May 8, 2007

Internet Radio running on uClinux

Final Project 2007 @ Columbia University

Embedded System Design: Computer Science & Electrical Engineering

PROFESSOR:	Stephen A. Edwards, PhD
TAS:	David Lariviere
	YingJian Gu
STUDENTS	Min Yang
	Sing Wang Ho
	Kai Li

1. Background

The internet is an increasingly popular communication media. The number of applications in the internet grows at an exponential rate. More and more applications are transferred to the internet, radio is one of them. There are many popular internet radio stations on the web already, a well-known server is http://www.shoutcast.com.

Our project aims to build an embedded system designed to get these radio streams and process it. Radio broadcasting over the internet is a lot different then broadcasting over internet. Unlike AM and FM broadcasting, where the signal is analog and is transmitted throught a carrier signal, internet radio transmit data as a digital signal encapsulated in a TCP/IP packet. Since bandwidth is costly with respect to the server (which is the equivalent of an antenna), the raw sound wave is rarely sent. In fact, in most cases these are transmitterd using the popular MPEG-I audio layer 3 or more commonly known as MP3 compression. This is a lossy compression techique, the original signal cannot be completely recovered. A waveform can be converted into a MP3 by converting it into frequency domain using discrete cosine transform, then filtering the signal, and finally, compressing it using huffman coding. The client, or receiver, performs the opposite: decoding, requantizing and perform inverse discrete cosine transform.

2. Table of Contents

1. Introduction

In this project, we design the client side for a shoutcast streaming server using Cyclone II DE2 boards. The board contains all the peripheral we will need. The board has an audio controller and an ethernet controller. The development tools we will use are Altera Quartus with SOPC builder. The objective is to be able to play up to 44kHz stereo MP3 from an internet server using the components built-in on DE2 board, namely DM9000A ethernet controller and WM8731 Wolfson Audio Codec. To accomplish this objective we also used nios uClinux and mpg123 mp3 player.

2. Overall Design

The operation of the internet radio can be divided into three operations: converting the TCP stream into MP3 frames, decoding the mp3, and playing the decoded mp3.

2.1 Converting the TCP/IP stream to MP3 frames

TCP/IP is a complicated protocol with many cases and many states. Programming TCP/IP in hardware makes a very fast controller but is very difficult due to the number of states and special cases it has to handle. Writing our own C code to manage the DM9000A is just as hard. The easiest way to accomplish this is to use uClinux's library and its pre-built controller. Hence, we implemented uClinux for nios onto the board to create internet capabilities. The stream that passes through is very similar to that of HTTP. It contains a header very similar to HTTP and the parsing method for the header is identical to that of HTTP. We use the mpg123 library to extract the mp3 frames from the TCP/IP stream. A sample of a TCP stream captured from www.shoutcast.com internet server is attached to the Appendix A. The mp3 frames can be obtained by splitting the frames at the mp3 header. Each mp3 frame always has a sync word which could be used to identify the headers. Using an open source program with uClinux, mpg123, we are able to retrieve an mp3 from the internet.

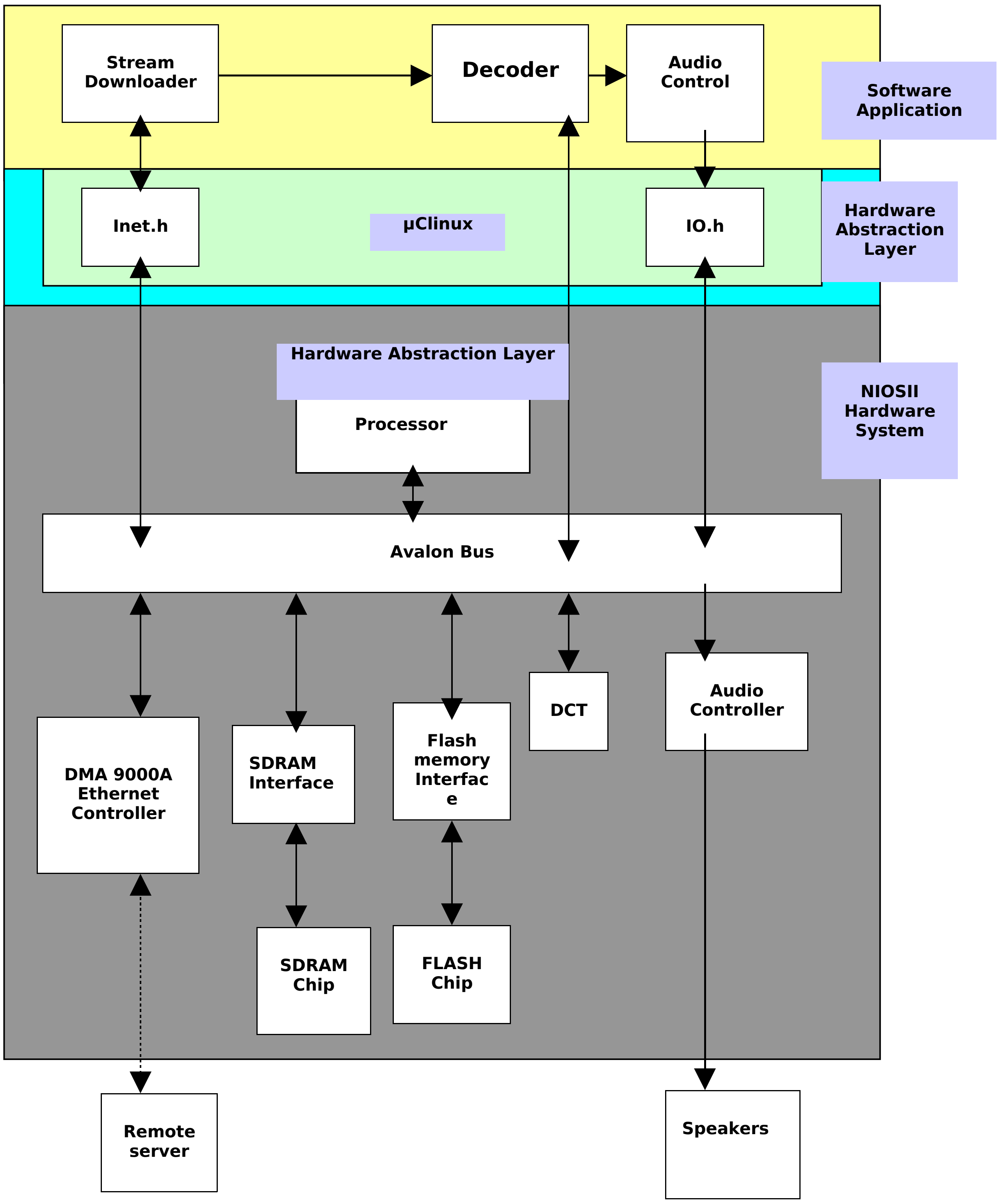
2.2 Mp3 Decoding

Mp3 decoding involves three main stages. Each frame is Huffman encoded with the code bits specified in the header. The first task is to decode the frame, which means decompressing the frame. There is a CRC checksum in the header as well, this can safely be ignored. Second, the decompressed frame is passed through a filterbank, which is a series of bandpass filters, requantized, and anti-aliasing is done in this stage. Finally, inverse discrete cosine transform (IDCT) is performed to convert the data back into time domain. Using mpg123 and a custom C audio driver, we can feed a 16-bit unsigned audio to the hardware and is ready to be played by the audio controller.

2.3 Stream to sound conversion

Finally, the decoded sound is passed to a custom audio controller which controls the audio signal of wm8731 sound controller. The audio controller also acts as a buffer for sound to be played to the speakers. This controller is built in hardware so we have the quickest and most direct access to the wm8731 controller.

3. System Architecture



4. Hardware Components

We used many components to build the hardware system. The hardware includes Cyclone-II level-3 Processor, Avalon Bus, 10/100 DM9000A Ethernet Controller with socket and 24-bit CD-Quality Audio Wolfson CODEC, 8 Mbits SDRAM, 4Mbits FLASH Memory, 16Kbits FIFO, one Custom Logic Unit dedicated to discrete Cosine Transform, Build-in USB Bluster for FPGA configuration and some LEDs for testing purpose.

