ARM® Cortex®-M0 DesignStart™ FPGA Testbench

Revision: r1p0

User Guide

ARM®

ARM DUI 0934B (ID062017)

ARM Cortex-M0 DesignStart FPGA Testbench
User Guide

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
28 September 2015	A	Non Confidential	First issue of User Guide for r1p0
25 January 2016	B	Non Confidential	Second issue of User Guide for r1p0

Proprietary Notice

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20348

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
ARM Cortex-M0 DesignStart FPGA Testbench User Guide

Appendix A Revisions

Preface

This preface introduces the *Cortex-M0 DesignStart Fpga Testbench Guide*. It contains the following sections:

About this book

This book is for the Cortex-M0 DesignStart FPGA Prototyping Kit.

Implementation obligations

This book is designed to help you implement an ARM product. The extent to which the deliverables can be modified or disclosed is governed by the contract between ARM and the Licensee. There might be validation requirements which, if applicable, are detailed in the contract between ARM and the Licensee and which, if present, must be complied with prior to the distribution of any devices incorporating the technology described in this document. Reproduction of this document is only permitted in accordance with the licenses granted to the Licensee.

ARM assumes no liability for your overall system design and performance. Verification procedures defined by ARM are only intended to verify the correct implementation of the technology licensed by ARM, and are not intended to test the functionality or performance of the overall system. You or the Licensee are responsible for performing system level tests.

You are responsible for applications that are used in conjunction with the ARM technology described in this document, and to minimize risks, adequate design and operating safeguards must be provided for by you. Publishing information by ARM in this book of information regarding third party products or services is not an express or implied approval or endorsement of the use thereof.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

Intended audience

This book is written for hardware engineers, software engineers, system integrators, and system designers, who might not have previous experience of ARM products, but want to run a complete example of a working system.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the Cortex-M0 DesignStart FPGA Prototyping Kit and its features.

Chapter 2 *Functional Description*

Read this for a description of the design and layout of the Cortex-M0 DesignStart FPGA Prototyping Kit.

Chapter 3 *Using the Simulation Environment*

Read this for a description of how to set up and run simulation tests.

Chapter 4 *Software Tests*

Read this for a description of the example software tests and the device drivers.

Appendix A *Revisions*

Read this for a description of the technical changes between released issues of this book.

Glossary

The *ARM Glossary* is a list of terms used in ARM documentation, together with definitions for those terms. The *ARM Glossary* does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

See *ARM Glossary* http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

Conventions

This book uses the conventions that are described in:

- *Typographical conventions*.

Typographical conventions

The following table describes the typographical conventions:

Style	Purpose
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace` *`italic`*	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter http://infocenter.arm.com, for access to ARM documentation.

See ARM CMSIS-Core http://www.arm.com/cmsis, for embedded software development resources including the *Cortex Microcontroller Software Interface Standard* (CMSIS).

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *ARM® Cortex®-M System Design Kit Technical Reference Manual* (ARM DDI 0479).
- *ARM® Cortex®-M0 Devices Generic User Guide* (ARM DUI 0497).
- *ARM® Cortex®-M0 Technical Reference Manual* (ARM DDI 0432).
- *ARM® ARMv6-M Architecture Reference Manual* (ARM DDI 0419).
- *ARM® AMBA®3 AHB-Lite Protocol (v1.0) Specification* (ARM IHI 0033).
- *ARM® Versatile™ Express Cortex®-M Prototyping System (V2M-MPS2) Technical Reference Manual* (ARM 100112).
- *ARM® Cortex®-M0 DesignStart FPGA Prototyping Kit* (DAI0387).
- *ARM® Cortex®-M0 DesignStart RTL Testbench User Guide* (DUI0926).

The following confidential books are only available to licensees:

- *ARM® Cortex®-M0 Integration and Implementation Manual* (ARM DII 0238).
- *ARM® Cortex®-M0 User Guide Reference Material* (ARM DUI 0467).

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- The title.
- The number, ARM DUI 0934B.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

This chapter introduces the Cortex-M0 DesignStart FPGA testbench. It contains the following sections:

1.1 About the ARM Versatile Express Prototyping System

The ARM Versatile Express Prototyping System, MPS2, is a small development board that contains:

- An ST Microcontroller.
- An Altera FPGA.
- Various memories.
- A color LCD screen.
- An ethernet and serial connectors.
- User switches and LEDs.

You can program the FPGA with:

- The ARM Cortex-M0 processor from DesignStart, provided as an encrypted partition.
- ARM CMSDK peripherals, provided as a user-modifiable partition.

The prototyping system enables hardware and software developers to rapidly design and test hardware and software components as part of a Cortex-M0 ecosystem.

The DesignStart Cortex-M0 hardware ecosystem includes the use of the standard CMSDK library of AHB and APB bus components. The DesignStart Cortex-M0 software ecosystem also makes use of the standard CMSIS software library. Use of these two standard techniques, coupled with a reprogrammable FPGA, allows for fast turnaround and proving of Cortex-M0 based hardware and software.

1.2 About the FPGA testbench

The FPGA testbench is a component part of the ARM Versatile Express Cortex-M Prototyping System, MPS2.

