Cortex-A5 DesignStart Cycle Model Fixed Virtual Platform

Version 1.0

User Guide

arm

101854_0100_00_en

Cortex-A5 DesignStart Cycle Model Fixed Virtual Platform

User Guide

Release Information

Document History

Issue	Date	Confidentiality	Change
0100-00	8 October 2019	Non-Confidential	First release

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

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.

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

www.arm.com

Contents

Cortex-A5 DesignStart Cycle Model Fixed Virtual Platform User Guide

Preface

This preface introduces the *Cortex-A5 DesignStart Cycle Model Fixed Virtual Platform User Guide*.

It contains the following:

About this book

This guide describes how to use the Cycle Model and Fast Model in the Cortex-A5 DesignStart Fixed Virtual Platform.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

This chapter introduces the Arm Cortex®-A5 DesignStart Cycle Model Fixed Virtual Platform (FVP).

Chapter 2 Running the DesignStart Cycle Model FVP

This section describes downloading and running the DesignStart Cycle Model FVP, including customizing the performance analysis data you want to collect.

Chapter 3 DesignStart Cycle Model FVP system details

This section provides configuration details for the components included in the Arm Cortex-A5 DesignStart Cycle Model FVP.

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 the *Arm® Glossary* for more information.

Typographic conventions

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.

`monospace`

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.

`<and>`

Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Feedback

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 *Cortex-A5 DesignStart Cycle Model Fixed Virtual Platform User Guide*.
- The number 101854_0100_00_en.
- If applicable, the page number(s) to which your comments refer.
- 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.

Other information

- *Arm® Developer*.
- *Arm® Information Center*.
- *Arm® Technical Support Knowledge Articles*.
- *Technical Support*.
- *Arm® Glossary*.

Chapter 1
Introduction

This chapter introduces the Arm Cortex®-A5 DesignStart Cycle Model Fixed Virtual Platform (FVP).

It contains the following sections:

1.1 Introduction to the Arm® Cortex®-A5 DesignStart Cycle Model FVP

This Arm DesignStart Cycle Model FVP is a virtual representation of a Cortex-A5 system.

The system has two representations, one based on Arm Cycle Models and the other based on Arm Fast Models. Software validation and debug is done using the Fast Models system. Then, simulate using the Cycle Models system to evaluate cycle-accurate performance metrics using software benchmarks.

A default application is included along with application source files, allowing you to make changes to the software and repeat the validation and performance evaluation process.

This workflow is represented as follows:

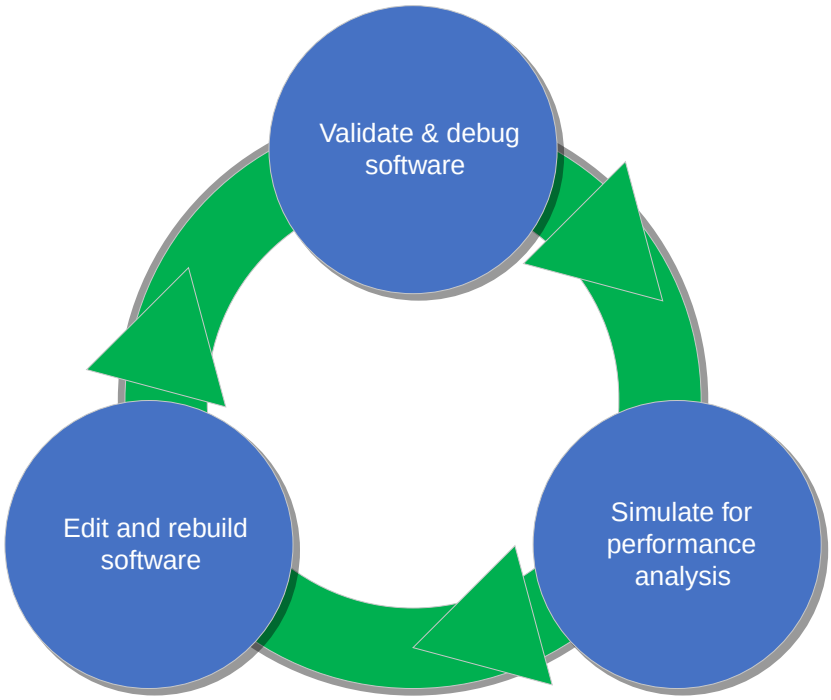

Figure 1-1 Cortex-A5 DesignStart Cycle Model FVP workflow

System components

The DesignStart Cycle Model FVP system includes a Cortex-A5 processor, cache controller, interconnect, memory, and UART (Trickbox).

