SmartFusion2 SoC FPGA - Cache Controller Configuration

Table of Contents

Purpose

This application note describes the cache controller features and how to configure the cache controller for various cacheable memories in SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA).

Introduction

SmartFusion2 devices integrate an 8 KB instruction cache. The following memories are cacheable:

- Embedded nonvolatile memory (eNVM)
- DDR/SDR SDRAM

To aid in system reliability, the instruction cache is constructed of single event upset (SEU) tolerant latches. This application note describes the configuration of cache controller for various cacheable memories.

Design Requirements

Table 1 shows the design requirements and details for this application.

Table 1 • **Design Requirements and Details**

Design Requirements and Details	**Description**
Hardware Requirements	
SmartFusion2 development kit. • 12 V adapter • FlashPro4 programmer • USB A to Mini-B USB cable *Note:* Refer the *SmartFusion2 Development Kit User Guide* for more information	Rev C or later
Host PC or Laptop	• Windows XP SP2 Operating System - 32-bit/64-bit • Windows 7 Operating System - 32-bit/64-bit

Table 1 • **Design Requirements and Details (continued)**

Design Requirements and Details	Description
Software Requirements	
Libero® System-on-Chip (SoC) for viewing the design files	11.3
SoftConsole	3.4
Host PC Drivers	USB to UART drivers
One of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	-

SmartFusion2 SoC FPGA Cache Controller Overview

Figure 1 shows the system level view of the cache controller in the SmartFusion2 SoC FPGA device.

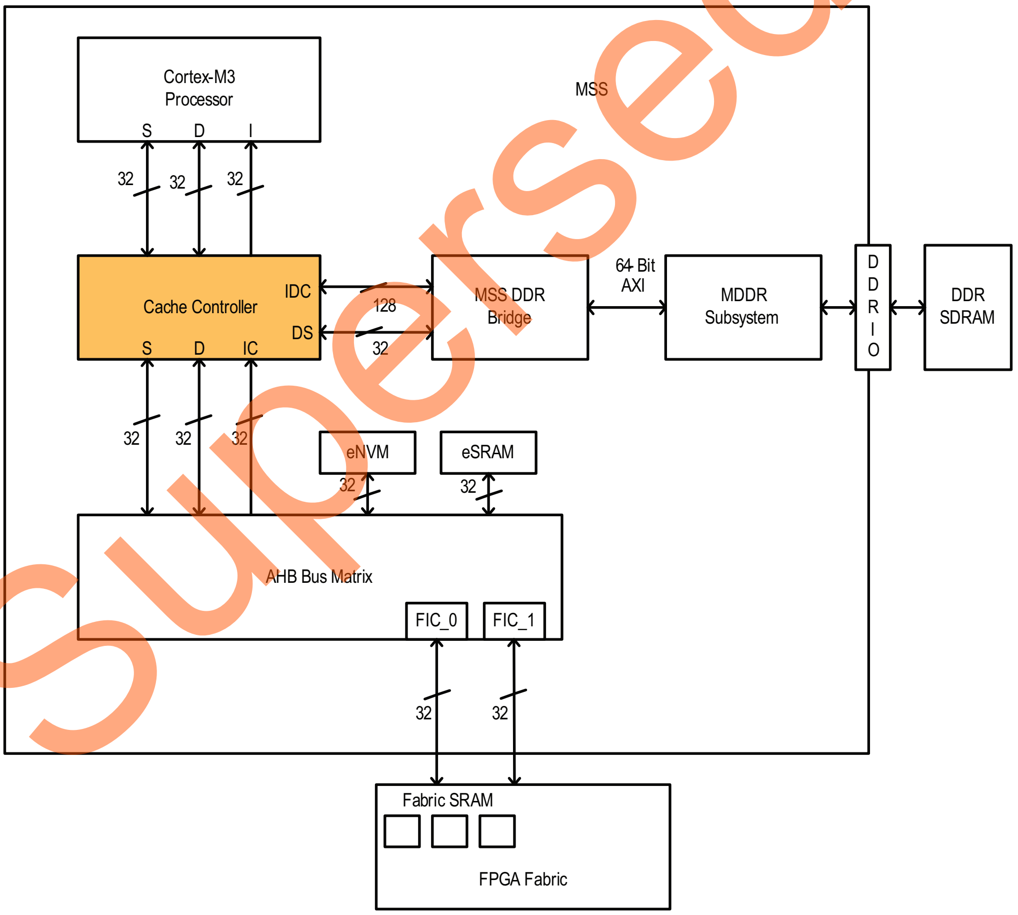


Figure 1 • **System Level View of Cache Controller in the SmartFusion2 SoC FPGA Device**

Figure 2 shows block diagram of the SmartFusion2 SoC FPGA cache controller. Refer to the *SmartFusion2 ARM Cortex-M3 and Subsystem User Guide* for more information on cache controller.

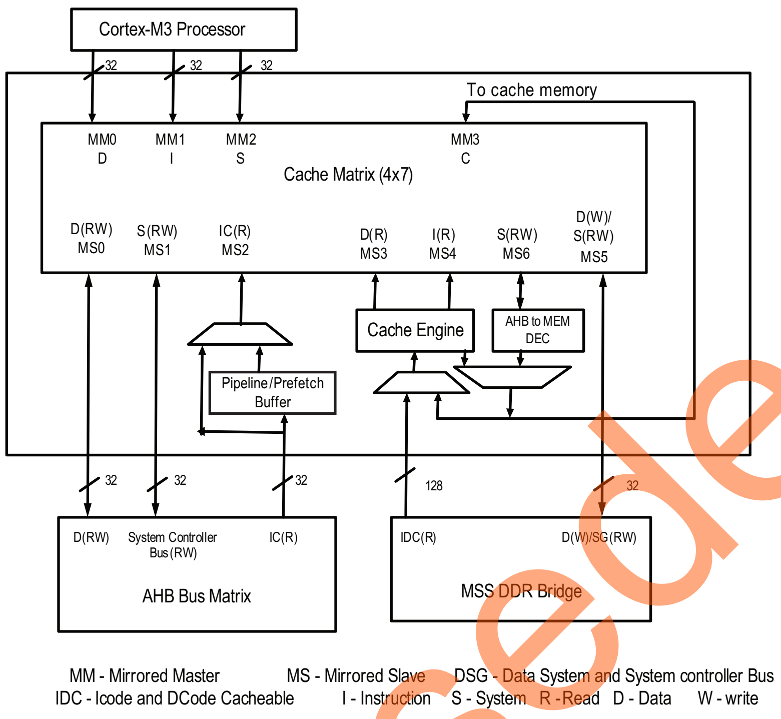


Figure 2 • **SmartFusion2 SoC FPGA Device Cache Controller Block Diagram**

Cacheable Memory Regions

The following sections explain memory mapping of eNVM or DDR/SDR SDRAM address space to cacheable memory regions. The code space of the Cortex-M3 processor ranges from 0x00000000 to 0x1FFFFFFF (0.5 GB). The address space of eNVM or DDR/SDR SDRAM can be mapped to code space of the Cortex-M3 processor to make the memory region cacheable. Design examples are provided in "Appendix A" on page 15.

