Application Note

Sitara™ AM62P Benchmarks

Qutaiba Saleh, Jonathan Bishop, Andrew Shutzberg, and Krunal Bhargav

ABSTRACT

This application note presents a series of benchmarks measuring performance of various components of the AM62Px family of devices. Some of the standard benchmarks are included in the Linux SDK while the others can be downloaded from their respective hosting websites. Instructions on how to execute some of the tests and analysis of the results are also included.

Table of Contents

List of Figures

List of Tables

Trademarks

Arm® and Cortex® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
CoreMark® is a registered trademark of Embedded Microprocessor Benchmark Consortium.
All trademarks are the property of their respective owners.

1 Introduction

AM62Px contains up to four Arm®-Cortex®-A53 cores with 64-bit architecture, a Cortex-R5F MCU core, a Cortex-R5F Device Management core, and various other features including: multi-high definition display support, 3D-graphics acceleration, 4K video acceleration, and extensive peripherals make the AM62Px well-suited for a broad range of automotive and industrial applications, including automotive digital instrumentation, automotive displays, industrial HMI, and more. It supports LPDDR4 32-bit width with speed of 3200 MT/s. Figure 1-1 is a functional block diagram for AM62Px. For details, see *AM62Px Sitara Processors Data Sheet*.

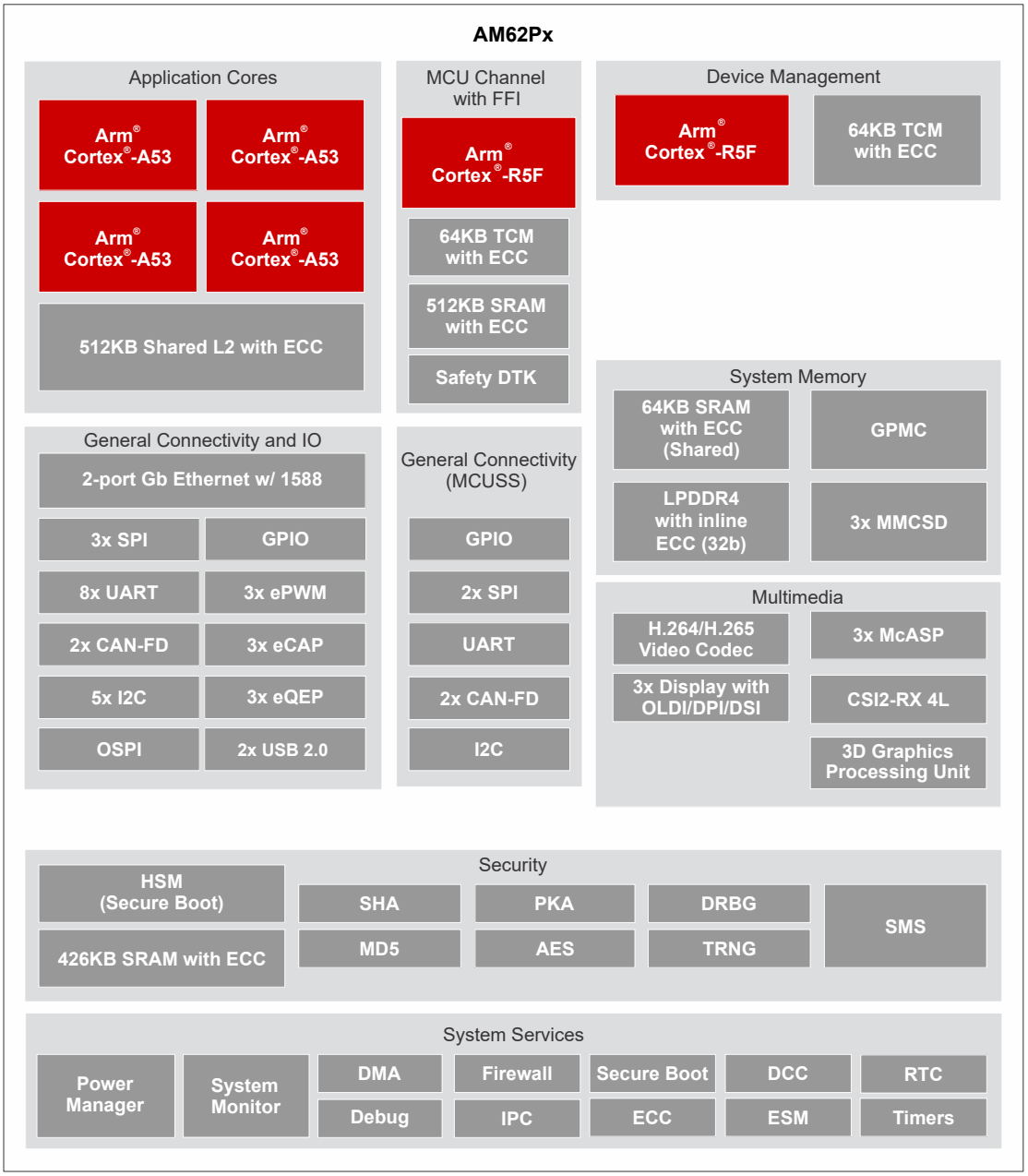

Figure 1-1. AM62Px Functional Block Diagram

This document presents a number of industry standard and application specific benchmarks measured on the AM62Px processor. The tests focus on the performance of the Arm-Cortex-A53 cores and the LPDDR4 memory with some application specific benchmarks for the Arm-Cortex-R5F MCU, GPU , and other components. The key parameters of the evaluation board are 1.25GHz and 1.4GHz clock speed for the Arm-Cortex-A53 cores, and a 32-bit wide LPDDR4 at a speed of 3200MT/s. Most of the standard benchmarks are already included in the SDK and can be directly executed while the other benchmarks can be downloaded from their respective official host websites. All of the benchmarks are implemented using Linux SDK 9.02.

1.1 Change Cortex-A53 Clock Frequency

Maximum frequency of A53 cores on AM62Px varies based on speed grade of the device and the VDD_CORE voltage applied. To obtain peak performance of the device, in some of the benchmarks, the setup was modified to run the A53 cores on AM62Px at 1.4GHz.

