Generating an AES11-compatible media clock in a System-on-Chip (SoC) for use in real-time audio streaming via AoIP

Zain Alabedin Alishammat
Capture and Display Systems (CDS)
Fraunhofer HHI
Berlin, Germany
zain.alabedin.alishammat@hhi.fraunhofer.de

Thorben Kron
Capture and Display Systems (CDS)
Fraunhofer HHI
Berlin, Germany
thorben.kron@hhi.fraunhofer.de

***Abstract*—The transmission of high-quality, low-latency audio over IP (AoIP) networks demands precise synchronization of distributed devices, particularly through alignment of local media clocks to a global reference via IEEE 1588 Precision Time Protocol (PTP). A media clock is a high-precision timing signal used to control the sampling, playback, or recording of digital media—typically audio or video—at a consistent and accurate rate. It ensures that all devices in a network play out or capture audio samples in perfect temporal alignment.**

The generation of media clocks for AoIP systems remains an open challenge, as no widely adopted standard or open-source reference design currently exists. Existing solutions are typically implemented on FPGAs, often relying on proprietary architectures with limited transparency and accessibility. The only established guideline is the AES11 standard, which specifies stringent requirements for such clocks—limiting jitter to ±5% of the sample period—and has been adopted in AES67 and RAVENNA. This lack of standardized or openly available implementations, however, leaves the practical derivation of a stable, PTP-synchronized media clock largely unclear.

To address this gap, this paper presents a hardware–software co-design methodology for deriving an AES11-compliant media clock from IEEE 1588 PTP. The proposed approach combines the Common Platform Time Sync (CPTS) module of Texas Instruments' AM62x System-on-Chip (SoC) with the LMK05318B jitter-cleaning clock generator. By leveraging a pulse-per-second (PPS) signal phase-locked to a PTP grandmaster, the system achieves sub-100 ns alignment accuracy and 35 ps RMS jitter, surpassing AES11 requirements.

The implementation demonstrates how cost-efficient embedded platforms without native media clocking capabilities can be adapted for professional AoIP systems such as AES67, while maintaining interoperability and scalability. Although validated on the AM62x and LMK05318B, the methodology is transferable to other embedded platforms featuring PTP-based timestamping and any clock synchronizer supporting IEEE 1588.

***Index Terms*—Media Clock, AoIP, AES11, clock synchronisation, IEEE-1588 PTP, wall clock.**

I. Introduction

The shift from legacy digital audio interfaces (e.g., AES3) to AoIP has transformed broadcasting, live sound, and telecommunications by enabling scalable, network-based audio distribution. Protocols like AES67 [1] and RAVENNA [2] standardize AoIP implementations, but their performance hinges on precise synchronization—a challenge exacerbated by the separation of audio payloads from timing references in IP networks. Unlike traditional interfaces like AES3, where clocks are embedded in the data stream [3], AoIP relies on external synchronization mechanisms, typically IEEE 1588 PTP [4], to align media clocks across devices with sub-microsecond accuracy. In AoIP systems, media clocks are not generated autonomously by the protocol stack; rather, they are externally supplied by the subnodes. If a node receives a media clock that fails to meet AES11 or other relevant timing specifications—particularly with respect to accuracy and jitter—it is considered non-compliant and may be excluded from participation in the audio network [5].

The AES11 [6] standard defines stringent requirements for such clocks, limiting jitter to $\pm 5\%$ of the sample period (at 48 kHz: $\pm 1\mu s$).

AoIP protocols such as RAVENNA call for $\pm\frac{1}{2}$ sample period [7] for sample-accurate stream alignment (at 48 kHz: $\pm 10\mu s$).

Meeting these constraints in embedded systems often necessitates specialized clock generators, as general-purpose SoCs like the AM62x [8] lack dedicated hardware for media clock synthesis. While the AM62x integrates PTP-capable Ethernet and timestamping peripherals (CPTS), its inability to directly generate audio-grade clocks poses a barrier to AoIP adoption in resource-constrained applications.

This work addresses this limitation by integrating the AM62x with the LMK05318B [9], a jitter-cleaning clock synchronizer, to generate an AES11-compliant 49.152 MHz media clock. The approach is highly flexible, as any desired media clock frequency can be obtained through the configuration of external audio PLLs or clock synthesizers. In this implementation, the generated clock drives a 64-channel MEMS microphone array [10] via the SoC's Multichannel Audio Serial Port (McASP) interface. Each channel operates

with 32-bit resolution at a 48 kHz sampling rate, resulting in the following overall clock frequency requirement:

$$64 \times 32 \times 48 \text{ kHz} = 49.152 \text{ MHz} \quad (1)$$

The key contributions of this work include:

- **A PPS-based synchronisation pipeline:** Generation a Pulse Per Second (PPS, 1 Hz) reference signal, custom modifications to the CPTS driver enable the derivation of a 1 kHz reference rather than 1 Hz from the AM62x's synchronised Physical Hardware Clock (PHC). This significantly reduces the lock time of the LMK05318B jitter cleaner from approximately 30 minutes (when using a 1 Hz input) to under one second, enabling fast and reliable media clock acquisition.
- **Phase-coherent clock multiplication:** The LMK05318B upconverts the 1 kHz reference to 49.152 MHz with 4 ns phase offset, achieving cumulative phase offset of 74 ns—well within AoIP end-to-end latency limits of $\pm\frac{1}{2}$ sample period.
- **Jitter performance validation:** Measurements confirm 35 ps RMS jitter at 49.152 MHz, compliant with AES67 and suitable for professional audio applications.

