Application Report

SPRAC45A–October 2016–Revised February 2017

Sitara AM437x DDR-Less System How-To Instructions and Benchmarks

Paula Carrillo

ABSTRACT

Embedded systems incorporate a memory hierarchy as shown in Figure 1. This is done to hide the performance gaps between the CPU and memory access. A typical memory hierarchy has the fast access, lower density closer to the CPU and the slow access, higher density far from the CPU. Most applications that run a high-level operating system (Windows, Linux, Android, and so forth) require a combination of both internal and external memory to manage the application and OS memory requirements. Depending on OS/application size (for example, RTOS), some of the components in the memory hierarchy can be eliminated. DDR memories increase system BOM, adds layout design complexity due to high speed signaling (minimum clock rate for DDR3 is 300 MHz), require dedicated supplies. Further, due to additional termination placement requirements, the overall PCB space/cost increases. This application report describes a system running EtherCAT slave with motor control leveraging the rich feature set of the Texas Instruments Sitara AM437x processor. The application on the Sitara AM437x Industrial Development Kit (IDK) EVM is demonstrated by utilizing internal SRAM, QSPI memory and eliminating external DDR completely. Benchmarking data is presented and shows no impact of the system performance with and without having DDR.

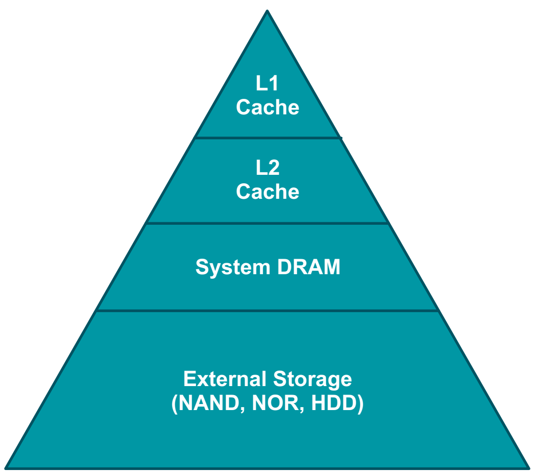

Figure 1. Memory Hierarchy

ARM, Cortex are registered trademarks of ARM Limited.
All other trademarks are the property of their respective owners.

Contents

List of Figures

List of Tables

1 TI Sitara AM437x Overview

The TI AM437x is a high-performance, scalable processor based on the ARM® Cortex®-A9 core that supports frequencies up to 1GHz. AM437x is highly integrated processor with enhanced Industrial communications and security features. The AM437x also has an enhanced 3D graphics acceleration core for enhanced user interface, support for single-cycle vector floating point (VFP) that is 10x better than ARM Cortex-A8, multiple 32-bit memory options, and plenty of I/O such as dual camera support, dual CAN, dual Gigabit Ethernet and much more. The AM437x also features a quad core programmable real-time unit for industrial communications that enables the integration of real-time industrial communications protocols and eliminates the need for an external ASIC or FPGA. AM437x also integrates crypto acceleration on all devices and enables secure boot with customer programmable keys.

The processors contain the subsystems shown in Figure 2.