2 Processor Core and Compute Benchmarks

This section contains benchmarks contained within an Arm Cortex processor core. Synthetic benchmarks included are for example Dhrystone.

2.1 Dhrystone

Dhrystone benchmark focuses on the processor core performance. It runs from warm L1 caches in all modern processors. It 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 it takes the benchmark loop to run by the reference 1 MIPS machine score of 1757. It 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 SDK. It can be performed by simply running the command *dhrystone*. Due to its short execution time, it is suggested to run the test for high number of iterations in order to measure accurate results. More than 100 million iterations are used in the tests implemented for Arm-Cortex-A53. The code block below shows a short version of the terminal printout for Dhrystone benchmark execution.

```
root@am62pxx-evm:~# dhrystone

Dhrystone Benchmark, Version 2.1 (Language: C)

Program compiled without 'register' attribute

Please give the number of runs through the benchmark: 100000000

Execution starts, 100000000 runs through Dhrystone
Execution ends

Final values of the variables used in the benchmark:
.
.
.

Microseconds for one run through Dhrystone:    0.1
Dhrystones per Second:                      7142857.0
```

Table 2-1 shows the results for this benchmark with the compiler and operating system details. The aggregate scores for AM62Ax with four A53 cores running at 1.25GHz and 1.4GHz are 14,229DMIPS and 16,261DMIPS, respectively.

Table 2-1. Dhrystone Benchmarks

	Arm-Cortex-A53(1.25GHz)	Arm-Cortex-A53 (1.4GHz)
Dhrystones/s	6,250,000	7,142,857
Normalized Dhrystones (divide by 1757 reference for 1MIPS)	3,557	4,065
DMIPS/MHz each core	~3	~3
Compiler and flags	GCC 11.4 -march=ARMv8 -O3	
Operating System	Linux 6.1.80 (2023)	

2.2 CoreMark-Pro

CoreMark®-Pro tests the entire processor, adding comprehensive support for multi-core technology, a combination of integer and floating-point workloads, and data sets for utilizing larger memory subsystems. The components of CoreMark-Pro utilizes all levels of cache with an up to 3MB data memory footprint. Many, but not all of the tests, are also using P threads to allow utilization of multiple cores. The score scales with the number of cores but is always less than linear (dual core score is less than 2x single core).

CoreMark-Pro should not be confused with the smaller CoreMark which, like Dhrystone, is a microbenchmark contained in L1 caches of a modern processor.