II. State fo the art

Despite the central importance of media clock generation in AoIP systems, there is a striking lack of openly available research or technical reports detailing how manufacturers and implementers realize this process in practice. Existing standards (AES67, RAVENNA, MILAN, etc.) deliberately refrain from prescribing implementation methods, and most of the available documentation is limited to datasheets or application notes provided by clock synthesizer vendors. To the best of our knowledge, no open-source reference designs or peer-reviewed publications systematically explain how AES11-compliant media clocks can be derived from PTP-synchronized references in embedded platforms. Commercial AoIP solutions, such as modules from ROSS, Dante, or XMOS, typically rely on external audio PLLs like the Cirrus Logic CS2300 or similar devices to generate the required media clock. However, these implementations do not disclose how the generated clock is synchronized to PTP, since their FPGA-based designs are proprietary and tied to vendor-specific development environments.

This gap underlines the relevance of the present work, as it provides a open-source , reproducible and validated methodology for achieving deterministic, low-jitter media clock generation in resource-constrained systems.

The synchronization of distributed audio devices in AoIP systems is predominantly achieved through the IEEE 1588 PTP, which establishes a common time reference across all networked nodes. All major AoIP protocols, including AES67, RAVENNA, and MILAN, share the same conceptual model: first, each device synchronizes its local system clock to the global PTP wall clock; second, a media clock must be derived from this synchronized reference, as shown in figure 1

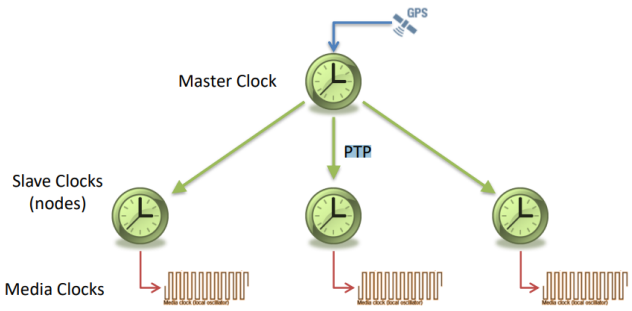

Fig. 1. Media clock generation scheme [1]

Crucially, none of these standards define, recommend, or even mention how the media clock should actually be generated. This task is entirely left to implementers.

As a result, several implementation approaches have emerged in practice. A common denominator among all of them is the use of external clock synthesizers or jitter-cleaning generators. General-purpose SoCs and FPGAs are unable to directly produce audio-grade media clocks in the required MHz range with sufficiently low jitter. Consequently, external devices—such as the Cirrus Logic CS2600 family or Texas Instruments' LMK series—are widely employed, both to multiply a low-frequency reference signal to the target audio frequency and to suppress jitter in order to meet AES11 timing specifications.

Two principal implementation strategies can be distinguished:

1) **Reference-fed architecture:** A low-frequency reference (e.g., PPS or a kHz-range signal) is provided by the SoC or FPGA to the external clock generator—an approach also applied in this work. This low-frequency must be generated from the timestamp generator of the used SoC or FPGA since the media clock and the network clock (PTP) shall share the IEEE 1588 epoch of 1 January 1970 00:00:00 TAI, as defined in IEEE 1588-2008 clause 7.2.2[11].
   The clock generator then multiplies this reference up to the required audio frequency (e.g., 24.576 MHz or 49.152 MHz) while simultaneously performing jitter cleaning and maintain a deterministic and minimum phase offset, as illustrated in figure 2
2) **Interface-controlled architecture [12]:** In this case, the external clock generator operates without an injected reference signal, functioning effectively as a Digitally Controlled Oscillator (DCO). Instead, it is programmed via serial interfaces such as I²C or SPI from the SoC or FPGA. The target output frequency and phase offset are continuously computed through the timestampe feature of the SoC or FPGA and updated through register configurations. To timestamp the output frequency, a small reference signal (e.g., 1 Hz) must be derived from the same

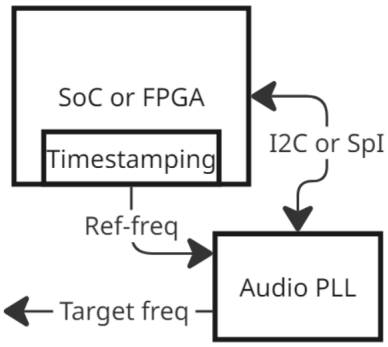

Fig. 2. Reference-fed architecture approche

audio PLL and phase-locked to the target frequency (not all audio PLLs support this). This is necessary because timestamping high frequencies can be impractical, often leading to buffer overflows and data loss. For example, the AM62x hardware timestamping inputs (Hardware Time Stamp Push Event) are not designed to capture high frequencies [8, p. 1324]. Figure 3 illustrates this approche. While this method offers greater flexibility, it is more complex: it requires ongoing calibration, offset compensation, and phase adjustment to maintain compliance with AES11 and AoIP timing requirements.

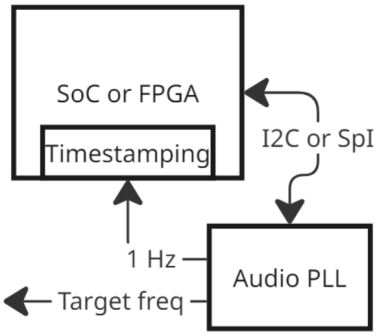

Fig. 3. Interface-controlled architecture approche

III. The SoC AM62x

The AM62x Sitara™ from Texas Instruments (TI) is a Linux®-capable SoC specifically designed for industrial and embedded applications. It is based on the Keystone 3 (K3) multicore architecture platform and offers a wide range of interfaces for real-time and multimedia applications. It comes in a 13 x 13 mm package (ALW) and meets the AEC-Q100 automotive standard in a 17.2 x 17.2 mm package (AMC).