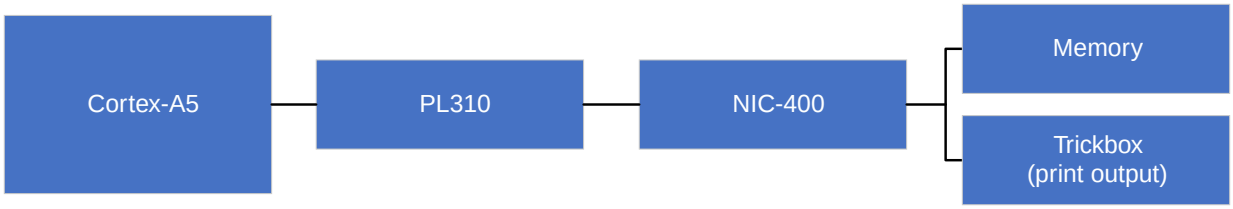

Figure 1-2 Cortex-A5 DesignStart Cycle Model FVP system components

See *3.1 Configuration reference* on page 3-27 for details about the system configuration.

Available performance metrics

The Cortex-A5 DesignStart Cycle Model FVP system supports:

- Standard output of cycle count and performance data for each bus.
- Logging of AXI bus transactions. See *2.6 AXI bus logging* on page 2-25 for details.
- Output of waveform data in VCD format for the Cycle Model simulation. The waveform file is output to `systems/cyclemodel/system/arm_cm_NIC400.vcd`.

Note

You can also enable the capture of specified Performance Monitoring events, and target specific areas of the software for performance monitoring. See *2.4 Setting performance monitoring behavior* on page 2-22 for instructions.

1.2 Package contents

The Cortex-A5 DesignStart Cycle Model FVP is composed of a pre-compiled virtual system and Dhrystone Benchmark software.

The Cortex-A5 DesignStart Cycle Model FVP is developed and tested on Red Hat Enterprise Linux Version 6.

System components

The Cortex-A5 DesignStart Cycle Model FVP consists of the following components. See *3.1 Configuration reference* on page 3-27 for details about the system configuration:

- Arm Cortex-A5 processor Cycle Model based on r0p1 RTL
- PL310 Arm PrimeCell Level 2 Cache Controller Cycle Model based on r3p3 RTL
- Arm NIC-400 Cycle Model based on r1p2 RTL
- System memory map that is aligned to the Cortex-A5 DesignStart RTL
- Trickbox (UART)

Default application

Dhrystone performance benchmark software is included in the Cortex-A5 DesignStart Cycle Model FVP tarball. Application binaries and sources are also included, so you can make modifications to the software. You can also create your own software to simulate with the Cycle Model FVP. See *Chapter 2 Running the DesignStart Cycle Model FVP* on page 2-15 for more information.

Simulation environment

The pre-built Cycle Model and Fast Model FVP systems simulate in an environment that includes:

- Fast Models version 11.7
- Accellera SystemC 2.3.1
- Cycle Model runtime version 11.0.3

1.3 Prerequisites

Before beginning to use the Arm Cortex-A5 DesignStart Cycle Model FVP, your environment must meet the requirements described in this section.

Simulation environment

The Cortex-A5 DesignStart Cycle Model FVP requires runtime libraries found in GCC version 6.4.0. Add the libraries of GCC version 6.4.0 to the environment variable `LD_LIBRARY_PATH`.

Licensing

You must have a valid, installed license for the Cortex-A5 DesignStart Cycle Model FVP. Visit the Arm licensing portal: *https://developer.arm.com/support/licensing/generate* and use your serial numbers to generate the licenses. Contact Arm Techical Support (support-esl@arm.com) if you need more information.

Debugging

Arm Development Studio (or another CADI-enabled debugger) is required for software debugging.

Software rebuild and development

To make changes to the default Dhrystone application, or to develop your own software to simulate with the Arm Cortex-A5 DesignStart Cycle Model FVP, you need:

- Arm Compiler 6

1.4 Instruction quick start

The command flow shows the instructions needed to get up and running quickly.