Remapping eNVM as Cacheable Region

The address range of eNVM_0 is from 0x60000000 to 0x6003FFFF and the address range of eNVM_1 is from 0x60040000 to 0x6007FFFF. By default the full eNVM memory from 0x60000000 to 0x6007FFFF is mapped as a cacheable region. The eNVM base address 0x60000000 is remapped to Cortex-M3 processor address space of 0x00000000. You can remap any offset of eNVM address to the Cortex-M3 processor address space of 0x00000000 by using the ENVM_CR, ENVM_REMAPSIZE, and ENVM_REMAP_BASR_CR system registers.

Refer to "Appendix A" on page 15 for eNVM as cacheable region design files and follow "Running the Design" on page 10 for executing the reference design.

Table 2 • **Memory Map of eNVM to Cortex-M3 Processor Code Region**

Data/Code Region	Space	Address Range
M3 Data Region	**RESERVED**	0xE000_0000 to 0xFFFF_FFFF
	DDR _SPACE 3 (256 MB)	0xD000_0000 to 0xDFFF_FFFF
	DDR _SPACE 2 (256 MB)	0xC000_0000 to 0xCFFF_FFFF
	DDR_ SPACE 1 (256 MB)	0xB000_0000 to 0xBFFF_FFFF
	DDR _SPACE 0 (256 MB)	0xA000_0000 to 0xAFFF_FFFF
	eNVM SFR, Remap Area etc (1 GB)	0x6000_0000 to 0x9FFF_FFFF
	Peripheral [SPI, UART, CAN, Fabric etc] (0.5 GB)	0x4000_0000 to 0x5FFF_FFFF
	RESERVED	0x2001_0000 to 0x3FFF_FFFF
	eSRAM-1 (32 KB)	0x2000_8000 to 0x2000_FFFF
	eSRAM-0 (32 KB)	0x2000_0000 to 0x2000_7FFF
M3 Code Region	**RESERVED**	0x0008_0000 to 0x1FFF_FFFF
	eNVM (Virtual View) [512 KB]	0x0000_0000 to 0x0007_FFFF

Remapping of External RAM as Cacheable Region

You can remap DDR or SDRAM memory address to the bottom (0x0000_0000) of the Cortex-M3 processor code region by using the DDR_CR system register and any portion of the mapped memory can be made as cacheable. The cacheable region can be configured to 128 MB, 256 MB or 512 MB dynamically by using CC_REGION_CR system register. Ensure that the stack and data/heap sections of the application are out of the cacheable memory region. Refer to *Remapping eNVM, eSRAM, and DDR/SDR SDRAM Memories* application note for more information on remapping techniques and linker script file generation.

Table 3 • **Memory Map of External RAM to Cortex-M3 Processor Code Region**

Data/Code Region	Space	Address Range
M3 Data Region	**RESERVED**	0xE000_0000 to 0xFFFF_FFFF
	DDR _SPACE 3 (256 MB)	0xD000_0000 to 0xDFFF_FFFF
	DDR _SPACE 2 (256 MB)	0xC000_0000 to 0xCFFF_FFFF
	DDR_ SPACE 1 (256 MB)	0xB000_0000 to 0xBFFF_FFFF
	DDR _SPACE 0 (256 MB)	0xA000_0000 to 0xAFFF_FFFF
	eNVM SFR, Remap Area etc (1 GB)	0x6000_0000 to 0x9FFF_FFFF
	Peripheral [SPI, UART, CAN, Fabric etc] (0.5 GB)	0x4000_0000 to 0x5FFF_FFFF
	RESERVED	0x2001_0000 to 0x3FFF_FFFF
	eSRAM-1 (32 KB)	0x2000_8000 to 0x2000_FFFF
	eSRAM-0 (32 KB)	0x2000_0000 to 0x2000_7FFF
M3 Code Region	**DDR _SPACE 1 (256 MB)**	0x1000_0000 to 0x1FFF_FFFF
	DDR _SPACE 0 (256 MB)	0x0000_0000 to 0x0FFF_FFFF

SmartFusion2 SoC FPGA Cache Controller Features

The following sections explain the various user configurable features of the cache controller in the SmartFusion2 SoC FPGA device:

- Cache memory Enable/Disable
- Cache flush
- Cache locked mode

Cache Memory Enable/Disable

Cache memory can be enabled or disabled dynamically by using CC_CR system register. Instructions will be cached when the cache memory is enabled. In Cache disabled mode, all transactions are treated as non-cacheable.

Use the following steps to enable/disable the cache memory dynamically using the application code.

- Set the cacheable region
- Enable cache memory
- Run the task
- Get the cache status information
- Disable cache memory

Refer to Table 4 on page 10 for APIs to enable/disable cache memory and to get cache status information.

Cache Flush

Cache memory can be flushed in the following two ways:

- Complete cache memory flush: When you flush the full cache memory, all the cached instructions are deleted.
- Index based cache memory flush: When you flush one index in the cache memory, it invalidates all tags of four sets at one index only.

The following example steps describe how to flush the cache memory:

1. Enable cache memory.
2. Run the task (the instructions will be cached).
3. Disable cache memory.
4. Flush the cache memory (the cached instructions will be deleted).

Refer to Table 4 on page 10 for APIs to flush the cache memory.

Cache Locked Mode

Cache locked mode is a special mode that provides predictable execution which is a requirement for some specific applications. Before enabling Cache locked mode, software ensures that the code is copied to cache memory by simulating a sequential location cache miss through I-code. After copying complete 8 KB, Cache locked mode is enabled. After Cache locked mode is enabled, any access from 0 to 8 KB is directly read from the cache and the cache is not invalidated or refilled for normal operations. Memory region beyond 8 kb is treated as non-cacheable and accessed as per the memory map.

