of Electrical & Computer Engineering
Virginia Polytechnic Institute & State University
Blacksburg, VA 24061

Abstract

Analog Device's Blackfin is a candidate processor for future software defined radio (SDR) systems. In this report we describe the Blackfin processor, including how it can be used with uClinux, a variant of the Linux operating system which is tailored toward embedded systems. To characterize the performance of the Blackfin/uClinux combination as a SDR processor, we developed a test application consisting of a simple finite impulse response (FIR) filter. The filter is implemented in C language and is compiled using the open-source GNU tool chain. Using the ADDS-BF537-STAMP Blackfin development board, we achieved a sample rate of 960 thousand samples per second (kS/s) for an 8-tap FIR, and almost 11 kS/s for a 1024 tap FIR. These results were obtained using a slightly modified kernel (cache features enabled) and integer arithmetic. Based on this finding, and taking into account other attractive features of the Blackfin part described in this report, the Blackfin/uClinux combination appears to be an excellent candidate for SDR processing of narrowband and some intermediate-bandwidth communications protocols.

Chapter I

Introduction

1.1 Background

An embedded system is a special-purpose computer system, which is completely encapsulated by the device it controls. Unlike general-purpose computers that are capable of running software to perform many different tasks, an embedded system has specific requirements and performs pre-defined tasks - for example, a dishwasher controller or flight navigation system. Usually the core of this system is a microprocessor or microcontrollers (MCUs), programmed to perform a few tasks [1].

Recently more audio, video and communications processing capabilities are used in portable devices and edge-client devices. Both MCUs and Digital Signal Processors (DSPs) have served these applications in the past, but embedded processors with improved computational capabilities are desired. MCUs are traditionally architected to enable efficient asynchronous control flow, whereas DSPs are architected to perform well for synchronous, constant-rate data flow (for example, audio or voice-band applications). Because so many embedded applications have intense requirements for both control and media processing, engineers often use DSPs and MCUs together, either at the board level or in system-on-chip (SoC) integration [2].

Thus, the need for a powerful “unified” microprocessor for embedded media applications has long been evident. Recently, Analog Devices and Intel jointly developed the high performance Micro Signal Architecture (on which all Blackfin devices are based) which seems powerful enough, inexpensive enough, and sufficiently optimized for both the complex, real-time world of media data flow and for the control-oriented tasks typically

handled by Reduced Instruction Set Computer (RISC) processors. The Blackfin architecture combines media processing attributes such as dual MACs (multiply-accumulate engines, commonly used for high performance DSP applications) with useful characteristics of RISC processors. Blackfin controllers have DSP features not found on any RISC microcontroller and important microcontroller characteristics not typically on DSPs. This paper provides an overview of the architectural and functional features of the Blackfin with some performance analysis [3]. Fig. 1.1 shows the power consumption of the Blackfin processors compare to other processors in the market.

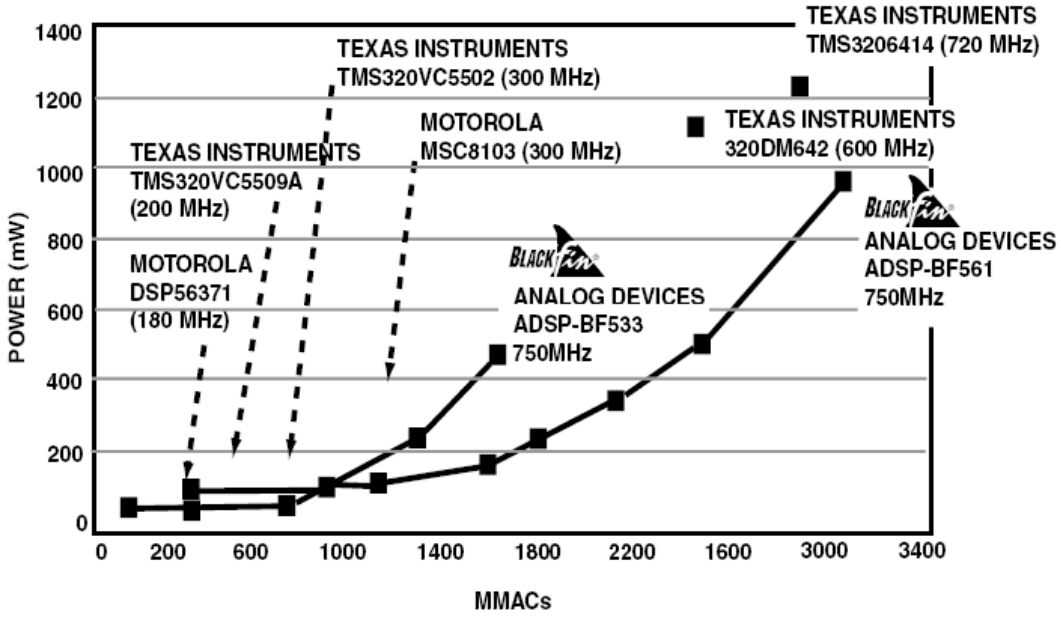

Figure 1.1: Power consumption versus speed for various devices [3].

1.2 Objectives

The objectives of this paper are (i) to present the basic architectural structure and functional features of the Blackfin processor, (ii) to provide a brief introduction to embedded Linux (uClinux) and other related software packages, (iii) to describe the features and capabilities of the hardware platform used for testing the Blackfin processor, (iv) to present the procedures of implementing the uClinux kernel into the Blackfin processor through the chosen hardware platform , and (v) to present a performance

analysis of the Blackfin processor in a common DSP task required for software-defined radio.

1.3 Outline of this Report

This report seeks to describe the architectural and functional features of the Blackfin processor. The description of the processor, hardware platform, software installation, application development, operation, tests, and performance analysis of the Blackfin processor are presented in five chapters. Chapter II presents a brief description of the Blackfin processor. This chapter also describes some of the key features of this processor. In Chapter III, the various software and hardware development tools required to build Blackfin-based application are presented with some brief discussion. Chapter IV presents a detailed performance analysis of a FIR filter running on a Blackfin using uClinux. The conclusions of the report are drawn in Chapter V.

Chapter II

Blackfin Processor- An Overview

This chapter presents the brief description of the Blackfin processor. It also describes the basic features, peripherals, memory layout, booting modes, and development tools required to develop a Blackfin-based application.

2.1 Blackfin Processor [5]

Intel and Analog Devices Inc.(ADI) jointly developed the Micro Signal Architecture (MSA) core and introduced it in December of 2000. Since then Intel has put this core in its cell phone chipsets, and ADI has put this core into it's Blackfin processor family of devices. This document will focus on ADI's ADSP-BF537 Blackfin device [4], which can be found on the BF-537 STAMP platform. The MSA core, found in the BF-537, has the advantages of a clean, orthogonal, RISC-like microprocessor instruction set. It combines a dual-MAC (Multiply/Accumulate) unit, a state-of-the-art signal processing engine, and single-instruction, multiple-data (SIMD) multimedia capabilities into a single instruction-set architecture. The DSP features include one instruction port and two separate data ports mapped to a unified 4GB memory space; two 16-bit, single-cycle throughput multipliers; two 40-bit split data ALUs; two 32-bit pointer ALUs with support for circular and bit-reversed addressing; two loop counters that allow nested, zero overhead looping; and hardware support for on-the-fly saturation and clipping.

The microcontroller features include arbitrary bit manipulation; mixed 16-bit and 32-bit instruction encoding for high code density; memory protection; stack pointers and scratch SRAM for context switching; flexible power management; and an extensible, nested, and prioritized interrupt controller for real-time control.

The multimedia features include four auxiliary 8-bit data ALUs and a rich set of alignment-independent, packed byte operation instructions. These instructions enable the acceleration of fundamental operations associated with video and imaging based applications. The block diagram of the Blackfin processor with peripheral interfaces is given in the Fig. 2.1.

