Application Note

Sitara™AM62Dx Benchmarks

TEXAS INSTRUMENTS

Qutaiba Saleh, Barath Ramesh, Jonthan Bishop and Andrew Shutzberg

ABSTRACT

This application note presents a series of benchmarks measuring performance of various components of AM62Dx family of devices. Some of the standard benchmarks are included in the FREERTOS-SDK while the others can be downloaded from the respective hosting websites. Instructions on how to execute the tests and analysis of the results are also included.

Table of Contents

List of Figures

List of Tables

Trademarks

1 Introduction

AM62Dx is a System-on-Chip (SOC) which contains up to four Arm®-Cortex®-A53 cores with 64-bit architecture and a C7x DSP (C7504) with Matrix Multiplication Accelerator (MMA). The DSP consists of scalar and vector compute units with 256-bit fixed and floating point vectors. The C7504 provides up to 40GFLOPS of vector compute while the MMA provides up to 2TOPS. The AM62D also contains a Cortex-R5F MCU core, a Cortex-R5F Device Management core, and extensive peripheral and networking options such as 3-port Gigabit Ethernet switch with Time-Sensitive Networking (TSN). This option can be used to enable audio networking features such as, Ethernet Audio Video Bridging (AVB) and Dante, while peripherals like the McASP, enable multi-channel I2S and TDM Audio inputs and outputs. AM62D supports LPDDR4 memory with 32-bit width at 3733MT/s. Figure 1-1 is a functional block diagram for AM62Dx. For details, see *AM62Dx Sitara Processors Data Sheet*.