CoreMark-Pro is not included in the SDK. It can be downloaded from the official host website. In this tests, the code is directly cloned and built in the AM62Px EVM. Next are the steps to clone, build, and run CoreMark-Pro directly on the target:

1. Clone the repository.

```
root@am62pxx-evm:~# git clone https://github.com/eembc/coremark-pro.git
```

2. Build CoreMark-Pro

```
root@am62pxx-evm:~# cd coremark-pro/
root@am62pxx-evm:~/coremark-pro# make TARGET=linux64 build-all
```

3. Run CoreMark-Pro: use "certify-all" to run all 9 benchmarks of CoreMark-Pro and "XCMD" to set the number of cores.

```
root@am62pxx-evm:~/coremark-pro# make TARGET=linux64 certify-all XCMD='-c4'
```

All official CoreMark-Pro rules have been satisfied such as making sure that the execution time of each workload is at least 1000 times the minimum timer resolution. Table 2-2 shows the CoreMark-Pro results for single, dual, and quad A53 cores at both 1.25GHz and 1.4GHz.

Table 2-2. CoreMark®-Pro Results

	Arm-Cortex-A53 At 1.25 GHz [iter/s]	Parallel Scaling	Arm-Cortex-A53 At 1.4 GHz [iter/s]	Parallel Scaling
Single Core	850	1	936	1
Dual Core	1,531	1.82	1,700	1.82
Quad Core	2,426	2.88	2,654	2.83

2.3 Fast Fourier Transform

Fast Fourier Transform (FFT) is on of the most common signal processing algorithms.Table 2-3 shows a 1024-point single precision floating point complex FFT execution time on Arm-Cortex-A53. The benchmark on Arm-Cortex-A53 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 it can be downloaded from the official Ne10 repository.

Table 2-3. NE10 CFFT Benchmark

	Arm-Cortex-A53 at 1.25 GHz (single thread / core)	Arm-Cortex-A53 at 1.4 GHz (single thread / core)
1024-point Complex FFT Execution Time	35 µs	31 µs

2.4 Cryptographic Benchmarks

The AM62Px Processor SDK Linux includes an openssl cryptographic library which provides optimized implementations of cryptographic functions. It is employed by some applications such as HTTPS, ssh, and netconf implementations. For the highest performance, the higher-level interface provided by the EVP library should be used. Table 2-4 shows a set of selected benchmarks of software observed performance run on AM62Px. Command run was *openssl speed -elapsed -evp <cryptographic mode> -multi 4*. This is utilizing all four A53 cores using four threads. In these tests, the Arm-Cortex-A53 is clocked at 1.4 GHz. The output of the openssl command is in KB/s. To meet desired industry standard, the results reported in the Table 2-4 are transformed to Mb/s.

Table 2-4. Symmetric Cryptography and Secure Hash in Mbit/s

	Frame Size (bytes)					
	16	**64**	**256**	**1024**	**8192**	**16384**
aes-128-gcm	2,035	6,082	12,285	17,755	20,806	21,033
aes-256-gcm	1,945	5,652	10,906	15,271	18,231	18,485
aes-128-ctr	3,205	9,477	19,306	27,385	31,360	31,597
sha256	412	1,531	4,950	11,190	17,774	18,538
sha512	240	956	2,083	3,496	4,365	4,441
chacha20-poly1305	1,306	2,802	5,519	6,876	7,288	7,313

Further benchmarks for public key cryptography are shown in Table 2-5. Tests can be run with command *openssl speed -elapsed -multi 4 <algorithm>* .

Table 2-5. Public Key Cryptography Benchmarks

RSA	size	512	1024	2048	3072	4096
	sign/second	16,359	3,419	520	171	76
	verify/second	202,801	67,920	19,394	8,950	5,124
ECDSA	curve	nistp224	nistp256	nistp521	nistk233	nistb233
	sign/second	1,884	23,010	240	1,467	1,395
	verify/second	2,124	7,859	315	743	707

3 Memory System Benchmarks

This section contains benchmarks involving the Arm-Cortex processor core and the memory system of the System-on-Chip (SoC). Synthetic benchmarks included are, for example, LMBench and CoreMark-Pro. Math function benchmarks include Fast Fourier Transforms (FFT).

3.1 Memory Bandwidth and Latency

A subset of LMBench and STREAM are benchmarks to measure achieved memory bandwidth and latency from software.

3.1.1 LMBench

LMBench is a suite of micro benchmarks for processor cores and operating system primitives. The memory bandwidth and latency related tests are most relevant for modern embedded processors. The results vary a little (< 10%) run to run.

