Application Note: AN00111

Optimizing start-up time in AVB endpoints

This application note provides background information and step-by-step instructions for minimizing the time from power on to streaming (start-up time) in AVB endpoints.

Using static configuration along with various boot time optimisations, this application note shows how a start-up time comfortably less than 500 milliseconds can be achieved.

The application note is based the XMOS Ethernet AVB Endpoint software. The code changes and experiments were conducted using two XMOS AVB-LC Kits connected back to back.

Information about the AVB Endpoint software and development boards used can be found on the XMOS website.[1]

Required tools and libraries

- xTIMEcomposer Tools - Version 13.2
- XMOS AVB Endpoint software - Version 6.1.1
- Start-up time optimized modified source files, to augment version 6.1.1 of the AVB Endpoint software. These are included in this application note.

Required hardware

The software described within this note is designed to run on an XMOS xCORE-L series devices.

The example firmware provided in the AVB Endpoint software has been implemented and tested using the low-cost AVB Audio Endpoint Platform hardware (XR-AVB-LC-BRD). Dual XR-AVB-LC-BRD boards are available as a kit from part number XK-AVB-LC-SYS.

Prerequisites

- This document assumes familiarity with the XMOS xCORE architecture, the IEEE AVB/TSN standards, the XMOS tool chain and the xC programming language. Documentation related to these aspects which are not specific to this application note are linked to in the references appendix at the end of this application note.
- For descriptions of XMOS related terms found in this document please see the *XMOS glossary*[2].
- For the full API listing of the XMOS AVB Audio Endpoint software please see the *AVB Endpoint Design Guide*[3].

[1] http://www.xmos.com/applications/audio/avb
[2] http://www.xmos.com/published/glossary
[3] https://www.xmos.com/published/avb-design-guide

1 Overview

1.1 Introduction

Ethernet AVB is useful in a broad range of environments, where streaming of time sensitive data over Ethernet is attractive. The Automotive industry is adopting Ethernet AVB as a high bandwidth time-sensitive, robust and cost effective network technology. However the Automotive application segment brings a unique set of requirements. One of the key requirements of the Automotive profile for AVB is a short power-on to streaming delay, here in referred to as “fast start-up”.

In fully dynamic, open, AVB networks there are a number of protocols which allow the dynamic insertion, connection and management of network infrastructure, endpoints and streams. These protocols can increase the start-up time to several seconds, which is beyond acceptable limits for some in-car uses. For example, the requirements for active noise cancellation, and the playback of warning “bongs” may be required to be operational less than one second after power-on.

To meet this requirement, a number of trade-offs can be made that take advantage of the pre-known AVB network configuration, by employing statically configured systems. This application note addresses implementation of various start-up optimisations applied to XMOS AVB Audio endpoints which, when combined, provide a power-on to streaming time considerably under 500 milliseconds.

2 Measuring Startup Time

2.1 Key events

In order to measure startup time, it is helpful to define events on the timeline. For the purpose of this application note, the events of interest are:

1. Power on. Power is applied to the AVB hardware to initiate a cold boot of the system.
2. Application start. The power rails have been energised and settled, the reset sequence has finished, the firmware has been fully loaded and compiler start-up code has been executed.
3. Ethernet Ready. The main() function is now being executed and the Ethernet PHY and MAC components are running. i.e. The Ethernet link is up.
4. AVB Sync. gPTP is operational and has received at least two sync messages, allowing the slave to lock to the master.
5. AVB Media Ready. The endpoint has an established media clock (i.e. it has recovered the media clock if it is input stream derived) and is ready to stream media.

From an application perspective, the measurement of practical importance is `Power on` to `AVB Media` Ready. i.e. All stages have been completed.

3 Testing Method

The system is tested by simultaneous starting of both endpoints. Measurement starts at the application of 3v3 to each of endpoint's power rail.

Oscilloscope measurements of the power rails, I/O transitions on test signals linked to internal events and timestamped debug print statements were used to quantify the timings.

3.1 Network overview

Figure 1 shows a overview of an AVB network consisting of a single Talker and Listener. To keep the system simple, and avoid the need to statically configure an AVB switch, a direct point to point connection was used.