The FPGA testbench includes the FPGA code, which is used to compile the Altera FPGA on the MPS2 system. This enables the testbench to simulate the full FPGA behavior and its interaction with the peripheral devices connected to the MPS2 board.

Before you can simulate the Cortex-M0 processor core as part of the FPGA testbench, you should first download the ARM Cortex-M0 DesignStart Design Kit. It includes obfuscated synthesizable code for the Cortex-M0 processor from DesignStart.

When the obfuscated code for the Cortex-M0 core has been downloaded and installed, you can compile standard C code for the Cortex-M0 processor from DesignStart to exercise both the internal FPGA peripherals and also the external MPS2 peripherals.

The Cortex-M0 DesignStart Design Kit is delivered as a compressed tar file that has to be decompressed and placed in the correct location.

For more information about:

- How to download the ARM Cortex-M0 DesignStart Design Kit, see *Prerequisites* on page 3-3.
- How to compile and run a simulation, see *Compiling a simulation in the simulation environment* on page 3-5 and *Running a simulation* on page 3-8

1.3 Cortex-M0 DesignStart FPGA Prototyping Kit directory structure

Table 1-1 describes the main directories of the Cortex-M0 DesignStart FPGA Prototyping Kit.

Table 1-1 Main directory descriptions

Directory name	Directory contents
`resources/cmsdk_r1p0`	Verilog components of the CMSDK library of AHB and APB peripherals
`resources/fpga_testbench`	FPGA testbench design files, software source and simulation scripts
`resources/smm_common`	Verilog components common to a number of Cortex-M series FPGA images
`RevB`	Verilog components and synthesis project files specific to the Cortex-M0 top level FPGA when targeting a MPS2 board.
`RevC`	Verilog components and synthesis project files specific to the Cortex-M0 top level FPGA when targeting a MPS2+ board.

Figure 1-1 on page 1-5 shows the location of the file directories in the Cortex-M0 DesignStart FPGA Prototyping Kit.