LMBench benchmark *bw_mem* measures achieved memory copy performance. With parameter *cp* it does an array copy and *bcopy* parameter uses the runtime glibc version of *memcpy()* standard function. The glibc uses a highly optimized implementation that utilizes, for example, SIMD resulting in higher performance. The size parameter equal to or smaller than the cache size at a given level measures the achievable memory bandwidth from software doing a typical for loop or *memcpy()* type operation. Typical use is for external memory bandwidth calculation. The bandwidth is calculated as byte read and written counts as 1, which is roughly half of STREAM copy result. The benchmark further allows creating parallel threads with -P parameter. To get the maximum multi-core memory bandwidth, create the same amount of threads as there are cores available for the operating system, which is 4 for AM62Px Linux (-P 4). To show full performance characterization of the AM62Px, the LMBench tests are implemented on full factorial combinations of number of cores and clock frequency. The code block below shows terminal printout of executing the LMBench commands.

```
root@am62pxx-evm:~# bw_mem 8M bcopy
8.00 1956.71
root@am62pxx-evm:~# bw_mem -P 2 8M bcopy
8.00 3122.66
root@am62pxx-evm:~# bw_mem -P 4 8M bcopy
8.00 3605.80

root@am62pxx-evm:~# bw_mem 8M cp
8.00 1012.27
root@am62pxx-evm:~# bw_mem -P 2 8M cp
8.00 1568.78
root@am62pxx-evm:~# bw_mem -P 4 8M cp
8.00 1874.32
```

Table 3-1 shows the measured bandwidth and the efficiency compared to theoretical wire rate. The wire rate used is the LPDDR4 MT/s rate times the width divided by two (read and write making up a copy both consume the bus).

$$Efficiency = \frac{Measured\ Speed}{\frac{LPDDR4\ MT/s \times width}{2}} = \frac{Measured\ Speed}{\frac{3200 \times 4\ B}{2}} = \frac{Measured\ Speed}{6400} \quad (1)$$

Table 3-1. LMBench Results

Command	Description	Arm-Cortex-A53 at 1.25 GHz, LPDDR4-3200MT/s-32 Bit [MB/s]	LPDDR4 Efficiency [%]	Arm-Cortex-A53 at 1.4 GHz, LPDDR4-3200MT/s-32 Bit [MB/s]	LPDDR4 Efficiency [%]
Bw_mem 8M bcopy	*Single core, glibc memcpy*	1,911	30	1,956	31
bw_mem -P 2 8M bcopy	*Dual core, glibc memcpy*	3,052	48	3,122	49
bw_mem -P 4 8M bcopy	*Quad core, glibc memcpy*	3,571	56	3,605	56
Bw_mem 8M cp	*Single core, inline copy loop*	1,003	16	1,012	16
bw_mem -P 2 8M cp	*Dual core, inline copy loop*	1,562	24	1,568	25
bw_mem -P 4 8M cp	*Single core, inline copy loop*	1,862	29	1,874	29

LMBench benchmark *lat_mem_rd* is used to measure the observed memory access latency for external memory (LPDDR4 on AM62Px) and cache hits. The two arguments are the size of the transaction (64 in the code block below) and the stride of the read (512). These two values are selected to measure the latency to caches and external memory, not the processor data prefetchers or other speculative execution. For access patterns, the prefetching will work, but this benchmark is most useful to measure the case when it does not.

The code block below shoes the terminal printout of executing *lat_mem_rd* command. The left column is the size of the data access pattern in megabytes, right column is the round trip read latency in nanoseconds. This command is executed for Arm-Cortex-A53 clock frequency of 1.25GHz and 1.4GHz.

```
root@am62pxx-evm:~# lat_mem_rd 64 512
"stride=512
0.00049 2.145
0.00098 2.145
0.00195 2.145
0.00293 2.145
0.00391 2.145
0.00586 2.145
0.00781 2.145
0.01172 2.145
0.01562 2.145
0.02344 2.289
0.03125 4.179
0.04688 6.494
0.06250 7.748
0.09375 8.626
0.12500 9.283
0.18750 9.764
0.25000 9.976
0.37500 11.156
0.50000 33.705
0.75000 94.007
1.00000 126.437
1.50000 142.957
2.00000 145.089
3.00000 146.336
4.00000 147.029
6.00000 147.626
8.00000 147.867
12.00000 148.112
16.00000 148.169
24.00000 148.243
32.00000 148.219
48.00000 148.284
64.00000 148.291
```