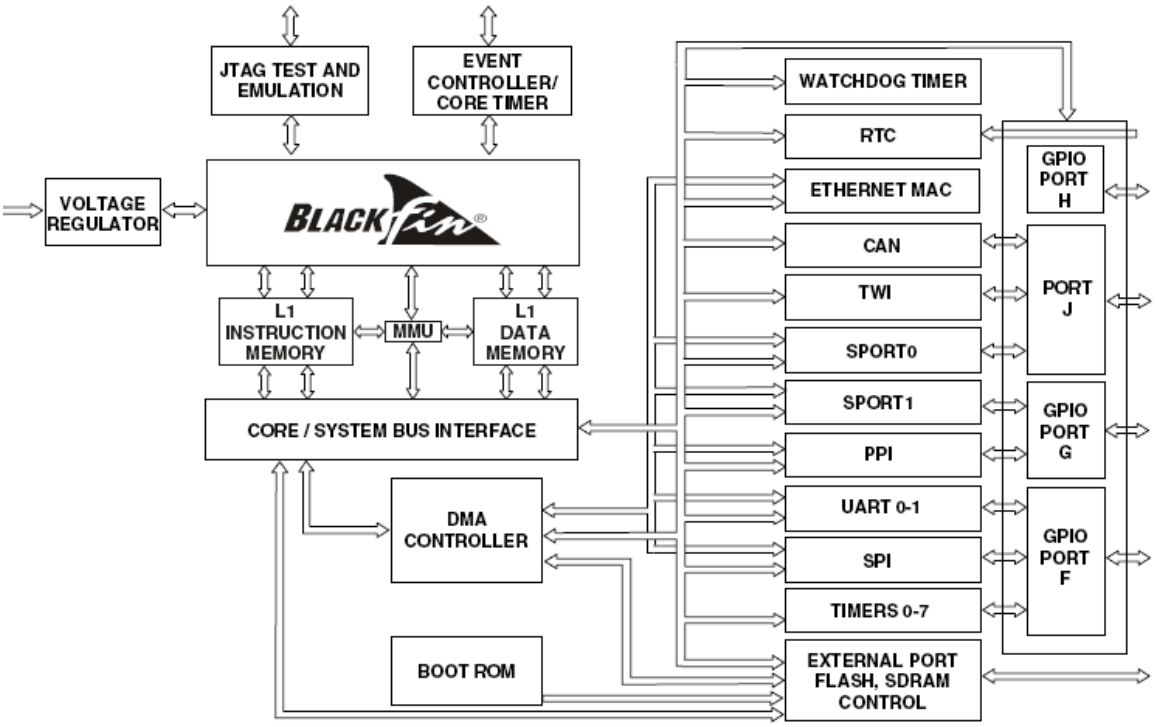

Figure 2.1: ADSP-BF537 processor block diagram [4].

2.2 Blackfin Peripherals

The Blackfin ADSP-BF537 device contains many on-chip peripherals. These include: a Parallel Peripheral Interface (PPI), Serial Ports (SPORTs), Serial Peripheral Interface (SPI), General-purpose timers, Universal Asynchronous Receiver Transmitter (UART), Real-Time Clock (RTC), Watchdog timer, General-purpose I/O (programmable flags), Ethernet MAC, Controller Area Network (CAN) Interface, 12 Peripheral DMAs, Two

Memory-to-Memory DMAs with Handshake DMA, Event Handler With 32 Interrupt Inputs, Two-Wire Interface (TWI) Controller, and a Debug/JTAG Interface. These peripherals are connected to the core via several high bandwidth buses, as shown in Fig. 2.1. Description of some of the peripherals is given below.

2.2.1 Parallel Peripheral Interface (PPI)

The Blackfin processor provides a Parallel Peripheral Interface (PPI) that can connect directly to parallel A/D and D/A converters, ITU-R 601/656 video encoders and decoders, and other general-purpose peripherals. The PPI consists of a dedicated input clock pin, up to 3 frame synchronization pins, and up to 16 data pins. The input clock supports parallel data rates up to half the system clock rate. Three distinct ITU-R 656 modes are supported- Active Video Only, Vertical Blanking Only, and Entire Field. General-purpose modes of the PPI are provided to suit a wide variety of data capture and transmission applications. These modes are divided into four main categories: Data Receive with Internally Generated Frame Syncs, Data Receive with Externally Generated Frame Syncs, Data Transmit with Internally Generated Frame Syncs, and Data Transmit with Externally Generated Frame Syncs.

2.2.2 Serial Ports (SPORTs)

This processor incorporates two dual-channel synchronous serial ports (SPORT0 and SPORT1) for serial and multiprocessor communications. The SPORTs support the following features:

- I^2S capable operation
- Bidirectional operation: Each SPORT has two sets of independent transmit and receive pins, enabling eight channels of I^2S stereo audio
- Buffered (eight-deep) transmit and receive ports
- Each transmit and receive port can either use an external serial clock or generate its own
- Each SPORT supports serial data words from 3 to 32 bits in length
- Each transmit and receive port can run with or without frame sync signals

- Each SPORT can perform A-law or μ-law companding in hardware
- DMA operations with single-cycle overhead
- Each transmit and receive port can generate an interrupt upon completing the transfer of data
- Multichannel Capability: Each SPORT supports 128 channels out of a 1024-channel window and is compatible with the H.100, H.110, MVIP-90, and HMVIP standards.

2.2.3 Serial Peripheral Interface (SPI) Port

The processor has an SPI-compatible port that enables the processor to communicate with multiple SPI-compatible devices. The SPI interface uses three pins for transferring data: two data pins and a clock pin. An SPI chip select input pin lets other SPI devices select the processor, and seven SPI chip select output pins let the processor select other SPI devices. Using these pins, the SPI port provides a full-duplex synchronous serial interface, which supports both master and slave modes and multi-master environments. The SPI port's baud rate and clock phase/polarities are programmable and it has an integrated DMA controller, configurable to support either transmit or receive datastreams.

2.2.4 Timers

There are 9 general-purpose programmable timer units in the processor. Eight timers have an external pin that can be configured either as a Pulse Width Modulator (PWM) or timer output, as an input to clock the timer, or as a mechanism for measuring pulse widths of external events. These timer units can be synchronized to an external clock input connected to the PF1 pin, an external clock input to the PPI_CLK pin, or to the internal SCLK. The timers can generate interrupts to the processor core to provide periodic events for synchronization, either to the processor clock or to a count of external signals. In addition to the 8 general-purpose programmable timers, a 9th timer is also provided. This extra timer is clocked by the internal processor clock and is typically used as a system tick clock for generation of operating system periodic interrupts.

2.2.5 Universal Asynchronous Receiver Transmitter (UART) Port

The processor provides two half-duplex Universal Asynchronous Receiver/Transmitter (UART) ports, which are fully compatible with PC-standard UARTs. The UART ports provide a simplified UART interface to other peripherals or hosts, providing half-duplex, DMA-supported, asynchronous transfers of serial data. The UART ports include support for 5 to 8 data bits; 1 or 2 stop bits; and none, even, or odd parity. The UART ports support two modes of operation: (1) Programmed I/O, in which the processor sends or receives data by writing or reading I/O-mapped UART registers, where the data is double buffered on both transmit and receive; and (2) Direct Memory Access (DMA), in which the DMA controller transfers both transmit and receive data, reducing the number and frequency of interrupts required to transfer data to and from memory.

2.2.6 Real-Time Clock (RTC)

The processor's Real-Time Clock (RTC) provides a robust set of time features, including current time, stopwatch, and alarm. The RTC is clocked by a 32.768 kHz crystal external to the processor. The RTC peripheral has dedicated power supply pins, so that it can remain powered up and clocked even when the rest of the processor is in a low power state. The RTC provides several programmable interrupt options. The 32.768 kHz input clock frequency is divided down to a 1 Hz signal by a prescaler. Like the other peripherals, the RTC can wake up the processor from Sleep mode or Deep Sleep mode.

2.2.7 Watchdog Timer

The processor includes a 32-bit timer that can be used to implement a software watchdog function. The programmer initializes the count value of the timer, enables the appropriate interrupt, then enables the timer. Thereafter, the software must reload the counter before it counts to zero from the programmed value.