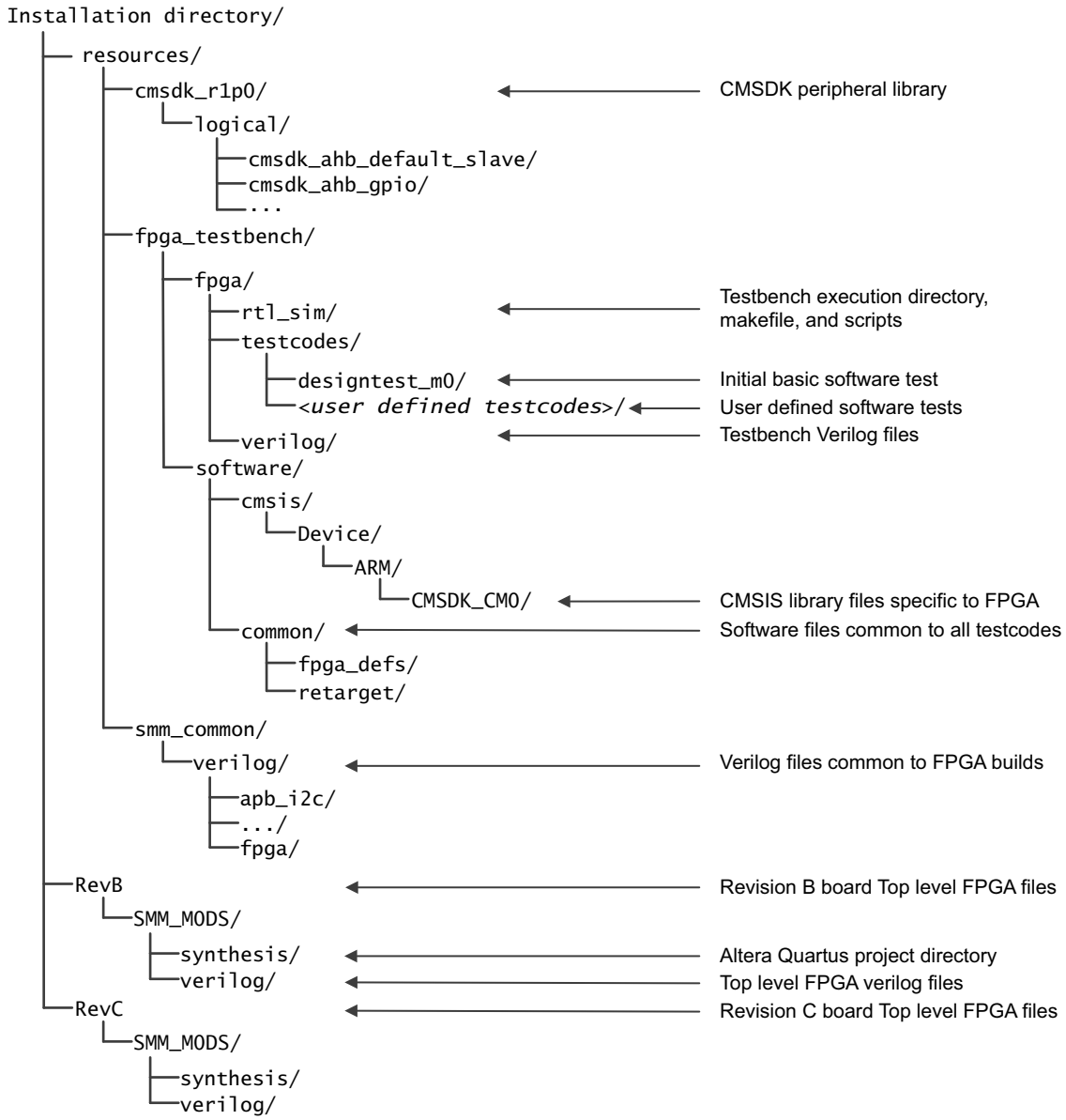

Figure 1-1 Directory structure

1.4 Limitations of the Cortex-M0 DesignStart FPGA Prototyping Kit

This section describes the limitations of the Cortex-M0 DesignStart FPGA Prototyping Kit. You should not use the processor technology or the supporting deliverables as an indicator of what is received under a full technology license of the ARM Cortex-M0 processor.

1.4.1 Deliverables

The Cortex-M0 DesignStart FPGA Prototyping Kit does not include software compilation tools. You must license these products separately.

1.4.2 Processor support

The Cortex-M0 DesignStart FPGA Prototyping Kit supports the Cortex-M0 processor from DesignStart.

Table 1-2 shows the differences in the features available in the full Cortex-M0 processor and the Cortex-M0 processor from DesignStart.

Table 1-2 Cortex-M0 processor and Cortex-M0 processor from DesignStart feature differences

Feature	Full Cortex-M0 processor	Cortex-M0 processor from DesignStart
Verilog code	Commented plain-text RTL	Flattened and obfuscated RTL
AMBA®3 AHB-Lite interface	Master and optional slave ports	Master port only
ARMv6-M instruction set	ARMv6-M instruction set support	ARMv6-M instruction set support
Multiplier options	Fast single-cycle or small 32-cycle	Fast single cycle multiplier
Nested vectored interrupt controller (NVIC)	1-32 interrupt inputs	32 interrupt inputs only
Wake-up Interrupt Controller (WIC)	Optional	None
Architectural clock gating	Optional	None
24-bit system timer, SysTick	Optional reference clock	Reference clock supported
Hardware debugger interface	Optional Serial-Wire or JTAG	Only available in the FPGA bitfile
Hardware debug support	Optional single step with up to four breakpoints, up to two watchpoints and PC sampling	Only available in the FPGA bitfile
Low-power signaling and domains	Optional state-retention power domains and power control signaling	SLEEPING, TXEV and RXEV signaling only

1.4.3 Endian support

The Cortex-M0 processor example system and its peripherals in the Cortex-M0 DesignStart FPGA Prototyping Kit are *little-endian*.

1.4.4 Platform

This release of the Cortex-M0 DesignStart FPGA Prototyping Kit supports Linux and Unix for the simulation process. If you use Keil MDK-ARM for software development, you can install the design kit in a location that is accessible from Linux, Unix, and Windows. Do this using one of the following procedures:

- Install the Cortex-M0 DesignStart FPGA Prototyping Kit on a network drive that:
  - A Linux or Unix terminal can access.
  - Is mapped to a network drive on a Windows machine.
- Use a personal computer to do the following:
  - Install virtualization software and install a guest *Operating System* (OS).
  - Set up a shared folder to access the Cortex-M0 DesignStart FPGA Prototyping Kit through the host OS.
  - Install the Cortex-M0 DesignStart FPGA Prototyping Kit in the shared folder.

Then compile the software with Keil MDK-ARM in the Windows environment, and run the simulations in the Linux or Unix environment.

To run the Cortex-M0 DesignStart FPGA Prototyping Kit on other operating systems, modify the makefiles to meet your specific requirements.

Chapter 2
Functional Description

This chapter describes the design and layout of the Cortex-M0 DesignStart FPGA Prototyping Kit. It contains the following sections:

2.1 System hierarchy

Figure 2-1 shows the FPGA testbench hierarchy, containing the Cortex-M0 processor from DesignStart and a modified CMSDK system. This indicates where the source code from the RTL testbench goes.

