Application Report

Sitara™ AM64x Benchmarks

Pekka Varis

ABSTRACT

This document contains benchmarks for the AM64x family of devices.

Table of Contents

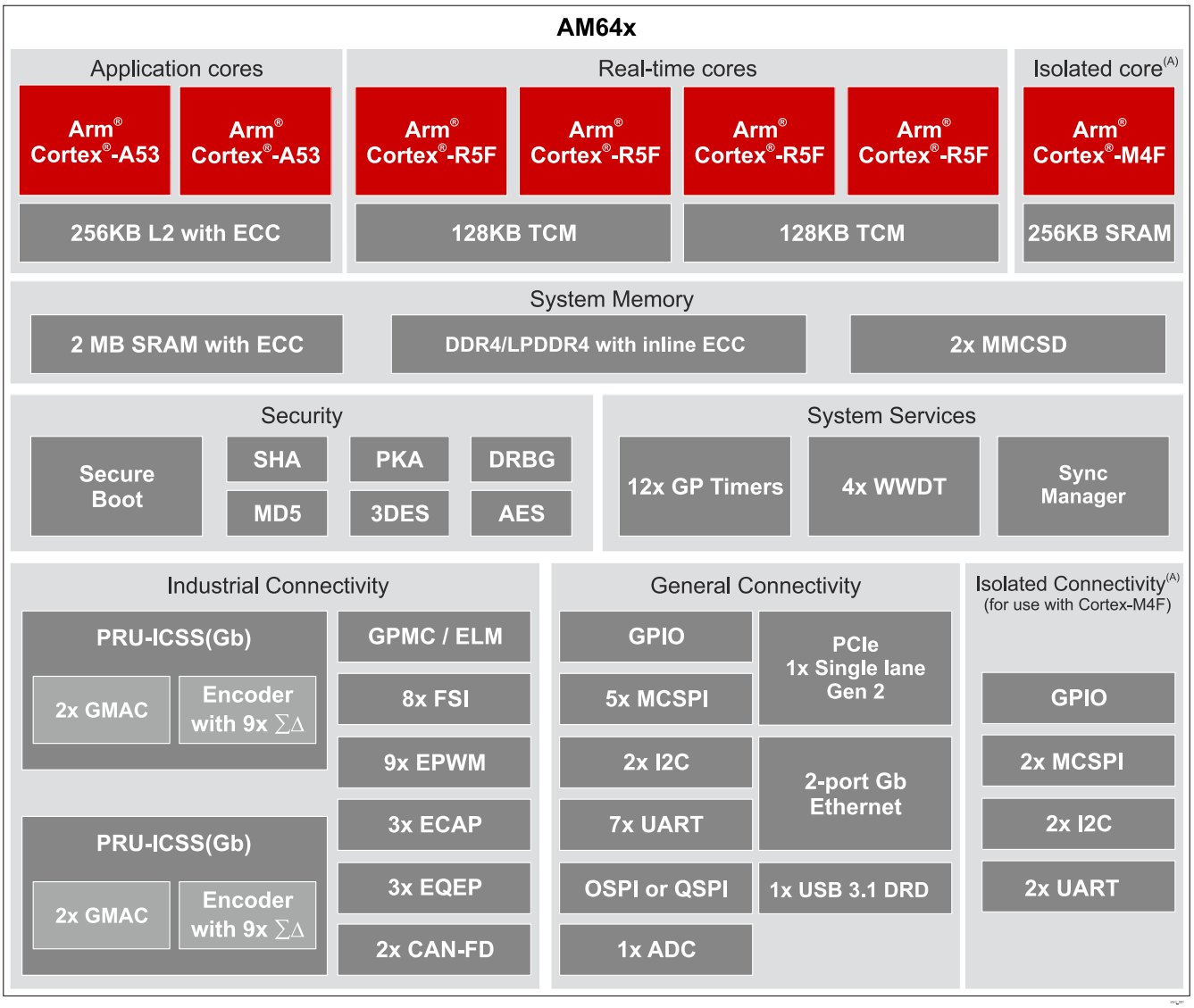
List of Figures

List of Tables

Trademarks

1 Introduction

Some of the benchmarks are measured on the Cortex®-A53 cores, and some are on Cortex-R5F cores. For updated benchmarks, the Processor SDK for AM64x Sitara™ Processors contains a set of benchmarks in the Performance Guide section and the Benchmark Demo application. The benchmarks have been run on the AM64x EVM. The key parameters of the evaluation board are 1 GHz clock speed for the Cortex-A53 cores, 800 MHz for the Cortex-R5F cores, and a 16-bit wide DDR4 at a speed of 1333MT/s.



A. Isolation of peripherals and M4F core is an optional feature. MCU domain resources are shared across SoC when in non-isolated configuration.

Figure 1-1. Functional Block Diagram

2 Processor Core Benchmarks

This section contains benchmarks contained within a Arm® Cortex processor core. Synthetic benchmarks included are for example Dhrystone. Math function benchmarks include several functions such as trigonometric functions as defined in math.h.

2.1 Dhrystone

Dhrystone is a core only benchmark that runs from warm L1 caches in all modern processors. It scales linearly with clock speed. The score calculated by normalizing the time it takes the benchmark loop to run by the reference 1 MIP machine score of 1757. Even though the benchmark was introduced in 1984 by Reinhold P. Weicker, Dhrystone still gets used in embedded processing. 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 will be identical with 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 aggregate score for AM6442 with two A53 cores at 1 GHz (6000 DMIPS) and four R5F cores at 800 MHz (6400 DMIPS) is 12400DMIPS.

	Cortex-A53 (1 GHz)	Cortex-R5F (800 MHz)