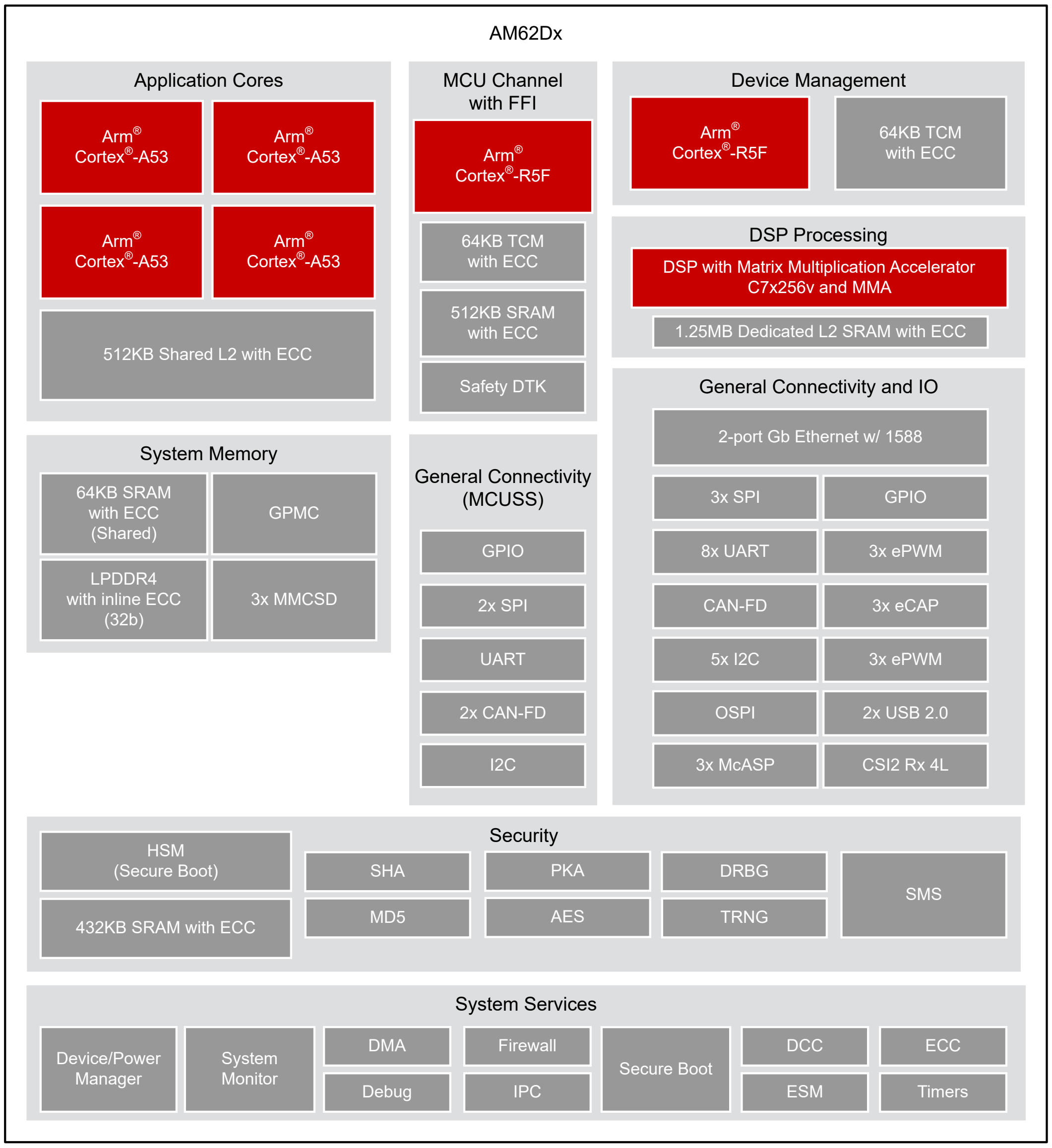

Figure 1-1. AM62Dx Functional Block Diagram

This document presents a number of industry standard and application specific benchmarks measured on the AM62Dx processor. The tests focus on the performance of the Arm-Cortex-A53 cores, the C7x DSP and the LPDDR4 memory. The key core clock and memory speed parameters are explained in each test. The tests in this document are executed on either FreeRTOS or bare-metal. Most of these benchmarks are already included in the FREERTOS-SDK for AM62D.

1.1 Load Binaries to AM62D

The documentation of FREERTOS-SDK for AM62D contains steps to setup the software and tools needed to load and run bare-metal and FreeRTOS™ binaries on the AM62D EVM. Listed below is the summary of the steps. For more details, refer to the FREERTOS-SDK documentation.

1. Download and install FREERTOS-SDK.
2. Download and install all required software and tools:
    a. SysConfig
    b. GCC AARCH64 Compiler
    c. C7000-CGT Compiler Toolchain
    d. Python3
    e. OpenSSL
    f. Mono Runtime
    g. Code Composer Studio™ (CCS)
    h. TI CLANG Compiler Toolchain
3. Build the SDK with make -sj4 all
4. EVM Setup:
    a. Setup up EVM to UART boot mode.
    b. Connect EVM to PC using the UART micro USB port.
    c. Use Unilflash to flash the initialization binaries. For running C7x kernels from libraries such as FFTLIB, MATHLIB, DSPLIB, and VXLIB, use SBL null.
5. Load .out binaries to EVM using CCS (this method is used to run bare-metal kernels on C7x DSP):
    a. Set up XDS110 target configuration for AM62D. This step is executed for the first time only.
    b. Change EVM to OSPI NOR boot mode.
    c. Connect EVM to PC using the JTAG micro USB port.
    d. Launch the XDS110 configuration created in previous step.
    e. Connect to the C7XSS_0.
    f. Load the .out binary file to C7x.
    g. Run.
6. For the freeRTOS example as in the Dhrystone, the .appimage file is flashed to the device.

2 Processor Core Benchmarks

This section contains benchmarks of the processing cores in AM62D including C7x DSP and Arm Cortex processor core. Synthetic benchmarks included are for example Dhrystone.

2.1 C7x DSP Benchmark

This section describes the benchmarks of the C7x DSP using kernels which are critical in the DSP algorithms. The C7x libs, such as FFTLIB and DSPLIB, are used for the benchmarks. These libraries contain optimized functions for most common used operations, such as finite impulse filter and dot product. These libraries are included in the FREERTOS-SDK of AM62D, distributed in source form. Instructions for building and running the modules are included in the documentation folder of each library.

The performance results presented for C7x DSP are all executed on bare-metal with warm L2 cache. Refer to Section 1.1 for information about loading and running bare-metal binaries on AM62D EVM. The benchmarks in the following subsections are organized based on the C7x libs containing them.

2.1.1 Fast Fourier Transform

Fast Fourier Transform (FFT) is on of the most common signal processing algorithms. The FFTLIB library contains a number of kernels for FFT with various batch size and data types. These kernels are implemented only utilizing the only the C7x. The kernels do not utilize the MMA, yet. The FFTLIB is included in the FREERTOS-SDK for AM62D. The documentation of the library contains instructions to build and run these kernels. Table 2-1 shows performance data of complex input/complex output FFT for 16-bit complex integer and 32-bit float complex data types for various vectors and batch sizes executed on C7x. The performance is measured in terms of the total number of C7x clock cycles to perform FFT and the duration to process each batch when C7x clocked at 1.0GHz.