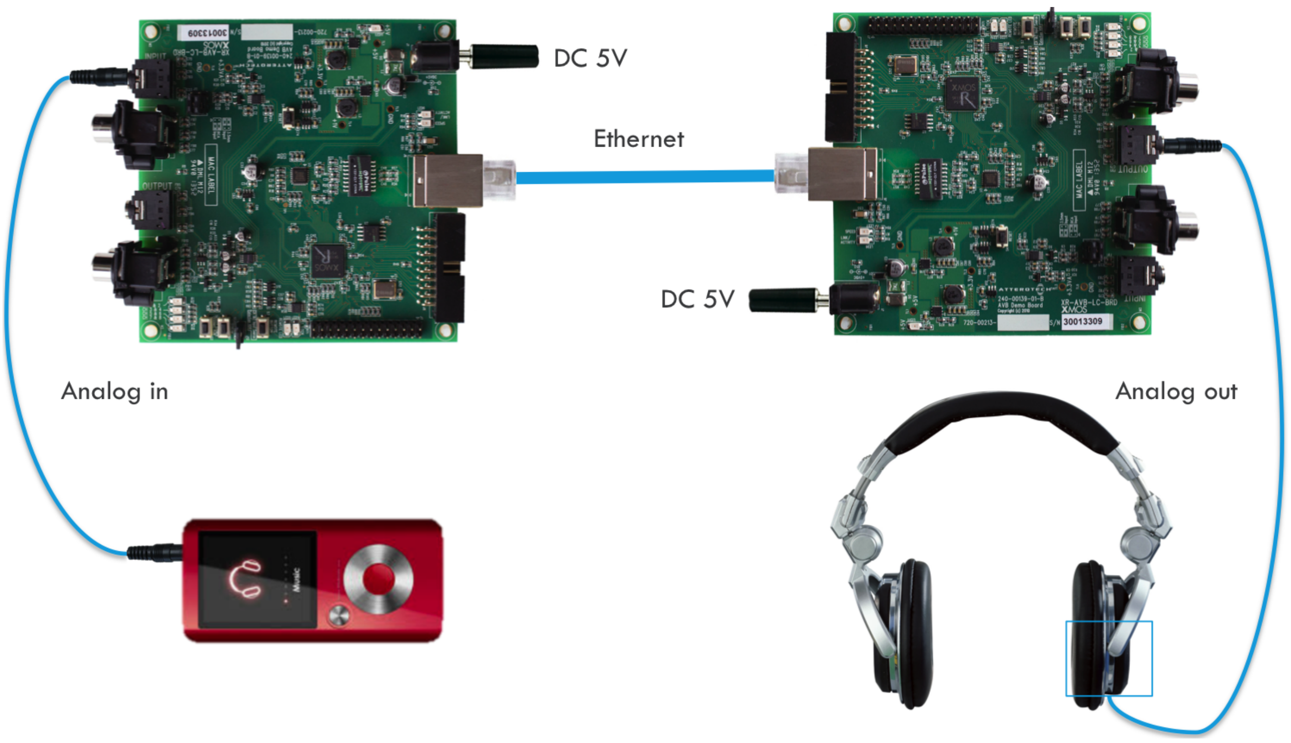

Figure 1: XMOS AVB Talker and Listener network overview

4 Unmodified AVB Endpoint software start-up timing

To provide a benchmark, the stock firmware start-up time was measured. The only change to the firmware was a modification to enable the built in 1722.1 controllers so that each endpoint would discover the other one and connect as both talker and listener. This results in an AVB system consisting of two endpoints connected back to back, which stream 8 channels of bi-directional audio at a sample rate of 48KHz. A summary of the AVB system configuration is as follows:

- 16-core XS1-L16-128 device
- 100Mbit PHY with auto-negotiation of PHY speed
- Single stream with 8 channels at 48KHz 24b
- gPTP with best master clock algorithm (BMCA) enabled using standard sync interval of 125ms
- Media clock master selection: GUID based (derived from MAC address)
- Talker and listener enabled on each endpoint
- Stream reservation protocol (SRP) enabled
- MAAP Enabled
- AVDECC discovery protocol (ADP) and AVDECC connection management protocol (ACMP) for controller and listner enabled. 1722.1 controller enabled on each endpoint to provide “plug and play” streaming system in a back to back system

The full process of booting an AVB endpoint is described in the XMOS AVB Design Guide referenced at the end of this document. However a graphical, simplified representation is included below in Figure 2. Please note that the exact relative timing of each of the boot processes is not shown; the diagram shows which processes occur relative to each of the event epochs described previously.