Before you begin, ensure your environment meets the *1.3 Prerequisites* on page 1-12.

Table 1-1 Instruction flow quick start

Task	Command
Untar the package.	`tar xzvf Cortex-A5_DesignStart_CM_FVP_1.0.tgz`
Access the directory `systems/fastmodel/system`.	`cd Cortex-A5_DesignStart_CM_FVP_1.0/systems/fastmodel/system/`
Run the Dhrystone application on the Fast Models system.	`./run.sh`
Access the directory `systems/cyclemodel/system`.	`cd Cortex-A5_DesignStart_CM_FVP_1.0/systems/cyclemodel/system/`
Run the Dhrystone application on the Cycle Models system.	`./run.sh`

See *Chapter 2 Running the DesignStart Cycle Model FVP* on page 2-15 for more information and next steps.

1.5 Related information

Additional documentation for the Arm Cortex-A5 DesignStart program is available.

You can find the following documents at *https://developer.arm.com*:

- *Arm® Cortex®-A5 DesignStart Getting Started Guide* (101857)
- *Arm® Cortex®-A5 Technical Reference Manual* (DDI0433)
- *Arm® CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual* (DDI0246)

Chapter 2
Running the DesignStart Cycle Model FVP

This section describes downloading and running the DesignStart Cycle Model FVP, including customizing the performance analysis data you want to collect.

It contains the following sections:

2.1 Downloading and unpacking the tarball

This section describes downloading the package and extracting its contents.

Download

Visit the Models for Cortex-A5 DesignStart page on the Arm Developer web site (*https://developer.arm.com/tools-and-software/simulation-models/designstart-simulation-models/cortex-a5-designstart-model*) and download the Cycle Model for Cortex-A5 DesignStart package.

The file is named `Cortex-A5_DesignStart_CM_FVP_1.0.tgz`.

Decompress the DesignStart Cycle Model FVP package

1. Untar the DesignStart Cycle Model FVP package:

```
$ tar xzvf Cortex-A5_DesignStart_CM_FVP_1.0.tgz
```

   The DesignStart Cycle Model FVP directory structure is created.

Next Steps

Proceed to *2.2 Validating the software* on page 2-17.

2.2 Validating the software

To validate your software, simulate with the Fast Models system provided with the DesignStart Cycle Model FVP package. After validation, you can debug the software if necessary, or simulate for performance analysis.

Before you begin

- Download and extract the DesignStart Cycle Model FVP package as described in *2.1 Downloading and unpacking the tarball* on page 2-16.
- Ensure that the environment variable `ARMLMD_LICENSE_FILE` is set to your license server; for example, `export ARMLMD_LICENSE_FILE=port@host`.

Validate the software

1. Change to the directory `systems/fastmodel/system`.

```
$ cd Cortex-A5_DesignStart_CM_FVP_1.0/systems/fastmodel/system/
```

2. Run the script. You may need to set `gcc:6.4.0` in your shell first (see *1.3 Prerequisites* on page 1-12 for details).

```
$ ./run.sh
```

Result

The Fast Model system runs the Dhrystone performance benchmark. The number of instructions executed may differ from the number shown in this example:

```
Fast Models [11.7.36 (Jun 25 2019)]
Copyright 2000-2019 ARM Limited.
All Rights Reserved.

Running on CPU0

Dhrystone Benchmark, Version 2.1 (Language: C)

Execution starts, 1000 runs through Dhrystone

Execution ends

Final values of the variables used in the benchmark:

Int_Glob:            5
        should be: 5
Bool_Glob:           1
        should be: 1
Ch_1_Glob:           A
        should be: A
Ch_2_Glob:           B
        should be: B
Arr_1_Glob[8]:       7
        should be: 7
Arr_2_Glob[8][7]:    1010
        should be: Number_Of_Runs + 10
Ptr_Glob->
Ptr_Comp:          22104
    should be:   (implementation-dependent)
Discr:             0
    should be:  0
Enum_Comp:          2
    should be:  2
Int_Comp:          17
    should be:  17
Str_Comp:          DHRYSTONE PROGRAM, SOME STRING
    should be:  DHRYSTONE PROGRAM, SOME STRING
Next_Ptr_Glob->
    Ptr_Comp:   22104
should be:   (implementation-dependent), same as above
Discr:             0
    should be:  0
Enum_Comp:          1
    should be:  1
Int_Comp:          18
    should be:  18
```

```
Str_Comp:        DHRYSTONE PROGRAM, SOME STRING
    should be:  DHRYSTONE PROGRAM, SOME STRING
Int_1_Loc:       5
    should be:  5
Int_2_Loc:       13
    should be:  13
Int_3_Loc:       7
    should be:  7
Enum_Loc:        1
    should be:  1
Str_1_Loc:       DHRYSTONE PROGRAM, 1'ST STRING
    should be:  DHRYSTONE PROGRAM, 1'ST STRING
Str_2_Loc:       DHRYSTONE PROGRAM, 2'ND STRING
    should be:  DHRYSTONE PROGRAM, 2'ND STRING


_________________________
Marked software performance monitor results:

Instructions Executed = 305854
Cycle Count (CCNT) = 305854
Loads = 52108
Average cycles per instruction = 1.000000

End of marked software performance monitor results.
_________________________
Main Complete, stopping simulation ...
simulation is complete

Info: /OSCI/SystemC: Simulation stopped by user.
$
```