Cache locked mode can only be used with either DDR or eNVM memory and the lock base address should be in the Cortex-M3 processor code region. The code image which is copied to cache memory is also present in eNVM or DDR memory. After executing the code from the cache, the execution control comes to the main memory to execute the remaining code image. You can enable or disable Cache locked mode dynamically.

Figure 3 shows simple program execution flow with Cache locked mode.

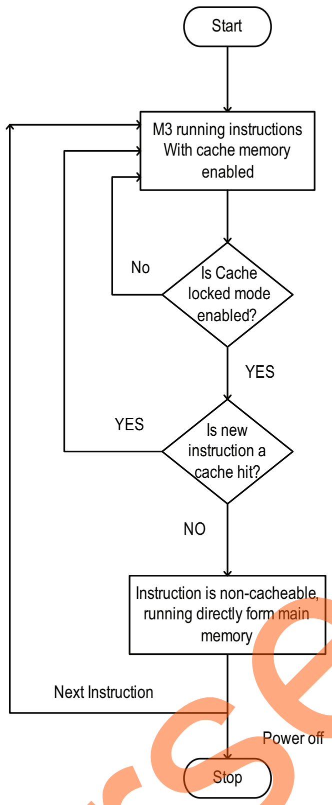

Start
M3 running instructions With cache memory enabled
No
Is Cache locked mode enabled?
YES
YES
Is new instruction a cache hit?
NO
Instruction is non-cacheable, running directly form main memory
Next Instruction
Power off
Stop

***Figure 3* • Simple Program Execution Flow with Cache Locked Mode**

Refer to Table 4 on page 10 for APIs to enable/disable Cache locked mode.

Design Description

The example designs use MMUART_0, eSRAM, DDR and eNVM memory controllers. In the design example, the MSS CCC is configured to run the M3_CLK at 111 MHz which drives the clock to Cortex-M3 processor. The cache controller can be configured either using cache controller block in System Builder configurator or through API's (Table 4 on page 10). The software application calculates the nth Fibonacci number with and without cache controller and compares the execution time. It also gets the cache status information like cache hit and cache miss to calculate the cache hit rate and cache miss rate. This application also supports cache memory flushing.

Hardware Implementation

The hardware implementation involves configuring MDDR, MMUART_0, Clocks using system builder. Figure 4 shows the top level SmartDesign of the cache controller configuration.

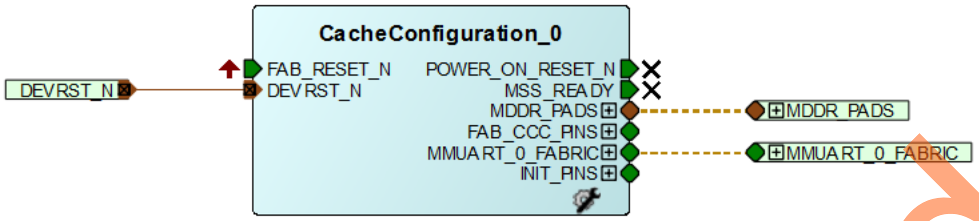
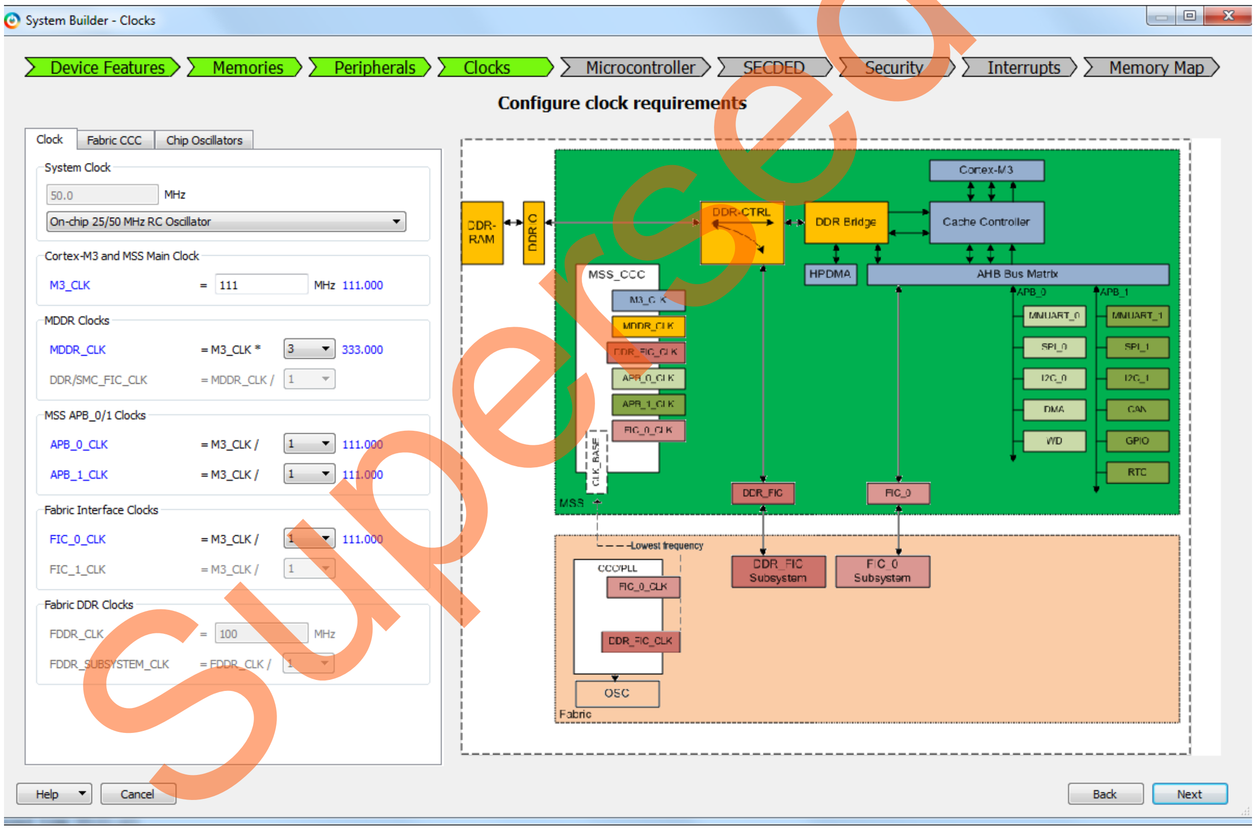


Figure 4 **• Top Level SmartDesign**

The MSS_CCC clock is sourced from the Fabric CCC. The Fabric CCC is configured to provide the 100 MHz clock using GL0. Figure 5 shows the system clocks configurations for the M3_CLK, MDDR_CLK, and APB_0_CLK/APB_1_CLK.