The audio controller is configured to 18.43MHz using a phase lock loop. Since uClinux does not support fork, meaning that when the system decode, it cannot play sound at the same time. This means we have to buffer a chunk of audio data long enough for the decoder to decode the next set. To do this, we implement a First-In-First-Out (FIFO) buffer to hold every chunk of sound data. To optimize speed and minimize space, we set the size of the buffer to match the size of each chunk received from the C program, which is 16kBytes. This creates the maximum delay after filling the buffer without using too much M4Ks on the board. To prevent a overflowing the buffer, the system performs a busy-wait (or polling) on the audio controller. The audio controller is responsible for returning the number of data left in buffer so the software system when to write without overflowing the buffer. The audio controller is configured to be able to run at three rates. These rates can be changed by writing to a pre-defined address the audio controller owns. The rates are 44kHz, 22kHz and 11kHz. The audio controller achieves these rates without changing the configuration of the wm8731 controller. The wm8731 is pre-configured to play at 44kHz. The other rates are then achieved by playing a sample multiple times.

We ran into several problems in implementing the audio controller. When the FIFO was initially implemented, it did not fit on the board. After looking at the fitter resource report, we realize that there is a large block of on-chip memory which used up most of our M4Ks. This block was not necessary and was removed from the project. Wiring the ports for the audio controller is also tricky. Since there are three clocks (which are the system clock, the audio clock and a external audio clock), it is easy to accidentally miss use clock as a system clock.

We also designed a custom component which calculates the discrete cosine transform (DCT) of a 128kBytes sample with each sample being 4Bytes long. The calculation is identical to dct64.c done in mpg123. The data is fed into an array of register with size 64 with each cell being 4Bytes long via the function writel(). The data is fed into to the address and then a write to (base_address + 64) is used tell the DCT that we're done sending data. Then we do a busy wait on the same address to wait for the process to finish. The user can then access the data by using readl().

5. uClinux Operation System

The uClinux was, in fact, not part of our original design to implement the radio controller. After finding out that we need to create a TCP/IP stream which requires writing a hardware driver to handle the Berkeley sockets in C library. We soon find out that it is very difficult and time consuming to write the IO control to handle the Berkeley socket in C library. When we try to resolve for an alternative solution, we discovered uClinux.

uClinux is a very compact version of linux that can fit into thumb drives. After some research, we discovered a nios version which also supports DM9000A. We decided to try it out and chose uClinux-dist-20070130 release as our operation system. This is the most recent stable release at the time this project is has started. This version has good online references and includes Ethernet driver (but no audio device driver). For reference check the website:

http://nioswiki.jot.com/WikiHome/OperatingSystems/%C2%B5Clinux/UClinuxDist

To compile an image running to run on the DE2 board, we have to download a cross-compiler toolchain. The purpose of the cross-compiler is to allow compilation of a platform other than the current system's platform. In our case, the cross-compiler compiles for a nios systems on Redhat linux. Since this toolchain is not part of our environment, we have to add it to the PATH variable in our environment before compilation. Suppose the tool chain is installed in: ~/download/opt, then we set the path by typing:

```
PATH=$PATH:~/download/opt/nios2/bin
```

We configure the kernel by selecting the following options which enables ethernet when compilation is completed:

```
Vendor/Product Selection --->
--- Select the Vendor you wish to target
(Altera) Vendor
--- Select the Product you wish to target
(nios2nommu) Altera Products

Kernel/Library/Defaults Selection --->
(linux-2.6.x) Kernel Version
(None) Libc Version
[ ] Default all settings (lose changes)
[*] Customize Kernel Settings
[ ] Customize Vendor/User Settings
[ ] Update Default Vendor Settings
```

We setup memory and io port address map of our Nios II board by:

```
make vendor_hwselect SYSPTF=~/download/NET2.ptf
```

Networking support is enabled on the DE2 board using the following kernel options:

```
Networking -->
[*] Networking support
Networking options --->
<*> Packet socket
<*> Unix domain sockets
[*] TCP/IP networking

Device Drivers -->Network device support —>
[*] Network device support
[*] Ethernet (10 or 100Mbit)

[ ] SMC 91C9x/91C1xxx support

[ ] Opencores (Igor) Emac support
[ ] MoreThanIP 10_100_1000 Emac support
[ ] DM9000 support
[*] DM9000A with checksum offloading
```

We type 'make' and 'make linux image' to build an image. Next we install the kernel and start the kernel by the following command:

```
PATH=$PATH:/usr/cad/altera/6.1/nios2eds/

sdk_shell

nios2-download -g ~/download/uClinux-dist/images/zImage

nios2-terminal
```

After booting the uClinux, we need to enable the ethernet connection. If there is dhcp, then we can type 'dhcpcd' after bring the ethernet control up. Assume that we do not, then we can assign a static IP to the device as follows:

```
ifconfig eth0 up

ifconfig eth0 192.168.1.2

route add default gw 192.168.1.1
```

Now the uClinux system is network enabled, the applications running on it can access internet by specifying the IP address, port number and file directory on the remote server:

```
mpg123 -@http://192.168.1.1:8000/ices.m3u
```

If we want to install a typical gcc program, which has the general 'configure', 'make' and 'make install' installation commands, we need to tell the system to invoke the cross-compiler instead of the standard gcc compiler. To do this, we need to set certain flags when configuring. In compiling mpg123, we used the following command and flags:

```
./configure –host=nios2-linux-uclibc –target=nios2-linux-uclibc –with-cpu=generic_nofpu

--with-audio=dummy CC=”nios2-linux-uclibc-gcc –D_KERNEL__ -elf2flt”

--with-optimization=4 --enable-gapless=no
```

The explanation for each flag is as follows:

- elf2flt: This flag converts the default elf file format for nios to flat format needed by uClinux
- with-cpu=generic_nofpu: This is an optimization flag to turn off floating point.
- D__KERNEL__: This flag causes the toolchain to include kernel libraries which gives us access to read and write ios to communicate with our hardware components
- with-opimization=4: This flag cause the kernel to perform optimization in compile stage.
- enable-gapless=no: This is a flag specific to mpg123 to turn off gapless which is a feature that removes 'blank' sound from mp3s but sacrificing speed.
- with-audio=dummy: Since mpg123 supports multiple sound drivers (alsa, oss, etc.), it also has a 'driver' with empty functions. We use this to write about custom sound driver.

When the configuration is finished, we use the 'make' command to compile the program. We should check that the system has invoked the correct compiler. When make is done, we grab the program executable and put it in ~/download/uClinux-dist/romfs/bin and then rebuild the uClinux's kernel image using 'make linux image'.

6. Software Architecture

The mpg123 is one of the two most popular open-source encoders available. mpg123's library is used by many programs such as xine. The other library that is frequently used is libMAD. We compared the size of mpg123 and libMAD and discover that mpg123 is significantly smaller in file size than mpg123. This usually means a smaller program size and hence a smaller kernel image. Furthermore, mpg123 has built-in ICY support which also plays shoutcast streams. This makes it the better library to use.

In mpg123, the program initializes two tables for synthesizing sound: cosine table and decwin table. The program then fetches the stream or file and add it to the playlist and play begin decoding the mp3. Mpg123 decompress the signal and perform filterbank operation together to optimize speed. Discrete cosine transform type-II using the cosine table generated at startup is and the well-known butterfly method is operated on the result. Lastly, we do windowing, which is a type of filter in filterbank, on the samples using the decwin table. The final output is stored into a temporary buffer which is later fed to the audio controller when 16kB of samples have been decoded.

Mpg123 provides multiple audio drivers. There is a dummy file which contains the function headers with an empty body. We decide that this is a good place to write our audio controller. The main function that is called is audio_play_samples() which is where the samples is played. We perform the busy-wait and writes into the buffer mentioned in the 'Hardware Implementation' section. Every time this function is called, the rate is written to the pre-defined address in the audio controller to configure the channel to play that the requested rate.

