Application Note

Sitara™ AM62Lx Benchmarks

Divyansh Mittal, Andrew Shutzberg, Qutaiba Saleh

ABSTRACT

This application note presents a series of benchmarks measuring performance of various components of the AM62Lx family of devices. Some of the standard benchmarks are included in the Linux SDK while the others can be downloaded from the respective hosting websites. Instructions on how to execute some of the tests and analysis of the results are also included.

Table of Contents

1 Introduction 2
2 Processor Core and Compute Benchmarks 3
2.1 Dhrystone 3
2.2 Whetstone 4
2.3 Linpack 4
2.4 NBench 5
2.5 CoreMark-Pro 6
2.6 Fast Fourier Transform 7
2.7 Cryptographic Benchmarks 7
3 Memory System Benchmarks 8
3.1 Memory Bandwidth and Latency 8
3.1.1 LMBench 8
3.1.2 STREAM 11
3.2 Critical Memory Access Latency 11
3.3 UDMA: DDR to DDR Data Copy 12
4 References 13

Trademarks

1 Introduction

AM62Lx is a System-on-Chip (SoC) containing dual Arm®-Cortex®-A53 cores with 64-bit architecture. The key parameters of the evaluation board are 1.25GHz clock speed on the Arm®-Cortex®-A53 cores and 16-bit wide LPDDR4 with speed of 1600 MT/s. It is equipped with several embedded features such as Multimedia DSI/DPI support, integrated ADC on chip, advanced lower power management modes, and extensive security options for IP protection. The device includes several peripherals which enable system-level connectivity such as USB, MMC/SD, OSPI, CAN-FD and an ADC. Figure 1-1 is a functional block diagram for AM62Lx. For details, see AM62Lx Sitara Processors Data Sheet.