Configure clock requirements

System Clock

50.0 MHz

On-chip 25/50 MHz RC Oscillator

Cortex-M3 and MSS Main Clock

M3_CLK	=	111 MHz	111.000

MDDR Clocks

MDDR_CLK	= M3_CLK *	3	333.000
DDR/SMC_FIC_CLK	= MDDR_CLK /	1	

MSS APB_0/1 Clocks

APB_0_CLK	= M3_CLK /	1	111.000
APB_1_CLK	= M3_CLK /	1	111.000

Fabric Interface Clocks

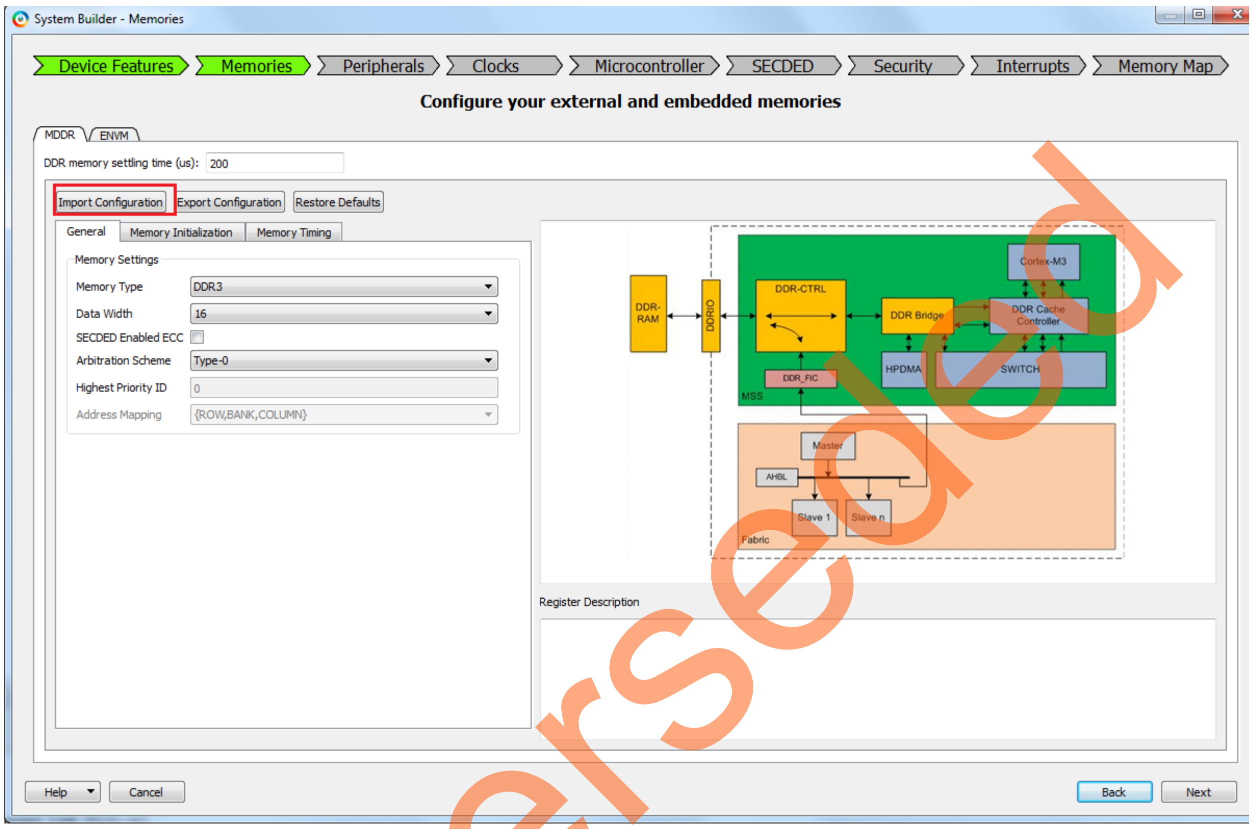
FIC_0_CLK	= M3_CLK /	1	111.000
FIC_1_CLK	= M3_CLK /	1	

Fabric DDR Clocks

FDDR_CLK	=	100 MHz	
FDDR_SUBSYSTEM_CLK	= FDDR_CLK /	1	

Figure 5 **• Clock Configurations**

The MMUART_0 is routed through FPGA fabric for communicating with the serial terminal program. The MDDR is configured for DDR3 at 333 MHz speed. Figure 6 shows MSS MDDR configuration settings. Click **Import Configuration** to import the Register configuration for DDR3 (refer to "Appendix A" on page 15 for DDR configuration file).



Configure your external and embedded memories

DDR memory settling time (us): 200

Memory Settings

Memory Type	DDR3
Data Width	16
SECDED Enabled ECC	☐
Arbitration Scheme	Type-0
Highest Priority ID	0
Address Mapping	{ROW,BANK,COLUMN}

Register Description

Figure 6 • **MSS External Memory Configurator**

Figure 7 shows the cache controller configuration through **System Builder** in Libero SoC software.