Dhrystones	5263158	2,962,962
Normalized Dhrystones (divide by 1757 reference for 1MIP)	2996	1686
DMIPS/MHz each core	3	2.1
Compiler and flags	GCC 9.2 -march=ARMv8 -O3	Arm Compiler 6.14 -mcpu=cortex-r5 -O3
Operating System	Linux 5.4 (2020 LTS)	Bare metal

2.2 Trigonometric Functions

Trigonometric functions are compute functions leveraged in for example motor control. The most commonly used ones are sine, cosine, arctan and arctan2. The standard C library *math.h* has highly accurate implementations of these functions, but the worst case execution time might not be acceptable. For Arm Cortex-R5F, the Arm CMSIS DSP library also has optimized implementations for some trigonometric functions. Sitara™ software development kit contains further optimized functions for some of the functions in a library called TI_R5FMATHLIB. Table 2-1 and Table 2-2 show the performance of the most commonly used trigonometric functions using the implementations found in C runtime library (math.h), CMSIS, and SitaraR5F library. The implementations trade off accuracy, shown in column max error, and compute time and sometimes additional memory consumption (a lookup table with constants in memory) as shown in column labeled Table size. This table needs to be in TCM or warm L1D cache for the documented performance.

Table 2-1. Sine and Cosine on Arm Cortex-R5F (32-bit float)

Function	Max Cycles	Avg Cycles	Max Error (abs rad)	Table Size	Library
ti_r5fmath_sin()	34	34	7.20E-07	polynomial	TI_R5FMATHLIB
ti_r5fmath_cos()	38	38	2.90E-07	polynomial	TI_R5FMATHLIB
ti_r5fmath _sincos()	51	51	7.20E-07	polynomial	TI_R5FMATHLIB
ti_r5fmath_fast_sincosB()	57	57	1.90E-07	polynomial	TI_R5FMATHLIB
arm_cos_f32()	66	66	1.80E-05	Table 2kbytes	CMSIS
sinf()	71	71	8.40E-08	polynomial	math.h
cosf()	81	81	9.70E-08	polynomial	math.h
arm_sin_cos_f32()	114	114	6.10E-07	Table 2kbytes	CMSIS
sin() (double)	340	296	3.00E-08	polynomial	math.h