The AM62x SoC is divided into several device domains to efficiently handle different computing, communication, and control tasks. These domains share various hardware resources and are interconnected via the CBASS module.

1) **MAIN Domain**: The MAIN domain includes the 1.4GHz Arm® Quad-core Cortex®-A53, which serve as the main processors for operating systems such as Linux or RTOS. It also contains the two instances of PRUSS (Programmable Real-Time Unit Subsystem) with 353MHz.
2) **MCU Domain**: The MCU domain hosts the 400MHz Arm® Cortex®-M4F core, used for general-purpose tasks or safety-critical functions.
3) **WKUP Domain**: The WKUP (Wake-Up) domain contains the R5F processor (R5FSS). This domain remains active during low-power or sleep modes.

IV. The Common Platform Time Sync (CPTS)

The CPTS module is used to facilitate host control of time sync operations [8, p. 1316]. It enables compliance with the IEEE 1588-2008 (PTPv2) standard for a precision clock synchronisation protocol and includes, among others, the following features:

- 4 hardware timestamp inputs (CPTS_HWn_TS_PUSH, n = 0–3), capable of capturing external timestamp events with 64-bit resolution.
- 2 timestamp generator outputs (CPTS_GENFn, n = 1–2), capable of generating signals such as PPS

V. Audio over IP (AoIP)

Audio over IP (AoIP) refers to the transmission of digital audio streams over IP-based networks and has become a standard technology in professional audio production, broadcasting, and telecommunications. By encapsulating multiple audio samples within individual IP packets, AoIP enables efficient and scalable transmission over local LAN-networks using standard Ethernet infrastructure.

A key advantage of AoIP is the ability to transport multichannel audio and control data over a single network cable, reducing cabling complexity and enabling flexible system architectures. Unlike legacy Audio-over-Ethernet systems based on Layer 2 protocols, AoIP operates at Layer 3 (IP), allowing for routing across complex network topologies and seamless integration into existing IT environments.

Core Components of Modern AoIP Systems:

- **IP-based Layered Communication:** AoIP systems leverage standardized IP networking protocols, such as UDP and RTP, for the transmission of audio data. The layered architecture—based on the OSI model—enables modular development, ensures interoperability across diverse network infrastructures, and facilitates integration into existing IT environments.
- **Real-Time Transport via RTP:** The Real-time Transport Protocol (RTP) is widely used for delivering audio in AoIP systems. It includes timestamps and sequence numbers to support low-latency, synchronized playback. RTP

is typically encapsulated over UDP and IP within Ethernet frames, with a maximum frame size of 1518 bytes (1522 with VLAN tagging). Each layer adds protocol-specific headers, with RTP carrying the actual audio payload.

- **Precise Synchronization with IEEE 1588 (PTP):** Synchronization is critical for maintaining phase coherence and minimizing latency in distributed audio systems. The Precision Time Protocol (PTP), as defined by IEEE 1588, enables sub-microsecond clock synchronization by establishing a Grandmaster-Slave hierarchy. Hardware timestamping ensures precise alignment between the local media clocks and the global reference time. This is essential for applications such as multichannel recording, live audio routing, and broadcast audio playout.

VI. Expermintal setup

Figure 4 illustrates the hardware and software components beeing used for the development, testing, and evaluation of the AM625 SoC to generate the media clock (49.152 MHz):

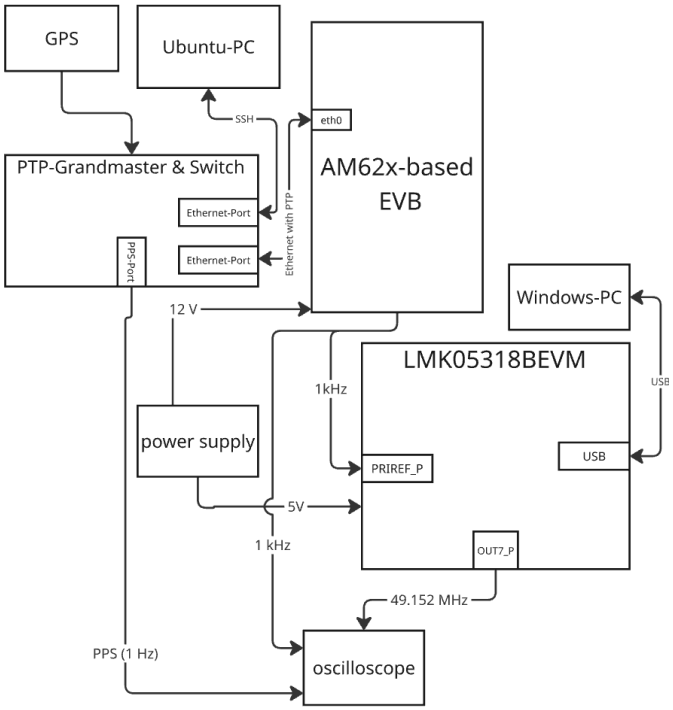

Fig. 4. Experimental setup for generating the 49.152 MHz

- **VAR-SOM-AM62 System-on-Module (SoM) [13] and Symphony V1.6 Evaluation Board** [14]
  The Symphony evaluation board, equipped with the VAR-SOM-AM62 module, served as the primary development platform for AM625.
- **Ubuntu 22.04 PC (Host System)**
  Used for building a Debian image with a `PREEMPT_RT` kernel for the AM62x (crucial for AoIP tasks)
- **Windows PC**
  Employed for configuring the LMK05318B clock generator via TI's *TICS Pro* (Texas Instruments Clocks and Synthesizers) software. TICS Pro allows programming of the registers and on-chip EEPROM of the LMK05318B. TICS Pro is available only for Windows OS.
- **PTP Grandmaster-Capable Ethernet Switch**
  Facilitated Precision Time Protocol (PTP) synchronization testing.
- **LMK05318BEVM Evaluation Module**
  For the evaluation of the LMK05318B, the development board LMK05318BEVM—which integrates the LMK05318B—was acquired. The LMK05318B is a network synchronization clock generator that offers jitter correction, clock generation, advanced clock monitoring, and glitch-free switching behavior to meet the strict timing requirements of IEEE 1588 PTP. The EVM features SMA connectors (Sub-Miniature Version A) for clock inputs, additional oscillator inputs, and clock outputs with $50\,\Omega$ impedance. Configuration of the board is performed via I²C or SPI using the on-board integrated microcontroller.
- **Oscilloscope or frequenz counter**
  Used for signal and jitter analysis