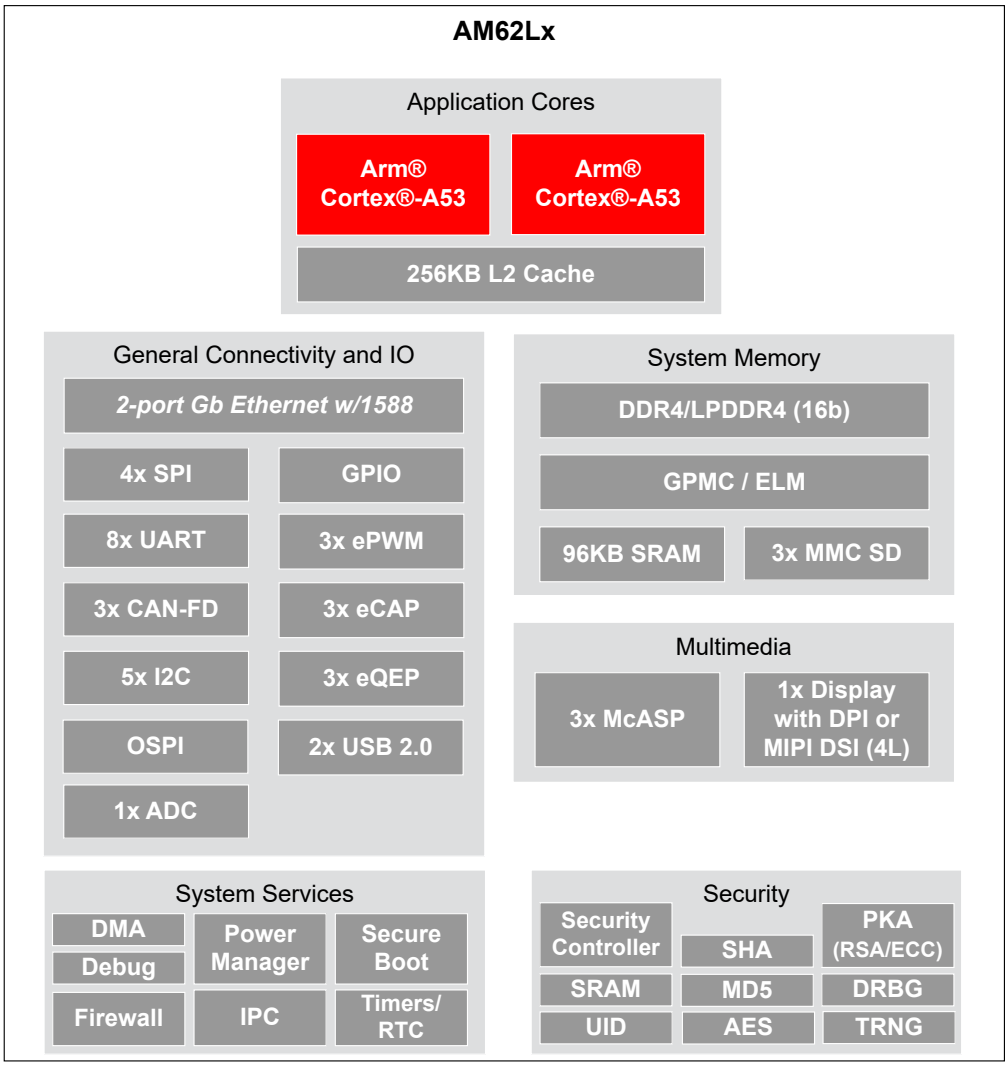

Figure 1-1. Functional Block Diagram

This document presents a number of industry standard and application specific benchmarks measured on the AM62Lx processor. The tests focus on the performance of the Arm®-Cortex®-A53 cores and the LPDDR4 memory with some application specific benchmarks. Most of the standard benchmarks are already included in the SDK and can be directly executed while the other benchmarks can be downloaded from the respective official host websites.

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. The benchmark 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 is calculated by normalizing the time the benchmark loop runs by the reference 1 MIPS machine score of 1757. A common issue is 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 and can be performed by simply running the command *dhrystone*. Due to the short execution time, TI recommends to run the test for high number of iterations to measure accurate results. 1 billion 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@am62lxx-evm:~# dhrystone

Dhrystone Benchmark, Version 2.1 (Language: C)

Program compiled without 'register' attribute

Please give the number of runs through the benchmark: 1000000000

Execution starts, 1000000000 runs through Dhrystone
Execution ends

Final values of the variables used in the benchmark:
.
.
.
Microseconds for one run through Dhrystone:    0.2
Dhrystones per Second:                      6410256.5
```

Table 2-1 shows the results for this benchmark with the compiler and operating system details. The aggregate score for AM62Lx with two A53 cores running at 1.25GHz is 7,296 DMIPS.

Table 2-1. Dhrystone Benchmarks

	Arm-Cortex-A53(1.25GHz)
Dhrystones/s	6,410,256
Normalized dhrystones (divide by 1757 reference for 1MIPS)	3648
DMIPS/MHz each core	~3
Compiler and flags	GCC 13.3.0 -march=ARMv8 -O3
Operating system	Linux 6.12.0

2.2 Whetstone

Whetstone is a benchmark primarily measuring floating-point arithmetic performance.

```
root@am62lxx-evm:~# runWhetstone



Whetstone running ...

Execution time approx 10 seconds

Loops: 100000, Iterations: 1, Duration: 2 sec.
C Converted Double Precision Whetstones: 5000.0 MIPS
```

Table 2-2 shows the results for this benchmark.

Table 2-2. Whetstone Benchmarks

	Arm-Cortex-A53(1.25GHz)
Whetstone (MIPS)	5,000

2.3 Linpack

Linpack measures peak double precision (64 bit) floating point performance in solving a dense linear system.

```
root@am62lxx-evm:~# runLinpack



Linpack running ...

