De1-SoC Board Audio Output Using the WM8731 Audio Codec

By: Adam Narten and Oliver Rarog; Winter 2018

Tools

Quartus Prime 17.0
Eclipse for DS-5 v.5.25.0

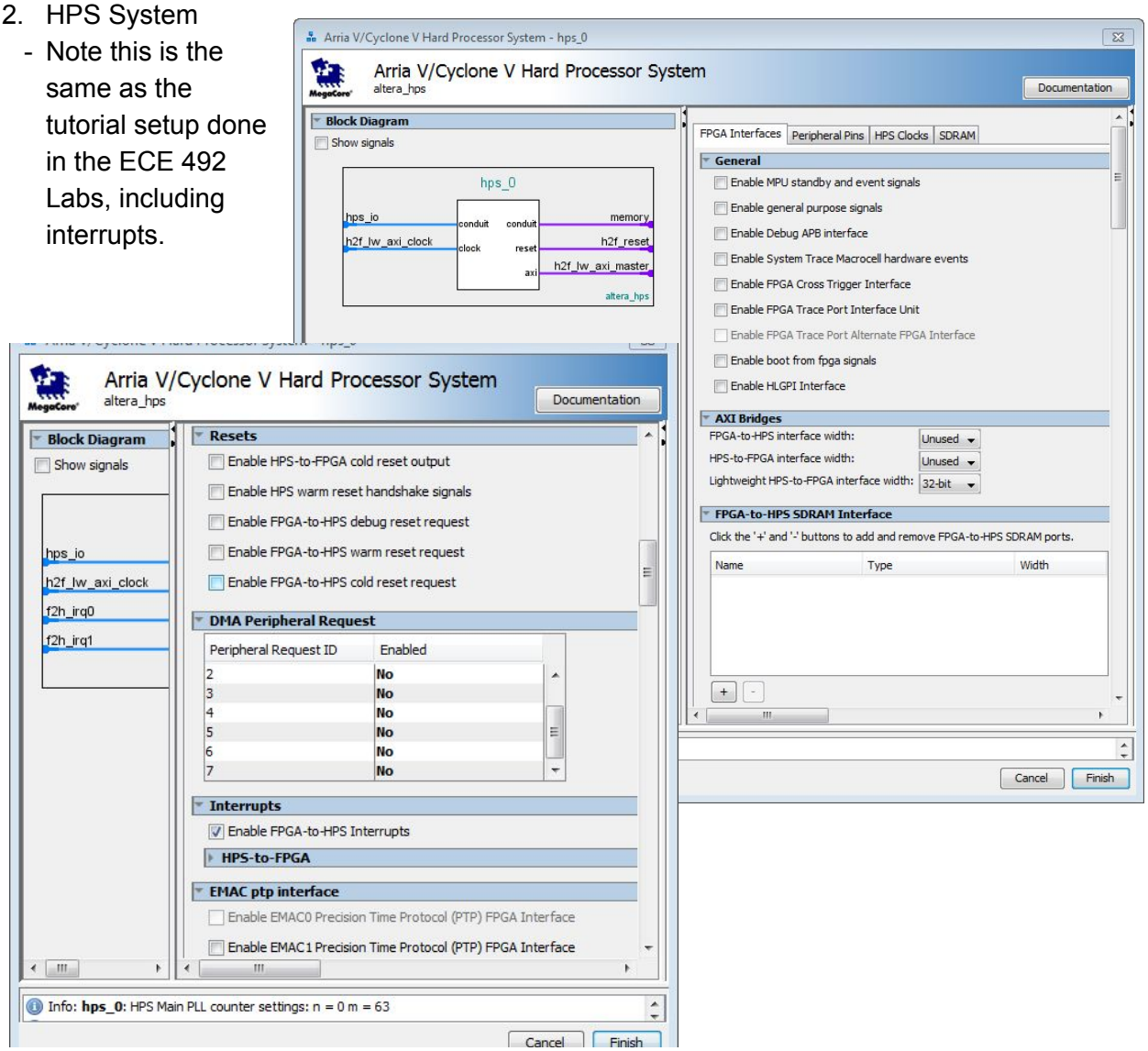
Introduction

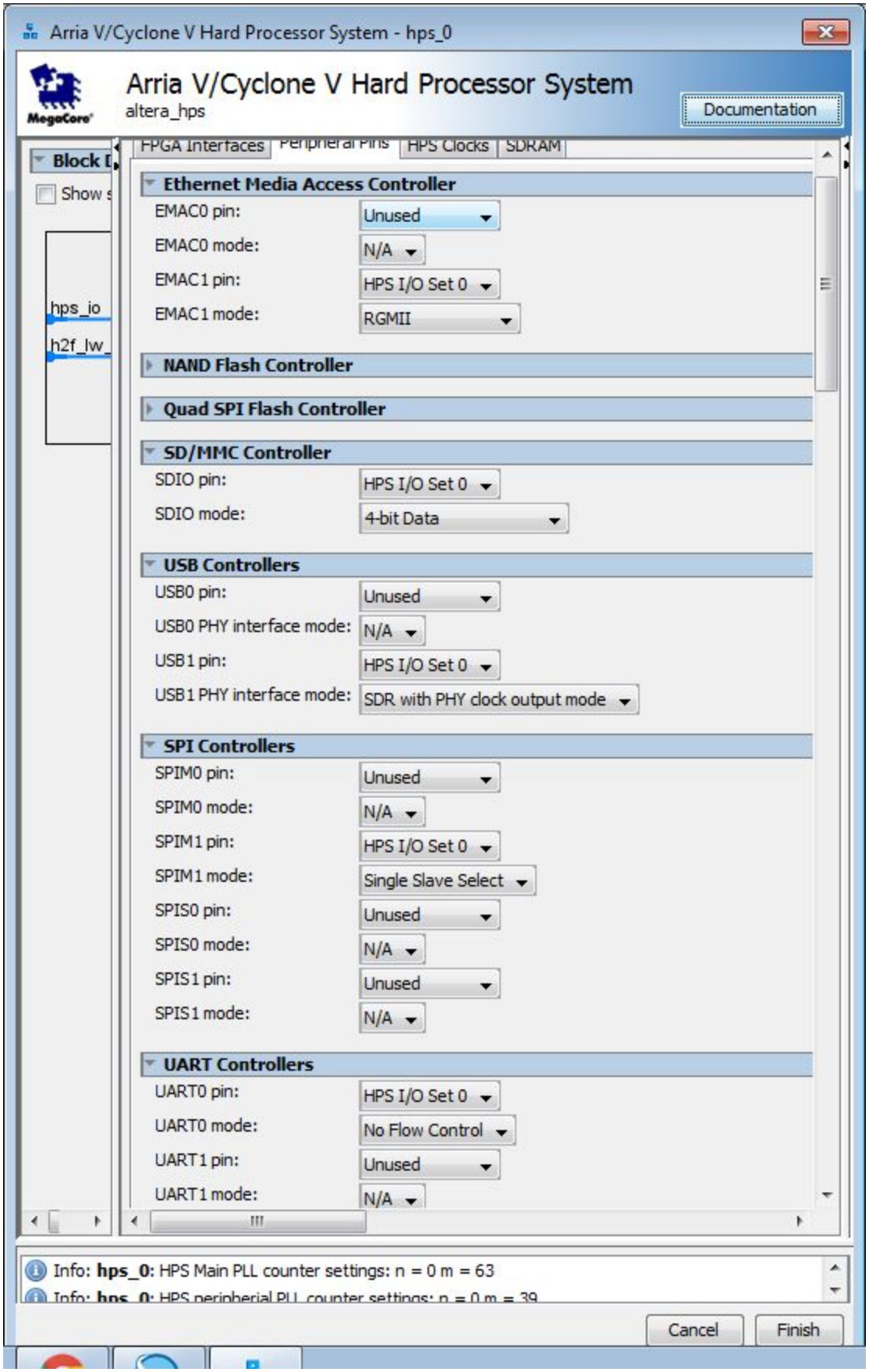
These notes will show how to get the WM8731 audio codec configured and running, as well as outputting a simple frequency sine wave.