VII. Generating the media clock

As mentioned before, the proposed methodology is transferable to other embedded platforms that provide PTP-based clock synchronization and timestamping functionality (e.g., CPTS or comparable modules). Likewise, any clock synchronizer supporting IEEE 1588 PTP can adopt this architecture to generate AES11-compliant media clocks suitable for professional AoIP applications.

TI's AM62x is not designed to generate a media clock, but it does feature two 1-Gigabit Ethernet interfaces with TSN (Time-Sensitive Networking) functionality. The following steps were taken sequentially to generate the media clock:

1) **Generating a PPS signal:** PPS is a precise timing signal that generates a short pulse exactly once per second (1 Hz), with extremely fast rise times (typically less than 5 ns, often around 2 ns). It is commonly used as a reference signal for the synchronization of clocks in PTP environments across various systems, particularly in networks.
   PPS is typically output by devices such as PTP Grandmasters via a dedicated connector as a separate signal. This allows the PPS signal generated by a slave device to be compared with the master's PPS signal using an oscilloscope, enabling precise verification of synchronization accuracy.
   The generation of the PPS signal on the AM62x platform is achieved exclusively through connecting one timestamp generator (CPTS_GENFn) with one hardware timestamp

input (HWn_TS_PUSH) via the Time Sync Router (TSR) [15], because CPTS doesn't have directly HW support for PPS signal generation or any signal that is synchronised to PTP, as illustrated in figure 5

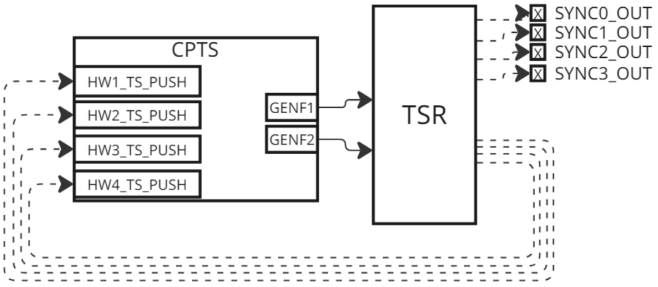

Fig. 5. CPTS and TSR integration in AM62x

The reason this modulation is necessary is that the GENFs, by default, derive their reference frequency from the CPTS reference clock, which in turn is driven by an internal PLL of the AM62x. This PLL is not synchronized to any external time source, as illustrated in the figures 6 and 7.

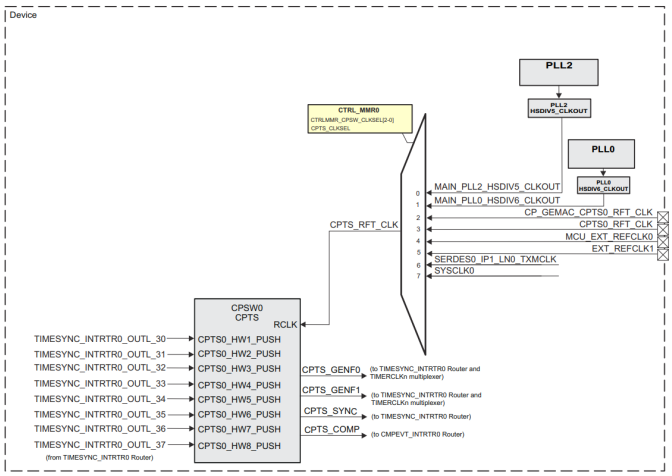

Fig. 6. CPTS integration [8, p. 1317]

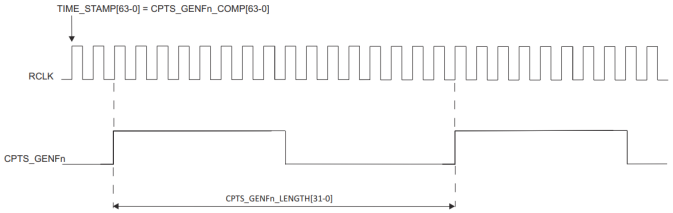

Fig. 7. CPTS_GENFn Output Signal Diagram [8, p. 1322]

Therefore, GENF must always be adjusted to output a signal that is synchronous with PTP. This is achieved by consistently capturing the GENF timestamps for each rising edge through the HWn_TS_PUSH inputs and then applying the necessary corrections.