Table 2-2. Arctan and Arctan2 on Arm Cortex-R5F (32-bit float)

Function	Max cycles	Avg Cycles	Max Error (abs rad)	Table Size	Library
ti_r5fmath_atanFast()	45	39	3.76E-03	function	TI_R5FMATHLIB
ti_r5fmath_atan2Fast()	54	54	3.76E-03	function	TI_R5FMATHLIB
ti_r5fmath_atan()	80	68	6.00E-07	poly	TI_R5FMATHLIB
atanf()	111	90	6.80E-08	poly	math.h
ti_r5fmath_atan2()	97	90	6.00E-07	poly	TI_R5FMATHLIB
atan2f()	204	171	2.00E-07	poly	math.h

3 Compute and Memory System Benchmarks

This section contains benchmarks involving the Arm Cortex processor core and the memory system of the SoC. Synthetic benchmarks included are for example LMbench and CoreMark-Pro. Math function benchmarks include functions such linear algebra and fast fourier transforms (FFT).

3.1 CoreMark®-Pro

CoreMark-Pro tests the entire processor, adding comprehensive support for multicore 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 also are using pthreads 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.

	Arm Cortex-A53
Single core	604
Dual core	1063 (1.66 times single core)

3.2 Fast Fourier Transform

Fast Fourier Transform (FFT) is a multiply accumulate heavy building block in many applications. Below table shows a 1024-point single precision floating point complex FFT execution time. The Arm Cortex-A53 benchmark uses the implementation from Ne10 library which leverages the Advanced SIMD or NEON acceleration of Cortex A53.

	1024pt float CFFT execution time (single thread / core)
Arm Cortex-A53 (1GHz)	27 microseconds
Arm Cortex-R5 (800MHz)	134 microseconds

3.3 Memory Bandwidth and Latency

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

3.4 LMBench

LMBench is a suite of microbenchmarks for processor cores and operating system primitivites. The memory bandwidth and latency related tests are most relevant for modern embedded processors.

LMBench benchmark *bw_mem* with parameter *cp* measures achieved memory copy 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 *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. The table below shows the measured bandwidth and the efficiency compared to theoretical wire rate. The wire rate used is the DDR MT/s rate times the width divided by two (read and write making up a copy both consume the bus).

	bw_mem (Blench)	Efficiency
Arm Cortex-A53, DDR4-1333MT/s-16bit	596MB/s	45%

LMBench benchmark *lat_mem_rd* is used to measure observed memory access latency for external memory (DDR4/LPDDR4 on AM64x) and cache hits. The two arguments are size of of the transaction (64 in the screenshot 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 some access patterns the prefetcing will work, but this benchmark is most useful to measure the case when it does not. The left column is the size of the data access pattern in megabytes, right column is the round trip read latency in nanoseconds. As a summary for Arm Cortex-A53 latency to L1D is 3 ns, L2 latency is 14 ns, and access to DDR4 is 196 ns.

```
root@am6x-evm:~# lat_mem_rd 64 512
"stride=512
0.00049 3.006
0.00098 3.006
0.00195 3.006
0.00293 3.006
0.00391 3.006
0.00586 3.006
0.00781 3.006
0.01172 3.006
0.01562 3.006
0.02344 3.009
0.03125 3.120
0.04688 9.212
0.06250 10.677
0.09375 12.269
0.12500 12.984
0.18750 13.651
0.25000 14.066
0.37500 115.226
0.50000 168.747
0.75000 189.919
1.00000 192.138
1.50000 193.431
2.00000 194.175
3.00000 194.870
4.00000 195.202
6.00000 195.463
8.00000 195.622
12.00000 195.700
16.00000 195.761
24.00000 195.876
32.00000 195.938
48.00000 196.001
64.00000 196.006
```

3.5 STREAM

STREAM is a microbenchmark for measuring data memory system performance without any data reuse. It is designed to miss on caches and exercise data prefetcher and speculative accesses. it uses double precision floating point (64bit) but in most modern processors the memory access will be the bottleneck. The four individual scores are copy, scale as in multiply by constant, add two numbers, and triad for multiply accumulate. 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 will show. The table below shows the measured bandwidth and the efficiency compared to theoretical wire rate. The wire rate used is the DDR MT/s rate times the width.