Table 2-1. FFT Performance on C7x

FFT Parameters		Performance for 16-bit int complex input and complex output [Cycles]		Performance for 32-bit float complex input and complex output [Cycles]	
FFT Size	Batch Size	Total [Cycles]	Time per batch [μs]	Total [Cycles]	Time per batch [μs]
128	128	12806	0.1	24221	0.18
256	64	13944	0.21	24850	0.38
512	32	13793	0.43	26826	0.83
1024	16	16012	1.0	29783	1.86
2048	8	15981	1.99	31650	3.95
4096	4	17112	4.27	34834	8.70
8192	2	17063	8.53	36304	18.15

To show performance gain when using C7x DSP over ARM, the execution time of FFT on C7x DSP is compared with execution time of the same type of FFT on Arm-Cortex-A53 core. The benchmark on Arm uses the implementation from Ne10 library, which leverages the Advanced SIMD or NEON acceleration of Cortex-A53. This library is not included in the SDK but can be downloaded from the official Ne10 repository. This library requires a Linux® OS, so this was implemented on AM62A, which has the same Cortex-A53 cores as AM62D. Table 2-2 shows a 1024-point single precision floating point complex FFT execution time on Arm-Cortex-A53 and C7x DSP. The results show approximately 11x speedup between the Cortex-A53 and C7x DSP.

Table 2-2. C7x DSP vs Arm CPU executing Complex FFT

	Arm-Cortex-A53 at 1.4GHz single thread / core	C7x at 1.0GHz	C7x (1.0GHz) to A53 (1.4GHz) Improvement
1024-point Complex FFT Execution Time	19.4μs	1.8μs	Approx. 11x

2.1.2 Digital Signal Processing

The DSPLIB library contains several low-level digital signal processing functions which are implemented on the C7x DSP. This library is included in the FREERTOS-SDK of AM62D. Instructions to reproduce and run these modules are included in the documentation folder of the library. The following subsections shows performance results for some of the frequently used functions in the digital signal processing domain.

2.1.2.1 FIR

Finite Impulse Response (FIR) is one of the most used filters in audio applications. The DSPLIB contains FIR implementation which can take various data and filter sizes. Table 2-3 shows performance results of executing FIR filter executed on C7x DSP on AM62D.

Table 2-3. FIR Filter Performance on C7x DSP

Data Type	Data Size	Output Size	Filter Size	EVM Cycles	Taps/Cycle
Float	2048	1025	1024	74483	14.09
Float	1151	1024	128	8803	14.89
Float	1087	1024	64	4707	13.92

2.1.2.2 Cascade Biquad

The DSPLIB_cascadeBiquad implements multichannel multistage cascade on input data. Table 2-4 shows performance results of implementing 32 channels cascade biquad on float input vectors. The table also shows performance improvments compared to older C66x DSP.

Table 2-4. Cascade Biquad Performance on C7x DSP

Data Type	Data Size	Num Channels	Num Stages	EVM Cycles	Cycles/Biquad	C66x to C7x Improvement
Float	512	32	3	14772	0.3	4.57x
Float	128	32	7	8494	0.3	7.46x

2.1.2.3 Dot Product

The DSPLIB_dotprod kernel calculates the dot products of two vectors. The kernel supports data types including int8, uint8, int16, uint16, int32, uint32, float and double. Table 2-5 shows performance results of executing the dot product kernel on C7x DSP on AM62D for various data types. While the kernel supports any input vector length, results of only 1024 vectors are reported in this document. Performance of other sizes is listed in DSPLIB's user guide.

Table 2-5. Dot Product Performance on C7x DSP

Data Type	Data Size	EVM Cycles	Cycle/Sample
Int8	1024	131	0.13
Int16	1024	132	0.13
Int16	16384	1112	0.07
Int	1024	199	0.19
Float	1024	229	0.22
Float	32768	4211	0.13
Double	1024	352	0.34

2.1.3 Mathematical Operations

The MATHLIB library contains several low-level functions for mathematical operations such as trigonometric, power and exponent which are implemented on the C7x DSP. This library is included in the FREERTOS-SDK of AM62D. Instructions to reproduce and run these modules are included in the documentation folder of the library. Table 2-6 shows performance results for some of the frequently used mathematical functions in MATHLIB library.