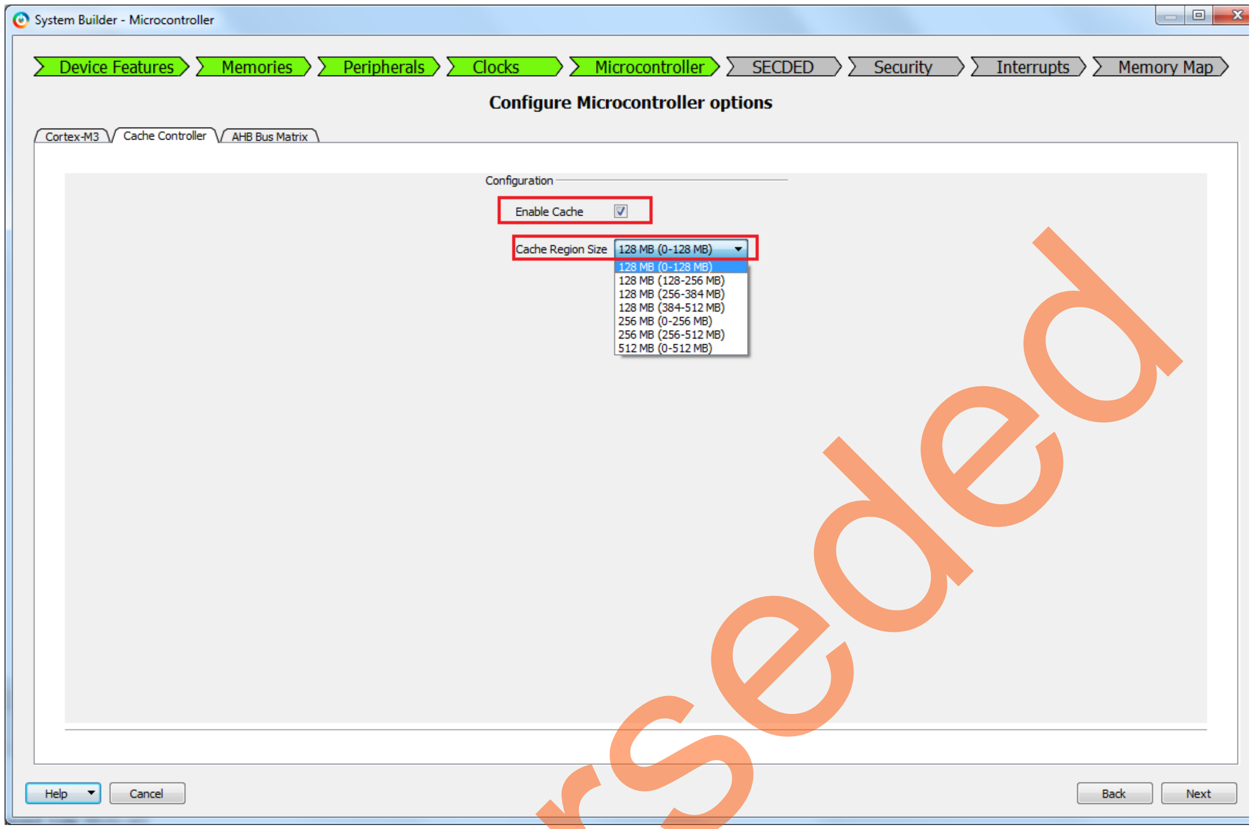


***Figure 7* • Cache Controller Configuration Through System Builder in Libero**

Note: Enabling cache through System Builder is not required in this application note design.

Software Implementation

The software design example performs the following operations:

- Enable or disable the cache controller
- Selecting cacheable region in case of DDR memory
- Cache memory flushing
- Initialization of Timer to measure execution time
- Calculating cache hit rate, cache miss rate, and task execution time
- Displaying results on serial terminal program (for example: HyperTerminal) using MMUART_0

The example software design uses UART based serial communication to communicate with serial terminal program on the Host PC. This example takes finding the nth Fibonacci number as the task and calculates the result with and without cache controller. This application selects the number randomly to find the Fibonacci number. It displays the cache hit rate, cache miss rate and execution time on serial terminal program.

In this design example, the following application images are created that can be remapped to the bottom (0x0000_0000) of the Cortex-M3 processor code region to execute the image:

1. eNVM as a cacheable region
2. DDR3 as a cacheable region with stack and data segment in the non-cacheable DDR3 memory.
   128 MB of DDR3 memory is selected as a cacheable region and following 32 KB of DDR3 is spared for stack and data/heap sections. The stack and data/heap sections of the application must be allocated in the non-cacheable DDR3 memory region.
3. DDR3 as cacheable region with stack and data segment in eSRAM.
   128 MB of DDR3 memory is selected as a cacheable region, and 32 KB of eSRAM is spared for stack and data/heap sections.

Firmware drivers:

The following firmware drivers are used in this application.

- MSS MMUART driver
  – To communicate with serial terminal program on the Host PC
- MSS Timer driver
  – To measure the task execution time.

List of APIs:

The following APIs in Table 4 are implemented in software design to configure the cache controller.

Table 4 • **APIs to Configure the Cache Controller**

API	Description	Input Parameters
MSS_CC_enable	Enables the cache memory	Void
MSS_CC_disable	Disables the cache memory	Void
MSS_CC_enable_lock	Enables Cache locked mode	Void
MSS_CC_disable_lock	Disables Cache locked mode	Void
MSS_CC_flush_index	Flushes one index in the cache memory which will be used to invalidate all tags of four sets at one index only	Index value
MSS_CC_flush	Flushes the cache memory which is used to invalidate all tags of four sets at the same time	Void
MSS_CC_set_region	Sets the cacheable region size to 128 MB, 256 MB, or 512 MB	Cacheable region value
MSS_CC_get_miss_cnt	Returns the total number of cache misses that occur in the Cacheable Region through ICode bus	Void
MSS_CC_get_hits_cnt	Returns the total number of cache hits occur in the Cacheable Region through ICode bus	Void
MSS_CC_get_trans_cnt	Returns the total number of transaction counts processed by Cache Engine	Void

Running the Design

This application note provides the following design files and describes the hardware and software requirements, board settings and steps to run the design.

- eNVM as cacheable region
- DDR3 as cacheable region with stack and data segment in non cacheable DDR3 memory
- DDR3 as cacheable region with stack and data segment in eSRAM

Board Settings

Connect the following jumpers on the SmartFusion2 SoC FPGA Development Kit, as described in Table 5. While making the jumper connections, the power supply switch SW7 on the board should be in OFF position.