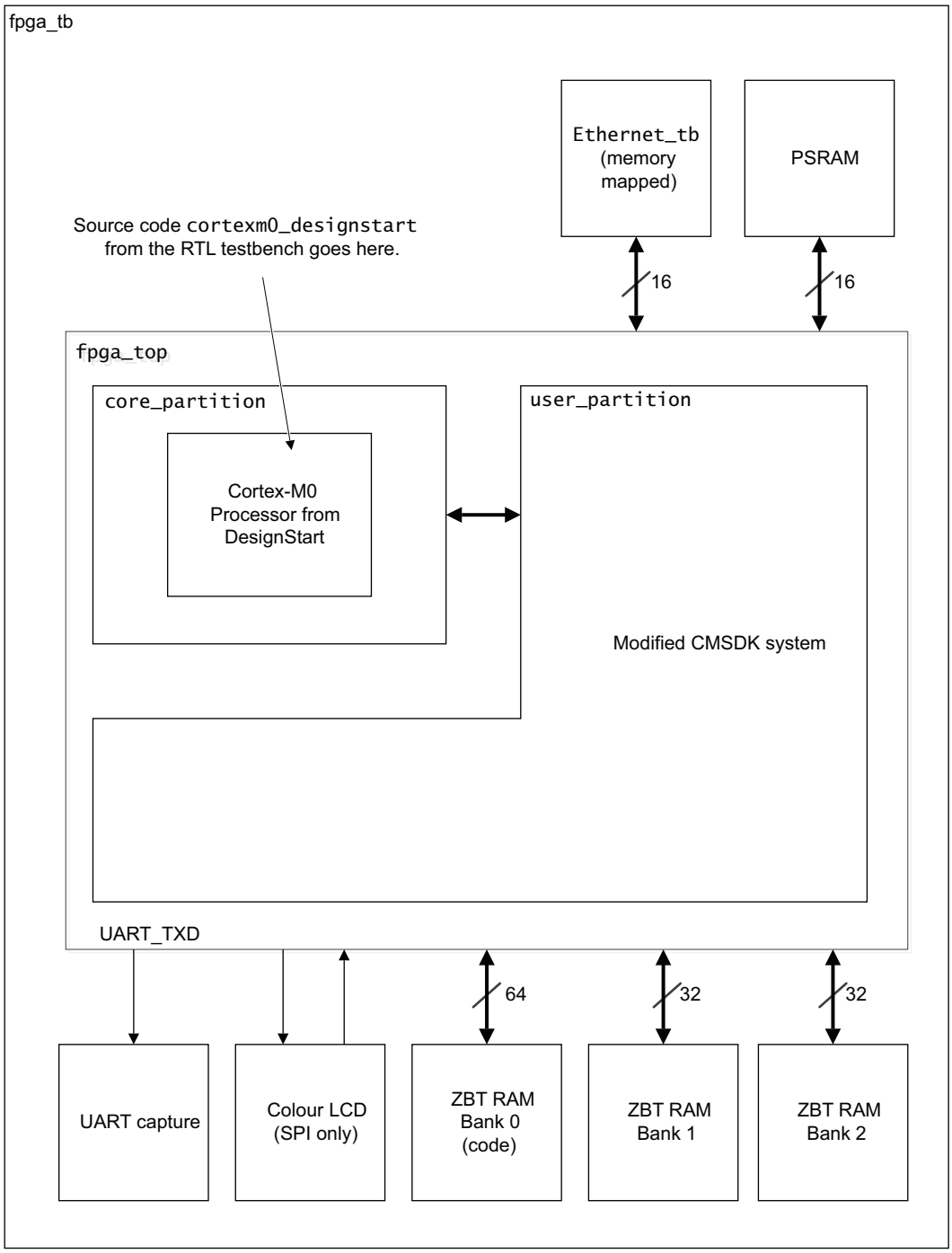

Figure 2-1 FPGA test structure

The FPGA includes extra peripherals and devices outside of the CMSDK system required to interface to the peripherals present on the MPS2 board. These peripherals include the:

- Zero Bus Turnaround RAMS (*ZBT RAMS*).
- SRAMs.
- *Pseudo-SRAM* (PSRAM).
- Color LCD screen.

The FPGA file structure and connectivity is different from the ARM Cortex-M0 DesignStart Design Kit because the FPGA supports partial reconfiguration. For the FPGA structure, the Cortex-M0 processor from DesignStart is delivered as an encrypted partial reconfiguration bit stream. To achieve this in the FPGA, the Cortex-M0 processor from DesignStart has been separated from the CMSDK system.

The differences in structure and connectivity are important when source code that might have been developed using the Cortex-M0 DesignStart Design Kit is applied to the FPGA for simulation and testing. Within the CMSDK system, it is seen that the AHB and APB bus structures, clocks and reset signals, are similar.

However, you should be aware that there is a change in connectivity between the Cortex-M0 processor from DesignStart and the CMSDK system, with the FPGA structure passing all the Cortex-M0 signals out of the CMSDK system. There are differences in the memory map too, with the FPGA having a number of extra peripherals present. You can see the differences by comparing the `cmsdk_mcu_addr_decode.v` files present in both CMSDK systems.

This means if you develop new functionality using the Cortex-M0 DesignStart Design Kit, then this functionality should be ported to the FPGA DesignStart testbench and the simulation rerun to ensure the functionality is maintained. After that you can build the FPGA and prototype with the new functionality using the MPS2 board.

2.2 Design files

The testbench files are located in `/resources/fpga_testbench/fpga/verilog`. See the directory structure in Figure 1-1 on page 1-5 for more information.

Table 2-1 shows the Verilog files that are included in the `verilog` folder.

Table 2-1 Verilog files for FPGA testbench

File name	Description
`tb_fpga.v`	Top level testbench. Instantiates peripherals below and FPGA top level.
`cmsdk_uart_capture.v`	Generates messages to simulator console from UART RS232 output.
`cyclone_crcblock.v`	Dummy placeholder for inbuilt silicon on FPGA.
`cyclone_prblock.v`	Dummy placeholder for inbuilt silicon on FPGA.
`GS8160Z36DT.v` `core.v`	Behavioral model of GSI SSRAM used on MPS2 board.
`IS66WVE409616BLL.v`	Behavioral model of ISSI PSRAM used on MPS2 board.
`scc_tb.v`	Behavioral model of SCC interface from Microcontroller to FPGA.

2.3 Cortex-M0 memory map

For information about the Cortex-M0 DesignStart memory map, see the *Cortex-M0 DesignStart RTL Testbench Guide* (DUI0926).