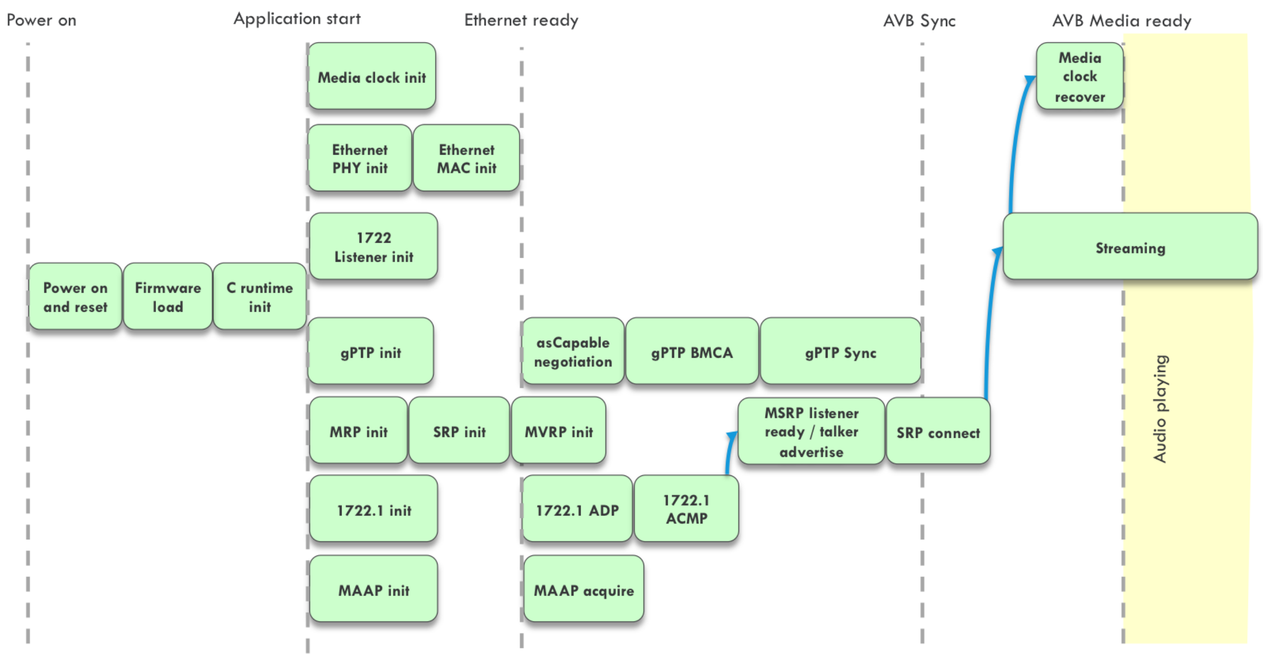

Figure 2: XMOS AVB Endpoint (unmodified) start-up sequence overview

The total time from power-on to audio streaming was measured to be, on average, 4835ms.

4.1 Start-up time summary of the un-optimized system

Start-up segment	**Time taken for segment**	**Elapsed time**
Power on - Application start	885ms	885ms
Application start - Ethernet ready	1500ms	2385ms
Ethernet ready - AVB Sync	1800ms	3885ms
AVB Sync - AVB Media Ready	950ms	4835ms

5 Optimised AVB Endpoint software start-up timing

A number of firmware modifications were made to the AVB Endpoint software to achieve the sub 500ms start-up value. A summary of these changes is given below:

- Power on reset circuit delay reduction
- Ethernet MAC initialisation delay reduction
- Disable PHY auto negotiation
- FLASH boot startup optimisations
- gPTP with static roles (BMCA disabled) with reduced sync interval of 15.625ms
- Static media clock role selection
- Static stream and endpoint configuration for back-to-back connection

The static stream configuration is implemented by making the following changes:

- Stream reservation protocol (SRP) disabled
- MAAP disabled - static talker MAC address assignment
- 1722.1 controller disabled

The combined effect of these changes results in a dramatically reduced start-up time. A simplified representation of the system timeline is included below in Figure 3. You can see that the number processes required to complete to reach audio streaming is dramatically reduced. In particular, it can be seen that streaming commences soon after `Ethernet` ready due to the pre-configured static stream configuration. Please note that the exact relative timing of each of the boot processes is not to scale; the diagram shows which processes occur relative to each of the event epochs described previously.