Table 5 • **SmartFusion2 SoC FPGA Development Kit Jumper Settings**

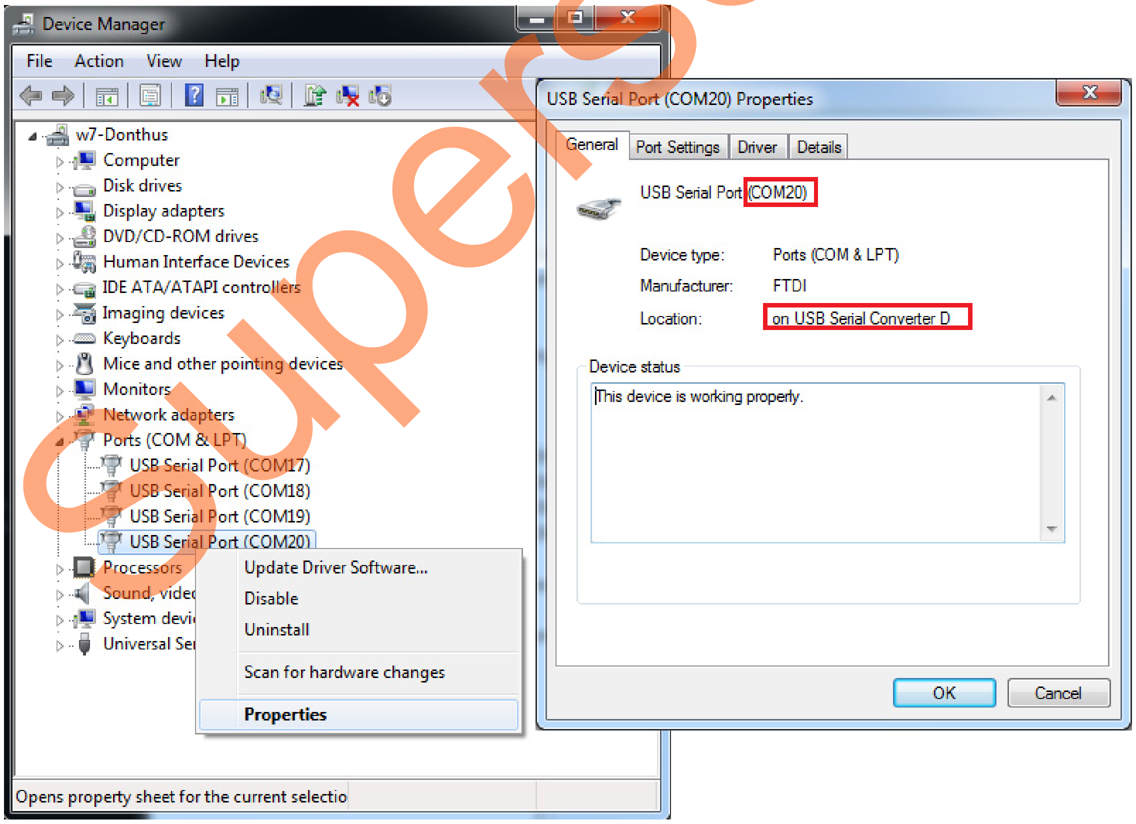
Jumper	Pin (From)	Pin (To)
J70, J93, J94, J117, J123, J142, J157, J160, J167, J225, J226, J227	1 (default)	2
J23	2 (default)	3
J129, J133	2	3

Steps to Run the Design

The following steps describe how to run the design:

1. Connect the FlashPro4 programmer to the J59 connector of SmartFusion2 SoC FPGA Development Kit. Connect one end of the USB mini-B cable to the J24 connector provided on the SmartFusion2 SoC FPGA Development Kit. Connect the other end of the USB cable to the host PC. Make sure that the USB to UART bridge drivers are automatically detected (can be verified in the Device Manager), as shown in Figure 8 on page 11.

Note: Copy the COM port number for serial port configuration. Ensure that the COM port location is specified as "on USB Serial Converter D", as shown in Figure 8.



USB Serial Port (COM20)

Device type: Ports (COM & LPT)

Manufacturer: FTDI

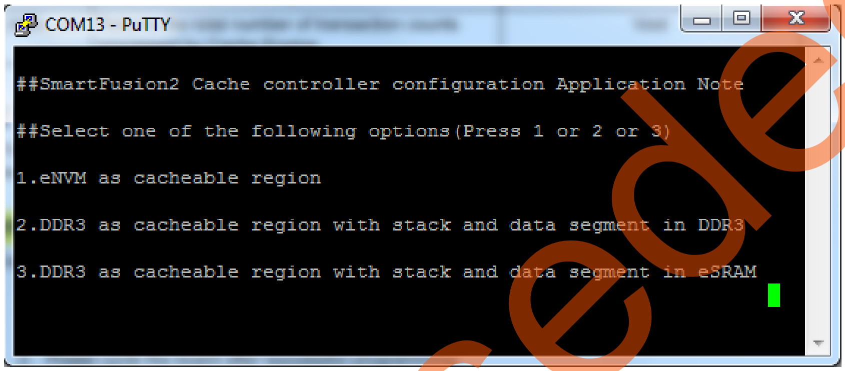
Location: on USB Serial Converter D

Device status

This device is working properly.

Figure 8 • **USB to UART Bridge Drivers**

2. If USB to UART bridge drivers are not installed, download and install the drivers from www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip.
3. Connect the power supply to the J18 connector and change the power supply switch SW7 to ON. Start HyperTerminal program with a baud rate of 57600, 8 data bits, 1 stop bit, no parity, and no flow control. If your computer does not have HyperTerminal program, use any free serial terminal emulation program such as PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.
4. Program the SmartFusion2 SoC FPGA Development Kit Board with the provided programming file (\M2S_AC389_DF\Programming File\CacheConfiguration.stp, refer to "Appendix A" on page 15) using the FlashPro software.
5. Press **SW9** switch to reset the board after successful programming.
6. The serial terminal program displays the user options, as shown in Figure 9.

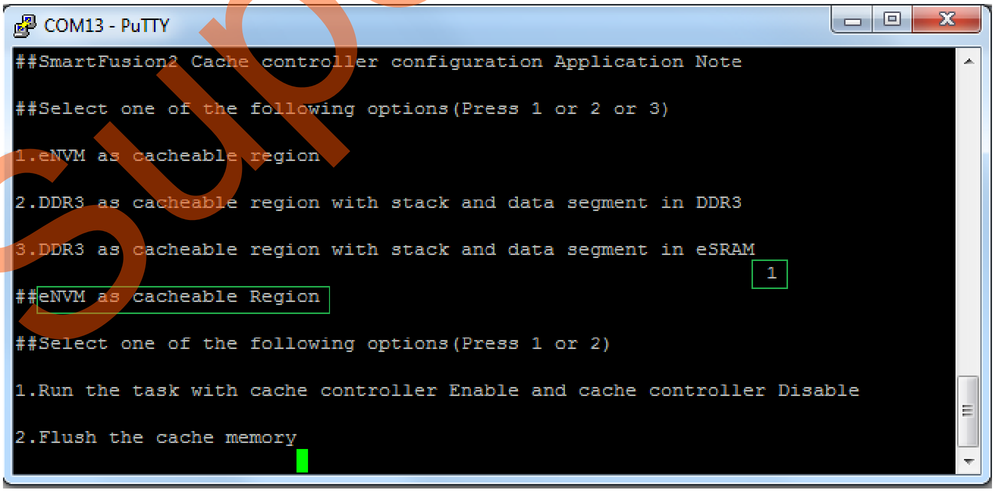


```
##SmartFusion2 Cache controller configuration Application Note

##Select one of the following options(Press 1 or 2 or 3)

1.eNVM as cacheable region

2.DDR3 as cacheable region with stack and data segment in DDR3

3.DDR3 as cacheable region with stack and data segment in eSRAM
```