	Bandwidth	Efficiency
copy, DDR4-1333MT/s-16bit	1138MB/s	43%
add, DDR4-1333MT/s-16bit	1030MB/s	39%
triad, DDR4-1333MT/s-16bit	1036MB/s	39%

4 Application Benchmarks

This section contains application level benchmarks such as machine learning inference and field oriented control (FOC) loop for motor drive control.

4.1 Machine Learning Inference

AM64x SDK has integrated open source TensorFlow Lite for deep learning inference at the edge. AM64x is not specifically targeting real-time image processing but is can execute machine learning inference for some edge applications. As examples below are to runs of TensorFlow Lite models for image classification (224x224 pixels 3 bytes for colors) based on imagenet database and 1000 object classes. The example image of Rear Admiral Grace Hopper is installed in the file system (also available at https://github.com/google-coral/edgetpu/raw/master/test_data/grace_hopper.bmp). The example *label_image* program will crop and resize the bmp image to the 224 x 224 pixels before calling the TensorFlow Lite. The inference time benchmark for the quantization aware trained Mobilenetv1 network (mobilenet_v1_1.0_224_quant.tflite) is 280 milliseconds. The example run is shown below can also be found in the Processor SDK :

```
root@am6x-evm:/usr/share/tensorflow-lite-1.15/examples# ./label_image -i grace_hopper.bmp -l
labels.txt -m mobilenet_v1_1.0_224_quant.tflite
Loaded model mobilenet_v1_1.0_224_quant.tflite
resolved reporter
invoked
average time: 280.587 ms
0.780392: 653 military uniform
0.105882: 907 Windsor tie
0.0156863: 458 bow tie
0.0117647: 466 bulletproof vest
0.00784314: 835 suit
```

The inference time for the floating point model based inference of an image of the exact same resolution (224 x 224 x 3) is 362 milliseconds. The console command and printout is shown below:

```
root@am6x-evm:/usr/share/tensorflow-lite-1.15/examples# ./label_image -i grace_hopper.bmp -l
labels.txt -mtest/mobilenet_v2_1.0_224.tflite
Loaded model test/mobilenet_v2_1.0_224.tflite
resolved reporter
invoked
average time: 362.22 ms
0.911345: 653 military uniform
0.014466: 835 suit
0.0062473: 440 bearskin
0.00296661: 907 Windsor tie
0.00269019: 753 racket
root@am6x-evm:/usr/share/tensorflow-lite-1.15/examples#
```

The numbers printed in the console below the inference time are the top-5 classification results as a number between 0 and 1 and the class out of the 1000 in imagenet labels.txt. The accuracy result is a benchmark of the model and input image, not the device running the inference.

All .tflite models will run on AM64x, a quantized Mobilenetv1 and floating point Mobilenetv2 were chosen as common benchmarks that can be used to interpolate the performance of an inference application.

4.2 Field Oriented Control (FOC) Loop

The Sitara AM64x SDK contains a benchmark demo application that includes an example FOC loop implementing the algorithm shown in Figure 4-1 and displaying the performance of the loop. Functions in the include Clark, Park, inverse Park, Sin, Cos, PI controllers and space vector generation.