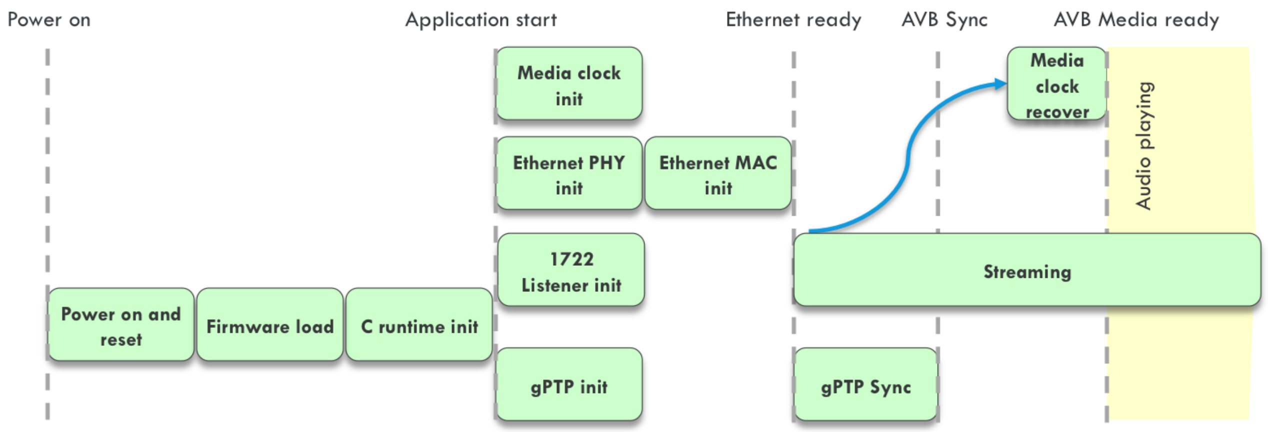

Figure 3: XMOS AVB Endpoint (optimized) start-up sequence overview

The gains are summarised below and further details of each of the changes is provided.

5.1 Start-up time summary

A power-on to audio streaming time of 409ms (average - experimental results ranged from 404 to 412ms) was achieved, comfortably exceed the 500ms target.

Start-up segment	Time from previous event	Elapsed time
Power on - Application start	147ms	147ms
Application start - Ethernet ready	75ms	222ms
Ethernet ready - AVB Sync	81ms	303ms
*AVB Sync - *AVB Media Ready*	106ms	**409ms**

An oscilloscope trace capture showing the 3v3 rail against the analog DAC output of the AVB board is shown below.