Table 2-6. Mathematical Operations Performance on C7x DSP

Function	Data Type	Data Size	EVM Cycles	Cycle/Sample
Sine	Float	1024	1161	1.133
Cos	Float	1024	1418	1.384
Tan	Float	1024	4752	4.640
Division	Float	1024	353	0.344
Square Root	Float	1024	611	0.596

2.2 Dhrystone on A53 cores

Dhrystone benchmark focuses on the processor core performance and runs from warm L1 caches in all modern processors. The benchmark scales linearly with clock speed. Even though the benchmark was introduced in 1984 by Reinhold P. Weicker, Dhrystone still gets used in embedded processing. The industry has adopted the VAX 11/780 as the reference 1 MIPS machine. The VAX 11/780 achieves 1757 Dhrystones per second. The score calculated by normalizing the time the benchmark loop takes to run by the reference 1 MIPS machine score of 1757. This is common to further normalize to DMIPS, MHz, core as the score scales linearly with clock speed. For standard Arm cores, the DMIPS, MHz is identical to the same compiler and flags. Dhrystone is a single core benchmark, a simple sum of multiple cores running the benchmark in parallel is sometimes used.

The Dhrystone (Version 2.1, C Language) benchmark is included in the FREERTOS-SDK as a freeRTOS project. The project is at <freertos-sdk root>/examples/kernel/freertos/dhrystone_benchmark. This project is executed on one of the Arm-Cortex-A53 cores. Due to the short execution time, TI suggests to run the test for high number of iterations to measure accurate results. By default, the example runs the test for 30 million iterations. The code is modified to run for 100 million iterations. Results showed that the results for 30 million and 100 million are comparable. The code block below shows a short version of the terminal printout for Dhrystone benchmark execution.

```
Image loading done, switching to application ...
Starting RTOS/Baremetal applications

[DHRYSTONE BENCHMARKING] Iterations                     : 100000000
[DHRYSTONE BENCHMARKING] Threads                        : 1
[DHRYSTONE BENCHMARKING] Dhrystones per second          : 7602786.5

[DHRYSTONE BENCHMARKING] Iterations                     : 100000000
[DHRYSTONE BENCHMARKING] Threads                        : 2
[DHRYSTONE BENCHMARKING] Dhrystones per second          : 7665291.5

[DHRYSTONE BENCHMARKING] Iterations                     : 100000000
[DHRYSTONE BENCHMARKING] Threads                        : 5
[DHRYSTONE BENCHMARKING] Dhrystones per second          : 7652734.0

[DHRYSTONE BENCHMARKING] Iterations                     : 100000000
[DHRYSTONE BENCHMARKING] Threads                        : 10
[DHRYSTONE BENCHMARKING] Dhrystones per second          : 7665313.

All tests have passed!!
```

Table 2-7 shows the results for this benchmark. The aggregate scores for AM62Ax with four A53 cores running at 1.4GHz is 17,308 DMIPS.

Table 2-7. Dhrystone Benchmarks

	Arm-Cortex-A53 (1.4GHz)
Dhrystones/s	7,602,786.5
Normalized Dhrystones (divide by 1757 reference for 1MIPS)	4,327
DMIPS/MHz each core	Approx. 3
Operating system	freeRTOS

3 Memory System Benchmarks

This section contains benchmarks focusing on the performance of the various memory systems of the System-on-Chip (SoC) including LPDDR4, OCSRAM and SRAM when accessed by the various processing cores such as Arm-Cortex A53, Arm-Cortex-R5F and C7x DSP.