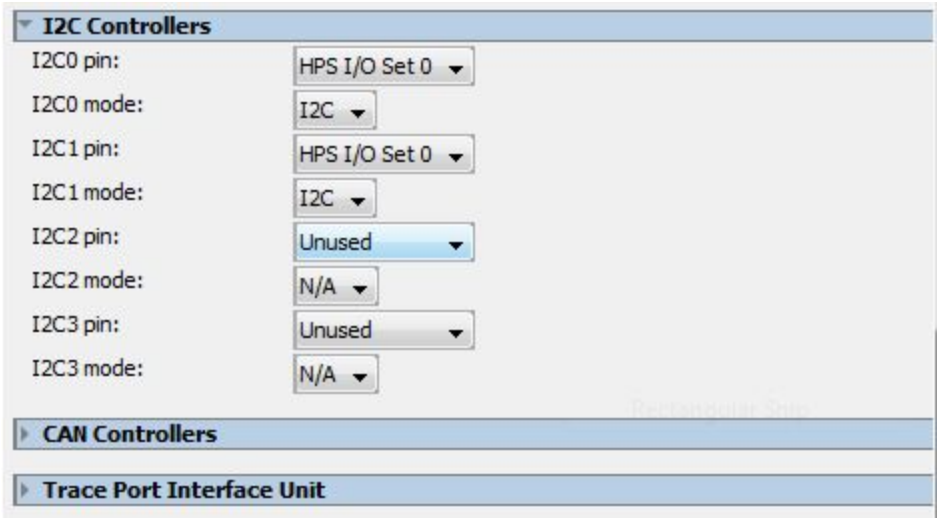
Hardware Procedure

The following images show the minimum required components in the Qsys system as well as the required settings.

1. Clock
    - Use the default clock source created by the New Project Wizard
2. HPS System
    - Note this is the same as the tutorial setup done in the ECE 492 Labs, including interrupts.