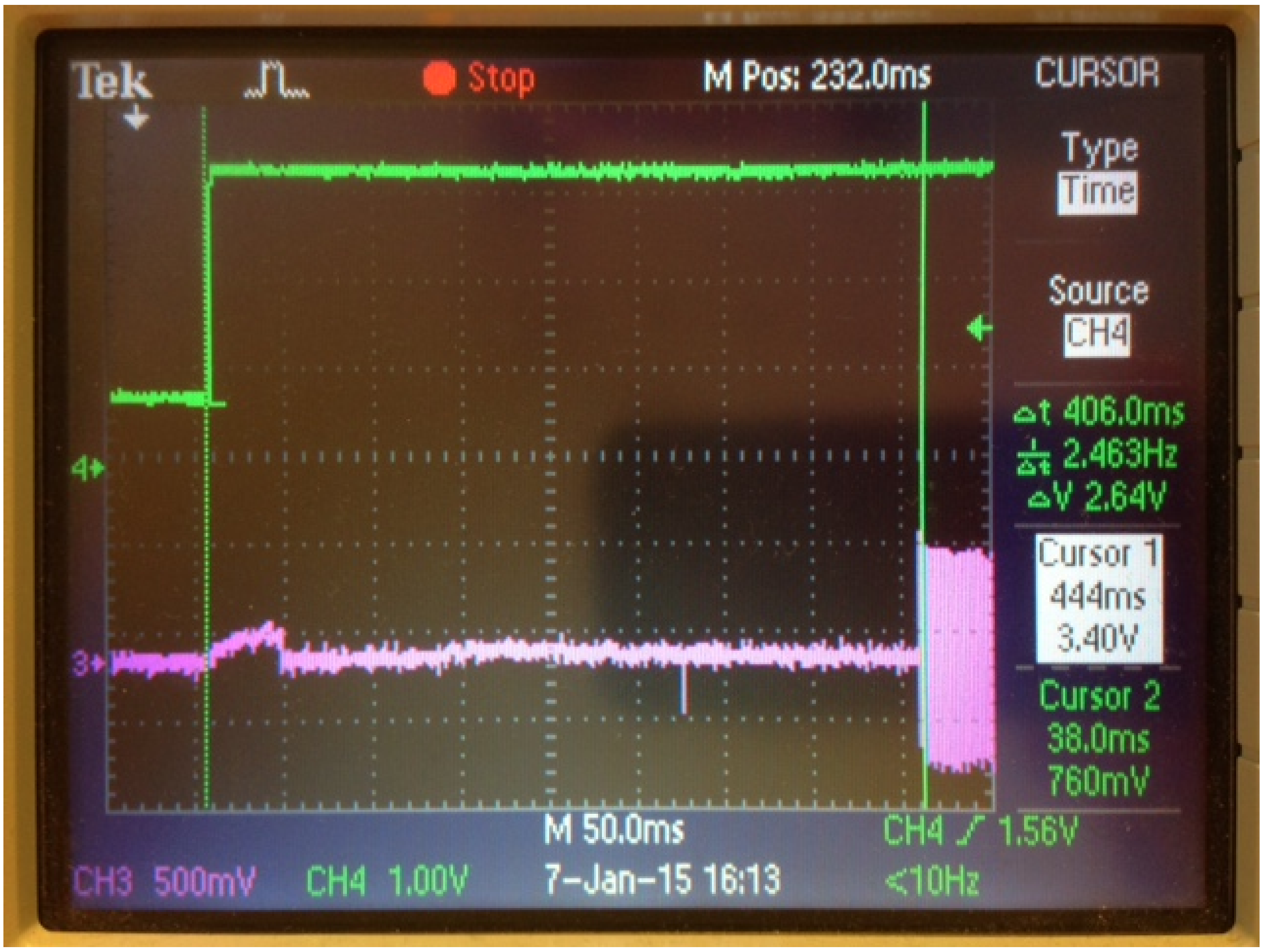

Figure 4: XMOS AVB Endpoint (optimized) start-up measurement

6 Optimization Details

The following section of this application note provides more detailed information on the individual start-up time optimizations used to achieve the reduced startup time.

6.1 Optimizing power-on to application start

This period is defined by the time taken for the power rails to be energised and settle, the reset sequence to complete, the firmware to be loaded and the pre-main() initialisation code to be executed.

6.1.1 Major Components

- Reset de-assertion from voltage supervisors
- Second stage bootloader load from SPI flash
- Firmware image boot from on-board SPI flash

Unoptimized, power-on and reset together with first (ROM boot), second (boot loader) and third (application image) stage boot from the SPI flash together constitute around 780ms of the total delay.

6.1.2 Minor Components

- Power supply start up
- L16 power-on-reset sequence
- Internal ROM boot execute
- Generic L16 initialization code including clock and xConnect network
- C run time and standard library start-up
- Clearing of global arrays designated as zero-initialized
- Spawning application processes on their respective cores

6.1.3 Optimization timing summary

Firmware profile	**Time**
Standard	885ms
Start-up time optimized	147ms

6.1.4 How to apply the optimization

- Increase flash SPI clock speed

Procedure: change clock divider from 3 (16.66MHz) to 2 (25MHz). Specify the clock divider in Flash Configurations ▶ Flash Options: –spi-div 2. Saving: 116ms[4]

Measurement: SPI activity on CS pin U2 1

- Reduce flash boot partition size

Procedure: Configure boot partition size of 256KB (262144). The default is the total size of given flash device, in this case 512KB. Specify the clock divider in Flash Configurations ▶ Flash Options: –boot-partition-size 262144.

[4]Version 13.2 of the tools has a bug which causes the divider to be fixed to 8 when using the winbond_25x40 definition, resulting in a 6.25MHz SPI clock and a load time of around 840ms. This will be fixed in subsequent releases and allow the –spi-div option to take precedence. The optimization has been confirmed to work using tools version 13.1 which does use the specified SPI divider.

Saving: 20ms

Measurement: SPI activity on CS pin U2 1

- Reduce delay between 3v3 and 1v0 and Reset Delay

Procedure: Remove caps C73 and C74 to change default delay to the minimum 20ms. Saving: Around 600ms

6.2 Optimizing Application start to Ethernet ready

The proportion of start-up time between application start to Ethernet ready is defined as the period between application start to execute main() and the PHY and MAC components running, providing an `Ethernet Link` up status. The length of period can be influenced by the hardware mode settings of the Ethernet PHY, but otherwise is completely defined by software. The software execution timing components are library initialization, application initialization and external hardware initialization.