Figure 3-1 shows connected scatter plots of memory latency results for both 1.25GHz and 1.4GHz. Based on memory block size (x-axis), the plot can be divided into three regions. The first region is when the accessed memory block is smaller than L1 cache. It is safe to assume that the data is completely inside the L1 and such the latency in this region is a close estimation of L1 cache latency. The second region is when the accessed memory block is bigger than L1 but smaller than L2 cache. The latency in this region is a mix of L1, L2, and LPDDR4 latency. The latency at the middle of that region can be assumed to be a close representation of L2 latency. The third region is when the access memory block is bigger than L2 cache. The last reading in this region reflects the LPDDR4 latency.

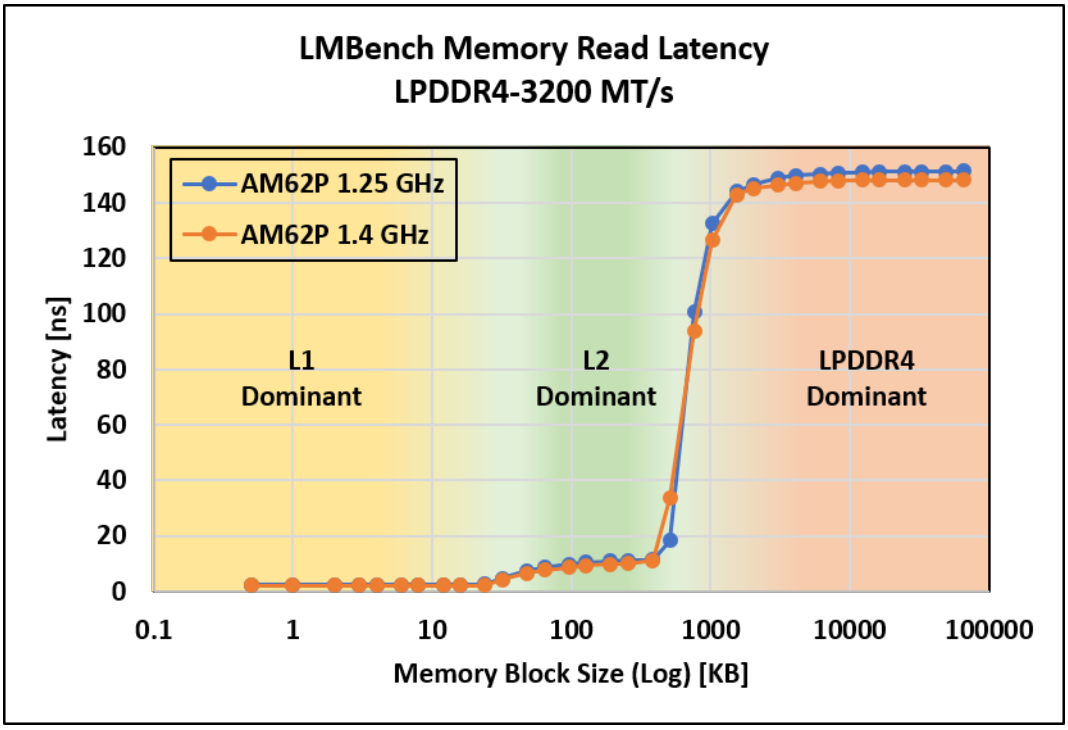

Figure 3-1. Memory Read Latency

Table 3-2 shows a summary for Arm-Cortex-A53 read latency.

Table 3-2. Memory Read Latency Results

Memory	Arm-Cortex-A53 at 1.25 GHz [ns]	Arm-Cortex-A53 at 1.4 GHz [ns]
L1 cache	2.4	2.1
L2 cache	10.4	9.2
LPDDR4-3200 MT/s	151.3	148.2

3.1.2 STREAM

STREAM is a microbenchmark for measuring data memory system performance without any data reuse. It is designed to miss on caches and exercise the data prefetcher and speculative accesses. It uses double precision floating point (64 bit), but in most modern processors the memory access is the bottleneck. The four individual scores are copy, scale as in multiply by constant, add two numbers, and triad for multiply accumulate.