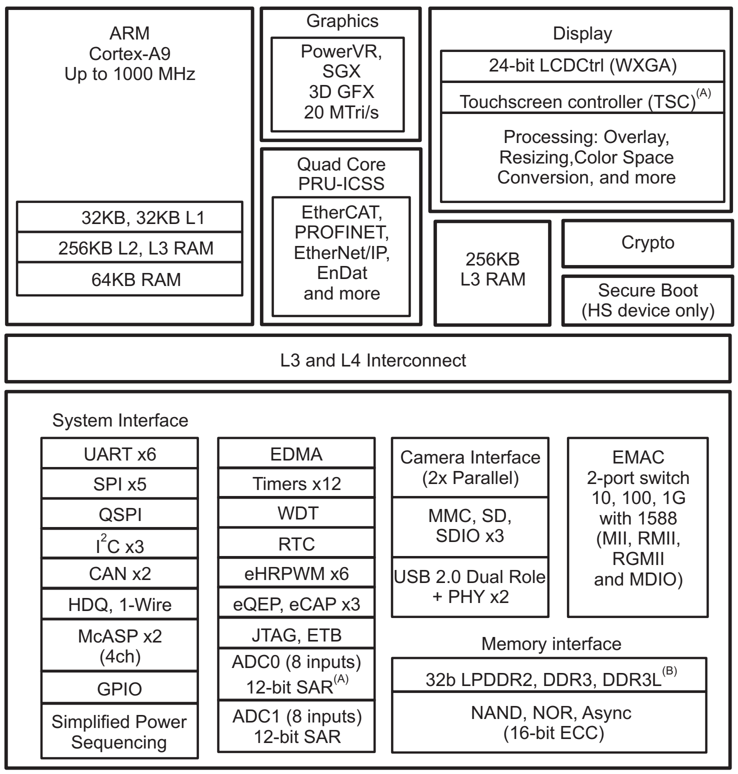

Figure 2. Block Diagram

Below some of the AM437x features highlight:

- Sitara ARM Cortex-A9 32-Bit RISC processor with processing speed up to 1000 MHz
- 256KB of L2 Cache or L3 RAM
- 256KB of general-purpose on-chip memory controller (OCMC) RAM
- General-purpose memory support (NAND, NOR, SRAM), supporting up to 16-Bit ECC
- One Quad-SPI. Supports eXecution In Place (XIP) from serial NOR flash
- High-performance interconnects provide high-bandwidth data transfers for multiple initiators to the internal and external memory controllers and to on-chip peripherals.
- Two Programmable Real-Time Units (PRUs) subsystems with two PRU cores each

For a complete list of features, see http://www.ti.com/product/AM4379.

2 Tested application: EtherCAT Slave With Motor Control Application Using AM437x

Sitara AM437x Cortex-A9 processors integrates an ARM Cortex-A9 and four PRU subsystems along with other peripherals and interfaces that make it an attractive device for building industrial automation equipment.

For this document, Sitara AM437x runs the EtherCAT slave with an integrated motor control, and uses TwinCAT as the EtherCAT PLC master. AM437x quadcore PRU performs the real-time processing for communication and data acquisition. Each of the two industrial-communication subsystems (ICSSs) contains two PRUs. PRU cores are in charge of performing EtherCAT, acquire EnDAT 2.2 master-interface position-feedback data, acquire ADC-current sense data, and communicate with ARM cortex-A9 via interrupts and shared memory. Due to the cycle nature of EtherCAT, all of this processing needs to be done before the next cycle scan. Figure 3 shows a block diagram of the application's main software and hardware components.

Sitara EtherCAT implementation encapsulates the entire EtherCAT MAC layer in the PRU subsystem through firmware. PRU EtherCAT firmware processes telegrams on-the-fly, parses them, decodes the address, executes EtherCAT commands, and communicates with the EtherCAT stack which runs on ARM cortex-A9. This implementation offers a low latency port to port communication (>700 ns), as shown in Figure 4, which is a necessary requirement for EtherCAT systems.