***Figure 9* • User Options**

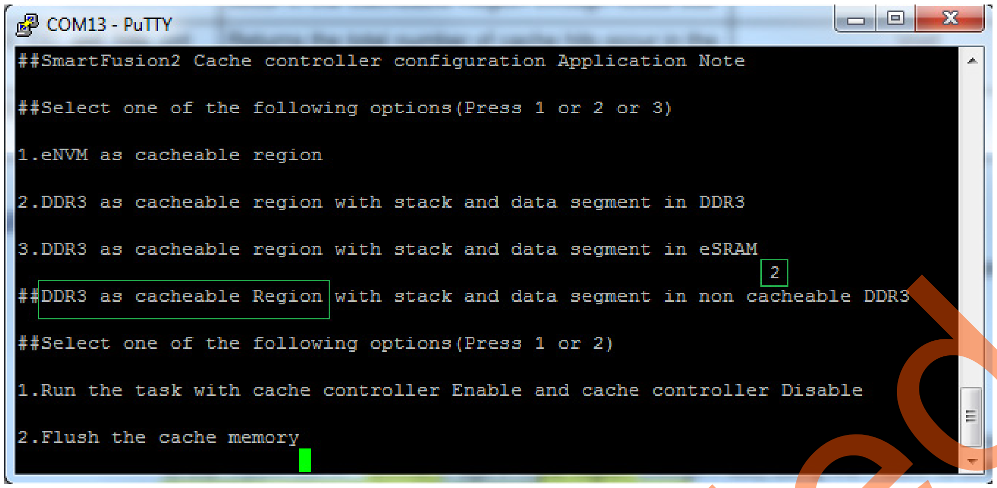
7. Select the option to remap the image to the bottom (0x0000_0000) of the Cortex-M3 processor code region to execute the code. Select option 1 to execute eNVM as cacheable memory application image as shown in Figure 10. Select 2 or 3 to execute DDR3 as cacheable memory application image as shown in Figure 11 on page 13 and Figure 12 on page 13.

Note: Reset the SmartFusion2 Development Kit board to switch among the application images.



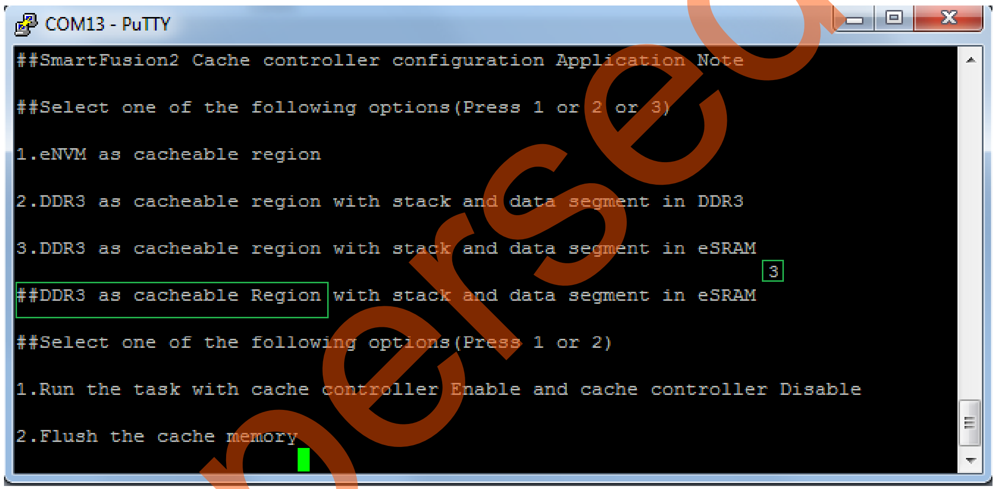
```
##SmartFusion2 Cache controller configuration Application Note

##Select one of the following options(Press 1 or 2 or 3)

1.eNVM as cacheable region

2.DDR3 as cacheable region with stack and data segment in DDR3

3.DDR3 as cacheable region with stack and data segment in eSRAM
                                                                  1
##eNVM as cacheable Region

##Select one of the following options(Press 1 or 2)

1.Run the task with cache controller Enable and cache controller Disable

2.Flush the cache memory
```

***Figure 10* • Executing eNVM as Cacheable Region Application**


COM13 - PuTTY

```
##SmartFusion2 Cache controller configuration Application Note

##Select one of the following options(Press 1 or 2 or 3)

1.eNVM as cacheable region

2.DDR3 as cacheable region with stack and data segment in DDR3

3.DDR3 as cacheable region with stack and data segment in eSRAM
                                                                2
##DDR3 as cacheable Region with stack and data segment in non cacheable DDR3

##Select one of the following options(Press 1 or 2)

1.Run the task with cache controller Enable and cache controller Disable

2.Flush the cache memory
```