Execution time approx 10 seconds
Unrolled Single Precision Linpack

Unrolled Single Precision Linpack

     norm. resid      resid           machep         x[0]-1        x[n-1]-1
       2.0        4.69621336e-05  1.19209290e-07 -1.31130219e-05 -1.30534172e-05
    times are reported for matrices of order   100
      dgefa      dgesl      total       kflops     unit      ratio
 times for array with leading dimension of  201
       0.00       0.00       0.00     506018       0.00       0.02
       0.00       0.00       0.00     507890       0.00       0.02
       0.00       0.00       0.00     511293       0.00       0.02
       0.00       0.00       0.00     511331       0.00       0.02
 times for array with leading dimension of 200
       0.00       0.00       0.00     559630       0.00       0.02
       0.00       0.00       0.00     563766       0.00       0.02
       0.00       0.00       0.00     563304       0.00       0.02
       0.00       0.00       0.00     564925       0.00       0.02
Unrolled Single  Precision 511331 Kflops ; 10 Reps
```

Table 2-3 shows the results for this benchmark.

Table 2-3. Linpack Benchmarks

	Arm-Cortex-A53(1.25GHz)
Linpack (Kflops)	511331

2.4 NBench

NBench which stands for Native Benchmark is used to measure macro benchmarks for commonly used operations such as sorting and analysis algorithms. More information about NBench at https://en.wikipedia.org/wiki/NBench and https://nbench.io/articles/index.html

```
root@am62lxx-evm:~# cd /usr/bin
root@am62lxx-evm:/usr/bin# nbench

BYTEmark* Native Mode Benchmark ver. 2 (10/95)
Index-split by Andrew D. Balsa (11/97)
Linux/Unix* port by Uwe F. Mayer (12/96,11/97)

TEST                : Iterations/sec.  : Old Index   : New Index
                    :                  : Pentium 90* : AMD K6/233*
--------------------:------------------:-------------:------------
NUMERIC SORT        :          560.92  :      14.39  :       4.72
STRING SORT         :          146.35  :      65.39  :      10.12
BITFIELD            :      1.7834e+08  :      30.59  :       6.39
FP EMULATION        :          192.44  :      92.34  :      21.31
FOURIER             :           20381  :      23.18  :      13.02
ASSIGNMENT          :          12.971  :      49.36  :      12.80
IDEA                :          3075.1  :      47.03  :      13.96
HUFFMAN             :          1057.3  :      29.32  :       9.36
NEURAL NET          :           7.741  :      12.44  :       5.23
LU DECOMPOSITION    :          472.71  :      24.49  :      17.68
==========================ORIGINAL BYTEMARK RESULTS==========================
INTEGER INDEX       : 40.569
FLOATING-POINT INDEX: 19.182
Baseline (MSDOS*)   : Pentium* 90, 256 KB L2-cache, Watcom* compiler 10.0
=============================LINUX DATA BELOW===============================
CPU                 : Dual
L2 Cache            :
OS                  : Linux 6.12.0-ti-g1a4dee21b1eb-dirty
C compiler          : aarch64-oe-linux-gcc
libc                : static
MEMORY INDEX        : 9.390
INTEGER INDEX       : 10.711
FLOATING-POINT INDEX: 10.639
Baseline (LINUX)    : AMD K6/233*, 512 KB L2-cache, gcc 2.7.2.3, libc-5.4.38
* Trademarks are property of their respective holder.
```

Table 2-4 shows the results for this benchmark.

Table 2-4. NBench Benchmarks

Benchmarks	Arm-Cortex-A53(1.25GHz)
assignment (Iterations)	12.97
fourier (Iterations)	20381
fp_emulation (Iterations)	192.44
huffman (Iterations)	1057.3
idea (Iterations)	3075.1
lu_decomposition (Iterations)	472.71
neural_net (Iterations)	7.74
numeric_sort (Iterations)	560.92
string_sort (Iterations)	146.35

2.5 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 must 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 and can be downloaded from the official host website. In this tests, the code is directly cloned and built in the AM62Lx EVM. Next are the steps to clone, build, and run CoreMark-Pro directly on the target:

1. Clone the repository.

```
root@am62lxx-evm:~# git clone https://github.com/eembc/coremark-pro.git
```

2. Build CoreMark-Pro

```
root@am62lxx-evm:~# cd coremark-pro/
root@am62lxx-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@am62lxx-evm:~/coremark-pro# make TARGET=linux64 certify-all XCMD='-c2'
```

Benchmark output:

```
root@am62lxx-evm:~/coremark-pro# make TARGET=linux64 certify-all XCMD='-c2'
.
.
WORKLOAD RESULTS TABLE