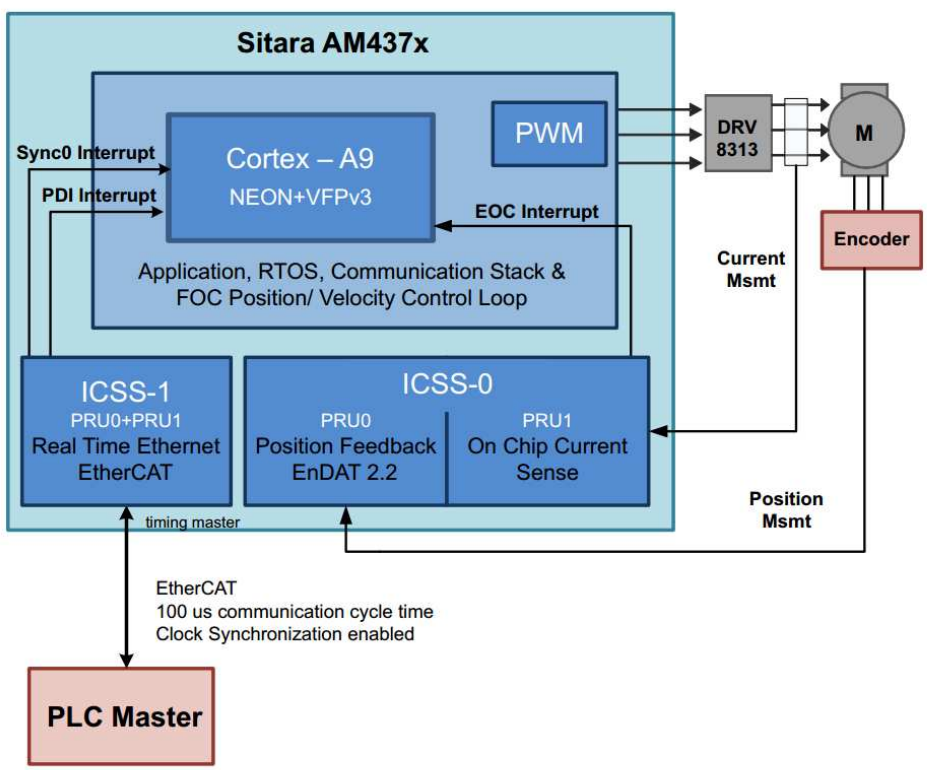

Figure 3. Sitara AM437x Block Diagram for EtherCAT Slave With Motor Control

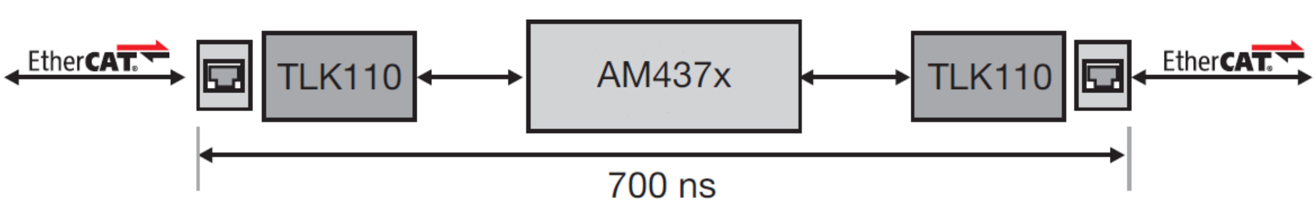

Figure 4. AM437x RX-TX Latency

3 Test Environment

The hardware and software used for testing and benchmarking a DDR-less Sitara EtherCAT slave with motor system are listed in the following subsections.

3.1 Hardware /Clock Configuration

- AM437X IDK EVM (http://www.ti.com/tool/tmdxidk437x)
  - Cortex A9. Benchmarking clock rates 288 MHz, 408 MHz, 600 MHz and 840 MHz
  - DDR clock 400 MHz
  - QSPI clock 48 MHz
- Permanent Magnet Motor (BLY171D-24V-4000, Anaheim Automation)

3.2 Software

- Sysbios Industrial SDK 2.1.1.2
- XDC 3.31.1.33_core
- TI-RTOS 6.41.4.54
- GNU v4.8.4 (Linaro)
- CCSv6.1.3

3.3 Application

- EtherCAT slave (full feature) + Motor control.

For more details about building and running EtherCAT slave example with motor control, see SYSBIOS Industrial SDK 02.01.01 User Guide “EtherCAT” and “Building Full Feature EtherCAT Application” sections or visit here.

Figure 5 shows connections required in order to run EtherCAT slave with motor control application on AM437x IDK.

3.3.1 Optional Hardware/Software

A computer with TwinCAT EtherCAT master installed.