2.2.8 General-Purpose I/O (GPIO)

The ADSP-BF537 processor has 48 bi-directional, general-purpose I/O (GPIO) pins allocated across three separate GPIO modules—PORTFIO, PORTGIO, and PORTHIO,

associated with Port F, Port G, and Port H, respectively. Each general-purpose port pin can be individually controlled by manipulation of the port control, status, and interrupt registers, which are as follows:

- GPIO Direction Control Register: Specifies the direction of each individual GPIO pin as input or output
- GPIO Control and Status Registers: The processor employs a “write one to modify” mechanism that allows any combination of individual GPIO pins to be modified in a single instruction, without affecting the level of any other GPIO pins
- GPIO Interrupt Mask Registers: The two GPIO Interrupt Mask registers allow each individual GPIO pin to function as an interrupt to the processor
- GPIO Interrupt Sensitivity Registers: The two GPIO Interrupt Sensitivity Registers specify whether individual pins are level- or edge-sensitive and specify—if edge-sensitive—whether just the rising edge or both the rising and falling edges of the signal are significant.

2.2.9 Ethernet MAC

The Ethernet Media Access Controller (MAC) peripheral for the ADSP-BF537 processors provide 10-100 Mb/s between a Media Independent Interface (MII) and the Blackfin peripheral subsystem. The MAC operates in both Half-Duplex and Full-Duplex modes. The MAC is clocked internally from the CLKIN pin on the processor.

2.3 Memory

The Blackfin processor architecture structures memory as a single unified 4GB address space using 32-bit addresses. All resources including internal memory, external memory, and I/O control registers occupy separate sections of this common address space. Level 1 (L1) memories are located on the chip and are faster than the Level 2 (L2) off-chip memories. The L1 memory system is the primary highest performance memory available to the core. The off-chip memory system, accessed through the External Bus Interface

Unit (EBIU), provides expansion with SDRAM flash memory, and SRAM, optionally accessing up to 132MB of physical memory. The memory DMA controller provides high bandwidth data movement capability. It can perform block transfers of code or data between the internal memory and the external memory spaces. Table 2.1 presents the memory configuration of the ADSP-BF537 processor and the memory map of the ADSP-BF537 is given in the Fig. 2.2. MMR is an acronym for Memory Mapped Register.

Table 2.1: Memory Configurations of ADSP-BF537 [4].

Type of Memory	ADSP-BF537
Instruction SRAM/Cache, lockable	16 KB
Instruction SRAM	48 KB
Data SRAM/Cache	32 KB
Data SRAM	32 KB
Data Scratchpad SRAM	4 KB
Total	132 KB

2.3.1 Internal Memory

The processor has three blocks of on-chip memory that provide high bandwidth access to the core:

- L1 instruction memory, consisting of SRAM and a 4-way set-associative cache, accessed at full processor speed
- L1 data memory, consisting of SRAM and/or a 2-way set-associative cache, accessed at full processor speed
- L1 scratchpad RAM, which runs at the same speed as the L1 memories but is only accessible as data SRAM and cannot be configured as cache memory.

2.3.2 External Memory

External (off-chip) memory is accessed via the External Bus Interface Unit (EBIU). This 16-bit interface provides a glue-less connection to a bank of synchronous DRAM

(SDRAM) and as many as four banks of asynchronous memory devices including flash memory, EPROM, ROM, SRAM, and memory-mapped I/O devices. The PC133-compliant SDRAM controller can be programmed to interface to up to 512M bytes of SDRAM. The asynchronous memory controller can be programmed to control up to four banks of devices. Each bank occupies a 1M byte segment regardless of the size of the devices used, so that these banks are only contiguous if each is fully populated with 1M byte of memory.

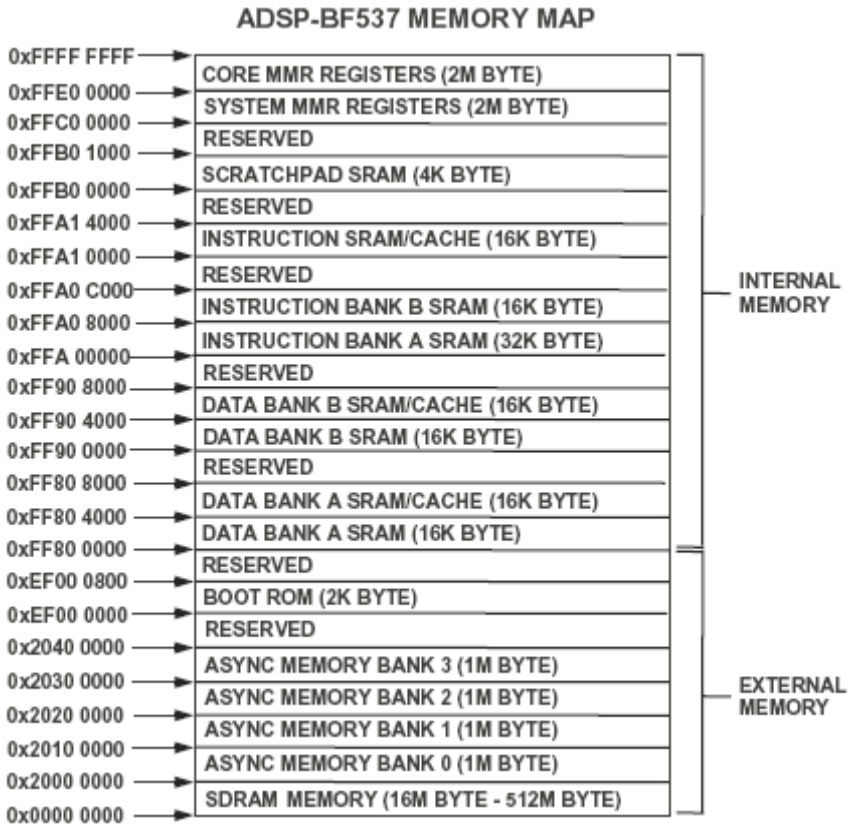

Figure 2.2: Memory map of the ADSP-BF537 processor [4].

2.3.3 I/O Memory Space

Blackfin processors do not define a separate I/O space. All resources are mapped through the flat 32-bit address space. Control registers for on-chip I/O devices are mapped into memory-mapped registers (MMRs) at addresses near the top of the 4G byte address space. These are separated into two smaller blocks: one contains the control MMRs for

all core functions and the other contains the registers needed for setup and control of the on-chip peripherals outside of the core. The MMRs are accessible only in Supervisor mode. They appear as reserved space to on-chip peripherals.

2.4 Boot Modes

The processor has six mechanisms for automatically loading internal L1 instruction memory after a reset. The boot modes are:

- Boot from 8-bit and 16-bit external flash memory,
- Boot from serial SPI memory (EEPROM or flash),
- Boot from SPI host device,
- Boot from UART,
- Boot from serial TWI memory (EEPROM/flash),
- Boot from TWI Host.
- Execute from 16-bit external memory, bypassing boot sequence

For each of the first six boot modes, a 10-byte header is first read from an external memory device. The header specifies the number of bytes to be transferred and the memory destination address. Multiple memory blocks may be loaded by any boot sequence. Once all blocks are loaded, program execution commences from the start of L1 instruction SRAM.

Chapter III

Development Tools

Several software and hardware tools are needed to develop a Blackfin-based DSP application. In this chapter the brief description of the selected software tools are given. We also describe the hardware platform which we use to evaluate the Blackfin processor.

3.1 Development Process

A typical embedded system development process consists of simulation, evaluation, and emulation. These phases are typically performed using a development environment consisting of an evaluation board including the target hardware and a development workstation, as shown in Fig. 3.1.

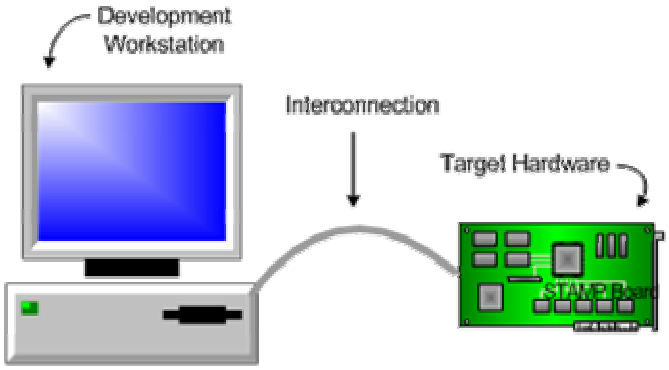

Figure 3.1: Basic components of a development environment.