7. System Optimization

7.1 Software Optimization

The mpg123 program is designed to be a very fast library. However, it also added features that slow it down such as gapless mentioned in section 5. uClinux Operating System. Another feature is the transfer of memory. This is not necessary and it tries to invoke a fork hence it is also removed from the simulation. TermIOs is also another feature that ran inside the play_frame() loop, which is also turned off. Next, all redundant codes in the system should be removed. For example, the equalizer is taken out of the program. After changing these options, the decoding speed of a benchmarked mp3 went from 5mins13secs to 4mins59secs, hence an improvement of 5%.

More importantly, the floating point should be turned off using the flags mentioned in section 5. uClinux Operating System. This converts all floating points into type long. This prevents the cpu from performing floating multiplication and division which is VERY slow. The imporvement is almost tripled (270%).

7.2 Hardware Optimization

Since we have a time constraint, converting the entire mp3 in hardware is infeasible. There are three areas in the code where we could choose to do optimization: Huffman decoding, filterbank and DCT. We chose to optimize the DCT because it performs we realize all its additions, subtraction, and multiplications can be done concurrently.

In theory, each cosine point can be calculated in one clock cycle if the butterfly method is used. This means we can multiply, subtract and add in the same cycle to get the results for the next point. However, multipliers are expensive in terms of lookup tables, we decided to only initiate one, and the multiplications becomes the longest delay in calculating each point for the cosine table. A modelSim of a cosine point being calculated is attached to the Appendix of this file. From the waveform, we can see using the addition and subtraction as dividers for each cosine point. After this optimization is completed, we benchmarked the time and found out the delay has been reduced by 20%.

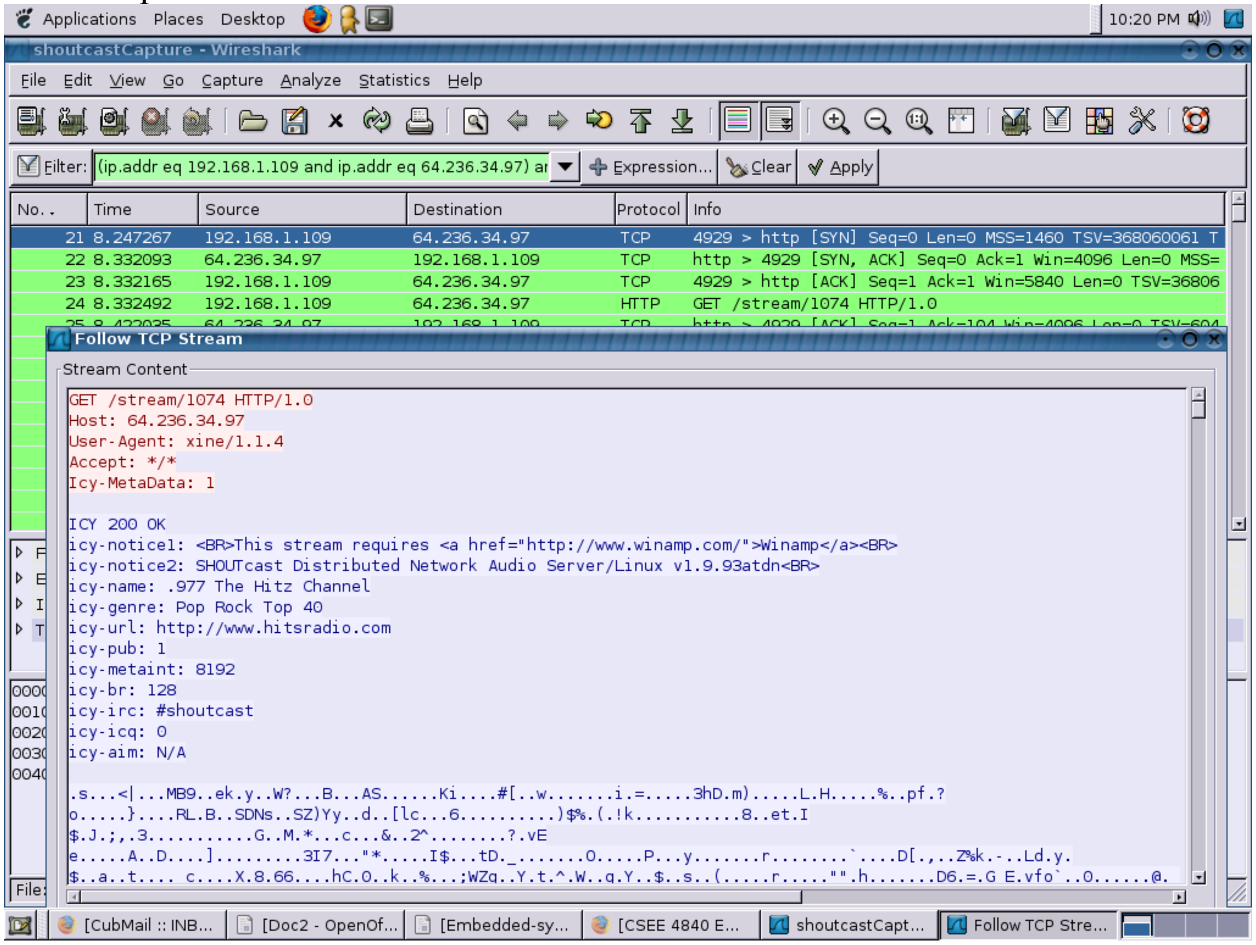
However, the audio controller still fails to play at the required rate of 44kHz. The reduction was not even significant enough for the sound to play at 22kHz. It was found that another bottleneck is windowing. This process performs more than 520 multiplications and would induce a huge delay. We done some experiments by removing windowing from the code and running it in test mode, which is a “decode only” option,and found out this is the bottleneck of decoding the MP3. The delay was reduced by about 220%.

8. Conclusion

In Conclusion, we set out to design and implement a peripheral that plays music from the internet at 44kHz. We could only play music at 11kHz seamlessly. This is because we fail to identify the biggest bottleneck. Though specialized multipliers are expensive in terms of lookup table's usage, they save a lot of CPU cycles and should be studied carefully when doing optimization.

APPENDIX - A

An example of TCP stream from www.shoutcast.com server



Applications Places Desktop
10:20 PM
shoutcastCapture - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: (ip.addr eq 192.168.1.109 and ip.addr eq 64.236.34.97) ar
Expression... Clear Apply
Follow TCP Stream
Stream Content
[CubMail :: INB... [Doc2 - OpenOf... [Embedded-sy... [CSEE 4840 E... shoutcastCapt... Follow TCP Stre...

```
No. .  Time        Source          Destination     Protocol  Info
    21 8.247267    192.168.1.109   64.236.34.97    TCP       4929 > http [SYN] Seq=0 Len=0 MSS=1460 TSV=368060061 T
    22 8.332093    64.236.34.97    192.168.1.109   TCP       http > 4929 [SYN, ACK] Seq=0 Ack=1 Win=4096 Len=0 MSS=
    23 8.332165    192.168.1.109   64.236.34.97    TCP       4929 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=36806
    24 8.332492    192.168.1.109   64.236.34.97    HTTP      GET /stream/1074 HTTP/1.0
```

```
GET /stream/1074 HTTP/1.0
Host: 64.236.34.97
User-Agent: xine/1.1.4
Accept: */*
Icy-MetaData: 1

ICY 200 OK
icy-notice1: <BR>This stream requires <a href="http://www.winamp.com/">Winamp</a><BR>
icy-notice2: SHOUTcast Distributed Network Audio Server/Linux v1.9.93atdn<BR>
icy-name: .977 The Hitz Channel
icy-genre: Pop Rock Top 40
icy-url: http://www.hitsradio.com
icy-pub: 1
icy-metaint: 8192
icy-br: 128
icy-irc: #shoutcast
icy-icq: 0
icy-aim: N/A

.s...<|...MB9..ek.y..W?...B...AS......Ki....#[..w.......i.=.....3hD.m).....L.H.....%..pf.?
o.....}....RL.B..SDNs..SZ)Yy..d..[lc...6..........)$%.(.!k...........8..et.I
$.J.;,.3...........G..M.*...c...&..2^........?.vE
e.....A..D....].........3I7..."*.....I$...tD._.......O.....P...y.......r........`....D[.,..Z%k.-..Ld.y.
$..a..t.... c....X.8.66....hC.O..k..%...;WZq..Y.t.^.W..q.Y..$..s..(.....r....."".h.......D6.=.G E.vfo`..O......@.
```