The TSR is designed to send one sync signal to multiple recipients. This sync signal allows multiple peripherals or cores within the processor to synchronise their counters to a single "master clock" [8, p. 833].
TSR signals can also be routed to processor pins (SYNCn_OUT, n = 0 - 3) if the synchronization signal needs to be received by an external device. Several module pins, including those of the CPTS, are directly connected to the TSR.
It should be noted that the two outputs GENF1 and GENF2 of the CPTS module are internally and permanently linked to the TSR. In addition, the TSR provides multiple outputs, including four CPTS_HW_PUSH lines, which can be internally routed to the CPTS module but must first be enabled. Once this internal connection is successfully established, the PPS signal must also be made available externally, so it can be analysed on an oscilloscope and eventually serve as a media clock source. To achieve this, the signal from GENFn must additionally be routed through the TSR to a SYNCn_OUT pin.
To enable this connection, the `timesync_router` node in the Device Tree (DT) file of the evaluation board (`k3-am625-var-somsymphony.dts`) has to be added and extended, as shown below:

```
#define K3_TS_OFFSET(pa, val)   (0x4+(pa)*4)
    (0x10000 | val)
&timesync_router {
.
.
    cpsw_cpts: cpsw-cpts {
    pinctrl-single,pins = <
        /* pps [cpsw cpts genf0] in16 ->
            out10 [cpsw cpts hw1_push] */
        K3_TS_OFFSET(10, 16)
        /*pps [cpsw cpts genf0] in16 -> out21
            [SYNC1_OUT Pin] */
        K3_TS_OFFSET(21, 16)
        >;
    };
};
```

Here GENF1 was routed to the HW1_TS_PUSH and SYNC1_OUT pins. The `ti,pps` property must also be added to the `cpts@3d000` node to enable GENF1 to be mapped to HW3_TS_PUSH, as described in the CPTS's binding (`ti,k3-am654-cpts.yaml`).

```
cpts@3d000 {
        /* MAP HW1_TS_PUSH to GENF0 */
        ti,pps = <0 1>;
};
```

The `ti,pps` property follows the format `<(HWx_TS_PUSH - 1) (GENFy)>`, meaning 0 for `HW1` (1-1=0) and 1 for `GENF1`.

2) **Implementing PTP in linux:** Once the DT has been recompiled and the AM62x restarted, the user must start `ptp4l` to implement PTP in the AM62x, as well as start

pch2sys to synchronise the Linux time with the PTP. This ensures that the AM62x's PPS is phase-locked to the grandmaster's PPS.

```
ptp4l -E -4 -H -i eth0 -s -m -q -p /dev/ptp0
phc2sys -s /dev/ptp0 -c CLOCK_REALTIME -O 0 -
    m -r
```

When the CPTS module is successfully enabled in the kernel, it exports a kernel interface for specific clock drivers and a PTP clock API user space interface, enabling support for SIOCSHWTSTAMP and SIOCGHWTSTAMP socket ioctls. PTP exposes the PHC as a character device with standardised ioctls, which can usually be found at the following path:

```
/dev/ptpN
```

If this interface doesn't exist, then the CPTS driver weren't enabled properly in the kernel.
Once the ptp4l has chosen the best PTP grandmaster and synchronised the AM62x's PHC (ptp0) to it, the following command can be run to enable the PPS signal:

```
echo 1 > /sys/class/ptp/ptp0/pps_enable
```

If the pps_enable cannot be found, then the ti,pps property is probably not configured or is not present in the DT.

Figure 8 shows the PPS signals from both the AM62x and the PTP Grandmaster as observed on the oscilloscope.

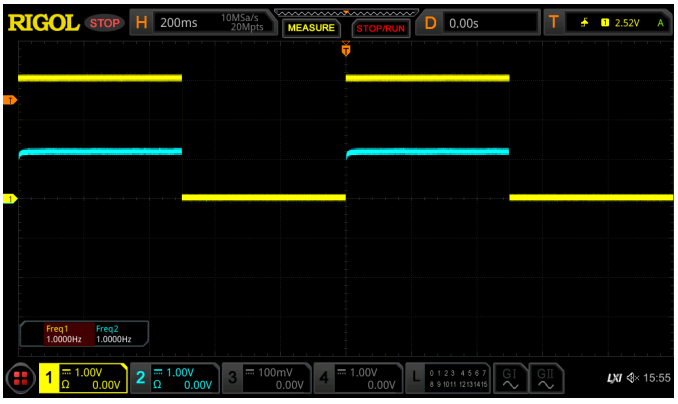

Fig. 8. PPS signal of the PTP Grandmaster (yellow) and the AM625 (blue)

The PPS signal of the AM62 (blue) is phase-locked with the PPS signal of the PTP Grandmaster (yellow). However, there is a phase offset of approximately 70 ns, as shown in Figure 9. A PPS signal was successfully generated and phase-locked to the PTP Grandmaster.

3) **Adapting the CPTS device driver:** After adapting the CPTS device driver (am65-cpts.c), the PPS signal is increased from 1 Hz to 1 kHz, so that the clock synthesizer LMK05318B can phase-synchronously upconvert the 1 kHz signal to the target frequency of 49.152 MHz in less than one second. With a frequency

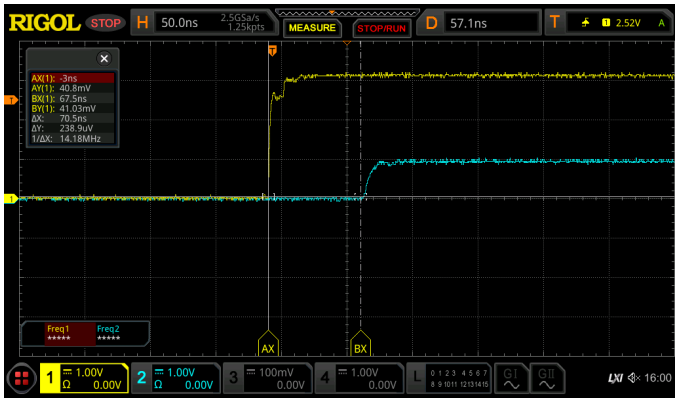

Fig. 9. Phase offset of PPS Grandmaster and PPS AM625 (70ns)

of 1 Hz, it takes the LMK05318B almost 30 minutes to generate a frequency of 49.152 MHz with deterministic phase locking. This prolonged lock time is due to the fact that the DPLL's Time-to-Digital Converter (TDC) operates directly from the low-frequency reference (1 Hz), resulting in very slow phase correction. the same experiment was performed. Figure 10 shows the 1 kHz output of the GENF0, rather than the 1 Hz output.