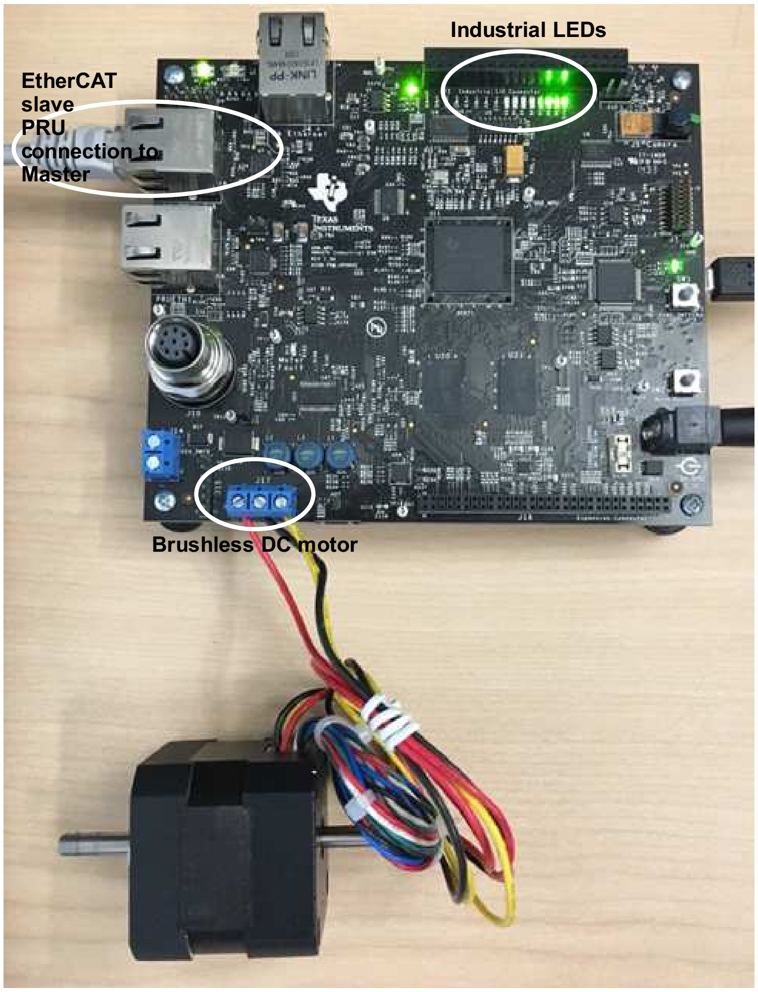

Figure 5. Hardware Test Setup

4 Software Changes

4.1 Platform Changes

A DDR-less system can be obtained using only on-chip memory or a non-DDR external type of memory such as NOR or NAND. This document explores both DDR-less system options and compares them with a DDR system. In order to compare, three different platform definitions and builds were created, which are referenced throughout this document: as “L3 build”, “QSPI XIP build”, and “DDR build”. For all three test platforms, the EtherCAT slave binary application was flashed into NOR memory for a non-volatile storage. During booting, the EtherCAT application image is copied either to L3 or DDR, or eXecute In Place (XIP). For more details, see the following subsections.

4.1.1 L3 Build (L2 as SRAM + OCMC RAM)

- Platform definition:

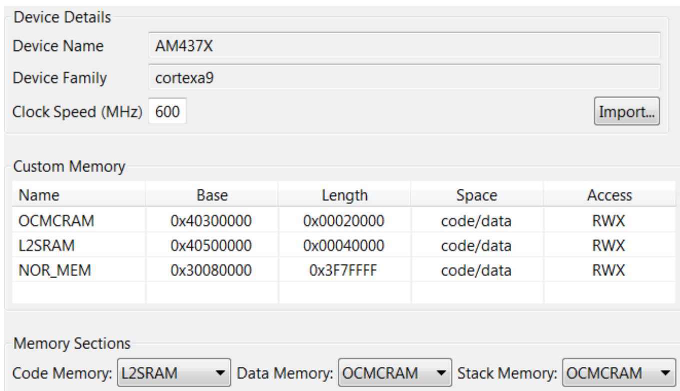

Figure 6. L3 Build - Platform Definition

4.1.2 QSPI XIP Build (NOR + OCMC RAM)

- Platform definition:

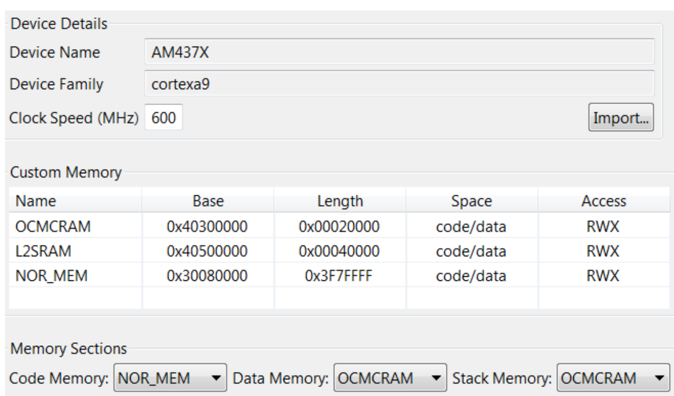

Figure 7. QSPI XIP Build - Platform Definition

4.1.3 DDR Build

- Platform definition:

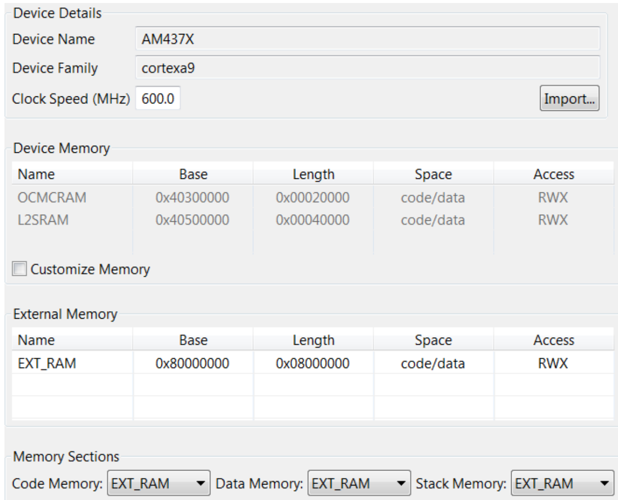

Figure 8. DDR Build - Platform Definition

4.2 Application Changes

For correctly build and run the application on L3 or from QSPI XIP, few changes need to be done in CCS configuration file project, inside the application and in the bootloader project

4.2.1 Changes in CCS ecat_appl Configuration File

In the CCS configuration file, the program load address needs to be changed according with the test.

- DDR build test
  - Cache.configureL2Sram = false;
  - Program.linkTemplate = java.lang.System.getenv("IA_SDK_HOME") + "/protocols/ethercat_slave/ecat_appl/ecat_appl.xdt";
  - Program.sectMap[".c_int00"].loadAddress = 0x80000000; //DDR mem addr
- L3 build test
  - Cache.configureL2Sram = true;
  - Program.linkTemplate = java.lang.System.getenv("IA_SDK_HOME") + "/protocols/ethercat_slave/ecat_appl/ecat_appl.xdt";
  - Program.sectMap[".c_int00"].loadAddress = 0x40500000; //L2SRAM mem addr
- QSPI XIP build test
  - Cache.configureL2Sram = false;
  - Program.linkTemplate = java.lang.System.getenv("IA_SDK_HOME") + "/protocols/ethercat_slave/ecat_appl/ecat_appl_modif_QSPI_XIP.xdt"

    For instructions on creating this xdt, see Section 4.2.2.
  - Program.sectMap[".c_int00"].loadAddress = 0x30080000; //NOR mem addr

4.2.2 Changes in ecat_appl Application Files

The changes shown in the application files are only required for the “QSPI XIP build” test.

- Cacheable bufferable MMU attr for NOR memory
  - In order to increase performance, NOR memory MMU attributes can be set to "Cacheable and bufferable".
  - File: ecat_app_cnfg.h

```
{(void *)0x30000000, SYS_MMU_CACHEABLE | SYS_MMU_BUFFERABLE},
{(void *)0x30100000, SYS_MMU_CACHEABLE | SYS_MMU_BUFFERABLE},
{(void *)0x30200000, SYS_MMU_CACHEABLE | SYS_MMU_BUFFERABLE},
{(void *)0x30300000, SYS_MMU_CACHEABLE | SYS_MMU_BUFFERABLE},
```