6.2.1 Major Components

- Ethernet PHY reset and configuration
- Setup of MAC filtering and packet queue size
- Configuration of media clock and talker streams and channels

6.2.2 Minor Components

- Obtain endpoint MAC address from the XMOS device's OTP memory

6.2.3 Other initialization functions triggered at Application start, that do not affect time to Ethernet ready

- Initialization of gPTP, MSRP, MVRP, media clock server, 1722 MAAP
- Global 1722.1 initialization
- External PLL or clock multiplier configuration
- Audio CODEC configuration
- I2S I/O driver start
- Initialization of media FIFOs

6.2.4 How to apply the optimizations

- Reduce Ethernet PHY init delay

Procedure: Change PHY_INIT_DELAY in mii.xc from 100ms (10000000) to 1ms (100000). This can be achieved by setting the REDUCE_MAC_INIT_DELAY #define in startup_time_customisations.h

Saving: 99ms

Measurement: Print statements in top level main and Ethernet ready (in ethernet_server_full after mii_init_full).

```
#define REDUCE_MAC_INIT_DELAY       1 //Reduces Ethernet MAC inititialisation delay time
```

- Disable Ethernet speed auto-negotiation

The PHY Autonegotiation can be disabled with a register write.

Procedure: Call the function eth_phy_config_noauto instead of eth_phy_config in avb_ethernet.xc

```
#if FAST_STARTUP
eth_phy_config_noauto(1, ports.smi);
#else
eth_phy_config(1, ports.smi);
#endif
```

Please note that this modification will disable auto-negotiation and so will only achieve link up when connected to fixed 100Mb peers.

Saving: 1-2sec

Measurement: Print statements/IO toggle in top level main and link-up (status message in avb_process_control_packet).

6.2.5 Optimization timing summary

Firmware profile	**Time**
Standard	~1500ms
Start-up time optimized	75ms

6.3 Optimizing Ethernet Ready to AVB Sync

This period is defined by the time to initialise gPTP and transfer at least two sync messages, allowing the slave to lock to the master. In the unoptimized firmware, there is an additional initial delay before sync message transfer while BMCA role selection protocol and asCapable negotiation decide which endpoint is the gPTP master and which is the slave.

Due to the parallel nature of the stack and the XMOS processor, other protocols used within the optimized firmware including SRP domain join and ADP discovery also commence at link up. This means some of the latency from the next phase (AVB Sync to AVB Media Ready) is hidden within this phase.

6.3.1 Major components

- Immediately after link-up, gPTP performs a BMCA role selection (assigns gPTP clock roles dynamically) and negotiates asCapable (determines if endpoint is capable of suitable 802.1AS performance). This communication with the peer is followed by delay reporting/reception which results in a sync[5] condition when the two clocks converge. The overall time from link-up to PTP sync is around 1800ms.

6.3.2 How to apply the optimization

- Statically configure gPTP roles so that one endpoint is designated the master and one is the slave.

This can be achieved by setting the STATIC_PTP_ROLES #define in startup_time_customisations.h. Setting this define disables dynamic PTP reset, disables calls to BMCA, sets the gPTP clock role based on Ethernet MAC address[6] and forces asCapable to 1.

[5] The PTP sync condition is defined by the implementation as a minimum of 5 (2 in the optimized case) FollowUp messages and a sub-0.01% change in the *adjust* value. The *adjust* value is the error ratio of the Sync-to-Sync times seen by both sides (see gPTP's `update_adjust` function for reference).

[6] The last octet of the board's ethernet MAC addresses must be set in startup_time_customisations.h. This provides a simple identification of each board allowing different paramenters for static configuration for each baord to be set, while using the same firmware binary image for both boards. This approach assumes that the last octet of the ethernet MAC address is different between the two boards. Please check this is the case (the address is printed in the console when running the firmware with

- Increase sync rate so that the clocks can converge more quickly and reduce the minimum number of sync lock messages required to meet the lock condition.

This optimizations can be applied by setting the SPEEDUP_PTP_SYNC_MESSAGES in startup_time_customisations.h.

```
#define STATIC_PTP_ROLES           1 //Disables BMCA and forces asCapable
#define SPEEDUP_PTP_SYNC_MESSAGES  1 //Increases PTP_SYNC rate and reduces minimum meesage lock count
```

6.3.3 Optimization timing summary

Firmware profile	Time
Standard	~1800ms
Start-up time optimized	81ms

6.4 Optimizing AVB Sync to AVB Media Ready

In order to allow AVB Media to stream, the endpoint must have an established media clock which is stable. If the media clock is derived from another talker, then it must already have locked to input stream[7] . Before streaming can finally take place, the following additional conditions must also be met:

- A valid talker stream MAC address, normally established by MAC address acquisition protocol (MAAP). The MAAP reservation represents a time of 1 second because the probing protocol going through two 500msec timeouts.
- Attribute declaration and join request via Multiple Attribute Declaration (MAD), Multiple Registration Protocol (MRP) and Multiple VLAN Registration Protocol (MVRP). These are necessary to advertise entities and join streams within the AVB network
- Advertisement, discovery and connection of endpoints via ADP and ACMP, part of the 1722.1 standards for audio/video discovery, enumeration, connection management and control (AVDECC). This is the process of actually discovering and connecting streams, the latter being implemented by Multiple Stream Reservation Protocol (MSRP).

However, in a start-up optimized system, we can use a static configuration for the VLAN ID, talker multicast MAC address and stream parameters. This removes the need for any of the above protocols which significantly speeds start-up; the only protocol related requirement for streaming audio to I2S is an established media clock.

6.4.1 Major components

- The talker's media input FIFO is enabled by a successful registration of a Listener Vector MSRP attribute. Successful registration is represented by a MAD Join indication. The indication causes the FIFO in the 1722 talker to be enabled.
- After enabling, talker's media input FIFO goes through about 1ms of prefill period.
- The Listener Vector MAD Join request is generated at the end of the previous start-up phase, when the ACMP connection sequence is completed and the *on talker connect* callback is invoked on the listener.
- The listener vector attribute declaration is communicated by sending a new message inside a transmission opportunity window, which runs on a 200ms interval (MRP join timer). The worst case timing

xscope printing enabled). The board with the last octet of it's MAC address matching this define will be assigned the slave role for gPTP.

[7] The definition of media clock lock condition is as a period of 32 media clock updates (one every 250 media output FIFO samples) during which the requested-to-actual presentation time only changes by the equivalent of 1 sample time or less and remains within ±308 (see media clock server's `manage_buffer` function for reference).

is just missing the transmission opportunity and delaying 200ms.
- Once the talker prefills its FIFO, it starts streaming 1722 data. The listener discards first 16 samples, which takes about 2ms. Then it clears its FIFO in about 7ms by stepping through it one sample at a time.
- With a zeroed-out FIFO, the listener enters a media recovery period, ending with a media clock lock condition. As soon as media clock lock is reached, the I2S output driver is provided with valid samples, starting audio output. Typical length of the media recovery period is within 180ms and 550ms.
- The media clock recovery period may get restarted, causing additional 180-550ms of delay[8].

6.4.2 How to apply the optimization

- Disable MAAP and assign a static multicast MAC address

Procedure: Add a new function to generate a static address (based on the Ethernet MAC address) at startup and remove the call to the MAAP periodic function. This can be enabled via the DISABLE_MAAP define in startup_time_customisations.h.

- Configure the stream statically

Procedure: Disable the 1722.1 AVDECC controller and endpoint entity by disabling functionality on talker and listener using avb_conf.h defines. Disable the SRP task and replace it with a static configuration for QAV bandwidth of the stream. Set the source and sink stream information statically at startup of the AVB application and then immediately enable the stream. Static stream information includes the sink address, media clock, channel count and map, stream ID and VLAN number. All of these optimizations can applied to the modified AVB Endpoint software using the STATIC_STREAM_CONFIG #define in startup_time_customisations.h[9].

```
#define DISABLE_MAAP                1 //Disables MAAP
#define STATIC_STREAM_CONFIG        1 //Disables SRP & 1722.1
```

6.4.3 Optimization timing summary

Firmware profile	**Time**
Standard	950ms
Start-up time optimized	106ms

[8]The reason could be the requested-to-actual presentation time offset being larger than 308 samples prior to reaching the lock condition (exceeding the media output FIFO's ability to adjust for the offset), or growing over 24 samples after reaching the lock condition (an *unlock* condition).

[9]The STATIC_STREAM_CONFIG optimization will only work if applied along when MAAP is also disabled.

7 Compiling and running the modified firmware

7.1 Obtaining the latest firmware

1. Log into xmos.com and access *My XMOS* ▶ *Reference Designs*
2. Request access to the *AVB Endpoint Software* by clicking the *Request Access* link under *AVB Audio Endpoint*. An email will be sent to your registered email address when access is granted.
3. A *Download* link will appear where the *Request Access* link previously appeared. Click on 'All versions' and expand out the list to reveal all AVB software versions.
4. Click on version 6.1.1 and download the firmware zip.