- Copy: Measures memory transfer rate without arithmetic operation, a[i] = b[i]
- Scale: Includes a simple arithmetic operation, a[i] = k × b[i]
- Add: Includes three memory access in addition to arithmetic operation, a[i] = b[i] + c[i]
- Triad: Combines scale and add in one operation, a[i] = b[i] + k × c[i]

For bandwidth, a byte read counts as one and a byte written counts as one resulting in a score that is double the bandwidth LMBench. Table 3-3 shows the measured bandwidth and the efficiency compared to theoretical wire rate. The wire rate used is the LPDDR4 MT/s rate times the width. To get overall maximum achieved throughput the command used is *stream -M 16M -P 4-N 10*, which means four parallel threads and 10 iterations. The Arm-Cortex-A53 clock frequency is setup to 1.4 GHz in this test.

Table 3-3. Stream Benchmarks

	LPDDR4-3200MT/s-32-Bit Bandwidth [MB/s]	LPDDR4-3200MT/s-32-Bit Efficiency [%]
copy	7,316	57
scale	7,274	57
add	6,401	50
triad	6,059	47

3.2 Critical Memory Access Latency

This section provides round-trip read latency measurements for processors in AM62Px to various memory destinations in the system. The measurements where made on the AM62Px platform using bare-metal silicon verification tests. The tests execute on A53 and R5F processor out of LPDDR4. Each test includes a loop of 8192 iterations to read a total of 32 KiB of data. The number of cycles for each access were counted and divided by the respective processor clock frequency to obtain latency time. Table 3-4 shows the average latency results.

Table 3-4. Critical Memory Access Latency of A53, R5F MCU, and R5F WKUP

Memory	Arm-Cortex-A53 (Avg ns)	Arm-Cortex-R5F MCU (Avg ns)	Arm-Cortex-R5F WKUP (Avg ns)
LPDDR4	129	207	173
OCSRAM MAIN	56	120	76
OCSRAM MCU	120	55	80
OCSRAM WKUP	207	193	153
R5F MCU TCM	140	1	180
R5F WKUP TCM	104	111	1

Tests were done at 0.75V VDD_CORE, 1.25Hhz A53 cores, 800MHz R5F cores, and 3200MT/s LPDDR4. Tightly-Coupled Memory, or TCM, is RAM that is directly attached to an ARM Cortex core. ARM architecture provides a local internal low latency path and also allows external access to the memory through SoC bus infrastructure.

3.3 UDMA: DDR to DDR Data Copy

This section provides test results and observations for DDR to DDR block copy, using both the High Capacity (HC) & Normal Capacity (NC) UDMA channels, detailed in Table 3-5.

Table 3-5. UDMA Channel Classes

	Description
Normal Capacity (NC)	Provides baseline amount of descriptor and TR prefetch and Tx/Rx control and data buffering. Suitable 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.
High Capacity (HC)	Provides an elevated amount of descriptor and TR prefetch and custom Tx/Rx control and data buffering. Suitable for applications which require moderate per-channel bandwidth with significantly increased data throughput. With an increased buffer size of 512B, this FIFO depth allows for 8 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, 1.25Hhz A53 cores, 800MHz R5F cores, and 3200MT/s LPDDR4. Transfer sizes range from 1KiB to 512KiB.

Table 3-6. UDMA: DDR to DDR Block Copy

Buffer Size (KiB)	HC Channel Bandwidth (MiB/s)	NC Channel Bandwidth (MiB/s)	HC Channel Latency (µs)	NC Channel Latency (µs)
1	121.92	96.21	8.01	10.15
2	188.16	157.51	10.38	12.40
4	369.56	237.32	10.57	16.46
8	542.16	312.75	14.41	24.98
16	711.20	381.94	21.97	40.91
32	895.93	426.91	34.88	73.20
64	985.03	452.31	63.45	138.18
128	1049.36	464.93	119.12	268.86
256	1087.10	472.64	229.97	528.94
512	1105.71	476.06	452.20	1050.29

Table 3-6 shows the transfer capacity of both HC & NC channels, and demonstrates an up to 2.3-fold increase in bandwidth, obtained by the High Capacity channel over the Normal Capacity channel.

4 Graphics Processing Unit Benchmarks

AM62P is equipped with a GPU with 3D graphics capabilities as shown in Table 4-1.