3.1 Critical Memory Access Latency

This section provides the read memory-access latency from processors in AM62Dx to various memory destinations in the system. The measurements were made on the AM62Dx platform using bare-metal silicon verification tests that are not currently included in the SDK. The tests executed on A53, C7x and R5F processor out of the LPDDR4. Each test includes a loop of 8192 iterations to read a total of 32KB of data. The number of cycles for each test were counted and divided by the respective processor clock frequency to obtain latency time. Table 3-1 shows the average latency results. Note that "Local Path" and "Extrefers to accessing

Table 3-1. Critical Memory Access Latency of A53, C7x, R5F MCU, and R5F WKUP

Memory	Arm-Cortex-A53 [Avg ns]	C7x DSP [Avg ns]	Arm-Cortex-R5F MCU [Avg ns]	Arm-Cortex-R5F WKUP [Avg ns]
LPDDR4	137	154	202	172
OCSRAM MAIN	59	57	122	77
OCSRAM MCU	120	118	58	85
OCSRAM WKUP	210	189	203	156
C7X SRAM - Local Path (C7X accessing internal memory)	NA	20	NA	NA
C7X SRAM - External Path (C7X's internal memory accessed by an external core)	80	NA	151	103
R5F MCU TCM - Local Path (R5F MCU accessing internal memory)	NA	NA	1	NA
R5F MCU TCM - External Path (R5F MCU internal memory accessed by an external core)	143	144	NA	120
R5F WKUP TCM - Local Path (R5F WKUP accessing internal memory)	NA	NA	NA	1
R5F WKUP TCM - External Path (R5F WKUP internal memory accessed by other cores)	112	108	120	NA

Tests were done at 0.75V VDD_CORE settings (A53 : 1.25GHz, C7x DSP: 1.0GHz and R5: 800MHz) and LPDDR4 at 3200MT/s.

3.2 UDMA: DDR to DDR Data Copy

This section provides test results and observations for DDR to DDR block copy, using the Normal Capacity (NC) UDMA channel, detailed in Table 3-2.

Table 3-2. UDMA Channel Class

	Description
Normal Capacity (NC)	Provides baseline amount of descriptor and TR prefetch and Tx/Rx control and data buffering. An excellent choice for most peripheral transfers which are communicating with on-chip memories and DDR. With a buffer size of 192B, this FIFO depth allows for 3 read transactions, of 64B data bursts, per flight.

The following measurements are collected using bare-metal silicon verification tests on A53 executing out of DDR. Transfer descriptors and rings in DDR. Tests were done at 0.75V VDD_CORE, 1.25Ghz A53 cores, and 3200MT/s LPDDR4. Transfer sizes range from 1KB to 512KB.

The transfer capacity and latency of the NC UDMA channel, for buffer sizes up to 512 KB, is shown in Table 3-3.

Table 3-3. UDMA: DDR to DDR Block Copy

Buffer Size [KB]	NC Channel Bandwidth [MB/s]	NC Channel Latency [µs]
1	108.99	8.96
2	182.88	10.68
4	262.69	14.87
8	341.90	22.85
16	403.02	38.77
32	448.35	69.7
64	472.91	132.16
128	485.30	257.57
256	491.92	508.21
512	495.12	1009.86

3.3 C7x DRU Performance: Block Copy with DMA

The Data Routing Unit (DRU) available within the C7x is employed to transfer data between DDR and L2SRAM of the C7x, effectively allowing for DMA. The Texas Instruments Signal Processing (TISP) middle-ware library provides several examples on how to wrap various kernels from DSPLIB and FFTLIB of the C7x with DMA. TISP is included in the FREERTOS-SDK of AM62D with documentation to build and run the examples. The TISP_blockCopy example within TISP provides performance results when moving data between DDR and L2SRAM of the C7x. In the TISP_blockCopy example, we read data from DDR into L2SRAM of the C7x while simultaneously writing data from L2SRAM to DDR. There is a block copy kernel that copies the same data, read into L2SRAM via DRU, from one location in L2SRAM to another location in L2SRAM. The block copy kernel employs the streaming engine (SE) to read data from the L2SRAM. To write the same data read via SE, the kernel employs the write path of the C7x into L2SRAM, while the address offsets for the write are generated via the Streaming Address (SA) generator. Few notes on this example are listed below:

- DDR spec: 3200 MT/s, 32 bits per transaction, which results in a peak theoretical DDR bandwidth of 12.8GB/s (4B x 3200MT/s).
- The DRU transfer property was setup to be 4D.
- One channel to read data from DDR into L2SRAM while another channel to writes data from L2SRAM into DDR, simultaneously. Figure 3-1 shows the details of the TISP_blockCopy example. In the example, we DMA 16 MB of data to and from the DDR, simultaneously. Amounting for a total data movement of 32 MB.
- Note that there is no computation involved.