2.4 MPS2 Cortex-M0 DesignStart implementation memory map

For information about the MPS2 Cortex-M0 DesignStart implementation memory map, see the application note *Cortex-M0 DesignStart FPGA Prototyping Kit* (DAI0387).

Chapter 3
Using the Simulation Environment

This chapter describes how to set up and run simulation tests. It contains the following sections:

3.1 About the simulation environment

The simulation environment in this example system enables you to start an MPS2 board based simulation quickly. The simulation environment includes software files and simulation setup makefiles.

The simulation environment supports the following Verilog simulators:

- Mentor ModelSim or QuestaSim.
- Altera-Modelsim.
- Cadence NC Verilog.
- Synopsys VCS.

The `makefile` for setting up the simulation environment, which runs on `make.exe`, is created for the Linux platform.

Note

- On most Unix and Linux platforms `make.exe` is readily available. If you use a Windows simulation platform, you must download and install a suitable version of `make.exe` as required.
- If you have already installed Altera Quartus, which is required to rebuild the FPGA, `make.exe` is installed as part of the `cygwin` shell. To access this, you should add the path `C:/<quartus_installation_directory>/quartus/bin64/cygwin/bin` to your path variable.

You can compile the example software using the following:

- ARM *Development Studio 5* (DS-5).

The simulation requires one of the supported Verilog simulators and tools, for compiling and assembling the software code.

3.2 Setting up the simulation environment

The ARM Cortex-M0 DesignStart FPGA Prototyping Kit contains the FPGA testbench and all the FPGA design files with the exception of the Cortex-M0 processor. The following sections describe how to set up the simulation environment:

- *Prerequisites*.
- *Setting up with a different directory structure*.

3.2.1 Prerequisites

When you are ready to access the RTL for the design and simulation phase, you should register on designstart.arm.com. After your registration is approved, download the package ARM Cortex-M0 DesignStart Design Kit.

When you have downloaded the ARM Cortex-M0 DesignStart Design Kit, you should install the ARM Cortex-M0 DesignStart Design Kit into a working directory, for example:

```
ARM_project\cortexm0_designstart
```

See the *Cortex-M0 DesignStart RTL Testbench User Guide* for more information about how to complete the ARM Cortex-M0 DesignStart Design Kit installation.

For the FPGA design and testbench files, from the ARM Cortex-M0 DesignStart FPGA Prototyping Kit, copy the path `DVD/app_notes/AN387/designstart` to your working directory, for example:

```
/designstart_FPGA
```

Figure 3-1 shows the recommended directory structure for your simulation environment.

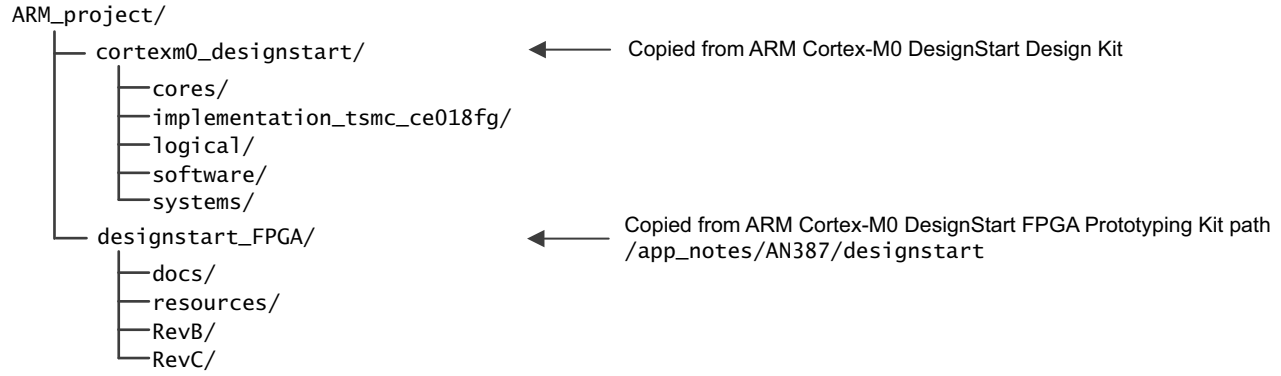

Figure 3-1 Simulation environment directory

3.2.2 Setting up with a different directory structure

The makefiles for both Verilog compilation and software compilation work with the directory structure that Figure 3-1 shows. If you require a different structure you must modify the supplied software compilation makefile and Verilog compilation makefile.

Software compilation makefile

Each software testcode requires its own specific `makefile`. The DesignStart FPGA testbench comes with one example testcode, located in `/designstart_FPGA/resources/fpga_testbench/fpga/testcodes/designtest_m0/makefile`.