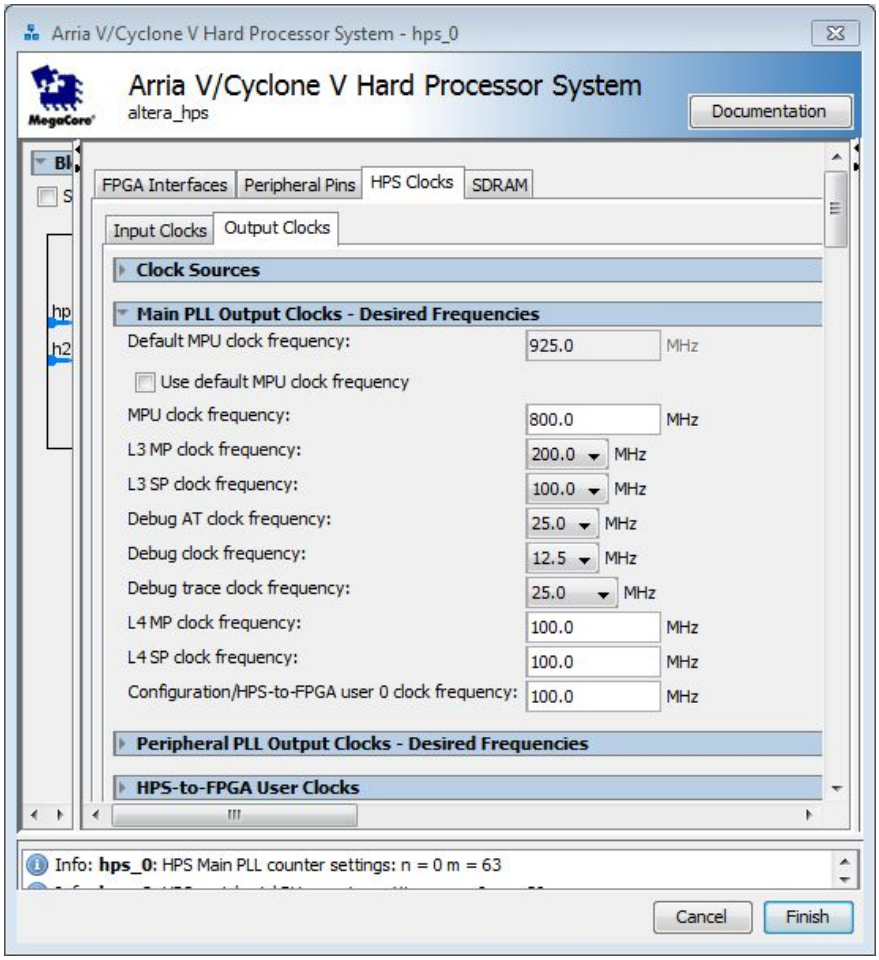


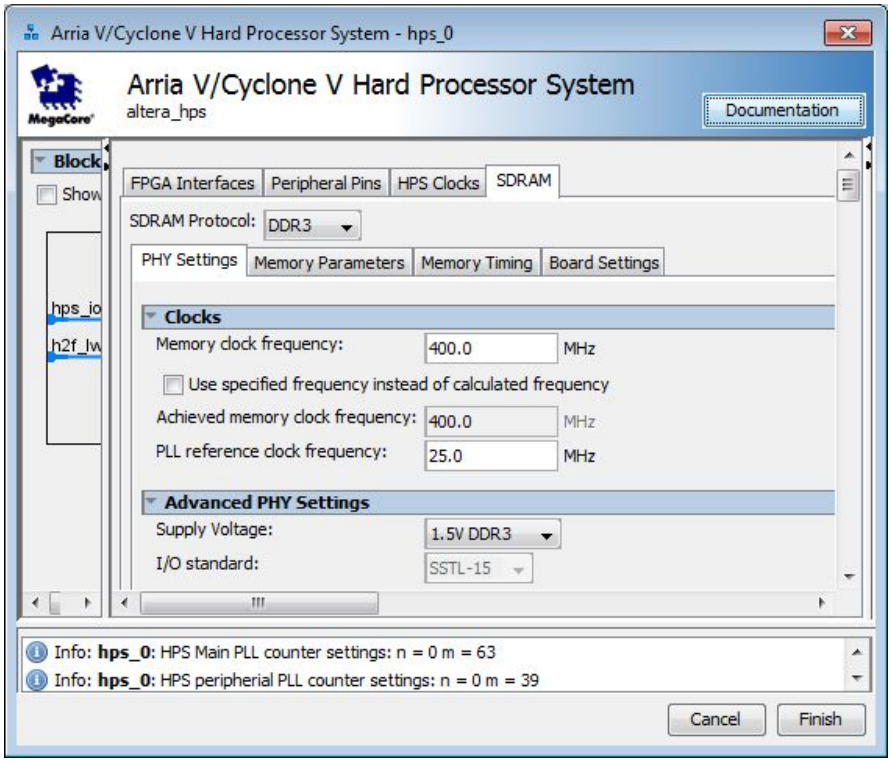

Arria V/Cyclone V Hard Processor System - hps_0
Arria V/Cyclone V Hard Processor System
altera_hps
Documentation
FPGA Interfaces
Peripheral Pins
HPS Clocks
SDRAM
Block D
Show s
hps_io
h2f_lw_
Ethernet Media Access Controller
EMAC0 pin: Unused
EMAC0 mode: N/A
EMAC1 pin: HPS I/O Set 0
EMAC1 mode: RGMII
NAND Flash Controller
Quad SPI Flash Controller
SD/MMC Controller
SDIO pin: HPS I/O Set 0
SDIO mode: 4-bit Data
USB Controllers
USB0 pin: Unused
USB0 PHY interface mode: N/A
USB1 pin: HPS I/O Set 0
USB1 PHY interface mode: SDR with PHY clock output mode
SPI Controllers
SPIM0 pin: Unused
SPIM0 mode: N/A
SPIM1 pin: HPS I/O Set 0
SPIM1 mode: Single Slave Select
SPIS0 pin: Unused
SPIS0 mode: N/A
SPIS1 pin: Unused
SPIS1 mode: N/A
UART Controllers
UART0 pin: HPS I/O Set 0
UART0 mode: No Flow Control
UART1 pin: Unused
UART1 mode: N/A
Info: hps_0: HPS Main PLL counter settings: n = 0 m = 63
Cancel
Finish