Appendix B
dct64.vhd

```
library IEEE;

use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity dct is
  port (
    reset_n: in std_logic;
    clk: in std_logic;
    waitrequest : out std_logic;
    byteenable: in std_logic_vector(3 downto 0);
    begintransfer: in std_logic;
    chipselect: in std_logic;
    read: in std_logic;
    write: in std_logic;
    address: in std_logic_vector(18 downto 0);
    writedata: in std_logic_vector(31 downto 0);
    readdata : out std_logic_vector(31 downto 0)
  );
end dct;

architecture behav of dct is

type buf_type  is array (31 downto 0) of std_logic_vector (31 downto 0);
type buf_type0 is array (15 downto 0) of std_logic_vector (31 downto 0);
type buf_type1 is array (7 downto 0) of std_logic_vector (31 downto 0);
type buf_type2 is array (3 downto 0) of std_logic_vector (31 downto 0);
type buf_type3 is array (1 downto 0) of std_logic_vector (31 downto 0);

signal pnt0  : buf_type0;
signal pnt1  : buf_type1;
signal pnt2  : buf_type2;
signal pnt3  : buf_type3;
signal pnt4  : std_logic_vector (31 downto 0);

signal buf1,buf2 : buf_type;
signal index, counter  : integer;
signal start,start_pre : std_logic;
signal end_flg     : std_logic;

signal add_a : buf_type;
signal add_b : buf_type;
signal add_result : buf_type;

signal sub_a : buf_type;
signal sub_b : buf_type;
signal sub_result : buf_type;

signal mult_a: std_logic_vector (31 downto 0);
signal mult_b: std_logic_vector (31 downto 0);
signal mult_result: std_logic_vector (31 downto 0);

signal i,j,m,n : integer;
signal operation_done : std_logic;

component add is
      port
      (
            dataa       : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            datab       : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            result            : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
      );
end component;

component sub IS
      PORT
      (
            dataa       : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            datab       : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            result            : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
      );
end component;

component mul_wrapper IS
      PORT
      (
            dataa       : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            datab       : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            result64          : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)

     );
end component;

begin
waitrequest <= '0';
-----------------------------------------------------
--  Make 16 adders
-----------------------------------------------------
loop_add_inst:
for i in 0 to 16 generate
      add_inst : add PORT MAP (
            dataa  => add_a(i),
            datab  => add_b(i),
            result       => add_result(i)
      );
end generate;
-----------------------------------------------------
--  Make 16 subtractors
-----------------------------------------------------
loop_sub_inst:
for j in 0 to 16 generate
      sub_inst : sub PORT MAP (
            dataa  => sub_a(j),
            datab  => sub_b(j),
            result       => sub_result(j)
      );
end generate;

-----------------------------------------------------
--  Make only 1 multiplier
--  64bits*64bits=128bits, only the lower 64bits of mult_result are used
```

```
-----------------------------------------------------
mult_inst : mul_wrapper PORT MAP (
            dataa  => mult_a,
            datab  => mult_b,
            result64     => mult_result

      );

-----------------------------------------------------
--  save previous value of "start" signal
-----------------------------------------------------
process (clk,reset_n)
begin
  if reset_n='0' then
      start_pre <= '0';
  elsif (clk'event and clk='1') then
      start_pre <= start;
  end if;
end process;


-----------------------------------------------------
--  counter begin from "start" signal
--  end @ 146
--  generate "end-flg" signal
--  end-flg <= '1' when counter=146
-----------------------------------------------------

operation_done <= '1' when counter = 89 else '0';

process (clk,reset_n)
begin
  if reset_n='0' then
    start <= '0';
  elsif (clk'event and clk='1') then
      if(chipselect='1' and write='1' and index=64) then
        start<=writedata(0);
    elsif ( operation_done = '1' ) then
        start <= '0';
    end if;
  end if;
end process;

process (clk,reset_n)
begin
  if reset_n='0' then
      counter <= 0;
      end_flg <= '0';
  elsif (clk'event and clk='1') then
      if (start_pre='0' and start='1') then
            counter <= 1;
          end_flg <= '0';
      elsif (operation_done = '1' ) then
            counter <= 89;
          end_flg <= '1';
      elsif ( start='1') then
          counter <= counter +1;
      end if;
  end if;
end process;


-----------------------------------------------------
--  transfer address from avalon to integer
-----------------------------------------------------
index <= conv_integer(address(7 downto 0));

-----------------------------------------------------
--  output "end_flg" to avalon bus
-----------------------------------------------------
process(chipselect, read, index, buf1, buf2)
begin
      if (chipselect='1' and read='1') then
        for i in 0 to 31 loop
          if(index = i)   then readdata <= buf1(i)(31 downto 0); end if;
        end loop;
        for i in 0 to 31 loop
          if(index = i +32) then readdata <= buf2(i)(31 downto 0); end if;
        end loop;
      if(index = 64) then
        readdata <= "0000000000000000000000000000000" & end_flg;
      end if;
    else
      readdata <= (others => '0');
    end if;
end process;

process (clk,reset_n)
begin
  if reset_n='0' then
     for i in 0 to 31 loop
          buf1(i) <= (others=>'0');
          buf2(i) <= (others=>'0');
       end loop;
  elsif (clk'event and clk='1') then
      if ( chipselect='1' and write='1' ) then
        for i in 0 to 31 loop
          if (index = i ) then
            buf1(i)(31 downto 0) <= writedata;
              buf2(i)(31 downto 0) <= writedata;
            end if;
        end loop;
    elsif (start_pre='1' and start='1') then
-----------------------------------------------------------------------------
-- pnt0
-----------------------------------------------------------------------------
    if (counter = 1) then
      for i in 0 to 15 loop
        buf1(i)    <= add_result(i);
        buf1(i+16) <= sub_result(i);
      end loop;
    end if;
      for i in 0 to 15 loop
        if (counter = i+2) then buf1(i+16) <= mult_result; end if;
      end loop;
------------------------------------------------------------------------------
```

```
-- pnt1
-----------------------------------------------------------------------------
    if(counter = 18) then
      for i in 0 to 7 loop
        buf2(i     )<= add_result(i);
        buf2(i+8   )<= sub_result(i);
        buf2(i  +16)<= add_result(i+8);
        buf2(i+8+16)<= sub_result(i+8);
      end loop;
    end if;
    for i in 0 to 7 loop
       if (counter = i+19) then buf2(i+8   ) <= mult_result; end if;
    end loop;
    for i in 0 to 7 loop
       if (counter = i+27) then buf2(i+8+16) <= mult_result; end if;
    end loop;
-----------------------------------------------------------------------------
-- pnt2
-----------------------------------------------------------------------------
    if(counter = 35) then
      for i in 0 to 3 loop
        buf1(i       )<= add_result(i);
        buf1(i+4     )<= sub_result(i);
        buf1(i  +8   )<= add_result(i+4);
        buf1(i+4+8   )<= sub_result(i+4);
        buf1(i    +16)<= add_result(i+8);
        buf1(i+4  +16)<= sub_result(i+8);
        buf1(i  +8+16)<= add_result(i+12);
        buf1(i+4+8+16)<= sub_result(i+12);
      end loop;
    end if;
    for i in 0 to 3 loop
       if (counter = i+36) then buf1(i+4     ) <= mult_result; end if;
    end loop;
    for i in 0 to 3 loop
       if (counter = i+40) then buf1(i+4+8   ) <= mult_result; end if;
    end loop;
    for i in 0 to 3 loop
       if (counter = i+44) then buf1(i+4  +16) <= mult_result; end if;
    end loop;
    for i in 0 to 3 loop
       if (counter = i+48) then buf1(i+4+8+16) <= mult_result; end if;
    end loop;
-----------------------------------------------------------------------------
-- pnt3
-----------------------------------------------------------------------------
    if(counter = 52) then
      for i in 0 to 1 loop
        buf2(i         )<= add_result(i);
        buf2(i+2       )<= sub_result(i);
        buf2(i  +4     )<= add_result(i+2);
        buf2(i+2+4     )<= sub_result(i+2);
        buf2(i    +8   )<= add_result(i+4);
        buf2(i+2  +8   )<= sub_result(i+4);
        buf2(i  +4+8   )<= add_result(i+6);
        buf2(i+2+4+8   )<= sub_result(i+6);
        buf2(i      +16)<= add_result(i+8);
        buf2(i+2    +16)<= sub_result(i+8);
        buf2(i  +4  +16)<= add_result(i+10);
        buf2(i+2+4  +16)<= sub_result(i+10);
        buf2(i    +8+16)<= add_result(i+12);
        buf2(i+2  +8+16)<= sub_result(i+12);
        buf2(i  +4+8+16)<= add_result(i+14);
        buf2(i+2+4+8+16)<= sub_result(i+14);
      end loop;
    end if;
    for i in 0 to 1 loop
       if (counter = i+53) then buf2(i+2   ) <= mult_result; end if;
    end loop;
    for i in 0 to 1 loop
       if (counter = i+55) then buf2(i+2+4 ) <= mult_result; end if;
    end loop;
    for i in 0 to 1 loop
       if (counter = i+57) then buf2(i+2+8 ) <= mult_result; end if;
    end loop;
    for i in 0 to 1 loop
       if (counter = i+59) then buf2(i+2+12) <= mult_result; end if;
    end loop;
    for i in 0 to 1 loop
       if (counter = i+61) then buf2(i+2+16) <= mult_result; end if;
    end loop;
    for i in 0 to 1 loop
       if (counter = i+63) then buf2(i+2+20) <= mult_result; end if;
    end loop;
    for i in 0 to 1 loop
       if (counter = i+65) then buf2(i+2+24) <= mult_result; end if;
    end loop;
    for i in 0 to 1 loop
       if (counter = i+67) then buf2(i+2+28) <= mult_result; end if;
    end loop;
----------------------------------------------------------------------------
-- pnt4
----------------------------------------------------------------------------