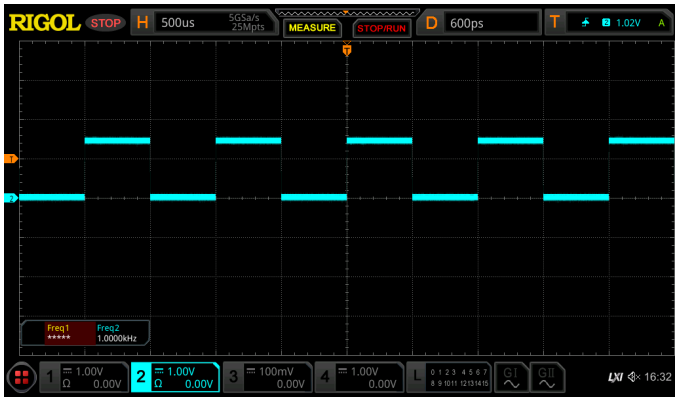

Fig. 10. CPTS driver with 1 kHz

4) **Using the LMK05318B:** To meet the target frequency of 49.152 MHz, the clock synchroniser LMK05318B was used to upconvert the 1 kHz while maintaining phase alignment.
The LMK05318B was configured in three modes: DPLL mode, Zero Delay Mode (ZDM), and cascaded mode. In DPLL mode, the device employs its internal digital PLL to lock to the 1 kHz reference frequency. In ZDM, a deterministic input–output phase delay of ±1 VCO cycle can be achieved between the selected DPLL reference input (1 kHz) and the OUT7 (one of seven ouput channels), which outputs the 49.152 MHz. Figure 4 illustrates the experimental setup used for generating the 49.152 MHz signal.

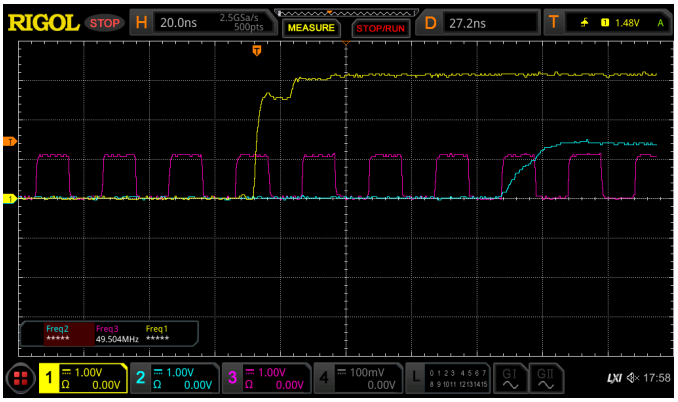

Fig. 11. 49.152 MHz (violet), 1 kHz (cyan) and Grandmaster PPS in sync

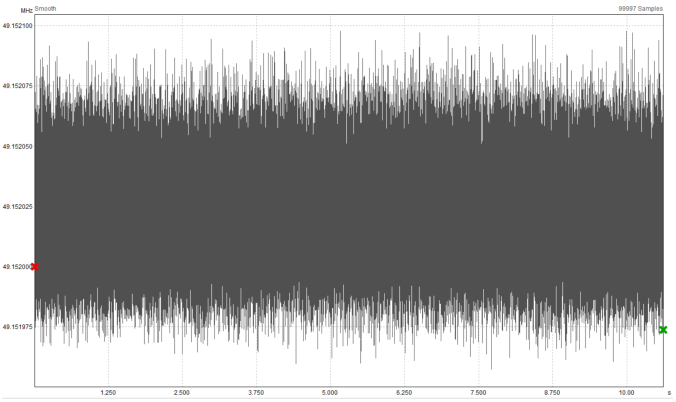

Fig. 12. Verification of the 49.152 MHz output frequency using a spectrum analyzer

VIII. Results

The same experimental setup described in figure 4 was employed to generate the 49.152 MHz media clock. Figure 11 illustrates the phase alignment between three key frequency signals in the system: the yellow signal represents the PPS signal from the PTP Grandmaster, the cyan signal corresponds to the 1 kHz reference signal generated by the AM62x's CPTS module, and the violet signal shows the 49.152 MHz media clock output from the LMK05318B jitter cleaner. Although the LMK05318B is configured to generate an output frequency of 49.152,MHz, this exact value is not accurately displayed on the oscilloscope. Instead, the measured frequency appears to deviate slightly, which may initially suggest an inaccuracy in the clock generation. However, this discrepancy is not due to the LMK05318B itself, but rather a limitation of the oscilloscope's timebase and resolution, especially at high frequencies and when using standard acquisition modes. Oscilloscopes are generally not optimized for precise frequency measurements over extended periods, particularly in the presence of minimal jitter and stable clock sources.

To achieve a more precise verification of the output frequency, a spectrum analyzer was employed. The spectral analysis results confirm that the LMK05318B generates a frequency that closely matches the nominal value of 49.152 MHz, as illustrated in Figure 12.

Figure 12 presents the measured frequency based on one million samples collected over a 10-second interval. The results indicate that the output frequency varies within a narrow range of 49.152000 MHz (red x) to 49.151973 MHz (green x) along the y-axis, demonstrating excellent precision and stability.

All three signals are phase-synchronized. The measured phase offset between the 1 kHz and 49.152 MHz signals is approximately 4 ns, MHz to demonstrycle alignment and tight phase synchronization.