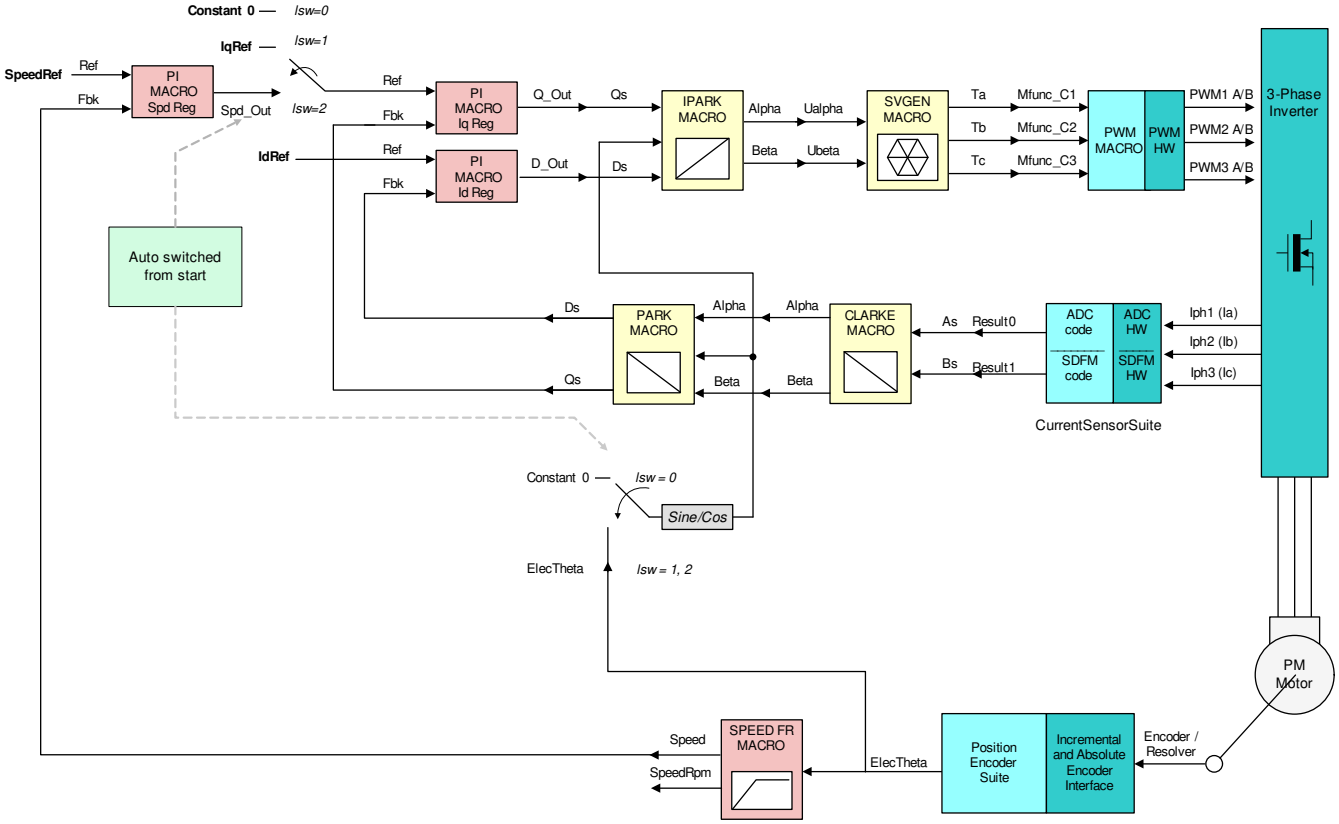


Figure 4-1. Speed Closed Field Oriented Control Loop

The performance of the FOC loop on a single Arm Cortex -R5F core of AM64x is shown in the table below. Each of the up to four Arm Cortex-R5F core can independently achieve this performance for the motor it is controlling.

	FOC Loop Execution Time (microseconds)
Average	0.4

5 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
- CMSIS library

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (https:www.ti.com/legal/termsofsale.html) 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.

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