    if(counter = 69) then
      for i in 0 to 15 loop
        buf1(i*2  )<= add_result(i);
        buf1(i*2+1)<= sub_result(i);
      end loop;
    end if;
    for i in 0 to 15 loop
      if(counter = i+70) then buf1(i*2+1) <= mult_result; end if;
    end loop;
----------------------------------------------------------------------------
    if(counter = 86) then
      for i in 0 to 7 loop
        buf1(2+i*4) <= add_result(i);
      end loop;
    end if;
    if(counter = 87) then
      for i in 0 to 3 loop
        buf1(4+i*8) <= add_result(i*3);
        buf1(6+i*8) <= add_result(i*3+1);
        buf1(5+i*8) <= add_result(i*3+2);
      end loop;
    end if;
    if(counter = 88) then
```

```
      for i in 0 to 1 loop
        buf1(8+i*16)  <= add_result(i*7);
        buf1(12+i*16) <= add_result(i*7+1);
        buf1(10+i*16) <= add_result(i*7+2);
        buf1(14+i*16) <= add_result(i*7+3);
        buf1(9+i*16)  <= add_result(i*7+4);
        buf1(13+i*16) <= add_result(i*7+5);
        buf1(11+i*16) <= add_result(i*7+6);
      end loop;
    end if;

------------------------------------------------------------------------------
    end if; --start_pre start
  end if;   --clock'event
end process;

-----------------------------------------------------
--  assign input value to adder/sub/multiplier
-----------------------------------------------------
process(counter,add_a,add_b,sub_a,sub_b,mult_a,mult_b)
begin
loop_add_assign: for m in 0 to 31 loop
    add_a(m) <= (others=>'0');
    add_b(m) <= (others=>'0');
end loop;
loop_sub_assign: for n in 0 to 31 loop
    sub_a(n) <= (others=>'0');
    sub_b(n) <= (others=>'0');
end loop;
mult_a <= (others=>'0');
mult_b <= (others=>'0');

----------------------------------------------------------------------------
-- pnt0
----------------------------------------------------------------------------
  if(counter = 1) then
    for i in 0 to 15 loop
      add_a(i) <= buf1(i);
      add_b(i) <= buf1(31-i);
      sub_b(i) <= buf2(16+i);
      sub_a(i) <= buf2(15-i);
    end loop;
  end if;
  for i in 0 to 15 loop
    if(counter = i+2) then
      mult_a <= buf1(i+16);
      mult_b <= pnt0(15-i);
    end if;
  end loop;
-----------------------------------------------------------------------------
-- pnt1
-----------------------------------------------------------------------------
  if (counter = 18) then
    for i in 0 to 7 loop
      add_a(i) <= buf1(i);
      add_b(i) <= buf1(15-i);
      sub_b(i) <= buf1(8+i);
      sub_a(i) <= buf1(7-i);
      add_a(i+8) <= buf1(0+i +16);
      add_b(i+8) <= buf1(15-i+16);
      sub_b(i+8) <= buf1(7-i +16);
      sub_a(i+8) <= buf1(8+i +16);
    end loop;
  end if;
  for i in 0 to 7 loop
    if(counter = i+19) then
      mult_a <= buf2(i+8);
      mult_b <= pnt1(7-i);
    end if;
  end loop;
  for i in 0 to 7 loop
    if(counter = i+27) then
      mult_a <= buf2(i+8+16);
      mult_b <= pnt1(7-i);
    end if;
  end loop;
-----------------------------------------------------------------------------
-- pnt2
-----------------------------------------------------------------------------
  if (counter = 35) then
    for i in 0 to 3 loop
      add_a(i)    <= buf2(i);
      add_b(i)    <= buf2(7-i);
      sub_b(i)    <= buf2(4+i);
      sub_a(i)    <= buf2(3-i);

      add_a(i+4)  <= buf2(0+i +8);
      add_b(i+4)  <= buf2(7-i +8);
      sub_b(i+4)  <= buf2(3-i +8);
      sub_a(i+4)  <= buf2(4+i +8);

      add_a(i+8)  <= buf2(0+i +16);
      add_b(i+8)  <= buf2(7-i +16);
      sub_b(i+8)  <= buf2(4+i +16);
      sub_a(i+8)  <= buf2(3-i +16);

      add_a(i+12) <= buf2(i   +24);
      add_b(i+12) <= buf2(7-i +24);
      sub_b(i+12) <= buf2(3-i +24);
      sub_a(i+12) <= buf2(4+i +24);
    end loop;
  end if;
  for i in 0 to 3 loop
    if(counter = i+36) then
      mult_a <= buf1(i+4);
      mult_b <= pnt2(3-i);
    end if;
  end loop;
  for i in 0 to 3 loop
    if(counter = i+40) then
      mult_a <= buf1(i+4+8);
      mult_b <= pnt2(3-i);
    end if;
  end loop;
  for i in 0 to 3 loop
```

```
    if(counter = i+44) then
      mult_a <= buf1(i+4+16);
      mult_b <= pnt2(3-i);
    end if;
  end loop;
  for i in 0 to 3 loop
    if(counter = i+48) then
      mult_a <= buf1(i+4+8+16);
      mult_b <= pnt2(3-i);
    end if;
  end loop;
--------------------------------------------------------------------------------
-- pnt3
--------------------------------------------------------------------------------
  if (counter = 52) then
    for i in 0 to 1 loop
      add_a(i)    <= buf1(i);
      add_b(i)    <= buf1(3-i);
      sub_b(i)    <= buf1(2+i);
      sub_a(i)    <= buf1(1-i);

      add_a(i+2)  <= buf1(i   +4);
      add_b(i+2)  <= buf1(3-i +4);
      sub_b(i+2)  <= buf1(1-i +4);
      sub_a(i+2)  <= buf1(2+i +4);

      add_a(i+4)  <= buf1(0+i +8);
      add_b(i+4)  <= buf1(3-i +8);
      sub_b(i+4)  <= buf1(2+i +8);
      sub_a(i+4)  <= buf1(1-i +8);

      add_a(i+6)  <= buf1(0+i +12);
      add_b(i+6)  <= buf1(3-i +12);
      sub_b(i+6)  <= buf1(1-i +12);
      sub_a(i+6)  <= buf1(2+i +12);

      add_a(i+8)  <= buf1(0+i +16);
      add_b(i+8)  <= buf1(3-i +16);
      sub_b(i+8)  <= buf1(2+i +16);
      sub_a(i+8)  <= buf1(1-i +16);

      add_a(i+10) <= buf1(0+i +20);
      add_b(i+10) <= buf1(3-i +20);
      sub_b(i+10) <= buf1(1-i +20);
      sub_a(i+10) <= buf1(2+i +20);

      add_a(i+12) <= buf1(i   +24);
      add_b(i+12) <= buf1(3-i +24);
      sub_b(i+12) <= buf1(2+i +24);
      sub_a(i+12) <= buf1(1-i +24);

      add_a(i+14) <= buf1(0+i +28);
      add_b(i+14) <= buf1(3-i +28);
      sub_b(i+14) <= buf1(1-i +28);
      sub_a(i+14) <= buf1(2+i +28);
    end loop;
  end if;
  for i in 0 to 1 loop
```

```
    if(counter = i+53) then
      mult_a <= buf2(i+2);
      mult_b <= pnt3(1-i);
    end if;
  end loop;
  for i in 0 to 1 loop
    if(counter = i+55) then
      mult_a <= buf2(i+2+4);
      mult_b <= pnt3(1-i);
    end if;
  end loop;
  for i in 0 to 1 loop
    if(counter = i+57) then
      mult_a <= buf2(i+2+8);
      mult_b <= pnt3(1-i);
    end if;
  end loop;
  for i in 0 to 1 loop
    if(counter = i+59) then
      mult_a <= buf2(i+2+12);
      mult_b <= pnt3(1-i);
    end if;
  end loop;
  for i in 0 to 1 loop
    if(counter = i+61) then
      mult_a <= buf2(i+2+16);
      mult_b <= pnt3(1-i);
    end if;
  end loop;
  for i in 0 to 1 loop
    if(counter = i+63) then
      mult_a <= buf2(i+2+20);
      mult_b <= pnt3(1-i);
    end if;
  end loop;
  for i in 0 to 1 loop
    if(counter = i+65) then
      mult_a <= buf2(i+2+24);
      mult_b <= pnt3(1-i);
    end if;
  end loop;
  for i in 0 to 1 loop
    if(counter = i+67) then
      mult_a <= buf2(i+2+28);
      mult_b <= pnt3(1-i);
    end if;
  end loop;
-----------------------------------------------------------------------------
-- pnt4
-----------------------------------------------------------------------------
  if(counter = 69) then
    for i in 0 to 7 loop
      add_a(i*2)   <= buf2(i*4);
      add_b(i*2)   <= buf2(i*4+1);
      sub_b(i*2)   <= buf2(i*4+1);
      sub_a(i*2)   <= buf2(i*4);

      add_a(i*2+1)  <= buf2(i*4+2);
      add_b(i*2+1)  <= buf2(i*4+3);
      sub_b(i*2+1)  <= buf2(i*4+2);
      sub_a(i*2+1)  <= buf2(i*4+3);
    end loop;
  end if;
  for i in 0 to 15 loop
    if(counter = i+70) then
      mult_a <= buf1(i*2+1);
      mult_b <= pnt4;
    end if;
  end loop;
-----------------------------------------------------------------------------
  if(counter = 86) then
    for i in 0 to 7 loop
      add_a(i) <= buf1(2+i*4);
      add_b(i) <= buf1(3+i*4);
    end loop;
  end if;
  if(counter = 87) then
    for i in 0 to 3 loop
      add_a(i*3) <= buf1(4+i*8);
      add_b(i*3) <= buf1(6+i*8);
      add_a(i*3+1) <= buf1(6+i*8);
      add_b(i*3+1) <= buf1(5+i*8);
      add_a(i*3+2) <= buf1(5+i*8);
      add_b(i*3+2) <= buf1(7+i*8);
    end loop;
  end if;
  if(counter = 88) then
    for i in 0 to 1 loop
      add_a(i*7) <= buf1(8+i*16);
      add_b(i*7) <= buf1(12+i*16);
      add_a(i*7+1) <= buf1(12+i*16);
      add_b(i*7+1) <= buf1(10+i*16);
      add_a(i*7+2) <= buf1(10+i*16);
      add_b(i*7+2) <= buf1(14+i*16);
      add_a(i*7+3) <= buf1(14+i*16);
      add_b(i*7+3) <= buf1(9+i*16);
      add_a(i*7+4) <= buf1(9+i*16);
      add_b(i*7+4) <= buf1(13+i*16);
      add_a(i*7+5) <= buf1(13+i*16);
      add_b(i*7+5) <= buf1(11+i*16);
      add_a(i*7+6) <= buf1(11+i*16);
      add_b(i*7+6) <= buf1(15+i*16);
    end loop;
  end if;
end process;

-- cosine table
pnt0(0)(31 downto 0)<=conv_std_logic_vector(16403,32);
pnt0(1)(31 downto 0)<=conv_std_logic_vector(16563,32);
pnt0(2)(31 downto 0)<=conv_std_logic_vector(16890,32);
pnt0(3)(31 downto 0)<=conv_std_logic_vector(17401,32);
pnt0(4)(31 downto 0)<=conv_std_logic_vector(18124,32);
pnt0(5)(31 downto 0)<=conv_std_logic_vector(19101,32);
pnt0(6)(31 downto 0)<=conv_std_logic_vector(20398,32);
pnt0(7)(31 downto 0)<=conv_std_logic_vector(22112,32);
pnt0(8)(31 downto 0)<=conv_std_logic_vector(24396,32);
```

```
pnt0(9)(31 downto 0)<=conv_std_logic_vector(27503,32);
pnt0(10)(31 downto 0)<=conv_std_logic_vector(31869,32);
pnt0(11)(31 downto 0)<=conv_std_logic_vector(38320,32);
pnt0(12)(31 downto 0)<=conv_std_logic_vector(48633,32);
pnt0(13)(31 downto 0)<=conv_std_logic_vector(67429,32);
pnt0(14)(31 downto 0)<=conv_std_logic_vector(111660,32);
pnt0(15)(31 downto 0)<=conv_std_logic_vector(333906,32);

pnt1(0)(31 downto 0)<=conv_std_logic_vector(16463,32);
pnt1(1)(31 downto 0)<=conv_std_logic_vector(17121,32);
pnt1(2)(31 downto 0)<=conv_std_logic_vector(18577,32);
pnt1(3)(31 downto 0)<=conv_std_logic_vector(21195,32);
pnt1(4)(31 downto 0)<=conv_std_logic_vector(25826,32);
pnt1(5)(31 downto 0)<=conv_std_logic_vector(34756,32);
pnt1(6)(31 downto 0)<=conv_std_logic_vector(56441,32);
pnt1(7)(31 downto 0)<=conv_std_logic_vector(167154,32);

pnt2(0)(31 downto 0)<=conv_std_logic_vector(16704,32);
pnt2(1)(31 downto 0)<=conv_std_logic_vector(19704,32);
pnt2(2)(31 downto 0)<=conv_std_logic_vector(29490,32);
pnt2(3)(31 downto 0)<=conv_std_logic_vector(83981,32);

pnt3(0)(31 downto 0)<=conv_std_logic_vector(17733,32);
pnt3(1)(31 downto 0)<=conv_std_logic_vector(42813,32);

pnt4   (31 downto 0)<=conv_std_logic_vector(23170,32);

end behav;
```