For the current effort we have chosen the Analog Devices ADDS-BF537-STAMP board, which includes the ADSP-BF537 Blackfin processor as our target hardware. The details of this hardware and the resident software are given on the next sections of this chapter. The STAMP board includes various useful resources such as an Ethernet controller with RJ45 Ethernet jack, a serial port with DB9 serial connector, a power jack and on board

voltage regulator (7V-20V DC input), and is intended to be used with the uClinux operating system.

Our development workstation is a Dell 600m laptop computer using the Fedora Core 4.0 (Linux) operating system, and using Kermit as terminal emulator software. For the interconnection between the workstation and the target, there are two options: (1) RS-232 serial connection or (2) 10 Mbps Ethernet connection. The details about these connections and how they work are given on the next sections.

3.2 Software Development Tools [5]

Three sets of software development tools are available for the Blackfin processor architecture: (1) VisualDSP++ 4.0 from Analog Devices, (2) MULTI® from Green Hills Software, (3) The open source GNU tool chain and uClinux. Each offers advantages for different types of applications. We have chosen the open source tool chain and the uClinux for our application development.

Fig. 3.2 shows a block diagram of the necessary software tools for developing a Blackfin-based application using uClinux. The operating system of the development workstation is Fedora Core 4.0 from RedHat Linux.

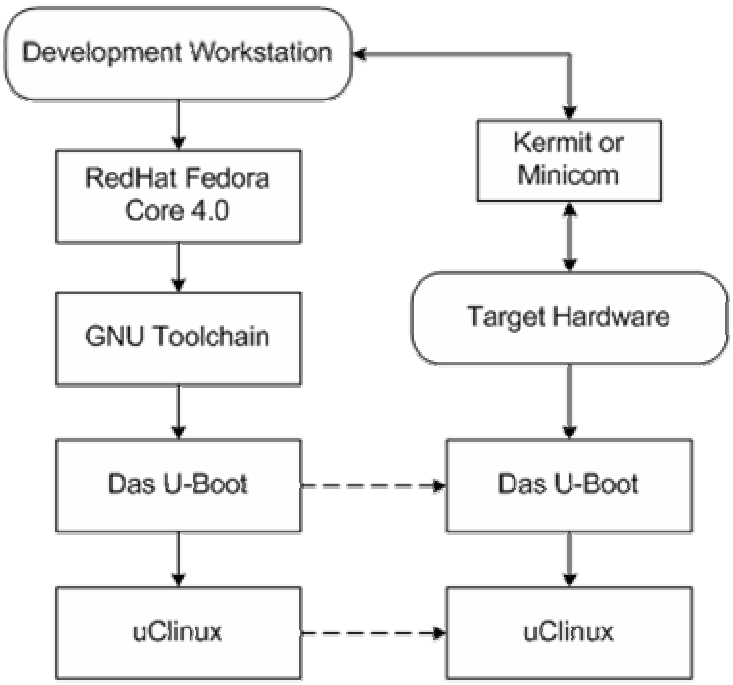

Figure 3.2: Block diagram of software tools for the GNU/uClinux approach.

The GNU tool chain is required for compiling, assembling and linking C source code. This is also required to build the uClinux kernel. So, uClinux should be installed after the installation of GNU toolchain. "*Das U-Boot*" is the necessary software for booting the Blackfin processor. The terminal program *Kermit* is used for communicating with the Blackfin processor through serial port. After building the uClinux kernel and *Das U-Boot* software the images of these are transferred into the Blackfin processor using *Kermit*. In summary, all software development is performed on the workstation and is transferred to the target hardware through serial port or Ethernet port.

3.2.1 uClinux

uClinux is an operating system that is derived from the Linux kernel. It is intended for microcontrollers without Memory Management Units (MMUs). It is available on many processor architectures, including the Blackfin processor. Factors which make uClinux an attractive choice include open source code availability, royalty-free licenses, open source

community support, tools availability, networking support, portability, and an extensive application base. The installation procedures for uClinux are described in detail in [6].

3.2.2 GNU Tool Chain

The GNU tool chain for the Blackfin consists of several utilities which accomplish the basic tasks required to create executable programs: compiling, assembling, and linking. First, standard C code is converted into Blackfin assembly code by the complier, *bfin-elf-gcc*. The assembler, *bfin-elf-as*, then takes this code and creates Executable and Linkable Format (ELF) object files. ELF files are linked together using the utility *bfin-elf-ld*, or are included in an archive library using the utility *bfin-elf-ar*. The ELF files must then be converted into a binary format compatible with uClinux. This conversion is done using the utility *bfin-elf-elf2flt* which converts ELF files into flat binary format files. Detailed installation procedures and details can be found in [7].

3.2.3 *Das U-Boot*

Das U-Boot is a boot loader program that is stored in flash memory on the target system. It allows the target hardware to load an ELF memory image from a serial connection, from a network connection, or from flash memory. When *Das U-Boot* is stored in flash memory on the target system, it can be loaded on system reset. It can then obtain a uClinux memory image and boot uClinux. The following is an overview of some of *Das U-Boot*'s features; see [8] for additional details.

- Autoboot: Automatically boots the system on power up or reset of the board after the timer counts down
- OS loading commands that can be executed from the *Das U-Boot* command line
- *Das U-Boot* allows itself to be upgraded with a new version from within *Das U-Boot*
- Support for the common network commands ping, tftp, and dhcp
- A memory image can be loaded through the serial port onto the target system.

3.2.4 Terminal Program

One of the first things that needs to be done before *Das U-Boot* or uClinux can be used is to setup a terminal program to communicate with the target device. The STAMP board has a DB9 serial connector for this purpose. The method used to configure *Kermit* for use with a STAMP board running *Das U-Boot* / uClinux is described in [9].

3.3 Target Hardware

The ADDS-BF537 STAMP board, shown in Fig. 3.3, is a low-cost (about US$200) development platform for the ADSP-BF537 Blackfin device. This board is part of the Blackfin/uClinux open source project. The schematics for the board as well as other related documents are available in [5]. The STAMP is designed to be used in conjunction with the GNU Tool Chain and supports advanced application code development features including:

- Create, compile, assemble, and link application programs written in C++, C and ADSP-BF537 assembly;
- Load, run, step, halt, and set breakpoints in the application program;
- Read and write data and program memory; and
- Read and write core and peripheral registers.

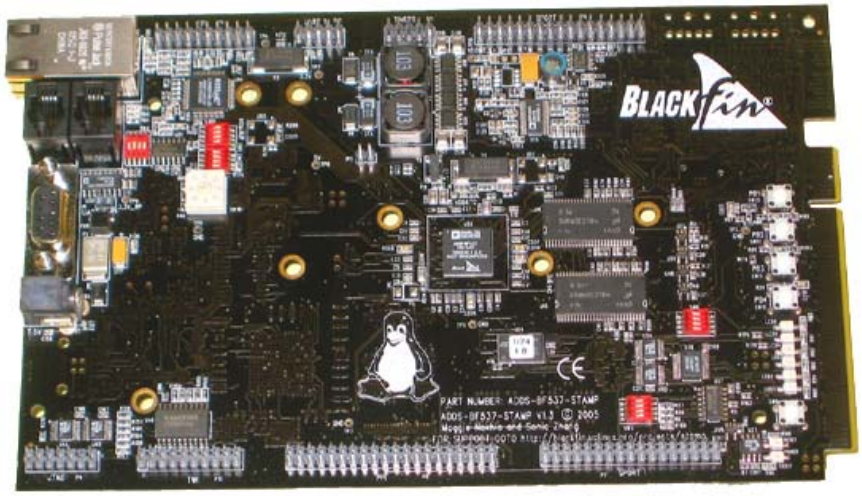

Figure 3.3: The BF537-STAMP board.