To replace the <cortexm0_designstart> path with the appropriate value in the makefile, edit the following line:

```
#============================================================
# EDIT the following to point to the CMSIS directory location
# downloaded as part of the DesignStart core
#============================================================
CMSIS_CORE_DIR    = ../../../../../../<cortexm0_designstart>/software/cmsis
#============================================================
```

Verilog compilation makefile

The Verilog compilation makefile is located in /designstart_FPGA/resources/fpga_testbench/fpga/rtl_sim/tbench.vc

To replace the <cortexm0_designstart> path with the appropriate value, edit the following four lines:

```
// ============= Cortex M0 DesignStart search path =============
-y ../../../../../<cortexm0_designstart>/cores/cortexm0_designstart_r1p0/logical/cortexm0ds/verilog
-y ../../../../../<cortexm0_designstart>/cores/cortexm0_designstart_r1p0/logical/cortexm0_dap/verilog
-y ../../../../../<cortexm0_designstart>/cores/cortexm0_designstart_r1p0/logical/cortexm0_integration/verilog
-y ../../../../../<cortexm0_designstart>/cores/cortexm0_designstart_r1p0/logical/models/cells
```

3.3 Compiling a simulation in the simulation environment

This section describes how to compile and run the various testcodes. Ensure you have completed *Setting up the simulation environment* on page 3-3 before you can compile and run the various testcodes.

The following sections describe how to compile the simulation files:

- *rtl_sim/makefile simulation options*.
- *Modifying the rtl_sim/makefile*.
- *Compiling the RTL*.
- *Setting up the testcode* on page 3-6.
- *Compiling the testcode* on page 3-6.
- *Modifying configuration files* on page 3-7.

3.3.1 rtl_sim/makefile simulation options

The `makefile` in the `rtl_sim` directory controls the simulation operation with the following options:

`compile`	Compile the RTL files.
`sim`	Start an interactive session with the simulator, to allow debugging of a testcode.
`run`	Run the specified testcode in batch mode.
`all`	Run all the testcodes in batch mode.

3.3.2 Modifying the rtl_sim/makefile

You must specify several variables inside `tl_sim/makefile`. Table 3-1 describes the variables.

Table 3-1 Makefile variables

Variable	Descriptions
`TESTNAME`	Name of software test to be executed, for example, `designtest_m0`. This name must match the software directory name inside the `fpga_testbench/fpga/testcodes` directory.
`TEST_LIST`	List of tests available.

Note

- You do not have to edit all of these variables every time you run a different test. You can override the makefile variables with command line options. For example, you can keep the `TESTNAME` variable unchanged, and override it only when you run a simulation.
- See *Run the simulation* on page 3-8 for example test programs.

3.3.3 Compiling the RTL

After you have configured the environment, you must compile the Verilog RTL in the `rtl_sim` directory. To do this, use the following command:

```
</designstart_FPGA/resources/fpga_testbench/fpga/rtl_sim> make compile
```

This starts the compilation process. The process uses a Verilog command file `tbench.vc`, which specifies the location of the all required verilog design files. It also specifies any simulation specific parameters and definitions. The compile stage ignores the `TESTNAME` setting.

You can use the command line to override variables in the makefile. For example, the following command line specifies that Modelsim is used for compilation:

```
<designstart_FPGA /resources/fpga_testbench/fpga/rtl_sim> make compile SIMULATOR=mti
```

3.3.4 Setting up the testcode

There is one testcode provided as part of the ARM Cortex-M0 DesignStart FPGA Prototyping Kit, `designtest_m0`. The testcode includes both C source code and the relevant `makefile`.

The ARM Cortex-M0 DesignStart FPGA Prototyping Kit has more makefiles that allow the FPGA testbench to compile and run the majority of the testcodes provided as part of the RTL simulation.

These makefiles use the `/CMSIS` directory within the RTL testbench structure in the ARM Cortex-M0 DesignStart Design Kit. If you use a different directory structure in your simulation environment, for each of the testcode directories, in the appropriate `makefile` you must change the following line:

```
#============================================================
# EDIT the following to point to the CMSIS directory location
# downloaded as part of the DesignStart core
#============================================================
DS_RTL_TESTBENCH  = ../../../../../../cortexm0_designstart/
#============================================================
```

3.3.5 Compiling the testcode

Before you compile the software code, you might want to change some of the settings for the software compilation. Each software test has a corresponding subdirectory in the `fpga_testbench/fpga/testcodes` folder. Inside each of these directories is a makefile for software compilation. The makefiles support ARM DS-5. Table 3-2 lists the settings contained in the makefiles.

Table 3-2 Makefile settings

Variable	Descriptions	
`TESTNAME`	Name of the software test. This must match the directory name.	
`COMPILE_MICROLIB`	**0**	Normal C runtime library. This is the default value.
	1	MicroLIB, a C runtime library optimized for microcontroller applications.
`USER_DEFINE`	A user defined C preprocessing macros. Set to `-DCORTEX_M0` for most test codes. This enables a piece of test code to include the correct header for the processor when multiplex example systems share the test code. You can add additional preprocessing macros for your applications.	
`SOFTWARE_DIR`	Shared software directory	
`CMSIS_DIR`	Base location of all CMSIS source code.	
`DEVICE_DIR`	Device specific support files, for example, header files, and device driver files.	
`STARTUP_DIR`	Startup code location.	
`ARM_CC_OPTIONS`	ARM C Compiler options.	
`ARM_ASM_OPTIONS`	ARM Assembler options.	
`ARM_LINK_OPTIONS`	ARM Linker options.	

Use makefiles to compile your software. You can use one of the following makefiles:

- *The makefile in testcodes/<testname>*.
- *The makefile in rtl_sim, software compilation only*.

The makefile in testcodes/<testname>

Execute the following:

`make all` This starts the software compilation process.