- Board init change: In order to avoid re-initializing Flash Memory inside the application file:
  - File tiescutils.c

```
#ifndef XIP_QSPI
     board_init(BOARD_LED_DIGOUT | BOARD_TRICOLOR0_GREEN | BOARD_TRICOLOR1_RED |
          BOARD_HVS_DIGIN | BOARD_FLASH_MEMORY);
#else
     board_init(BOARD_LED_DIGOUT | BOARD_TRICOLOR0_GREEN | BOARD_TRICOLOR1_RED |
          BOARD_HVS_DIGIN);
#endif
```

- Moving critical sections:

  Edit Program.linkTemplate = java.lang.System.getenv("IA_SDK_HOME") + "/protocols/ethercat_slave/ecat_appl/ecat_appl.xdt"

  In order to have critical .text sections closer to CPU, those sections need to be moved from NOR (REGION_TEXT) to OCMC (REGION_DATA)) inside linker template (*.xdt) file:

```
text : { ...*(EXCLUDE_FILE(knl_*.o) .text.ti_sysbios_knl*)
*(EXCLUDE_FILE(*Hwi*.o) .text.ti_sysbios*_Hwi*) ..... } > REGION_TEXT
.data : {…. *(.text.ti_sysbios_knl*)
*(.text.ti_sysbios*_Hwi*) …..} > REGION_DATA AT> REGION_TEXT
```

  Save as Program.linkTemplate = java.lang.System.getenv("IA_SDK_HOME") + "/protocols/ethercat_slave/ecat_appl/ecat_appl_modif_QSPI_XIP.xdt"

4.2.3 Bootloader Changes

Industrial SDK includes a CCS bootloader project that has different possible configurations. One of the bootloader project configurations is am43xx_boot_qspi_debug (or am43xx_boot_qspi _release). The changes below were done on top of this project configuration.

- QSPI XIP build
  - Fixing point of entrance to NOR. When eXecuting In Place is used, there is no need to copy binary image to other memory location. Therefore, there is no need to have a TI binary header which contains image application’s destination address. Then, for QSPI XIP build image copy is avoided and point of entrance is fixed.
  - File: sbl_qspi.c

```
#ifdef XIP_QSPI
*pEntryPoint = 0x30080000;//NOR memory address where app image is store
#else
status = SblQspiImageCopy(pEntryPoint); //Reads image app TI’s header. Copy image to L3 or
DDR, so program can execute from there.
#endif
```

- L3 build
  - Enabling L2 as SRAM. SYSBIOS can enable and configure L2 as SRAM, However, because the program application point of entrance, for L3 build, resides on SRAM, initialization of L2SRAM has to be done before application executes.
  - File: sbl_am43xx_platform.c

```
HW_WR_REG32((0x44E10654), 4);
HW_WR_REG32((0x44E101E0), 0x10000);
```

- CPU clocks test
  - Profiling benchmark. For benchmark purpose different CPU clocks were used
  - File: sbl_am43xx_platform_pll.c

  Test @288MHz: mpuDpllMult = 12; //12x24
  Test @408MHz: mpuDpllMult = 17; //17x24
  Test @600MHz: mpuDpllMult = 25; //25x24
  Test @840MHz: mpuDpllMult = 35; //35x24

5 Profiling and Benchmarking

With changes proposed in Section 4, the EtherCAT slave with motor can be built and run with control applications with or without DDR involved. In this section, profiling code and benchmarking results are presented. For profiling the application, focus is on profiling the Field Oriented Control (FOC) loop code, due it is the kernel function for motor control application. The FOC loop was measured and printed in the console. The cycles to microseconds can be converted by dividing by CPU clock (MHz).

5.1 Profiling Application

The snippet code below shows added time stamp code around FocLoop. In order to avoid disrupting motor control functionality, only an average value is printed every 100000 FocLoop cycles.