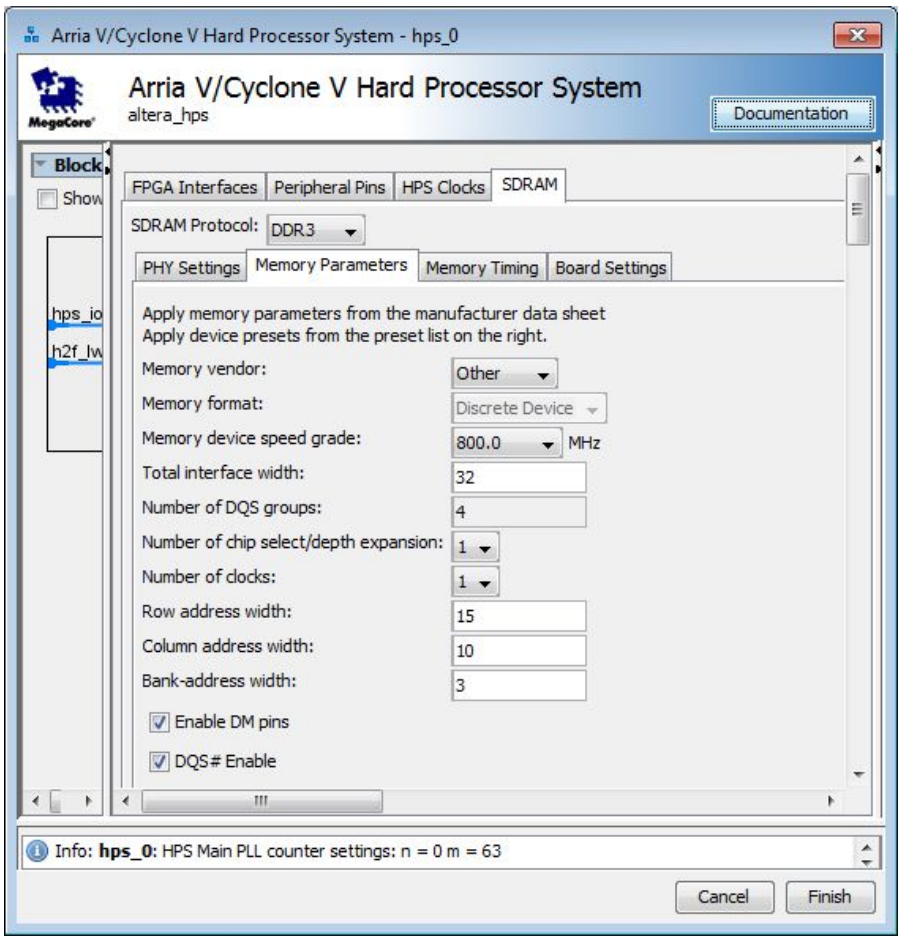


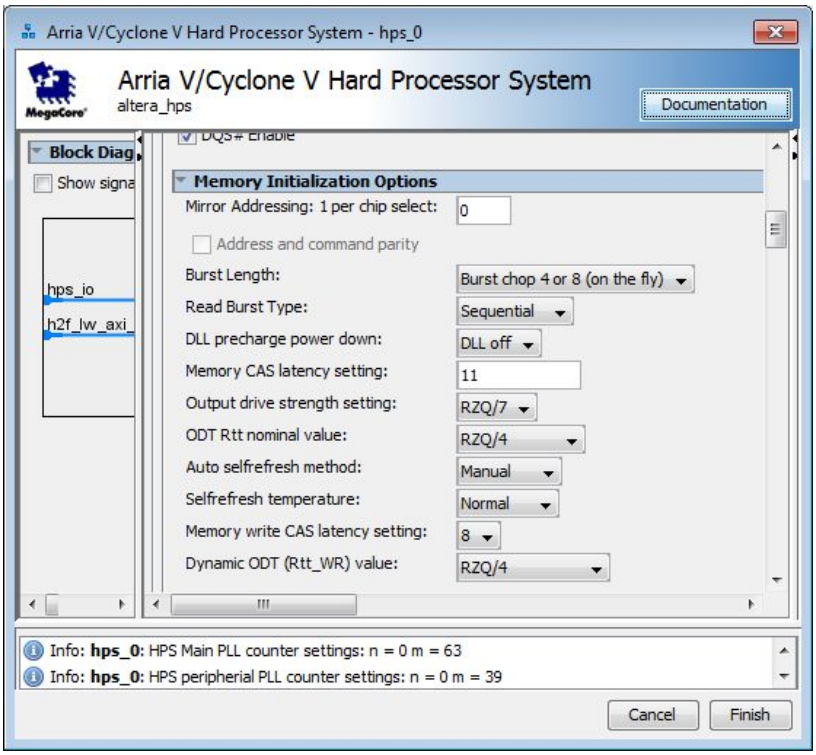
In the Peripheral Mux Table click on the GPIO09, GPIO35, GPIO40, GPIO48, GPIO53, GPIO54 and GPIO61 pins to appropriately map them manually. .

Under the HPS Clocks tab leave the Input Clocks tab as the default.
The output clocks are as follows:

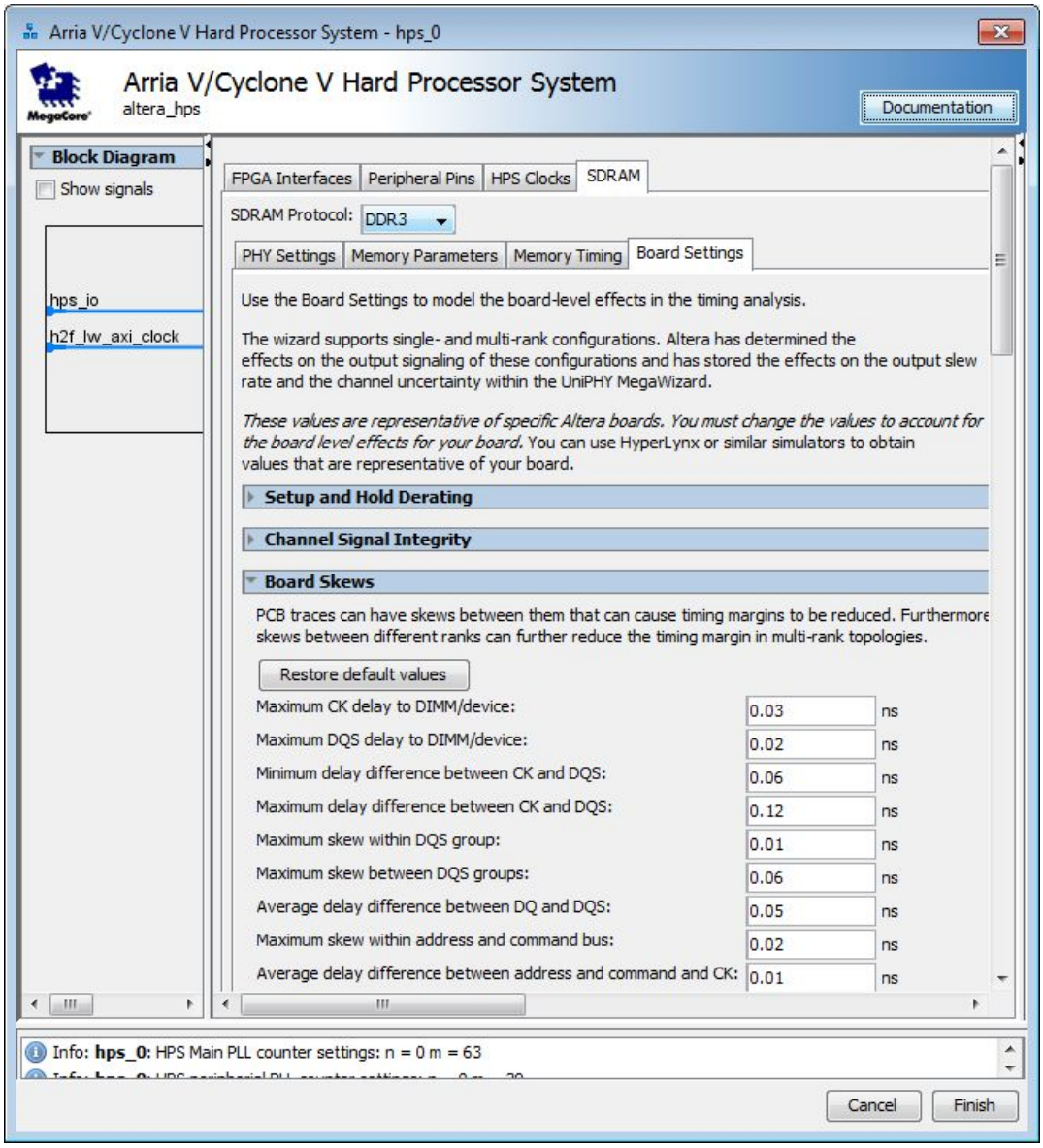



Arria V/Cyclone V Hard Processor System - hps_0
Arria V/Cyclone V Hard Processor System
altera_hps
Documentation
Block
Show
hps_io
h2f_lw
FPGA Interfaces
Peripheral Pins
HPS Clocks
SDRAM
SDRAM Protocol:
DDR3
PHY Settings
Memory Parameters
Memory Timing
Board Settings
Clocks
Memory clock frequency:
400.0
MHz
Use specified frequency instead of calculated frequency
Achieved memory clock frequency:
400.0
MHz
PLL reference clock frequency:
25.0
MHz
Advanced PHY Settings
Supply Voltage:
1.5V DDR3
I/O standard:
SSTL-15
Info: hps_0: HPS Main PLL counter settings: n = 0 m = 63
Info: hps_0: HPS peripherial PLL counter settings: n = 0 m = 39
Cancel
Finish

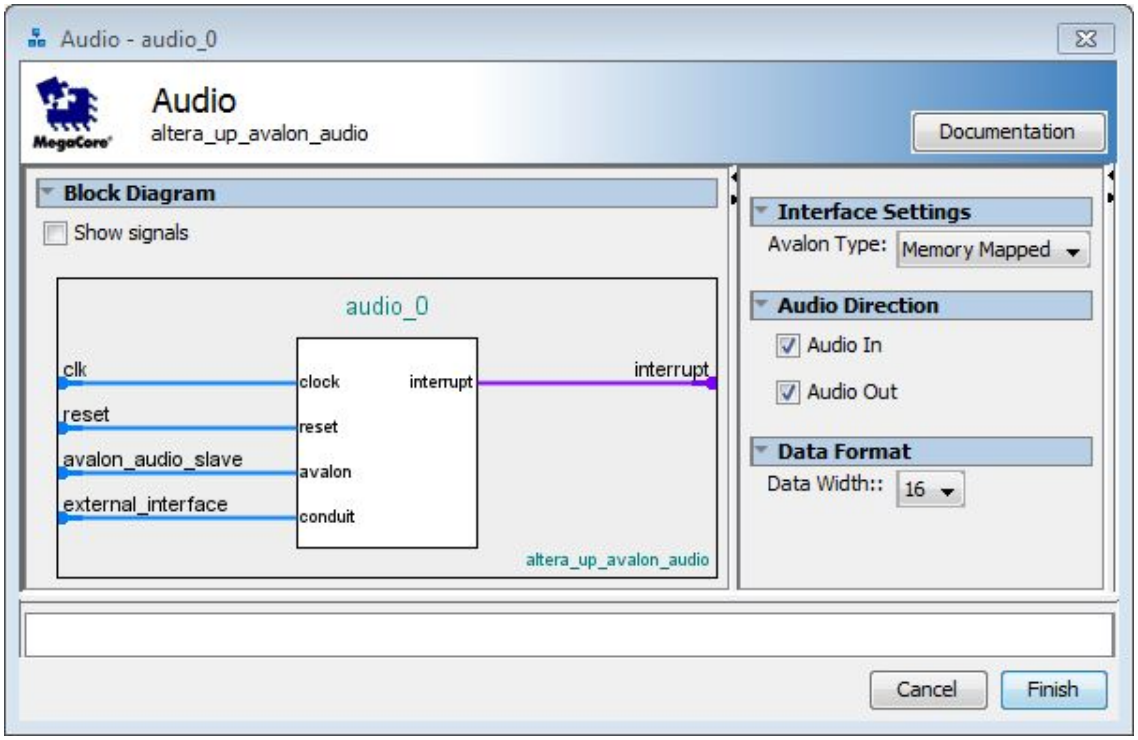

Arria V/Cyclone V Hard Processor System - hps_0
Arria V/Cyclone V Hard Processor System
altera_hps
Documentation
Block
Show
hps_io
h2f_lw
FPGA Interfaces
Peripheral Pins
HPS Clocks
SDRAM
SDRAM Protocol:
DDR3
PHY Settings
Memory Parameters
Memory Timing
Board Settings
Apply memory parameters from the manufacturer data sheet
Apply device presets from the preset list on the right.
Memory vendor:
Other
Memory format:
Discrete Device
Memory device speed grade:
800.0
MHz
Total interface width:
32
Number of DQS groups:
4
Number of chip select/depth expansion:
1
Number of clocks:
1
Row address width:
15
Column address width:
10
Bank-address width:
3
Enable DM pins
DQS# Enable
Info: hps_0: HPS Main PLL counter settings: n = 0 m = 63
Cancel
Finish


Arria V/Cyclone V Hard Processor System - hps_0
Arria V/Cyclone V Hard Processor System
altera_hps
Documentation
Block Diag
Show signa
hps_io
h2f_lw_axi_
DQS# Enable
Memory Initialization Options
Mirror Addressing: 1 per chip select:
0
Address and command parity
Burst Length:
Burst chop 4 or 8 (on the fly)
Read Burst Type:
Sequential
DLL precharge power down:
DLL off
Memory CAS latency setting:
11
Output drive strength setting:
RZQ/7
ODT Rtt nominal value:
RZQ/4
Auto selfrefresh method:
Manual
Selfrefresh temperature:
Normal
Memory write CAS latency setting:
8
Dynamic ODT (Rtt_WR) value:
RZQ/4
Info: hps_0: HPS Main PLL counter settings: n = 0 m = 63
Info: hps_0: HPS peripherial PLL counter settings: n = 0 m = 39
Cancel
Finish