                                              MultiCore SingleCore
Workload Name                                  (iter/s)   (iter/s)    Scaling
---------------------------------------------- ---------- ---------- ----------
cjpeg-rose7-preset                                71.43      37.04       1.93
core                                               0.52       0.27       1.93
linear_alg-mid-100x100-sp                         24.58      12.92       1.90
loops-all-mid-10k-sp                               0.72       0.43       1.67
nnet_test                                          1.96       1.01       1.94
parser-125k                                        6.85       7.09       0.97
radix2-big-64k                                    32.47      22.28       1.46
sha-test                                         138.89      73.53       1.89
zip-test                                          33.90      20.00       1.69

MARK RESULTS TABLE

Mark Name                                      MultiCore SingleCore    Scaling
---------------------------------------------- ---------- ---------- ----------
CoreMark-PRO                                     1189.42     710.32       1.67
```

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 2-5 shows the CoreMark-Pro results for single, and dual A53 cores at 1.25GHz.

Table 2-5. CoreMark®-Pro Results

	Arm-Cortex-A53 At 1.25GHz [iter/s]	Parallel Scaling
Single Core	710	1
Dual Core	1,189	1.67

2.6 Fast Fourier Transform

Fast Fourier Transform (FFT) is on of the most common signal processing algorithms. Table 2-6 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 can be downloaded from the official Ne10 repository.

Table 2-6. NE10 CFFT Benchmark

	Arm-Cortex-A53 at 1.25GHz (single thread / core)
1024-point Complex FFT Execution Time	37µs

2.7 Cryptographic Benchmarks

The AM62Lx Processor SDK Linux includes an openssl cryptographic library which provides optimized implementations of cryptographic functions. The AM62Lx Processor SDK Linux 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 must be used. Table 2-7 shows a set of selected benchmarks of software observed performance run on AM62Lx. Command run was *openssl speed -elapsed -evp <cryptographic mode> -multi 2*. This is utilizing both A53 cores using two threads. In these tests, the Arm-Cortex-A53 is clocked at 1.25GHz. The output of the openssl command is in KB/s. To meet desired industry standard, the results reported in the Table 2-7 are transformed to Mb/s.

Table 2-7. Symmetric Cryptography and Secure Hash in Mbit/s

	Frame Size (bytes)					
	16	**64**	**256**	**1024**	**8192**	**16384**
aes-128-gcm	893	2,673	5,379	7,776	9,090	9,212
aes-256-gcm	848	2,470	4,760	6,895	8,001	8,105
aes-128-ctr	1,279	3,860	8,178	11,782	13,742	13,800
sha256	176	651	2,124	4,860	7,772	8,091
sha512	98	392	881	1,494	1,905	1,942
chacha20-poly1305	571	1,245	2,413	3,014	3,192	3,212

Further benchmarks for public key cryptography are shown in Table 2-8. Tests can be run with command *openssl speed -elapsed -multi 2<algorithm>* .

Table 2-8. Public Key Cryptography Benchmarks

RSA	**size**	**512**	**1024**	**2048**	**3072**	**4096**
	sign/second	7,357	1,511	230	74	33
	verify/second	86,983	29,394	8,515	3,888	2,232
	encr./s	70,260	26,811	8,091	3,810	2,198
	decr./s	6,248	1,450	226	74	33
ECDSA	**curve**	**nistp224**	**nistp256**	**nistp521**	**nistk233**	**nistb233**
	sign/second	802	8,218	103	638	607
	verify/second	921	3,344	137	326	311

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.

3.1 Memory Bandwidth and Latency

A subset of memory system benchmarks are LMBench and STREAM used 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*, the benchmark 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 2 for AM62Lx Linux (-P 2). To show full performance characterization of the AM62Lx, 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@am62lxx-evm:~# bw_mem 8M bcopy
8.00 1000.00
root@am62lxx-evm:~# bw_mem -P 2 8M bcopy
8.00 1127.87
root@am62lxx-evm:~# bw_mem 8M cp
8.00 579.54
root@am62lxx-evm:~# bw_mem -P 2 8M cp
8.00 645.71
```

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).

$$Efficiency = \frac{Measured\ Speed}{\frac{LPDDR4\ MT/s \times width}{2}} = \frac{Measured\ Speed}{\frac{1600 \times 2\ B}{2}} = \frac{Measured\ Speed}{1600} \quad (1)$$