Furthermore, the phase shift between the PPS signal from the PTP Grandmaster and the 1 kHz signal is approximately 70 ns, which corresponds to the phase offset shown in Figure 8, where a 1 Hz signal is generated instead of 1 kHz.

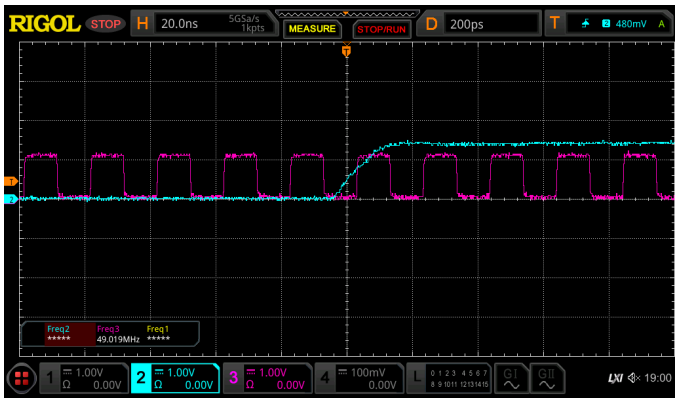

Fig. 13. 1 kHz and 49.152 MHz phase offset (4 ns)

This results in a total phase offset of approximately 74 ns between the PTP Grandmaster and the 49.152 MHz output frequency of the LMK05318B, which is well below the recommended AES11 phase offset of 1 µs for 48 kHz sample rate. This means the target frequency is AES11 compatible. The rise time of the 49.152 MHz frequency is approximately 265 ps, as shown in figure 14. For jitter analysis, the oscilloscope's built-in jitter analysis function was used. Figures 15

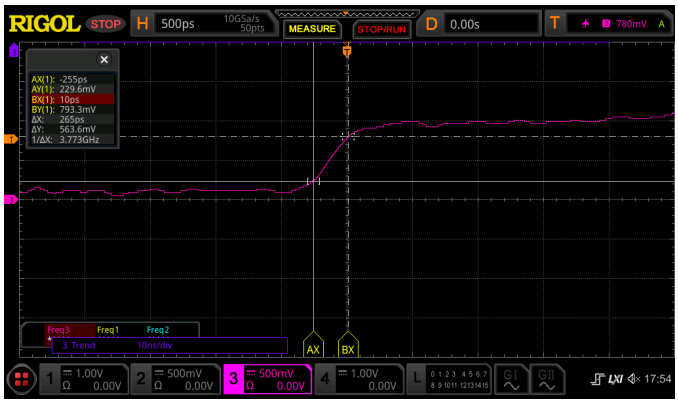

Fig. 14. Rise time of the 49.152 MHz

and 16 show the jitter analysis of the 49.152 MHz signal. In

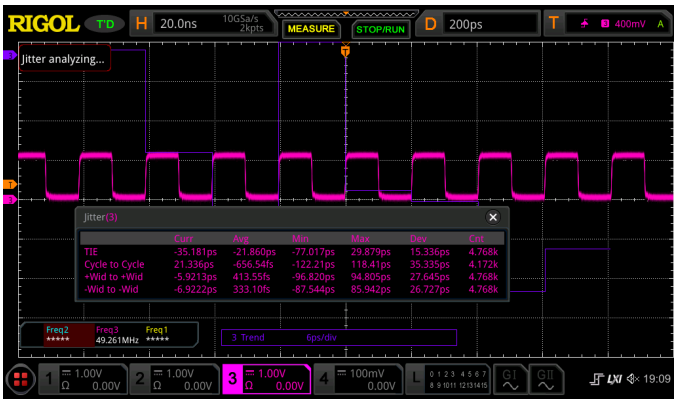

Fig. 15. The jitter analysis of the 49.152 MHz

Jitter(3)	Curr	Avg	Min	Max	Dev	Cnt
TIE	-35.181ps	-21.860ps	-77.017ps	29.879ps	15.336ps	4.768k
Cycle to Cycle	21.336ps	-656.54fs	-122.21ps	118.41ps	35.335ps	4.172k
+Wid to +Wid	-5.9213ps	413.55fs	-96.820ps	94.805ps	27.645ps	4.768k
-Wid to -Wid	-6.9222ps	333.10fs	-87.544ps	85.942ps	26.727ps	4.768k

Fig. 16. Jitter tabelle of 49.152 MHz from the oscilloscope

the second row of the Cycle-to-Cycle table in figure 16, the root mean square (RMS) jitter (denoted as Dev in the table, representing the standard deviation), is the relevant metric for evaluation. It amounts to 35 ps, based on more than 4,000 samples (Cnt).

Although this RMS jitter exceeds the value of 125 fs specified in the datasheet of the LMK05318B [9, p. 14], it remains significantly below the AES11 jitter threshold of 1 µs. Consequently, the output frequency of the LMK05318B at 49.152 MHz is well-suited for AoIP systems compliant with the AES67 standard.