- tiescutils.c

```
#ifdef FOC_PROFILING
   unsigned cnt1=0, cnt2=0, diff=0;
   count_foc_loops++;

   cnt1 = Timestamp_get32();
   FocLoop();
   cnt2 = Timestamp_get32();

   diff = cnt2 - cnt1;
   time_total = time_total + diff;

   if (count_foc_loops==100000)
   {
       time_avg= time_total/count_foc_loops;
       CONSOLEUtilsPrintf("\n FOC loop: %d",time_avg);
       count_foc_loops=0;
       time_total=0;
   }
#else
  FocLoop();
#endif
```

6 Benchmark Results

For benchmarking, every test runs about 10 minutes using a motor speed of “100”. As mentioned in Section 3.3.1, TwinCAT was used as EtherCAT master.

Table 1. FOC Loop Benchamark

	Build Location ecat_appl / FOC Performance		
CPU Clock (MHz)	QSPI+OCMC FOC Loop Time (µs)	L3 (L2+OCMC) FOC Loop Time (µs)	DDR FOC Loop Time (µs)
288	11.81	13.18	11.70
408	9.72	10.45	9.58
600	8.11	8.71	8.04
840	7.15	7.75	7.10

Results demonstrate that it is possible to run the EtherCAT slave with motor control functionality without a DDR. One observation (shown in Table 1) is that “QSPI XIP build” FOC time is close to “DDR build” time.

Another observation is that “L3 build” shows a slightly worse FOC time compared with “DDR build”. This result could partially be attributed to the fact that L2 as SRAM uses the same clock as L3 (<200 MHz).

On the other hand, due to the fact that the application size is 230KB (<256KB), it can fit in L2 cache. Having an L2 cache big enough to hold the application helps to improve performance when slower memories such as DDR or NOR are used.

7 References

- *AM437x Single-Chip Motor-Control Design Guide* (TIDU800)
- *EtherCAT ® on Sitara ™ Processors, White Paper* (SPRY187)

Steps for Flashing QSPI Using SD Card

QSPI flash can be done in different ways. In this application report, a microSD card is used for loading and running the “qspi_flash_writer” application. The “qspi_flash_writer” is in charge of flashing application image in NOR memory.

Steps for flashing QSPI are shown below:

1. Copy sysbios_ind_sdk_prebuilt_02_01_01_02\bootloaders\AM437X_IDK\mmcsd_release\MLO to SD card
2. Copy C:\TI\sysbios_ind_sdk_2.1.1.2\sdk\starterware\binary\qspi_app_flash_writer\bin\am43xx-evm\gcc\qspi_app_flash_writer_a9host_release_ti.bin and renamed as app
3. Copy bootloader from C:\TI\sysbios_ind_sdk_2.1.1.2\sdk\starterware\binary\bootloader\bin\am43xx-evm\ccs \bootloader_boot_qspi_a9host_release.bin, or your modified version, and renamed as boot
4. Convert .out in a .bin using command below:
   (a) arm-none-eabi-objcopy.exe -O binary -R .ARM.exidx -R .debug_aranges -R .debug_info -R .debug_abbrev -R .debug_line -R .debug_frame -R .debug_str -R .debug_loc -R .debug_ranges ecat_appl.out ecat_appl.bin

   NOTE: arm-none-eabi-objcopy.exe can be found at C:\TI\ccsv6\tools\compiler\gcc-arm-none-eabi-4_8-2014q3\bin

5. If you are testing “L3 build” or “DDR build”, add TI’s header to your application binary. TI’s header gives an application entrance point address to the bootloader. Below examples:
   (a) L3 build: tiimage.exe 0x40500000 NONE ecat_appl.bin ecat_appl_ti.bin
   DDR build: tiimage.exe 0x80000000 NONE ecat_appl.bin ecat_appl_ti.bin
6. Copy C:\TI\sysbios_ind_sdk_2.1.1.2\sdk\protocols\ethercat_slave\ecat_appl\am437x_release\ecat_appl.bin (or ecat_appl.bin ecat_appl_ti.bin) and renamed as image
7. Created a text file with name config (without file extension) and added contents

   boot 0x0
   image 0x80000
8. Insert SD card and turn on the board. Wait few seconds or check UART messages for “Flashing completed”.
9. Removed the SD card and boot the board. Wait for approximately 20 seconds and the ECAT application should start!. LEDs turn on.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (October 2016) to A Revision **Page**

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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