3.3.1 STAMP Board Features

Some of the STAMP board's features are:

- ADSP-BF537 Blackfin device with JTAG interface
- 500MHz core clock
- 133MHz system clock
- 32M x 16bit external SDRAM (64MBytes)
- 2M x 16bit external flash (4MBytes)
- 10/100 Mbps Ethernet Interface (via on chip MAC, connected via DMA)
- CAN TJA1041 transceiver with two modular connectors
- RS-232 UART interface with DB9 serial connector
- JTAG ICE 14-pin header
- Six general-purpose LEDs, four general purpose push-buttons
- Discrete IDC Expansion ports for all processor peripherals

3.3.2 Boot Configurations

It has already been discussed that the Blackfin processor has various boot modes. STAMP board provides a DIP switch labeled 'S1' which selects from among the various boot modes available. The boot modes are summarized in the table below:

Table 3.1: ADSP-BF537 Boot Modes [5].

BMode2	BMode1	BMode0	Description of Boot Modes
0	0	0	Execute from 16-bit external memory (Bypass Boot ROM)
0	0	1	Boot from 8-bit or 16-bit memory (EPROM/flash)
0	1	0	Reserved
0	1	1	Boot from serial SPI memory (EEPROM/flash)
1	0	0	Boot from SPI host (slave mode)
1	0	1	Boot from serial I2C memory (EEPROM/flash)
1	1	0	Boot from I^2C host (slave mode)
1	1	1	Boot from UART host (slave mode)

3.3.2 STAMP Board Quick Start [10]

The layout of the ADSP-BF537 STAMP board is shown in the Fig. 3.4. The kit contains a ADSP-BF537 STAMP board, a universal 7.5V DC power supply, and a CD-ROM containing uClinux for the ADSP-BF537 Processor with all the required software and the documents.

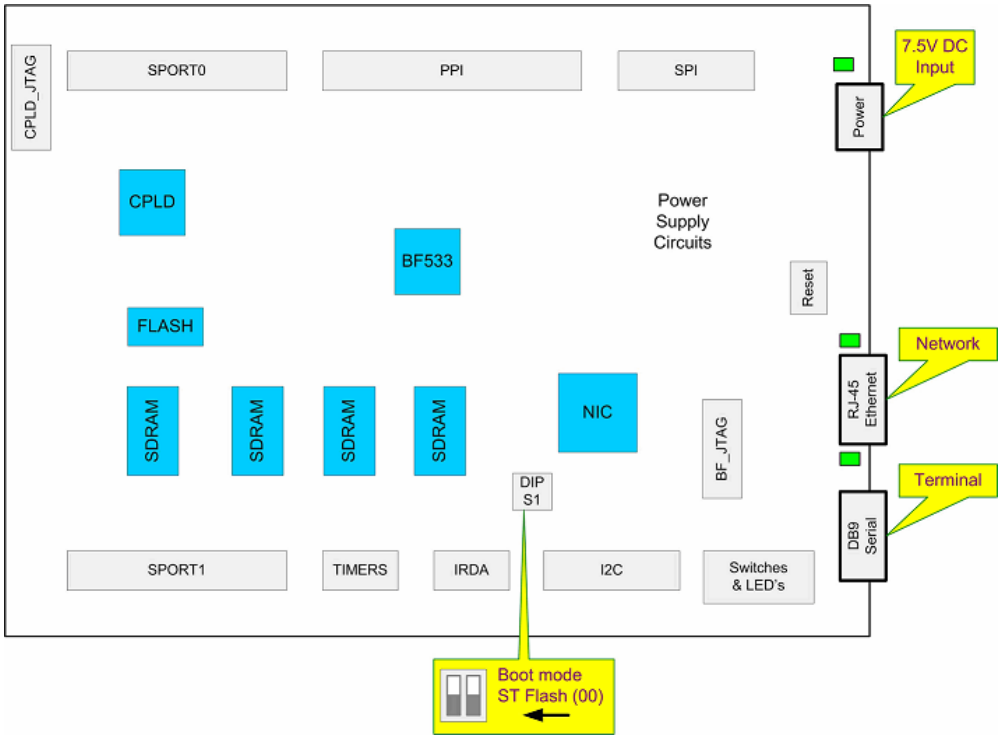

Figure 3.4: Layout of ADSP-BF537 STAMP board [10].

The following procedure can be followed by a new user to quickly start using the ADSP-BF537 STAMP board:

- Connect the serial port to the host computer from the STAMP board using straight-through serial cable.
- Connect straight-through Ethernet cable to the local network from the STAMP board or to the host computer Ethernet port via crossover Ethernet cable.

- Start the terminal emulation program (e.g., *Kermit*) on the host computer. From the configuration menu, set the serial port connection for 57600 bps, 8 data bits, no parity, 1 stop bit, and hardware flow control off.
- Connect DC power.
- After the STAMP board powers up a *Das U-Boot* start-up screen appears in the terminal window. At this point, a boot timer has started to count down. When it reaches zero, the board will automatically load the uClinux kernel from flash memory and boot the kernel. At this point, we are presented the command prompt of the shell program.

Chapter IV

FIR Filter Performance Analysis

The finite impulse response (FIR) filter is a common way to assess the performance of a DSP processor. In this chapter we use a FIR filter to measure the Blackfin processor's performance.

4.1 FIR Filter Basics

A FIR filter is usually implemented by using a series of delays, multipliers, and adders to create the filter's output. Fig. 4.1 shows the basic block diagram for an FIR filter of length N. The h_K's are the coefficients used for multiplication, so that the output at time *n* is the summation of all the delayed samples multiplied by the appropriate coefficients.

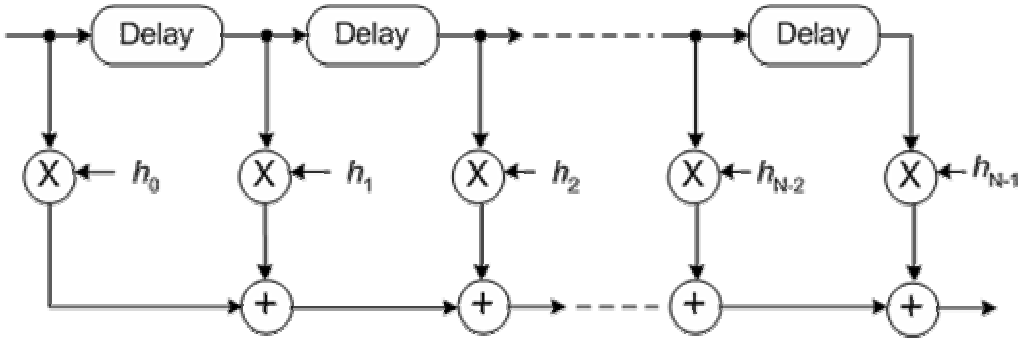

Figure 4.1: A FIR filter.

This output is given by the following equation:

$$y_n = h_0 x_n + h_1 x_{n-1} + h_2 x_{n-2} + ... + h_{N-1} x_{n-N-1} \quad (4.1)$$

4.2 FIR implementation

For our testing we have implemented a FIR filter using a simple function written in C. Table 4.1 shows the source code. The main C program calls this function periodically to

compute the filtered output based on the supplied input signal and the value of coefficients. This program was compiled, assembled, and linked using the GNU toolchain. The program is intended to execute from the uClinux kernel, thus we have used *bfin-uclinux-gcc* compiler, which automaticlly links the program with the appropriate uClinux run-time libraries, which in turn makes the appropriate calls to the uClinux operating system. *bfin-elf-gcc* is used to compile the uClinux operating system itself, and uses a different set of libraries. The switch (compiling option) '-elf2flt' has been used with the compiler to convert the ELF file to binary format to make it compatible with uClinux. After compilation, the output file was downloaded to the Blackfin processor using FTP, where it was executed and the time required to complete was measured.