Table 3-1. LMBench Results

Command	Description	Arm-Cortex-A53 at 1.25GHz, LPDDR4-1600MT/s-16 Bit [MB/s]	LPDDR4 Efficiency [%]
Bw_mem 8M bcopy	*Single core, glibc memcpy*	1,000	62
bw_mem -P 2 8M bcopy	*Dual core, glibc memcpy*	1,127	70
Bw_mem 8M cp	*Single core, inline copy loop*	579	36
bw_mem -P 2 8M cp	*Dual core, inline copy loop*	645	40

LMBench benchmark *lat_mem_rd* is used to measure the observed memory access latency for external memory (LPDDR4 on AM62Lx) 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 works, but this benchmark is most useful to measure the case when the prefetching 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.

```
root@am62lxx-evm:~# lat_mem_rd 64 512
"stride=512
0.00049 2.405
0.00098 2.404
0.00195 2.404
0.00293 2.405
0.00391 2.404
0.00586 2.404
0.00781 2.405
0.01172 2.404
0.01562 2.405
0.02344 2.525
0.03125 5.903
0.04688 7.172
0.06250 8.453
0.09375 9.674
0.12500 10.239
0.18750 10.838
0.25000 30.115
0.37500 106.020
0.50000 155.871
0.75000 178.770
1.00000 181.088
1.50000 182.258
2.00000 182.664
3.00000 182.958
4.00000 182.947
6.00000 183.280
8.00000 183.235
12.00000 183.265
16.00000 183.375
24.00000 183.149
32.00000 183.076
48.00000 183.513
64.00000 183.483
```

Figure 3-1 shows connected scatter plots of memory latency results for 1.25GHz. 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. 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.