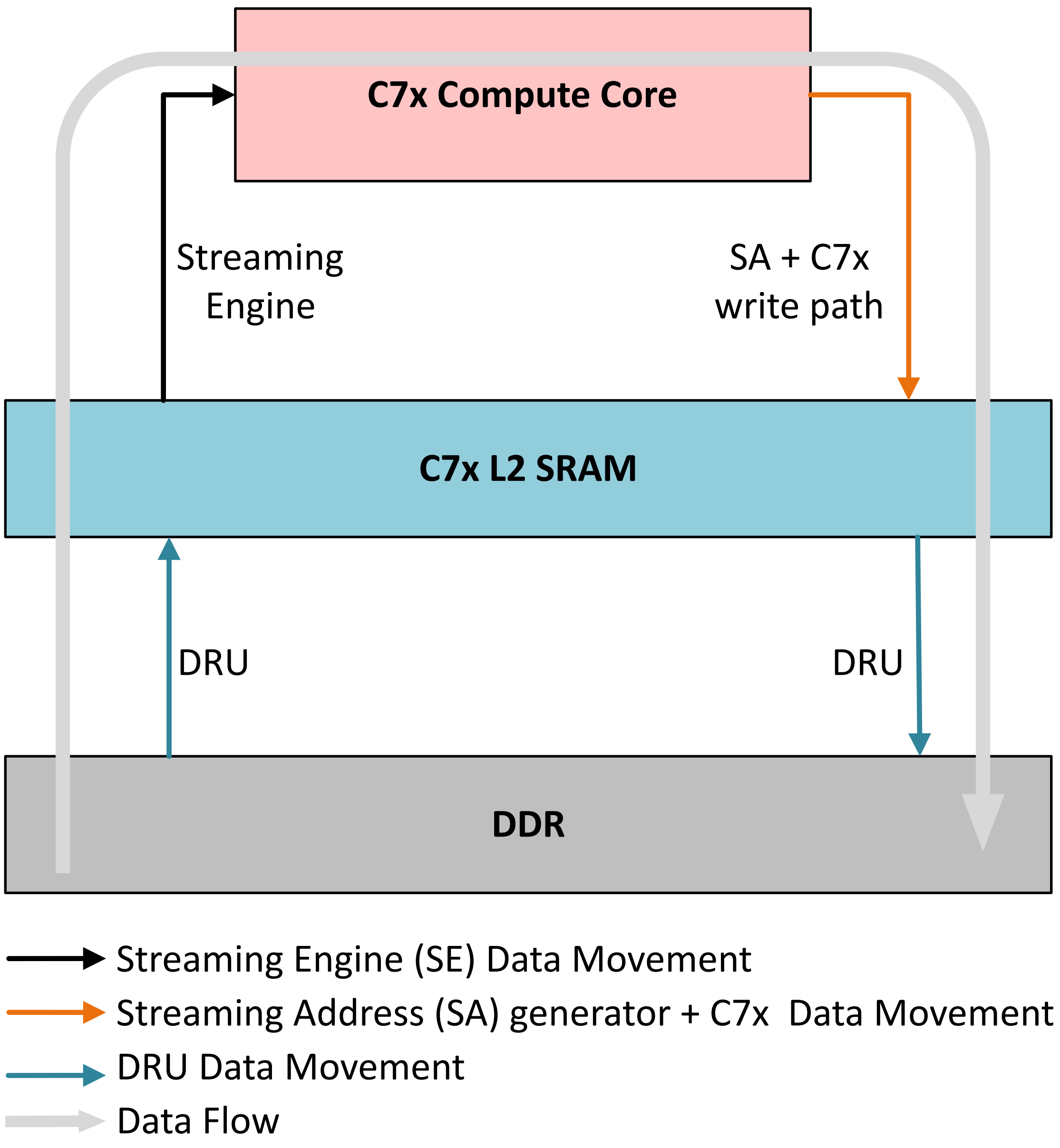

Figure 3-1. DRU, SE, and SA Date Movement Example

Table 3-4 shows performance measurement of moving 16MB of data achieving bandwidth of 10.4GB/s with efficiency of 81% of the total DDR bandwidth.

Table 3-4. DRU Performance: Data Movement from DDR to C7x and Bmoack to DDR

Data Type	Data Size	EVM Cycles	Data Transfer	Efficiency
Float	2048x2048x4=16MB	3185174	5.2x2 = 10.4GB/s	10.4/12.8 = 81%

4 Application Specific Benchmarks

The FREERTOS-SDK for AM62D provides several additional performance benchmarks for various HW IPs and software drivers such as IPC, OSPI, and McASP. The FREERTOS-SDK documentation provides results of these benchmarks. The tables in the following subsections provides performance results for some of the benchmarks. For latest numbers, visit the FREERTOS-SDK page.