You can override the variable in the makefile, for example, by executing the following:

```
make all COMPILE_MICROLIB=1
```

This causes the program to compile using DS-5 with the MicroLIB option enabled.

`make clean` This cleans all intermediate files created during the compilation process invoked by `make all`. If changes are made in code other than the testcode itself, for example, in the CMSIS header files, running `make clean` ensures that these changes are detected by a subsequent `make all`.

The makefile in rtl_sim, software compilation only

For example, in `fpga_testbench/fpga/rtl_sim/`, you can execute:

`make code` The makefile in the `rtl_sim` directory changes the current directory to the one specified by the `TESTNAME` variable. By default there is no `TESTNAME` specified in the makefile. If `make code` is executed without specifying a `TESTNAME` on the make command line or by editing the makefile, a message is printed requesting a `TESTNAME` to be specified.

You can use the command line to specify the software test that you want to run by executing the following:

```
make code TESTNAME=designtest_m0
```

This causes the `designtest_m0` test code to compile. The process then copies the compiled code images to the `rtl_sim` directory.

Note

Use the `make code` option to debug compilation errors because this option does not invoke simulation.

3.3.6 Modifying configuration files

The `/RevC/SMM_M0DS/fpga/verilog/cmsdk_mcu_defs.v` file specifies most of the configurations of the example system. This file is configured correctly to match the features available in the Cortex-M0 DesignStart deliverable.

3.4 Running a simulation

This section describes how to run a simulation in the design toolkit. It contains the following sections:

- *Run the simulation*.
- *Simulation run time command file*.

3.4.1 Run the simulation

After the RTL compilation, you can start the simulation in the `fpga_testbench/fpga/rtl_sim/` directory using one of the following commands:

`make sim` For interactive simulation.

`make run` For batch mode simulation.

The makefile in the `rtl_sim` directory automatically invokes the makefiles in the testcodes directories.

Note

The `make run` and the `make sim` steps automatically run the `make code` operation. Therefore, if you have previously compiled a test using `make code` with specific options, you must repeat the same options when you invoke `make run` or `make sim`.

For example:

- `make code TESTNAME=designtest_m0 TOOL_CHAIN=ds5 COMPILE_MICROLIB=1`
- `make sim TESTNAME=designtest_m0 TOOL_CHAIN=ds5 COMPILE_MICROLIB=1 SIMULATOR=vcs`

If you do not do this, the software test might be recompiled without the previous configuration settings. The command `make code` enables you to test that a program file compiles correctly. It does not start the simulation.

3.4.2 Simulation run time command file

For any simulation to load successfully, the simulator must first execute the simulation run time command file. There is a simulation run time command file for each supported simulator:

Modelsim `setup_mti.do`

VCS `setup_vcs.do`

NCSim `setup_nc.tcl`

The run time command file:

- Forces the address bus into any system memory connected to the AHB and APB buses to 0, runs for 1us, then releases the various address buses. This is done to remove warning messages from these memories that the address bus connected to them is X at the start of simulation.

  You can remove the force commands with no effect on the simulation. In this case, simulation time 0 warnings are issued.

- Preloads the external ZBT RAMS (bank 0), with the software code to emulate the process performed by the microcontroller on the MPS2 board. At boot-up the microcontroller:

  1. Programs the FPGA bitfiles.
  2. Loads the software code into the ZBT RAMS.

3. Sets the `zbt_boot_ctrl` to indicate the ZBT RAMS is to execute the emulation software code.

To speed up simulation time, this process is done in simulation as a direct load into RAMS.

If these commands are removed, the Cortex-M0 core does not have any software to execute and the simulation does not run.

- Forces the `zbt_boot_ctrl` signal active. This signals to the Cortex-M0 processor from DesignStart that the external ZBT RAMS (bank 0) is to execute software code. This signalling is normally set by the external microcontroller on the MPS2 board.

  If this command is removed, the Cortex-M0 processor from DesignStart does not load software correctly from the external ZBT RAMS.

Chapter 4
Software Tests

This chapter describes the example software test provided in the FPGA testbench. It contains the following sections:

4.1 About the software tests

The Cortex-M0 DesignStart FPGA testbench is able to run two forms of software test, both those written for directly for the FPGA, and also those targeted at the Cortex-M0 DesignStart RTL testbench. This is because both FPGA and RTL structures use the CMSDK with standardized memory map.

Note

The FPGA memory map is a superset of the CMSDK memory map and includes the peripherals fitted to the MPS2 board, outside of the core FPGA.

4.1.1 Available FPGA simulation tests

There is one software test, `designtest_m0`, provided as part of the FPGA testbench. It is located in the `/resources/fpga_testbench/fpga/testcodes/` directory.

Table 4-1 shows the example FPGA software tests that the Cortex-M0 DesignStart FPGA Prototyping Kit contains.

Table 4-1 Example FPGA software test list

TESTNAME	Descriptions
`Designtest_m0`	Tests that all external peripherals to the FPGA are connected correctly, read and writes to most as memory mapped devices. It uses the retargeting action that redirects `printf` to the UART output.

4.1.2 Available RTL simulation tests

Table 4-2 shows the example software tests that the RTL design kit contains.