Table 4.1: FIR filter function written in C language.

```
float fir_filter(float input, float *coef, int n, float *history)
{
    int i;
    float *hist_ptr,*hist1_ptr,*coef_ptr;
    float output;

    hist_ptr = history;
    hist1_ptr = hist_ptr;             /* use for history update */
    coef_ptr = coef + n - 1;          /* point to last coef */

/* form output accumulation */
    output = *hist_ptr++ * (*coef_ptr--);

    for(i = 2 ; i < n ; i++) {
        *hist1_ptr++ = *hist_ptr;            /* update history array */
        output += (*hist_ptr++) * (*coef_ptr--);
    }
    output += input * (*coef_ptr);           /* input tap */
    *hist1_ptr = input;                      /* last history */

    return(output);
}
```

Table 4.2 shows the FIR function in assembly language, which has been generated from the C function using the command '*bfin-uclinux-gcc –S*'.

Table 4.2: FIR filter function converted into assembly language.

```
.global _fir_filter;
.type _fir_filter, STT_FUNC;
_fir_filter:
      [--sp] = ( p5:5 );

      LINK 20;
      [FP+16] = R1;
      [FP+20] = R2;
      W [FP+12] = R0;
      R0 = [FP+24];
      [FP+-8] = R0;
      R0 = [FP+-8];
      [FP+-12] = R0;
      R0 = [FP+20];
      R1 = R0;
      R1 <<= 1;
      R0 = [FP+16];
      R0 = R1 + R0;
      R0 += -2;
      [FP+-16] = R0;
      P0 = FP;
      P0 += -8;
      R3 = [P0];
      P5 = FP;
      P5 += -16;
      R1 = [P5];
      P2 = R3;
      P1 = R1;
      R2 = W [P2] (X);
      R0 = W [P1] (X);
      R0 = R2.L * R0.L (IS);
      R1 += -2;
      [P5] = R1;
      R3 += 2;
      [P0] = R3;
      W [FP+-18] = R0;
      R0 = 2 (X);
      [FP+-4] = R0;
```

4.3 Performance Analysis Procedures

Our main goal is to measure the execution time for various FIR filters on the Blackfin. Since the Blackfin processor architecture was designed for fixed point computations,

floating point arithmetic is emulated, which results in slower computations compared to native fixed-point arithmetic. Also, L1 memory is the highest level memory in the Blackfin processor and supplies the highest level of performance in any computation. The cache controller allows larger instruction and data sections to exist in lower-level memory, and code and data that are used most frequently are brought into cache and thus are available for single-cycle access, just as though it was in L1 memory. Thus, for our analysis, we consider the effects of:

- Number of filter taps, N
- Native fixed-point calculation vs. emulated floating-point calculation
- Cache access strategy

A low-pass FIR filter was designed for a normalized cut-off frequency of 0.208. Fig. 4.2 shows the frequency response of this filter for N=64 taps. For different numbers of taps the coefficient values have been generated and saved to a file for supplying to our C program. In the same way, an input data file of 48000 samples, simulating a sum of two sine waves of different frequencies has been generated and saved.

Analog Devices Inc. has developed a formula, shown below, to estimate the number of execution cycles required of the Blackfin processor for a block FIR filter [12].

$$\text{Number of execution cycles} = \frac{x}{2}(2+N) \tag{4.2}$$

This the *maximum* number of execution cycles for the processor, and the processor may perform better than this. An "execution cycle" in this case 50 clock "tics", which is this case is the system clock rate (500 MHz) divided by 500. x is the number of input samples. We have used this formula to estimate the maximum execution time and compared these values with our results.

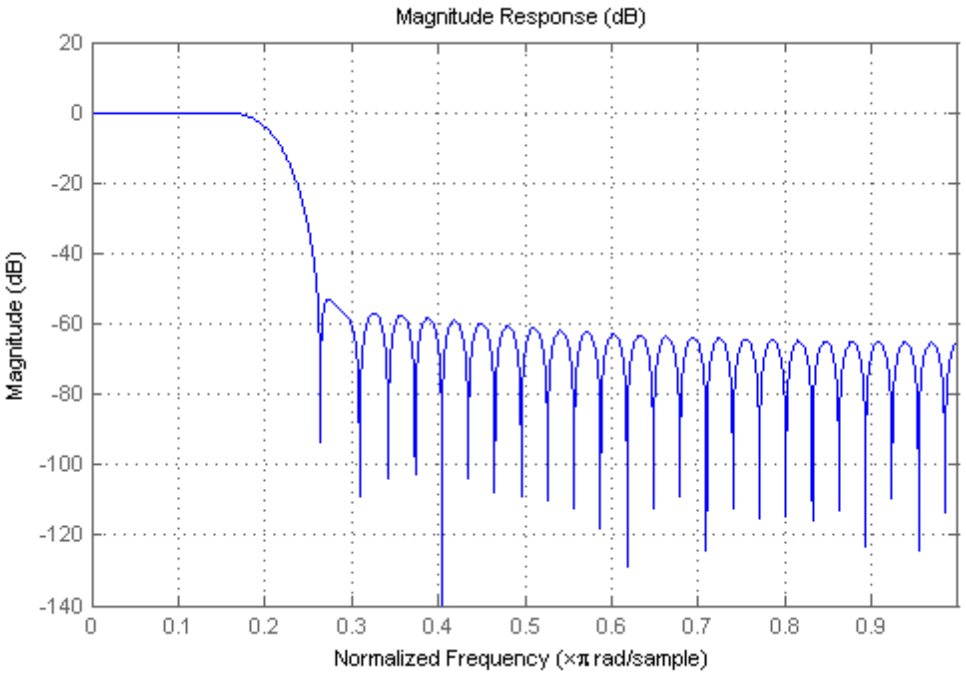

Figure 4.2: Frequency response of the 64-Tap FIR filter designed in MATLAB.

Fig. 4.3 shows the images of the experimental setup.

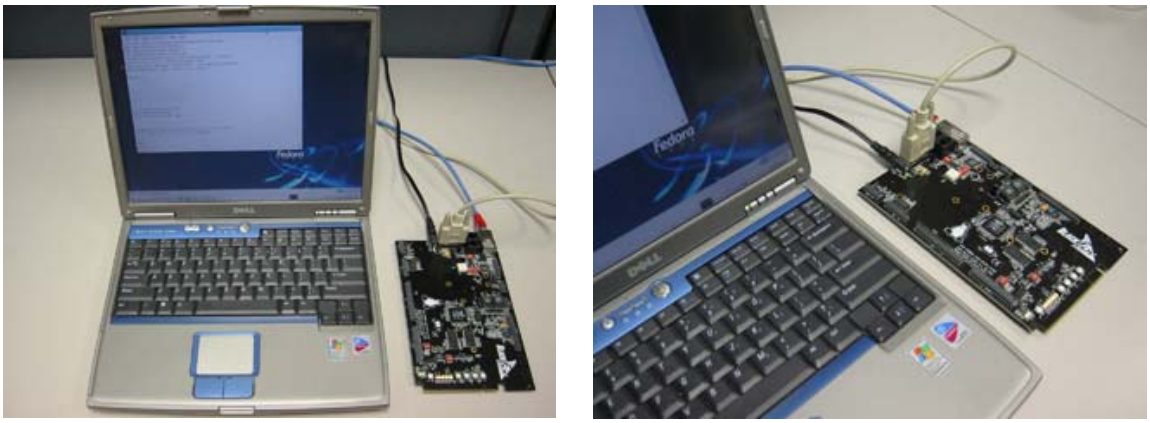

Figure 4.3: Experimental setup.

4.4 Results and Analysis

The results are summarized in Tables 4.3(a) and (b). Note that "S/s" stands for "samples per second" and "kS/s" stands for "thousand samples per second".

Table 4.3(a): FIR filter execution time for 48000 samples, in seconds.

N	**With floating point variables and cache turned OFF**	**With floating point variables and cache turned ON**	**With fixed point variables and cache turned ON**	**Estimated from formula**
8	4.04 s	0.72 s	0.05 s	0.24 s
16	7.78 s	1.41 s	0.09 s	0.43 s
32	15.26 s	2.79 s	0.16 s	0.82 s
64	30.23 s	5.59 s	0.29 s	1.58 s
128	60.02 s	11.19 s	0.57 s	3.12 s
256	120.45 s	22.48 s	1.12 s	6.19 s
512	240.65 s	45.07 s	2.23 s	12.34 s
1024	480.9 s	90.4 s	4.44 s	24.62 s

Table 4.3(b): Same as Table 4.3(a), except now expressed as effective sample rate.