Memory Read Latency

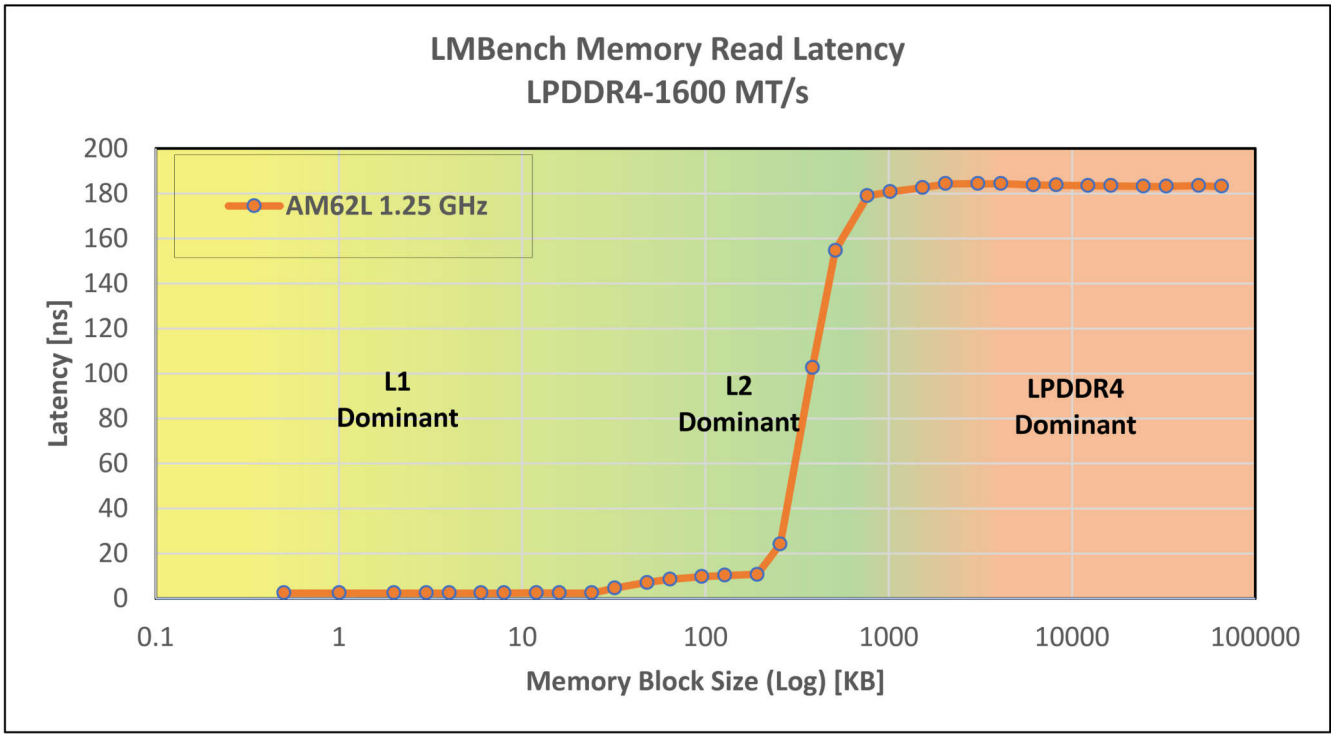

Figure 3-1. Memory Read Latency

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

Table 3-2. Memory Read Latency Results

Memory	Arm-Cortex-A53 at 1.25GHz [ns]
L1 cache	2.4
L2 cache	10.2
LPDDR4-1600 MT/s	183.5

Latency: lat_mem_rd-stride128-szN, where N is equal to or smaller than the cache size at given level measures the cache miss penalty. N that is at least double the size of last level cache is the latency to external memory.

Bandwidth: bw_mem_bcopy-N, where N is equal to or smaller than the cache size at a given level measures the achievable memory bandwidth from software doing a memcpy() type operation. Typical use is for external memory bandwidth calculation. The bandwidth is calculated as byte read and written counts as 1 which should be roughly half of STREAM copy result.

3.1.2 STREAM

STREAM is a microbenchmark for measuring data memory system performance without any data reuse. STREAM is designed to miss on caches and exercise the data prefetcher and speculative accesses. STREAM 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 2 -N 10*, which means two parallel threads and 10 iterations. The Arm-Cortex-A53 clock frequency is setup to 1.25GHz in this test.

```
root@am62lxx-evm:~# stream -M 16M -P 2 -N 10
STREAM copy latency: 13.64 nanoseconds
STREAM copy bandwidth: 2346.27 MB/sec
STREAM scale latency: 13.59 nanoseconds
STREAM scale bandwidth: 2354.55 MB/sec
STREAM add latency: 21.72 nanoseconds
STREAM add bandwidth: 2209.49 MB/sec
STREAM triad latency: 22.20 nanoseconds
STREAM triad bandwidth: 2162.58 MB/sec
```

Table 3-3. Stream Benchmarks

	LPDDR4-1600MT/s-16-Bit Latency [ns]	LPDDR4-1600MT/s-16-Bit Bandwidth [MB/s]	LPDDR4-1600MT/s-16-Bit Efficiency[%]
copy	13.64	2,346	73
scale	13.59	2,354	73
add	21.72	2,209	69
triad	22.20	2,162	67

3.2 Critical Memory Access Latency

This section provides round-trip read latency measurements for processors in AM62Lx to various memory destinations in the system. The measurements where made on the AM62Lx platform using bare-metal silicon verification tests. The tests execute on A53 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

Memory	Arm-Cortex-A53 (Avg ns)	SoC Address
LPDDR4	166	0x80000000
OCSRAM MAIN	49	0x70800000
OCSRAM WKUP	123	0x707f0000

Tests were done at 0.75V VDD_CORE, 1.25Ghz A53 cores and 1600MT/s LPDDR4. 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 the Normal Capacity (NC) UDMA channel, detailed in Table 3-5.

Table 3-5. 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 1600MT/s LPDDR4. Transfer sizes range from 1KiB to 512KiB.

The transfer capacity and latency of the NC UDMA channel, for buffer sizes up to 512 KiB, are shown in Table 3-6.

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

Buffer Size (KiB)	NC Channel Bandwidth (MiB/s)	NC Channel Latency (µs)
1	204.73	4.77
2	283.47	6.89
4	349.40	11.18
8	387.91	20.14
16	420.48	37.16
32	436.33	71.62
64	444.49	140.61
128	448.77	278.54
256	450.82	554.54
512	452.10	1105.96

4 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: AM62Lx Sitara Procesors *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