Appendix C
mul64_with_shift.vhd

```
-- mult128_wrapper is used to get the lower 63bits of the 128bits output from
component mult128 defined inmult128.vhd
-- mult128_wrapper is a component in dct64-final-5-2.vhd
-- created at May 3th. by kai

library IEEE;

use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity mul_wrapper is
  PORT
      (
            signal dataa            : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            signal datab            : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            signal result64         : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)

      );

end mul_wrapper;

architecture datapath of mul_wrapper is

component mul IS
      PORT
      (
            dataa       : in STD_LOGIC_VECTOR (31 DOWNTO 0);
            datab       : in STD_LOGIC_VECTOR (31 DOWNTO 0);
            result            : out STD_LOGIC_VECTOR (63 DOWNTO 0)

      );
END component;

signal mult_result: std_logic_vector (63 downto 0);

begin

  mult_inst : mul PORT MAP (
            dataa  => dataa,
            datab  => datab,
            resul  => mult_result

      );

result64 <= mult_result(31)&mult_result(31)&mult_result(31)&mult_result(31)&
mult_result(31)&mult_result(31)&mult_result(31)&mult_result(31)&mult_result(3
1)&mult_result(31)&mult_result(31)&mult_result(31)&mult_result(31)&mult_resul
t(31)&mult_result(31)&mult_result(31 downto 15);

end datapath;
```