Table 4-2 Example software test list

TESTNAME	Descriptions
`sleep_demo`	Demonstration of sleep features.
`dhry`	Simple Dhrystone test.
`self_reset_demo`	Demonstration of the self reset feature that uses the signal **SYSRESETREQ**.
`dualtimer_demo`	Demonstration of the APB Dual Timer.
`watchdog_demo`	Demonstration of the APB Watchdog.
`timer_tests`	Tests the simple APB timer.
`uart_tests`	Tests the simple APB UART.
`default_slaves_tests`	Tests the default slave activation. It accesses invalid memory locations.
`timer_driver_tests`	Simple test for the simple timer device driver functions.
`apb_mux_tests`	Simple test for the APB slave multiplexer.

Some of the tests are timing dependent and are written for a system with zero wait states. The test might fail if you change the wait states of the system.

The `config_id.h` header file in the `cortex_m0_mcu/testcodes/generic` directory contains the defines for each of the available functions in the Cortex-M0 processor. The values are set to match the fixed configuration of the CortexM0 processor from DesignStart.

4.2 Creating a new test

You can add new tests to the testcodes directory. Use the `designtest_m0` test as a guide to the format you can use. For example, you can use the following process to create a new test:

1. Create a new directory in the `testcodes/` directory. For example:
   a. `cd <installation_directory>/designstart_FPGA/resources/fpga_testbench/fpga/testcodes`
   b. `mkdir mytest`
2. Copy the files that are located in the `designtest_m0/` directory to your new test directory, and then rename the test file. For example:
   a. `cd mytest`
   b. `cp ../designtest_m0/* .`
   c. `mv designtest_m0.c mytest.c`
3. Edit the makefile to rename `designtest_m0.c` to `mytest.c`.
4. Ensure that the output `.elf` file has the same name as the directory name, for example `mytest.elf`. This enables the `makefile` in `rtl_sim/` directory to copy the `.elf` file to the `rtl_sim/` directory before the simulation starts.
5. If required, you can add the name of your new test to the `TEST_LIST` variable in the `makefile` located in the `rtl_sim/` directory.

4.3 Example header files and device driver files

The example software uses header files that are based on the *Cortex Microcontroller Software Interface Standard* (CMSIS). The example software includes the following types of files:

- Generic Cortex-M0 processor header files, located in directory `software/cmsis/CMSIS/Include/`.
- Device-specific header files, located in directory `software/cmsis/Device/ARM/CMSDK_CM0/` or `software/cmsis/Device/ARM/CMSDK_CM0plus/`.
- Device-specific startup codes, located in directory `cmsis/Device/ARM/CMSDK_CM0/Source/`.
- Device-specific example device drivers, located in directory `cmsis/Device/ARM/CMSDK_CM0/`.

Note

You must update to the latest version of the CMSIS-Core files when preparing your own CMSIS software packages. See ARM CMSIS-Core `http://www.arm.com/cmsis`.

Table 4-3 shows the generic Cortex-M0 processor support files.

Table 4-3 Generic Cortex-M0 processor support files

Filename	Descriptions
`core_cm0.h`	CMSIS 3.0 compatible header file for processor peripheral registers definitions.
`core_cmInstr.h`	CMSIS 3.0 compatible header file for accessing special instructions.
`core_cmFunc.h`	CMSIS 3.0 compatible header file for accessing special registers.
`system_CMSDK_CM0.h`	CMSIS 3.0 compatible header file for system functions.
`system_CMSDK_CM0.c`	CMSIS 3.0 compatible program file for system functions.

Table 4-4 shows the device-specific header files. These files contain changes specific to the FPGA and are located in the `fpga_testbench/software/CMSIS` directory.

Table 4-4 Device-specific header files

Filename	Descriptions
`CMSDK_CM0.h`	CMSIS compatible device header file including register definitions.

Table 4-5 shows the device-specific startup codes. These files contain changes specific to the FPGA and are located in the `fpga_testbench/software/CMSIS` directory.

Table 4-5 Device-specific startup codes

Filename	Descriptions
`ARM/startup_CMSDK_CM0.s`	CMSIS compatible startup code for ARM DS-5 or Keil MDK
`GCC/startup_CMSDK_CM0.s`	CMSIS compatible startup code for ARM GCC

Table 4-6 shows the device-specific example device drivers.

Table 4-6 Device-specific example device drivers

Filename	Descriptions
CMSDK_driver.h	Header file for including driver code
CMSDK_driver.c	Driver code implementation

To use these header files, you only have to include the device-specific header file CMSDK_CM0.h. This file imports all the required header files. Because some of the shared program files in the software/common directory also support different types of processor, these programs include the following header code:

```
#ifdef CORTEX_M0
#include "CMSDK_CM0.h"
#endif
```

The makefile in directory fpga_testbench/fpga/testcodes/<testname> contains the USER_DEFINE variable that defines the C preprocessing directive CORTEX_M0. This ensures that the simulation uses the correct version of the header file.

Appendix A
Revisions

This appendix describes the technical changes between released issues of this book.

Table A-1 Issue A

Change	Location	Affects
First issue	-	-

Table A-2 Changes between issue A and issue B

Changes	Locations	Affects
Remove references to ARM GCC.	*About the simulation environment* on page 3-2. *Compiling the testcode* on page 3-6.	All versions.