N	**With floating point variables and cache turned OFF**	**With floating point variables and cache turned ON**	**With fixed point variables and cache turned ON**	**Estimated from formula**
8	11.9 kS/s	66.7 kS/s	960.0 kS/s	200.0 kS/s
16	6.2 kS/s	34.0 kS/s	533.3 kS/s	111.6 kS/s
32	3.1 kS/s	17.2 kS/s	300.0 kS/s	58.5 kS/s
64	1.6 kS/s	8.6 kS/s	165.5 kS/s	30.4 kS/s
128	800 S/s	4.3 kS/s	84.2 kS/s	15.4 kS/s
256	398 S/s	2.1 kS/s	42.8 kS/s	7.7 kS/s
512	199 S/s	1.1 kS/s	21.5 kS/s	3.9 kS/s
1024	100 S/s	531 S/s	10.8 kS/s	1.9 kS/s

4.4.1 Effect of Cache Accessing

Blackfin processors have three levels of memory, providing a trade-off between capacity and performance. Level-1 (L1) memory provides the highest performance with the lowest capacity. Level-2 (L2) and Level-3 (L3) memory provide much larger memory sizes, but typically have multiple cycle access times. Blackfin processors have on-chip data and instruction memory banks, which can be independently configured as cache [13]. There are two basic options when writing to a cache: (1) “write-through”, in which the information is written to both the block in the cache and to the block in the source memory, and (2) “write-back”, in which the information is written only to the block in the cache and the modified cache block is written to the main memory only when it is replaced. Write-back is much faster than the write-through. Instruction cache and data cache are turned off in the uClinux kernel by default. To turn on the cache, we have to “menuconfig” the kernel with cache features, recompile it, and reload it to the Blackfin processor. For our experiment, the write-back method has been used when the kernel was configured to use cache.

Fig. 4.4 shows the effect of cache access on the FIR filter performance. When the cache is available, the FIR filter gives approximately five times better performance (lower execution time). So, the cache should be used for DSP algorithms.

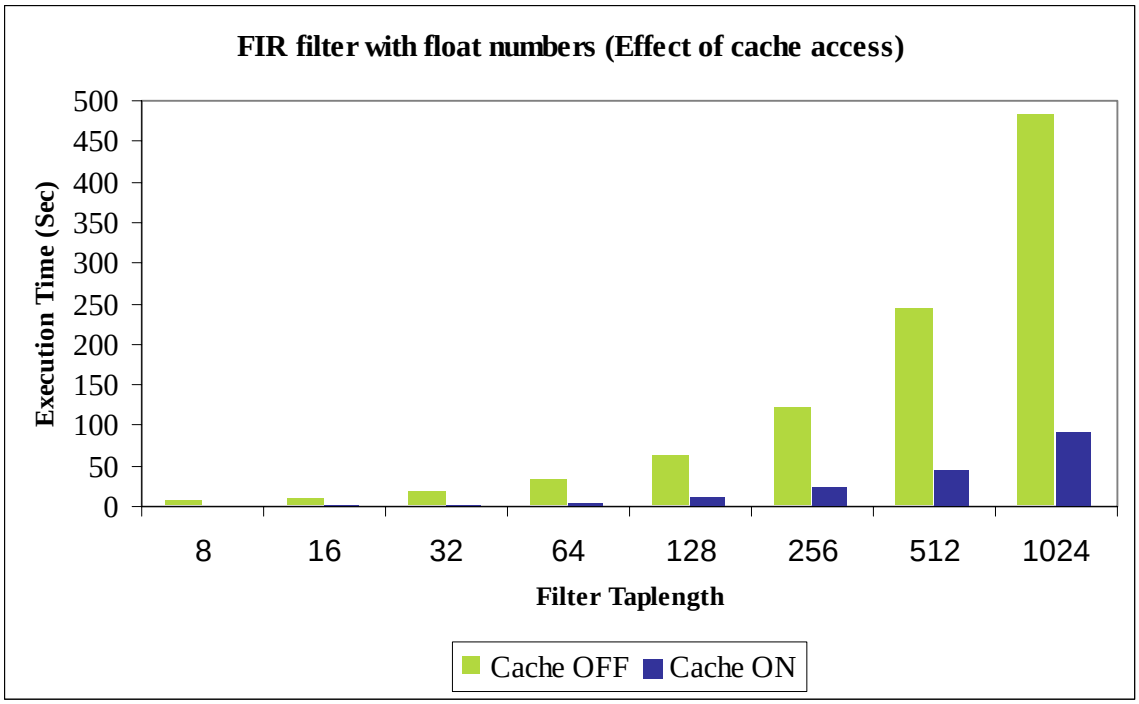

Figure 4.4: FIR filter performance as a function of *N* and cache use.

4.4.2 Effect of Data Types

Digital Signal Processors are divided into two broad categories: fixed-point and floating-point. Fixed-point families tend to be fast, low power, and low cost, while floating-point processors offer high precision and wide dynamic range in hardware [14]. In fixed-point number representation, the radix point is always at the same location which simplifies the numeric operations and conserves memory. But in situations that require a large range of numbers or high resolution, a changeable radix point is desirable. The Blackfin processor follows a fast floating-point emulation technique to improve floating-point computational efficiency on the fixed-point Blackfin processor platform [15]. However, it is still much slower than the fixed-point arithmetic. Fig. 4.5 shows how the performance improves when we change the type data from float to integer. It is noticeable that the execution time is around 17 times faster in fixed-point compare to floating-point operations.

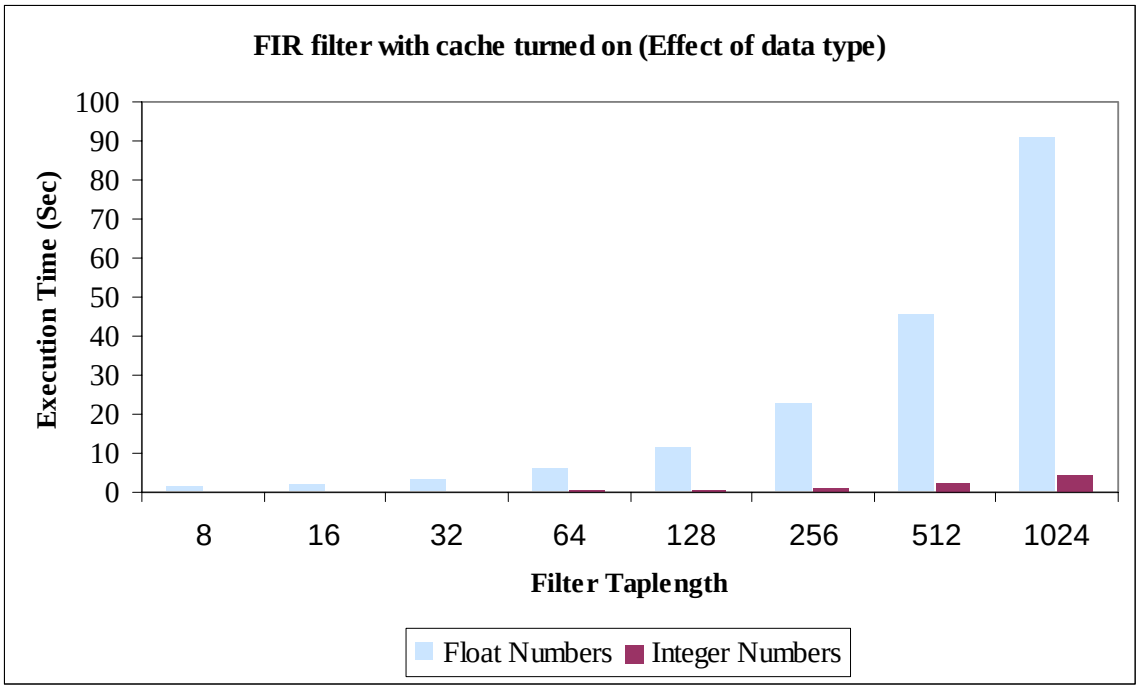

Figure 4.5: FIR filter performance by changing data types.