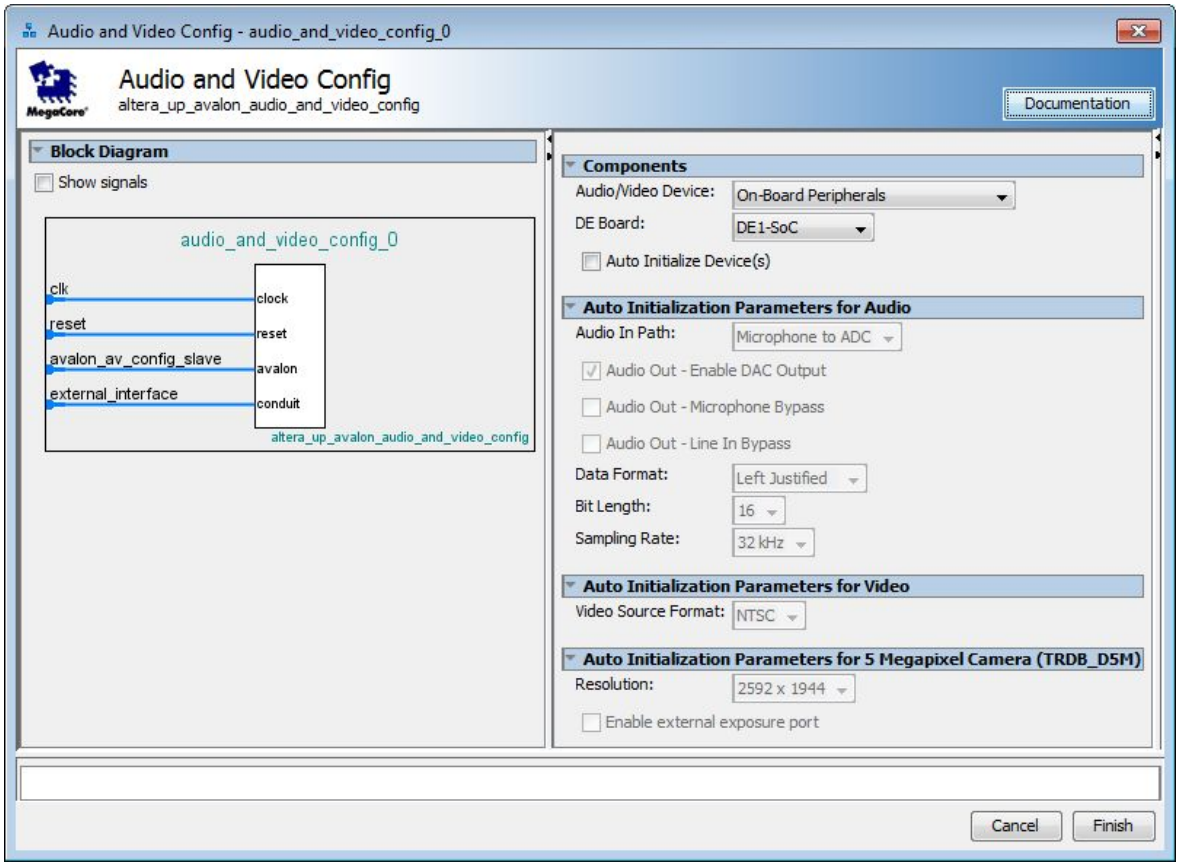
Under Board Settings -> Setup and Hold Derating and Channel Signal Integrity, leave Altera's default settings checked.
The remaining configuration is as follows:



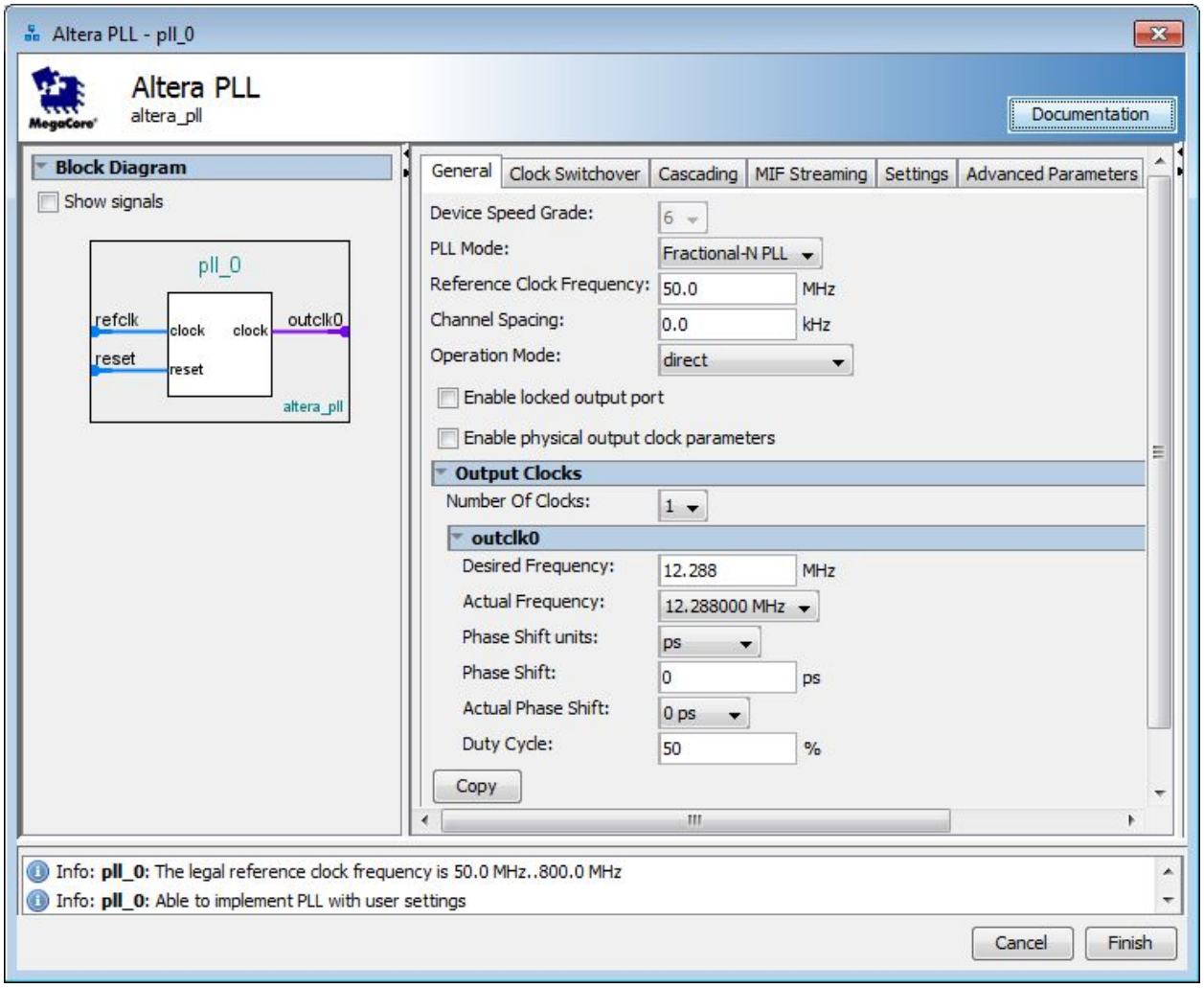
3. Audio Core (University Program -> Audio and Video -> Audio)