Appendix D
audio_wrapper.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_SIGNED.all;

-----------------------------------------------------------------------------
--
-- entity
-----------------------------------------------------------------------------
--

entity audio_wrapper is
port (
    clk : in std_logic;
    reset_n : in std_logic;

    -- Bus master signals
    address      : in std_logic_vector (7 downto 0);
    byteenable   : in std_logic_vector (1 downto 0);
    writedata    : in std_logic_vector (15 downto 0);
    read         : in std_logic;
    write        : in std_logic;
    chipselect   : in std_logic;

     -- Slave signals
    readdata     : out std_logic_vector (15 downto 0);
    waitrequest  : out std_logic;

    -- Audio interface signals
    klcoidua    : in std_logic;                     --    18.43MHz audio
clock AUD_XCK
    AUD_ADCLRCK  : out std_logic;                    --    Audio CODEC ADC LR
Clock
    AUD_ADCDAT   : in  std_logic;                    --    Audio CODEC ADC
Data
    AUD_DACLRCK  : out std_logic;                    --    Audio CODEC DAC LR
Clock
    AUD_DACDAT   : out std_logic;                    --    Audio CODEC DAC
Data
    AUD_BCLK     : inout std_logic;                   --    Audio CODEC Bit-
Stream Clock

    -- Test signals --
    ledr    : out std_logic_vector (17 downto 0)
  );
end entity audio_wrapper;

-----------------------------------------------------------------------------
--
-- architecture
-----------------------------------------------------------------------------
--

architecture imp of audio_wrapper is
```

```
--component fifo
  component aud_fifo
  port (
            data        : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
            rdclk       : IN STD_LOGIC ;
            rdreq       : IN STD_LOGIC ;
            wrclk       : IN STD_LOGIC ;
            wrreq       : IN STD_LOGIC ;
            q                 : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
            rdempty           : OUT STD_LOGIC ;
            wrempty           : OUT STD_LOGIC ;
            wrusedw           : OUT STD_LOGIC_VECTOR (12 DOWNTO 0)
  );
  end component;

  component de2_wm8731_audio
    port (
        clk : in std_logic;
        reset_n : in std_logic;
        test_mode : in std_logic;
        audio_request : out std_logic;
        data : in std_logic_vector(15 downto 0);

        -- Audio interface signals
        AUD_ADCLRCK  : out std_logic;
        AUD_ADCDAT   : in  std_logic;
        AUD_DACLRCK  : out std_logic;
        AUD_DACDAT   : out std_logic;
        AUD_BCLK     : inout std_logic
      );
  end component;

  signal reset : std_logic;

  signal sound_request : std_logic;
  signal write_fifo : std_logic;
  signal data_from_bus : std_logic_vector (15 downto 0);
  signal mdata : std_logic_vector (15 downto 0);
  signal rdempty : std_logic;
  signal wrempty : std_logic;
  signal writefifo   : std_logic;

  signal counter : std_logic_vector (13 downto 0);
  signal sin_counter : std_logic_vector (5 downto 0);
  signal sin_out : std_logic_vector (15 downto 0);

  signal rate : std_logic_vector(2 downto 0); -- "XX1" 11kHz, "01X" 22kHz,
"1XX" 44kHz
  signal rate_counter : std_logic_vector (2 downto 0);
  signal req_fifo : std_logic;

  signal used_buf : std_logic_vector (12 downto 0);

  --test
  signal ledr_count : std_logic_vector (16 downto 0);
begin

  -- assignments
```

```
ledr(16 downto 0) <= ledr_count(16 downto 0);
ledr(17) <= wrempty;

process (clk, reset_n)
begin
  if (clk'event and clk = '1') then
    writefifo <= '0';
    if(write = '1' and chipselect = '1') then
      if(address = "0000") then
        data_from_bus <= writedata;
        writefifo <= '1';
      else
        rate <= writedata(13 downto 11);
      end if;
    end if;
    if(read = '1' and chipselect = '1' and address = "0000") then
      readdata <= "000"& used_buf;
    end if;
  end if;
end process;

process (klcoidua)
begin
  if(klcoidua'event and klcoidua = '1') then
    req_fifo <= '0';
    if(sound_request = '1') then
      if((rate_counter(2) = '1' and rate(2) = '1') or
         (rate_counter(1) = '1' and rate(1) = '1') or
         (rate_counter(0) = '1' and rate(0) = '1')) then
        rate_counter <= "100";
        req_fifo <= '1';
      else
        rate_counter <= '0' & rate_counter (2 downto 1);
      end if;
    end if;
  end if;
end process;

process(wrempty)
    begin
      if(wrempty'event and wrempty = '1') then
      ledr_count <= ledr_count + '1';
    end if;
end process;

  audio : de2_wm8731_audio
  port map (
    clk     => klcoidua,
    reset_n => reset_n,
    test_mode  => '0',
    audio_request => sound_request,
    data => mdata,

    AUD_ADCLRCK  => AUD_ADCLRCK,
    AUD_ADCDAT   => AUD_ADCDAT,
    AUD_DACLRCK  => AUD_DACLRCK,
    AUD_DACDAT   => AUD_DACDAT,
    AUD_BCLK     => AUD_BCLK
    );

    the_fifo : aud_fifo
    port map (
            data        => data_from_bus,
            rdclk       => klcoidua,
            rdreq       => req_fifo,
            wrclk       => clk,
            wrreq       => writefifo,
            q                 => mdata,
            rdempty           => rdempty,
            wrempty           => wrempty,
            wrusedw           => used_buf
      );