4.4.3 Comparison with Estimated Performance

Fig. 4.6 shows the comparison between the measured time and the calculated time. It shows that we are getting much better performance than is predicted by the Analog Devices formula, which is reasonable since that formula is intended to lower-bound performance.

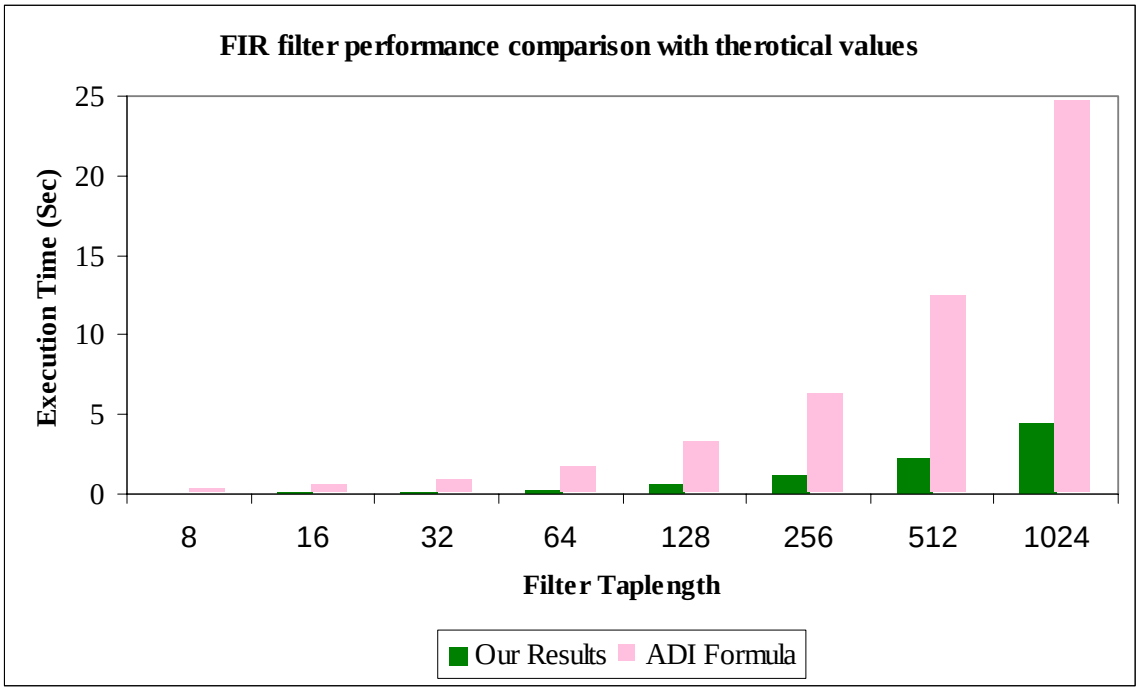

Figure 4.6: FIR filter performance comparison with theoretical values.

4.4.4 Execution Time Vs Tap-length

By analyzing all the results above it is evident that to get the better performance of the Blackfin processor, we need to use fixed-point operation with cache features enabled. Fig. 4.7 shows this ultimate performance for various tap-lengths.

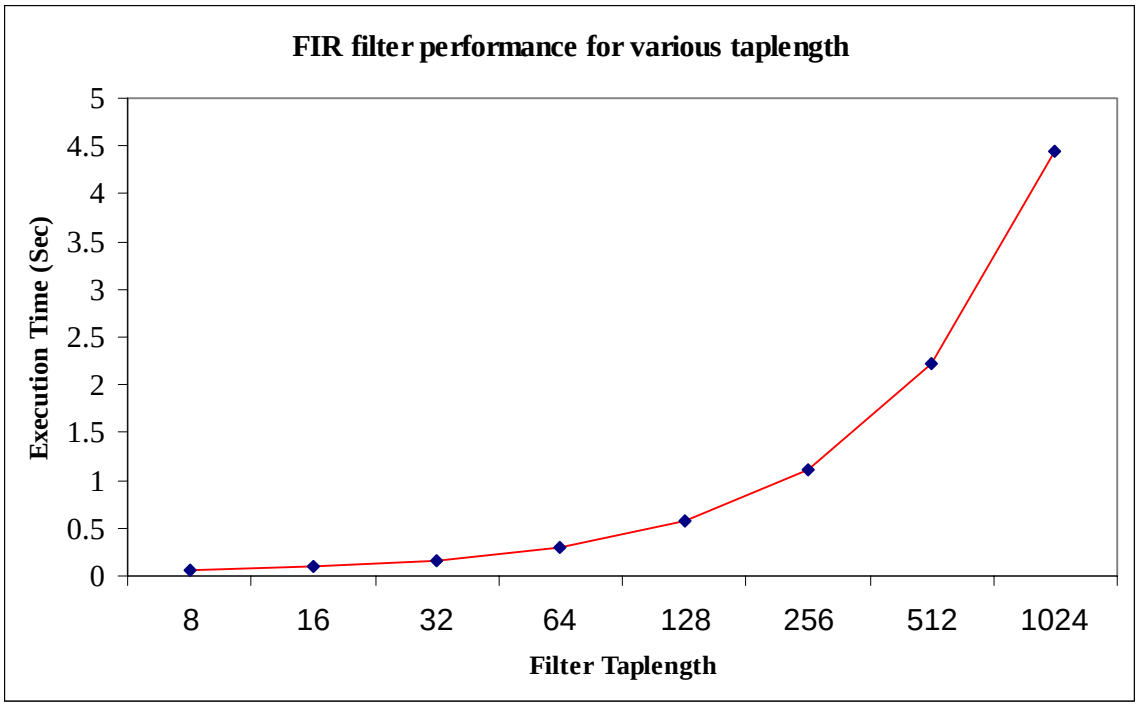

Figure 4.7: FIR filter performance for various tap-lengths.

References

[1] C. Hollabaugh, *Embedded Linux: Hardware, Software, and Interfacing*, Addison-Wesley Pearson Education, 2002, ISBN 0-672-32226-9.

[2] White paper, 'Unifying Microarchitecture for Embedded Media Processing', Analog Devices Inc., 2003, http://www.analog.com/UploadedFiles/White_Papers/.

[3] Blackfin processor family manuals, 'Getting Started with Blackfin Processors', Analog Devices Inc., September, 2005, http://www.analog.com/processors/processors/blackfin/.

[4] Blackfin processor family manuals, 'ADSP-BF537 Blackfin Processor Hardware Reference', Analog Devices Inc., January, 2005, http://www.analog.com/Processors/Processors/blackfin/technicalLibrary/manuals/blackfinIndex.html.

[5] Blackfin uClinux documentation website, http://docs.blackfin.uclinux.org.

[6] uClinux on Blackfin, Blackfin uClinux documentation website, http://docs.blackfin.uclinux.org/doku.php?id=uclinux_on_blackfin.

[7] Compilers and assemblers, Blackfin uClinux documentation website, http://docs.blackfin.uclinux.org/doku.php?id=compilers_and_assemblers .

[8] Das U-Boot on Blackfin, Blackfin uClinux documentation website, http://docs.blackfin.uclinux.org/doku.php?id=das_u-boot_on_blackfin.

[9] Terminal programs, Blackfin uClinux documentation website, http://docs.blackfin.uclinux.org/doku.php?id=terminal_programs.

[10] BF-537 Quick Start, Blackfin uClinux documentation website, http://docs.blackfin.uclinux.org/doku.php?id=bf537_quick_start.

[11] J. G. Ganssle, M. Barr, J. Ganssle, *Embedded Systems Dictionary*, CMP Books, 2003, ISBN: 1578201209.

[12] Benchmarks Comparison, http://www.analog.com/processors/processors/blackfin/benchmarks/comparison.html.

[13] Application Note, ‘Using Cache Memory on Blackfin Processors’, EE-271, Analog Devices Inc.

[14] Application Note, ‘Extended-Precision Fixed-Point Arithmetic on the Blackfin Processor Platform’, EE-186, Analog Devices Inc.

[15] Application Note, ‘Fast Floating-Point Arithmetic Emulation on the Blackfin Processor Platform’, EE-185, Analog Devices Inc.