4.1 SBL Boot Time

The SBL setup details are shown in Table 4-1.

Table 4-1. SBL Setup Details

Property	Detail
Software or application used	sbl_ospi_multistage, ipc_rpmsg_echo, and HSM App Images
Cores booted by stage1 SBL	r5f0-0
Cores booted by stage2 SBL	hsm-m4f0-0 mcu-r5f0-0 a530-0 c75ss0
Size of image loaded by stage1: r5f0-0	199KB
Size of HSM-M4F image loaded by stage2	7.81KB
Size of MCU-R5F image loaded by stage2	39.06KB
Size of a53 image loaded by stage 2	73.92KB
Size of C7x image loaded by stage 2	144.82KB
Total size of images loaded by stage2	144.82KB

The boot times for stage1 and stage2 using both OSPI and EMMC on HS-FS device are shown in Table 4-1. Note that most of time in Stage1 is attributed to DDR initialization.

Table 4-2. SBL Boot Time

Stage	OSPI at 166.667MHz [ms]	EMMC at 200.0MHz [ms]
Stage1	40.861	63.044
Stage2	33.912	52.372

4.2 IPC Performance

The IPC notify performance is benchmarked by sending 10,000 notifications among the various processing cores while measuring the latency. All cores are running from DDR with the exception of MCU-R5 from MSRAMTable 4-3 shows the average notify latency.

Table 4-3. IPC Message Notify Latency

Local Core	Remote Core	Average Message Latency [ns]
mcu-r5f0-0	c75ss0	2094ns
mcu-r5f0-0	a530-0	1169ns
mcu-r5f0-0	r5f0-0	1689ns
a530-0	c75ss0	2082ns
c75ss0	r5f0-0	2065ns
a530-0	r5f0-0	1000ns

The IPC RPMSG performance is benchmarked by sending 1000 messages between processors while measuring message latency. Table 4-4shows the average and maximum message latency for various message sizes.

Table 4-4. IPC Message Transfer Latency

Local Core	Remote Core	Message Size	Average Message Latency [µs]	Max Latency [µs]
r5f0-0	a530-0	4	6.842	10
r5f0-0	mcu-r5f0-0	4	8.933	12
r5f0-0	c75ss0	4	79.916	94
r5f0-0	a530-0	32	9.659	12
r5f0-0	a530-0	64	12.655	15
r5f0-0	a530-0	112	17.544	21
r5f0-0	mcu-r5f0-0	32	15.526	18
r5f0-0	mcu-r5f0-0	64	22.628	25
r5f0-0	mcu-r5f0-0	112	33.379	36
r5f0-0	c75ss0	32	89.926	104
r5f0-0	c75ss0	64	92.618	127
r5f0-0	c75ss0	112	113.281	128

4.3 Flash

Table 4-5 shows the write and read speed for EMMC and OSPI for various data sizes. The EMMC is setup with HS200 mode while the OSPI is using FLASH_CFG_PROTO_8D_8D_8D protocol with both PHY and DMA enabled.

Table 4-5. EMMC and OSPI Performance

Mode	Data Size [MB]	Write Speed [MBps]	Read Speed [MBps]
EMMC	1	53.48	157.51
EMMC	4	109.84	169.34
EMMC	6	108.03	169.78
OSPI	1	0.43	283.77
OSPI	5	0.43	248.83
OSPI	10	0.43	284.96

4.4 Application Specific Latency

Table 4-6 shows latency for various functionalities which are critical in audio applications.

Table 4-6. Application Specific Latency Performance

Test	Detail	Latency
McASP (Audio)	Operating at 48KHz, I2C mode RX to TX pin to pin	792µs
Boot Latency (MCU R5)	GPIO toggle time from MCUR5 main measured from MCU_PORz	112ms
C7x audio chime time	Measured from MCU_PORz Includes DAC configuration	185ms

5 Summary

This document presents a set of benchmarks for AM62D processor. The tests focuses on processing cores and memory performance with emphasis on the C7x DSP. The document also includes steps to reproduce the results.

6 References

- Texas Instruments, *AM62Dx Sitara Processors*, data sheet
- Ne10 math library, webpage