The RMS jitter of the 1 kHz signal is illustrated in Figures 17 and 18. It is approximately 3.2 ns, clearly demonstrating that the LMK05318B effectively attenuates the jitter of the 1 kHz from 3.2 ns to 35 ps

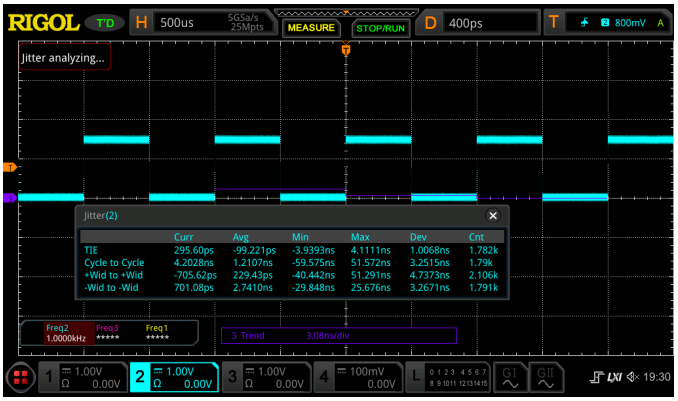

Fig. 17. 1 kHz Jitter value

Jitter(2)	Curr	Avg	Min	Max	Dev	Cnt
TIE	295.60ps	-99.221ps	-3.9393ns	4.1111ns	1.0068ns	1.782k
Cycle to Cycle	4.2028ns	1.2107ns	-59.575ns	51.572ns	3.2515ns	1.79k
+Wid to +Wid	-705.62ps	229.43ps	-40.442ns	51.291ns	4.7373ns	2.106k
-Wid to -Wid	701.08ps	2.7410ns	-29.848ns	25.676ns	3.2671ns	1.791k

Fig. 18. Jitter tabelle of the 1 kHz from the oscilloscope

IX. Summary

This work demonstrates a method for generating an AES11-compliant media clock on the TI AM62x SoC for real-time AoIP streaming applications. While the proposed methodology supports arbitrary media clock frequencies, this paper presents the approach using a 49.152 MHz clock as a representative example. In this implementation, the media clock is derived from an external audio PLL—in this case, the LMK05318B. Although the AM62x lacks dedicated hardware for audio-grade clock generation, its PTP-capab CPTS module can produce a 1 kHz reference signal (also PPS). Custom driver modifications of the CPTS allow derivation of this 1 kHz reference from the synchronized Physical Hardware Clock (PHC), significantly reducing the lock time of the LMK05318B jitter cleaner from 30 minutes (1 Hz input) to under one second.

The LMK05318B then upconverts the 1 kHz reference to 49.152 MHz with a cumulative phase offset of only 74 ns, well within AoIP latency constraints. Measurements confirm a RMS jitter of 35 ps, demonstrating tight phase coherence and suitability for AES11 and professional AoIP applications compliant with AES67. This approach provides a flexible, reproducible pipeline for high-quality media clock generation on general-purpose embedded platforms

Acknowledgment

The authors would like to thank the team at h7r for providing background information on their AES67-based audio-streaming technology, as described on their website https://h7r.net

References

[1] A. Hildebrand. “AES67-2013: AES standard for audio applications of networks - High-performance streaming audio-over-IP interoperability”. In: *NAB Broadcast Engineering Conference* (2014).

[2] A. N. GmbH. “RAVENNA & AES67”. In: *White Paper* (2014).

[3] N. Audio. “AES3, AES/EBU”. In: *NTi Audio Applicaiton Note* (2012).

[4] I. S. 1588-2008. *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. 2008.

[5] A. N. GmbH. “RAVENNA, Operating Principles”. In: *Draft 1.0* (2011).

[6] AES11-2003. “AES recommended practice for digital audio engineering - Synchronization of digital audio equipment in studio operations (Revision of AES11-1997)”. In: *Audio Engineering Society* (2003).

[7] A. Hildebrand. “Comparison of Clocking Synchronization in RTP, RAVENNA/AES67 ST2110, AVB and IPMX”. In: (2024). Accessed: 14.09.2025. URL: https://aimsalliance.org/wp-content/uploads/2022/11/Clocking-Synchronization-in-RTP-RAVENNA_AES67-ST2110-AVB-and-IPMX-Andreas-Hildebrand.pdf.

[8] T. I. Incorporated. *AM62x Processors Silicon Revision 1.0 Texas Instruments Families of Products*. 2023.

[9] T. I. Incorporated. “LMK05318B Ultra-Low Jitter Clock Generator”. In: *SNAS801B* (2020).

[10] T. Kron et al. “AES67 based microphone array for sound source localization”. In: *INTER-NOISE and NOISE-CON Congress and Conference Proceedings* 270.5 (2024), pp. 6015–6024. DOI: 10.3397/IN_2024_3674. URL: https://www.researchgate.net/publication/384862900_AES67_based_microphone_array_for_sound_source_localization.

[11] A. E. S. S. C. (AESSC). “Call for comment on DRAFT REVISED AES standard for audio applications of networks - High-performance streaming audio-over-IP interoperability”. In: (2018.02.14). Accessed: 14.09.2025.

[12] H. Zheng. “eAVB Media Clock Synchronization Using CDCE6214-Q1 Clock Generator”. In: *SNAA333* (2020). Accessed: 2025-09-13. URL: https://www.ti.com/lit/an/snaa333/snaa333.pdf?ts=1757502051200&ref_url=https%253A%252F%252Fwww.google.com%252F.

[13] V. Ltd. *VAR-SOM-AM62_V1.x Datasheet*. Version Rev. 1.10. Accessed: 1.07.2025. 2025. URL: https://www.variscite.de/wp-content/uploads/2023/01/VAR-SOM-AM62_Datasheet.pdf.

[14] V. Ltd. “Symphony-Board Rev 1.x Datasheet”. In: *Symphony-Board Datasheet* (2025). Accessed: 20.06.2025. URL: https://www.variscite.de/wp-content/uploads/2019/07/Symphony-Board_Datasheet.pdf.

[15] G. Strashko. “[PATCH net-next 2/5] net: ethernet: ti: am65-cpts: add pps support”. In: (2023). Accessed: 2025-09-14. URL: https://lists.infradead.org/pipermail/linux-arm-kernel/2023-January/802065.html.