Table 4-1. AM62P GPU Specifications

Features	Values
GPU Core	IMG BXS-4-64 with 256KB cache
GFLOPS	50
3D API	OpenGL ES 3.2 and Vulkan 1.2

4.1 Glmark2

Glmartk2 is used to benchmark the performance of the GPU on AM62P. The test is included in the SDK. This tests is implemented for DRM and Wayland display types. Table 4-2 shows the commands to run the tests and their results.

Table 4-2. Glmark2 Results

Test	Score
glmark2-es2-drm --off-screen	298
glmark2-es2-wayland --off-screen	744

4.2 GFXBench5

The results shown in Table 4-3 were gathered from the GFXBench5 performance tests.

Table 4-3. GFXBench5 Results

Test	FPS
Manhattan Offscreen	14.72
Trex Offscreen	27.56
Egypt Offscreen	49.15

5 Video Codec

This section introduces benchmarks for the video codec in AM62P. Specifically, the benchmark focuses on using of the H264 encoder and decoder. The results are collected from a close to real use-case scenario which includes transmitting of a live video stream over UDP using gstreamer pipeline. The setup uses a USB camera with resolution of 1920x1080 and rate of 30FPS. The transmitter side captures the live video stream from the camera, encodes it using codec accelerator with H264 format and transmits the video over UDP. The other side receives the video stream over UDP, decodes and display it on the screen. Both the transmitter and receiver side are executed on the same AM62P device. This shows the codec capability of executing both the encoder and decoder at the same time. The latency of each component in the pipeline can be measured using the gstreamer tracer that outputs live logs of the measurements to a file that you specify.

Following is the gstreamer pipeline for the encoder sider and UDP transmitter side with the gstreamer tracer configured to log the latency measurement at "/run/trace_encode.log".

```
GST_TRACERS="latency(flags=pipeline+element)" GST_DEBUG=GST_TRACER:7 GST_DEBUG_FILE="/run/
trace_encode.log" \
gst-launch-1.0 \
v4l2src device=/dev/video2 ! image/jpeg, width=1920, height=1080, framerate=30/1 ! jpegdec !
videoconvert ! v4l2h264enc ! h264parse ! rtph264pay ! udpsink host=$1 port=5000 sync=false
```

Following is the gstreamer pipeline for the decoder, UDP receiver and display side with the gstreamer tracer configured to log the latency measurement at "/run/trace_decode.log".

```
GST_TRACERS="latency(flags=element+pipeline)" GST_DEBUG=GST_TRACER:7
GST_DEBUG_FILE=/run/trace_decode.log \
gst-launch-1.0 -v \
udpsrc port=5000 ! 'application/x-rtp, encoding-name=H264, payload=96' !
rtpjitterbuffer latency=50 ! rtph264depay ! h264parse ! v4l2h264dec !
queue ! kmssink driver-name=tidss sync=false plane-id=31
```

The python script provided at /opt/edgai-gst-apps/seripts/gst_tracers/parse_gst_tracers.py can be used to calculate the average of the latency measurements recorded in the .log files for each component in pipeline. This script can be executed in parallel with the gstreamer pipeline to show live updates of the latency measurements. For instance this is the printout for the decoder side.

```
root@am62pxx-evm:/opt/edgeai-gst-apps/scripts/gst_tracers/# parse_gst_tracers.py /run/
trace_decode.log
.
.
+-------------------------------------------------------------------------------------+
|element            latency        out-latancy   out-fps     frames |
+-------------------------------------------------------------------------------------+
|capsfilter0         0.16              27.54       36        30938  |
|rtpjitterbuffer0    0.36              27.53       36        30938  |
|rtph264depay0       0.20              35.86       27        23752  |
|h264parse0          0.26              35.86       27        23752  |
|v4l2h264dec0        48.12             35.86       27        23751  |
|udpsrc0             49.73             35.86       27        23751  |
|queue0              0.75              35.86       27        23751  |
+-------------------------------------------------------------------------------------+
```

The average latency measurements for the encoder and decoder are shown inVideo Codec Latency.

Table 5-1. Video Codec Latency

Codec H264	Resolution	Latency [ms]
Encoder	1920x1080	10.58
Decoder	1920x1080	48.12

6 References

- CoreMark-Pro
- STREAM McCalpin, John D. "STREAM: Sustainable Memory Bandwidth in High Performance Computers", a continually updated technical report (1991-2007), available at: http://www.cs.virginia.edu/stream/
- Ne10 math library
- OpenSSL
- Texas Instruments: *AM62Px Sitara Processors Data Sheet*.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265