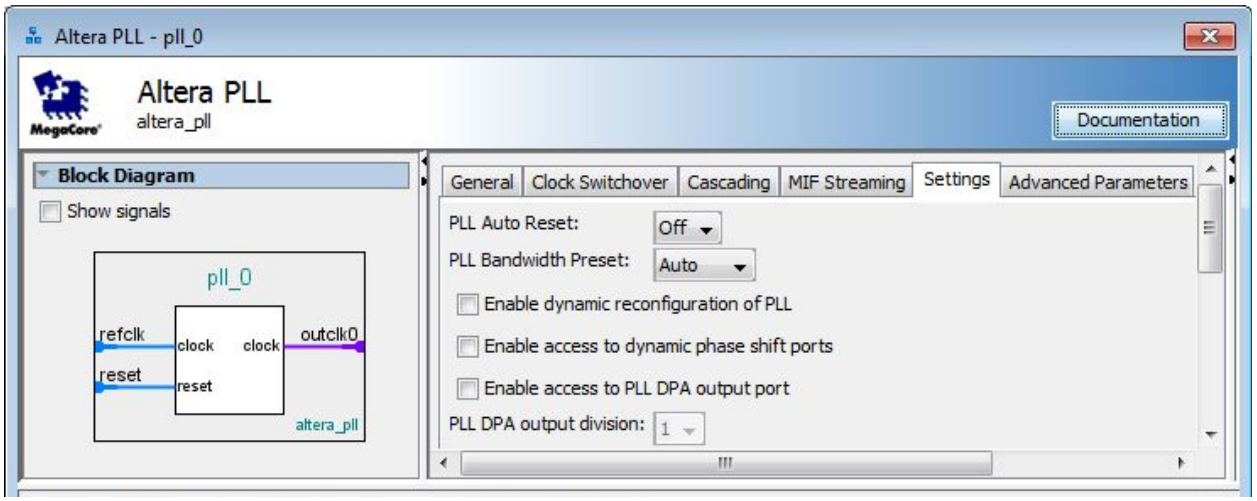
4. Audio and Video config (University Program -> Audio & Video -> Audio and Video Config)



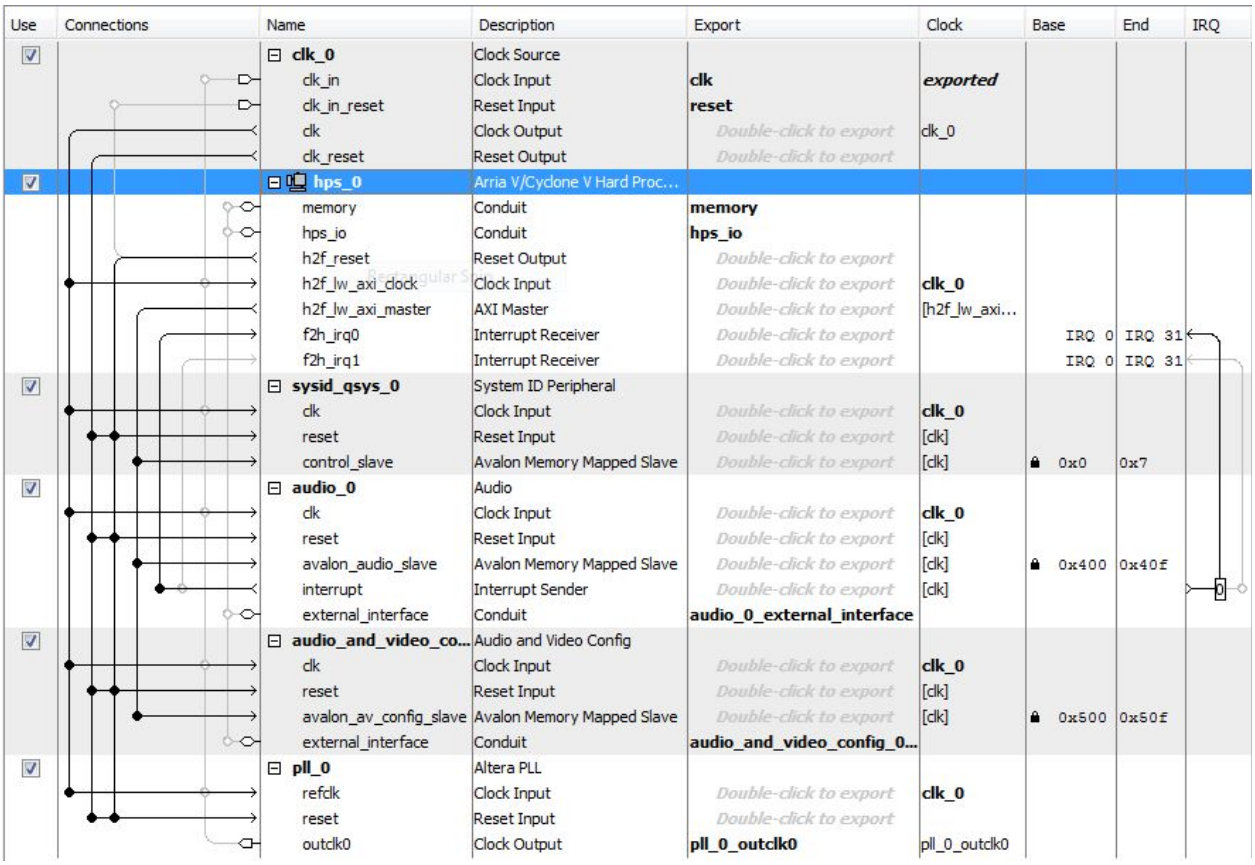
5. Altera PLL (Basic Functions -> Clocks; PLLs and Resets -> PLL -> Altera PLL)



Under the Clock Switchover, Cascading, and MIF Streaming tabs, nothing needs to be enabled
Finally, the Settings tab is as follows:



Once these components are added, connect them according to the following image and generate the VHDL in Qsys. Be sure to also include the sysid_qsys_0 component with the default configurations of 0x0.