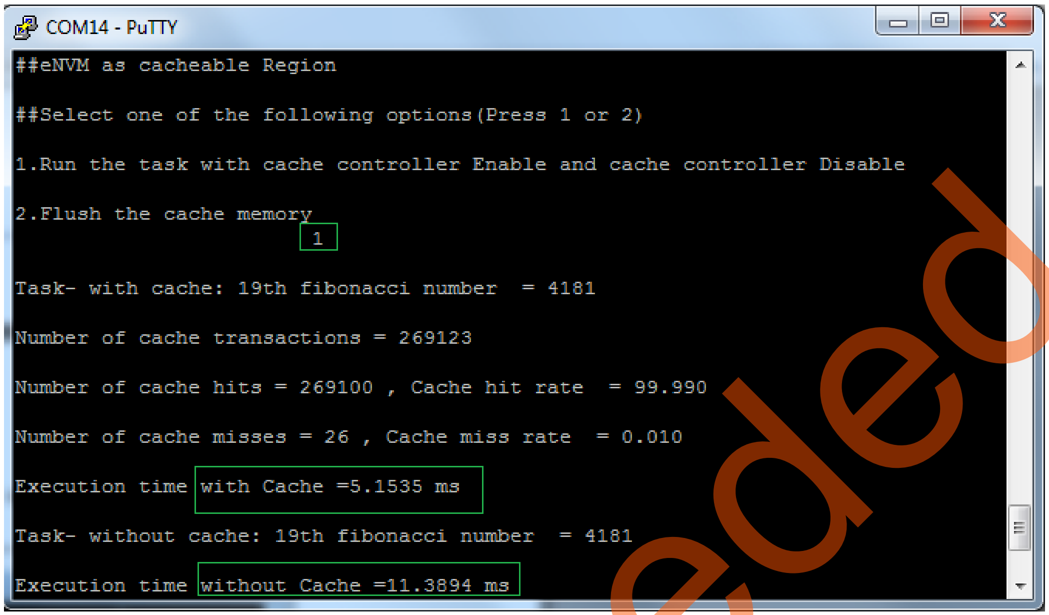
Figure 11 **• Executing DDR3 as Cacheable Region Application**


COM13 - PuTTY

```
##SmartFusion2 Cache controller configuration Application Note

##Select one of the following options(Press 1 or 2 or 3)

1.eNVM as cacheable region

2.DDR3 as cacheable region with stack and data segment in DDR3

3.DDR3 as cacheable region with stack and data segment in eSRAM
                                                                3
##DDR3 as cacheable Region with stack and data segment in eSRAM

##Select one of the following options(Press 1 or 2)

1.Run the task with cache controller Enable and cache controller Disable

2.Flush the cache memory
```

Figure 12 **• Executing DDR3 as Cacheable Region Application**

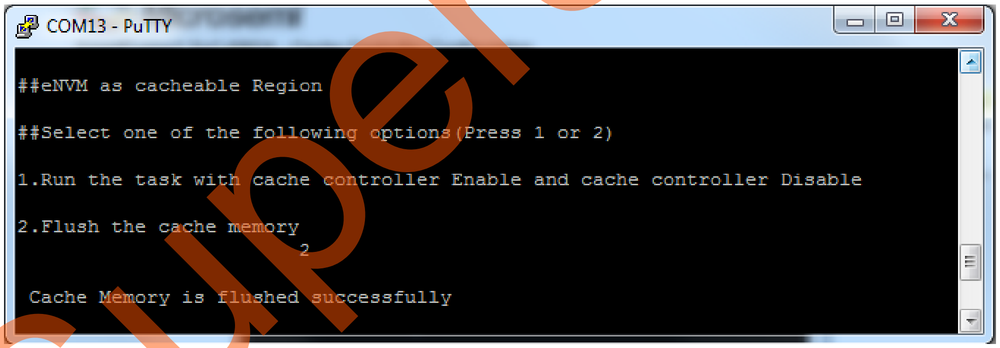
8. Selecting option 1 executes the task with the cache controller enabled. The application program calculates the task execution time with cache memory and without cache memory and also displays the cache status information, as shown in Figure 13.


COM14 - PuTTY

```
##eNVM as cacheable Region

##Select one of the following options(Press 1 or 2)

1.Run the task with cache controller Enable and cache controller Disable

2.Flush the cache memory
                         1

Task- with cache: 19th fibonacci number  = 4181

Number of cache transactions = 269123

Number of cache hits = 269100 , Cache hit rate  = 99.990

Number of cache misses = 26 , Cache miss rate  = 0.010

Execution time with Cache =5.1535 ms

Task- without cache: 19th fibonacci number  = 4181

Execution time without Cache =11.3894 ms
```

Figure 13 **• Cache Status Information With Cache Memory Enabled**

9. Selecting option 2 flushes the cache memory completely.


COM13 - PuTTY

```
##eNVM as cacheable Region

##Select one of the following options(Press 1 or 2)

1.Run the task with cache controller Enable and cache controller Disable

2.Flush the cache memory
                         2

 Cache Memory is flushed successfully
```

Figure 14 **• Cache Memory Flush**

Conclusion

This application note explains the cache controller configuration in eNVM and DDR memory remap modes supported by the SmartFusion2 SoC FPGA devices.

Appendix A

You can download the design files from the Microsemi SoC Products Group website: http://soc.microsemi.com/download/rsc/?f=M2S_AC389_DF.

The design file consists of Libero Verilog, SoftConsole software project, programming files (*.stp) for SmartFusion2 SoC FPGA Development Kit. Refer to the Readme.txt file included in the design file for the directory structure and description.

You can download the programming files (*.stp) in release mode from the Microsemi SoC Products Group website: http://soc.microsemi.com/download/rsc/?f=M2S_AC389_PF.

The programming zip file consists of the STAPL programming file (*.stp) for SmartFusion2 SoC FPGA Development Kit.

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 6 (May 2014)	Updated the document for Libero SoC v11.3 software release (SAR 57102).	NA
Revision 5 (November 2013)	Updated the document for Libero SoC v11.2 software release (SAR 52966).	NA
Revision 4 (November 2013)	Updated note (SAR 51331).	9
	Updated Figure 5 and Figure 6 (SAR 51331).	7, 8
	Updated Table 5 and Table 6 with latest version of s/w v11.1 SP2 and the latest silicon Rev D (SAR 51331).	11
	Deleted Jumper J2 from the Table 5 (SAR 51331).	11
Revision 3 (May 2013)	Updated the document for Libero SoC v11.0 software release (SAR 47616).	NA
Revision 2 (March 2013)	Updated for Libero SoC v11.0 beta SP1 release (SAR 45274).	NA
Revision 1 (November 2012)	Modified "Remapping eNVM as Cacheable Region" section (SAR 42936).	3
	Modified "Remapping of External RAM as Cacheable Region" section (SAR 42936).	4
	Modified "Cache Flush" section (SAR 42936).	5
	Modified "Software Implementation" section (SAR 42936).	9
	Modified "Running the Design" section (SAR 42936).	10
	Updated Figure 9, Figure 13 and Figure 14 (SAR 42936).	12, 14
	Modified "Conclusion" section (SAR 42936).	14
	Modified "Appendix A" section (SAR 42936).	15

Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com

51900257-6/05.14