7.2 Installing xTIMEcomposer Studio

The AVB-LC software requires xTIMEcomposer version 13.2, available for download at xmos.com[10].

7.3 Importing the firmware

To import and build the firmware, open xTIMEcomposer Studio and follow these steps:

1. Choose *File* ▶ *Import*.
2. Choose *General* ▶ *Existing Projects into Workspace* and click **Next**.
3. Click **Browse** next to **'Select archive file'** and select the firmware .zip file downloaded in section 1.
4. Make sure that all projects are ticked in the *Projects* list.
5. Click **Finish**.

7.4 Applying the start-up time software modifications

1. Copy the start-up time optimized files over the fresh v6.1.1 AVB Endpoint software. The files are provided with the correct directory structure so unzipping recursively inside the top level of the project directory will apply all of the modifications correctly in a single operation.
2. Set the MAC addresses of both boards in the Software. This is achieved by changing the following define values in startup_time_customisations.h to match the MAC addresses of your two Boards.

```
// Example Mac Address of one Endpoint
#define LC_BOARD_0_MAC_ADDR {0x00, 0x22, 0x97, 0x80, 0x41, 0x11}

// Example Mac Address of the other Endpoint
#define LC_BOARD_1_MAC_ADDR {0x00, 0x22, 0x97, 0x80, 0x41, 0x42}
```

The start-up time optimized code uses the unique MAC address of the boards to define different roles to each of the endpoints. The MAC address of each endpoint is printed out over xScope at startup by the modified AVB Endpoint software.

1. Check startup_time_customisations.h to ensure that the chosen optimizations are enabled. The instructions to this are in the previous sections.

7.5 Building the firmware

1. Select the app_avb_lc_demo project in the Project Explorer and click the **Build** icon in the main toolbar.

[10] http://www.xmos.com/support/xtools

7.6 Installing the application onto flash memory

1. Connect the xTAG-2 debug adapter (XA-SK-XTAG2) to the first AVB endpoint board.
2. Plug the xTAG-2 into your development system via USB.
3. Plug in the 5V power adapter and connect it to the AVB endpoint board.
4. In xTIMEcomposer, right-click on the binary within the *app_avb_lc_demo/bin* folder of the project.
5. Choose *Flash As* ▶ *Flash Configurations*.
6. Double click *xCORE Application* in the left panel.
7. Choose *hardware* in *Device options* and select the relevant xTAG-2 adapter.
8. Click on **Apply** if configuration has changed.
9. Click on **Flash**. Once completed, reset the AVB endpoint board using the reset button.
10. Repeat steps 1 through 8 for the second endpoint using the same binary.

7.7 Setting up the hardware

Other than applying power at the same time, no special hardware instructions are necessary. Both endpoints are talker/listners, so the audio is bi-directional. Consequently the analog input should be used on one board, and the output on the other. It does not matter which is which. Audio will automatically start streaming within the start-up time period.

8 Further Work

The optimizations discussed here are only some of the potential ideas to further reduce start-up time. It is anticipated that around a further 100ms could be saved if all of the below suggestions were applied.

1. Use non volatile storage and readback of gPTP parameters. By storing previously measured and configured gPTP delays and configurations in flash, configuration can be flexible yet still offer fast start-up.
2. Firmware boot loader (stage 2) optimization. Use PLL parameters that give shorter PLL lock time, optimized secondary node boot and a customized initial stage to reduce the time spent at the default SPI (slow) clock speed.
3. Faster boot using wider SPI bus. Implement quad SPI boot mechanism.
4. Selection of faster reset supervisors. Currently, the reset supervisor imposes a minimum 20ms delay before the 1v0 rail is enabled, and a further 20ms before reset is de-asserted. These delays can be significantly reduced as long as the “3v3 first” power sequencing required by the XMOS chip is preserved. The DC-DC converters used take only around 300us to power up, so do not introduce appreciable delay and are already optimal.

8.1 References

XMOS Tools User Guide

http://www.xmos.com/published/xtimecomposer-user-guide

XMOS xCORE Programming Guide

http://www.xmos.com/published/xmos-programming-guide

XMOS AVB Design Guide:

http://www.xmos.com/published/avb-design-guide

IEEE 1722-2011:

http://standards.ieee.org/findstds/standard/1722-2011.html

IEC 61883-4:

http://webstore.iec.ch/Webstore/webstore.nsf/ArtNum_PK/32987

EN 50083-9 2002

https://www.dvb.org/resources/public/standards/En50083-9.2002.pdf