Note you will have four warnings pertaining to interrupts upon successful generation. This is fine and will still provide functional audio.
Also note, the base addresses for the audio components are set at 0x400 and 0x500. You may change these to suit your design as long as they are updated in the audio.h and audio_cfg.h files (discussed later).

Next, ensure your toplevel VHDL file in Quartus is connected as follows. A template is provided in audio.vhd for reference.

A. Declare these pins in the entity (in addition to all of the hps signals):

```
-- I2C Interface
        FPGA_I2C_SCLK                   : out std_logic;
        FPGA_I2C_SDAT                   : inout std_logic := 'X';
-- Audio
        AUD_ADCDAT                              : in std_logic := 'X';
        AUD_ADCLRCK                             : in std_logic := 'X';
        AUD_BCLK                                : in std_logic := 'X';
        AUD_DACDAT                              : out std_logic;
```

```
AUD_DACLRCK                    : in std_logic := 'X';
AUD_XCK                        : out std_logic
```

B. Add these ports in the architecture's component declaration

```
audio_0_external_interface_ADCDAT       : in    std_logic                  := 'X';          -- ADCDAT
audio_0_external_interface_ADCLRCK      : in    std_logic                  := 'X';        -- ADCLRCK
audio_0_external_interface_BCLK         : in    std_logic                  := 'X';          -- BCLK
audio_0_external_interface_DACDAT       : out   std_logic;                                  -- DACDAT
audio_0_external_interface_DACLRCK      : in    std_logic                  := 'X';        -- DACLRCK
audio_and_video_config_0_external_interface_SDAT : inout std_logic    := 'X';          -- SDAT
audio_and_video_config_0_external_interface_SCLK : out   std_logic;                 -- SCLK
pll_0_outclk0_clk                       : out   std_logic;                                  -- clk
```

C. Port map the pins to the ports in the instantiation:

```
audio_and_video_config_0_external_interface_SDAT => FPGA_I2C_SDAT,
audio_and_video_config_0_external_interface_SCLK => FPGA_I2C_SCLK,
audio_0_external_interface_ADCDAT                => AUD_ADCDAT,
audio_0_external_interface_ADCLRCK               => AUD_ADCLRCK,
audio_0_external_interface_BCLK                  => AUD_BCLK,
audio_0_external_interface_DACDAT                => AUD_DACDAT,
        audio_0_external_interface_DACLRCK               => AUD_DACLRCK,
        pll_0_outclk0_clk                                => AUD_XCK
```

Ensure that the .qip and .tcl scripts have also been added to the project before compiling. Also be sure to run the appropriate .tcl scripts beforehand. Then you may compile the design.

A zipped archive of a template project is included with this report for your reference.

Software Procedure

While only using the LINEOUT to the speakers, the audio core has four writable registers starting at the base address. For a full description of what each register does, see the first reference. In order to send audio out, you must write to both the left and right audio channels, if only one channel is written to and one FIFO buffer remains empty, the codec will not output audio. Although the audio and video config component can be setup in Qsys, this configuration can be overridden by writing to the components registers. These can be written to by the function write_audio_cfg_register(REGISTER ADDRESS, REGISTER VALUE). Below is the full register configuration used in the example code. However, something to note in the 0x8 register, is that bit 6 must be set to 0 as to enable the Slave Mode.

```
write_audio_cfg_register(0x0, 0x17);
write_audio_cfg_register(0x1, 0x17);
write_audio_cfg_register(0x2, 0x7F);
write_audio_cfg_register(0x3, 0x7F);
write_audio_cfg_register(0x4, 0x15);
```

```
write_audio_cfg_register(0x5, 0x06);
write_audio_cfg_register(0x6, 0x00);
write_audio_cfg_register(0x7, 0x4D);
write_audio_cfg_register(0x8, 0x18);
write_audio_cfg_register(0x9, 0x01);
```

This should enable all of the correct settings for the audio core and it should be ready to be written to according to the 12.288 MHz clock specified in the Altera PLL.

Finally, to ensure the software can properly communicate with the hardware, specify the base addresses near the top of the audio.h and audio_cfg.h files (0x400 and 0x500 from qsys). Import all of the audio files into your project and use them as follows.

1. Generate a buffer of audio data of type INT32S*
2. Pass the buffer and frequency to the write function using write_audio_data(buffer, freq);

An example is shown in the attached app.c file.

Please take note that the SAMPLING_RATE specified in app.c of 32000 Hz is dependent on the 12.288 MHz clock in the PLL. Changing the sampling rate to be different (ex. 44100 Hz) requires changing the Altera PLL outclk0 Desired Frequency to match according to pg. 39 of the datasheet [2]. For the example of a 44100 Hz sampling frequency, the desired PLL frequency must be set to 11.2896 MHz. The configuration done in software will also change based on the datasheet since bits 5:2 in register 8 are based on sampling rate. Use 0x18 for 32kHz and 0x20 for 44.1kHz. (`write_audio_cfg_register(0x8, 0x20);`)

References

[1] University of Toronto. *Digital Embedded Systems Lab: Audio Core.* [Online]. Available:http://www-ug.eecg.utoronto.ca/desl/nios_devices_SoC/dev_audio.html

[2] WOLFSON MICROELECTRONICS plc. (2005). *Portable Internet Audio CODEC with Headphone Driver and Programmable Sample Rates.* [Online]. Available:https://www.rockbox.org/wiki/pub/Main/DataSheets/WM8731_8731L.pdf

Attached Files

audio.c: Provides communication driver for writing to the audio buffers.
audio.h: Header file for audio core driver.
audio_cfg.c: Provides configurations for audio and video cores.
audio_cfg.h: Header file for audio and video core drivers.
app.c: Provides an example main.c and AppTask to produce audio output.
audio.qar: Archived Quartus/Qsys project example to provide functional audio.
pin_assignment_DE1_SoC.tcl: Pin Assignment tcl script from Terasic
WM8731_8731L_DATASHEET.pdf: Downloaded datasheet from [2]