end architecture imp;
```

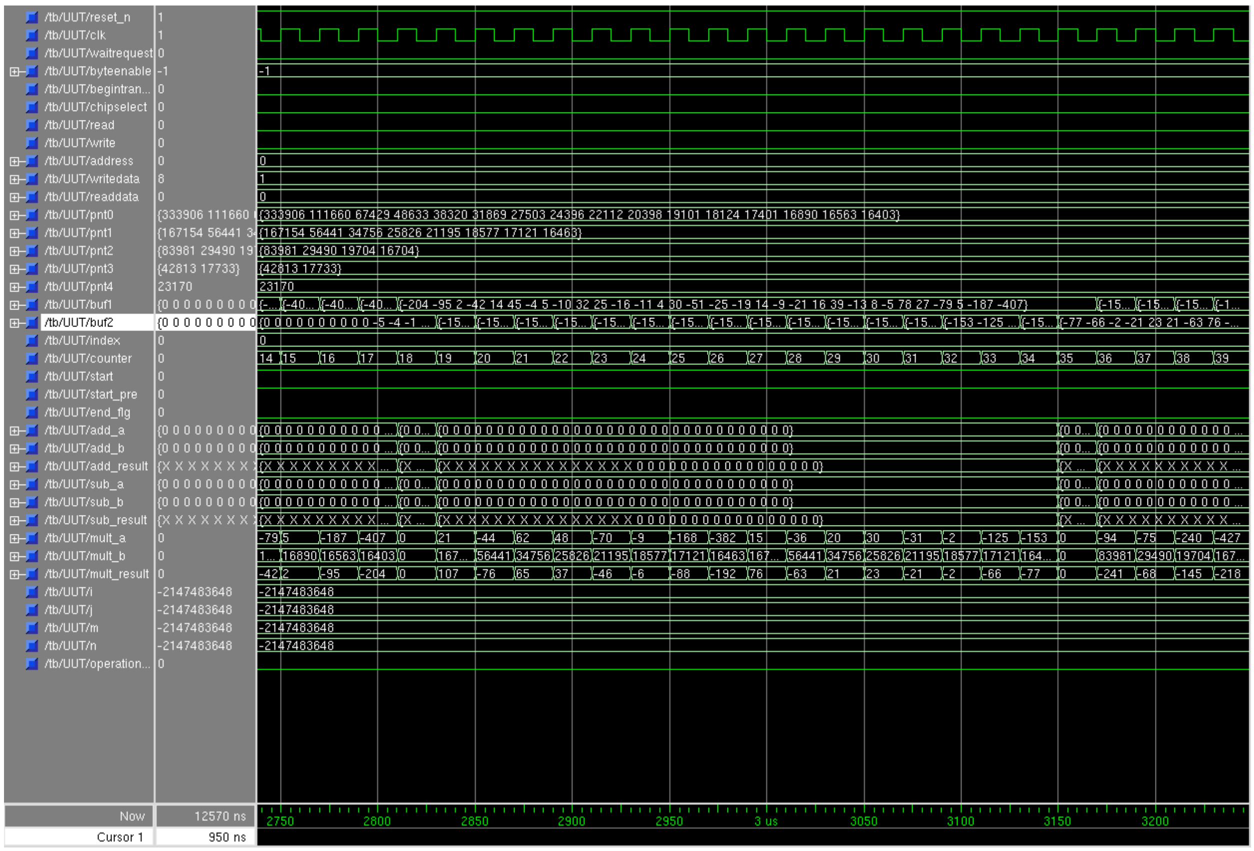
Appendix E
dct64.c

```
#include "config.h"
#include "mpg123.h"
#include <asm/io.h>

#include <stdio.h>

void dct64(real *out0,real *out1,real *samples)
{
  register int i;
  register real *b1;
  real bufs[64];
  b1 = bufs;
  for(i=0; i<32; i++)
    writel(*b1++, 0x00600000+i);
  writel(0xffffffff, 0x00600000+64);
  while(readl(0x00600000+64) != 1);

  for(i=0; i<64; i++)
    bufs[i] = readl(0x00600000+i);

  out0[0x10*16] = bufs[0];
  out0[0x10*15] = bufs[16+0]  + bufs[16+8];
  out0[0x10*14] = bufs[8];
  out0[0x10*13] = bufs[16+8]  + bufs[16+4];
  out0[0x10*12] = bufs[4];
  out0[0x10*11] = bufs[16+4]  + bufs[16+12];
  out0[0x10*10] = bufs[12];
  out0[0x10* 9] = bufs[16+12] + bufs[16+2];
  out0[0x10* 8] = bufs[2];
  out0[0x10* 7] = bufs[16+2]  + bufs[16+10];
  out0[0x10* 6] = bufs[10];
  out0[0x10* 5] = bufs[16+10] + bufs[16+6];
  out0[0x10* 4] = bufs[6];
  out0[0x10* 3] = bufs[16+6]  + bufs[16+14];
  out0[0x10* 2] = bufs[14];
  out0[0x10* 1] = bufs[16+14] + bufs[16+1];
  out0[0x10* 0] = bufs[1];

  out1[0x10* 0] = bufs[1];
  out1[0x10* 1] = bufs[16+1]  + bufs[16+9];
  out1[0x10* 2] = bufs[9];
  out1[0x10* 3] = bufs[16+9]  + bufs[16+5];
  out1[0x10* 4] = bufs[5];
  out1[0x10* 5] = bufs[16+5]  + bufs[16+13];
  out1[0x10* 6] = bufs[13];
  out1[0x10* 7] = bufs[16+13] + bufs[16+3];
  out1[0x10* 8] = bufs[3];
  out1[0x10* 9] = bufs[16+3]  + bufs[16+11];
  out1[0x10*10] = bufs[11];
  out1[0x10*11] = bufs[16+11] + bufs[16+7];
  out1[0x10*12] = bufs[7];
  out1[0x10*13] = bufs[16+7]  + bufs[16+15];
  out1[0x10*14] = bufs[15];
  out1[0x10*15] = bufs[16+15];
}
```

Appendix F
audio_dummy.c

```
#include "config.h"
#include "mpg123.h"
#include <stdlib.h>
#include <stdio.h>
#include <asm/io.h>

#include <asm/ioctl.h>

int audio_open(struct audio_info_struct *ai)
{
  ai->handle = NULL;
  return 0;
}

int audio_get_formats(struct audio_info_struct *ai)
{
  return AUDIO_FORMAT_SIGNED_16;
}

#define CAN_WRITE 220
int audio_play_samples(struct audio_info_struct *ai,unsigned char *buf,int
len)
{
  writew(ai->channels, 0x00500004);
  writew(ai->rate, 0x00500002);
  for(i=0; i<8192; i++)
    if(readw(0x00500000) < 8190)
       writew((buf[i*2+1]<<8)|(buf[i*2]), 0x00500000);
  return len;
}

int audio_close(struct audio_info_struct *ai)
{
  return 0;
}

void audio_queueflush(struct audio_info_struct *ai)
{
printf("We don't queue damnit!\n");
}
```

Appendix G
Modelsim result for dct64.vhd



```
/tb/UUT/reset_n        1
/tb/UUT/clk            1
/tb/UUT/waitrequest    0
/tb/UUT/byteenable     -1
/tb/UUT/begintran...   0
/tb/UUT/chipselect     0
/tb/UUT/read           0
/tb/UUT/write          0
/tb/UUT/address        0
/tb/UUT/writedata      8
/tb/UUT/readdata       0
/tb/UUT/pnt0           {333906 111660 6
/tb/UUT/pnt1           {167154 56441 34
/tb/UUT/pnt2           {83981 29490 19
/tb/UUT/pnt3           {42813 17733}
/tb/UUT/pnt4           23170
/tb/UUT/buf1           {0 0 0 0 0 0 0 0 0
/tb/UUT/buf2           {0 0 0 0 0 0 0 0 0
/tb/UUT/index          0
/tb/UUT/counter        0
/tb/UUT/start          0
/tb/UUT/start_pre      0
/tb/UUT/end_flg        0
/tb/UUT/add_a          {0 0 0 0 0 0 0 0 0
/tb/UUT/add_b          {0 0 0 0 0 0 0 0 0
/tb/UUT/add_result     {X X X X X X X
/tb/UUT/sub_a          {0 0 0 0 0 0 0 0 0
/tb/UUT/sub_b          {0 0 0 0 0 0 0 0 0
/tb/UUT/sub_result     {X X X X X X X
/tb/UUT/mult_a         0
/tb/UUT/mult_b         0
/tb/UUT/mult_result    0
/tb/UUT/i              -2147483648
/tb/UUT/j              -2147483648
/tb/UUT/m              -2147483648
/tb/UUT/n              -2147483648
/tb/UUT/operation...   0

Now        12570 ns
Cursor 1   950 ns
```