Next steps

- (Optional). Configure a debug session using Arm Development Studio. See *https://developer.arm.com/docs/101469/latest/tutorials/tutorial-hello-world/configure-your-debug-session* for instructions.
- Simulate using the Cycle Model system. See *2.3 Simulating for performance analysis* on page 2-19.

2.3 Simulating for performance analysis

This section describes how to simulate the Dhrystone application with the Cycle Models FVP system, and shows example output.

Before you begin

- Run the Fast Model system as described in *2.2 Validating the software* on page 2-17.
- Ensure that the environment variable `ARMLMD_LICENSE_FILE` is set to your license server; for example, `export ARMLMD_LICENSE_FILE=`*`port@host`*.

Simulate with the Cycle Model system

1. Change to the directory `systems/cyclemodel/system`.

```
$ cd Cortex-A5_DesignStart_CM_FVP_1.0/systems/cyclemodel/system/
```

2. Run the script. You may need to set `gcc:6.4.0` in your shell first (see *1.3 Prerequisites* on page 1-12 for details).

```
$ ./run.sh
```

Result

The Cycle Model simulation runs with the Dhrystone benchmark application. Performance statistics may differ from those shown in the example:

```
Arm Cycle Model SystemC setup completed.
Starting Simulation
Running on CPU0
<0d>
<0d>Dhrystone Benchmark, Version 2.1 (Language: C)
<0d>
<0d>Execution starts, 1000 runs through Dhrystone
<0d>
<0d>Execution ends
<0d>
<0d>Final values of the variables used in the benchmark:
<0d>
<0d>Int_Glob:            5
<0d>        should be:   5
<0d>Bool_Glob:           1
<0d>        should be:   1
<0d>Ch_1_Glob:           A
<0d>        should be:   A
<0d>Ch_2_Glob:           B
<0d>        should be:   B
<0d>Arr_1_Glob[8]:       7
<0d>        should be:   7
<0d>Arr_2_Glob[8][7]:    1010
<0d>        should be:   Number_Of_Runs + 10
<0d>Ptr_Glob->
<0d>  Ptr_Comp:          218712
<0d>        should be:   (implementation-dependent)
<0d>  Discr:             0
<0d>        should be:   0
<0d>  Enum_Comp:         2
<0d>        should be:   2
<0d>  Int_Comp:          17
<0d>        should be:   17
<0d>  Str_Comp:          DHRYSTONE PROGRAM, SOME STRING
<0d>        should be:   DHRYSTONE PROGRAM, SOME STRING
<0d>Next_Ptr_Glob->
<0d>  Ptr_Comp:          218712
<0d>        should be:   (implementation-dependent), same as above
<0d>  Discr:             0
<0d>        should be:   0
<0d>  Enum_Comp:         1
<0d>        should be:   1
<0d>  Int_Comp:          18
<0d>        should be:   18
<0d>  Str_Comp:          DHRYSTONE PROGRAM, SOME STRING
<0d>        should be:   DHRYSTONE PROGRAM, SOME STRING
<0d>Int_1_Loc:           5
<0d>        should be:   5
<0d>Int_2_Loc:           13
```

```
<0d>        should be:   13
<0d>Int_3_Loc:           7
<0d>        should be:   7
<0d>Enum_Loc:            1
<0d>        should be:   1
<0d>Str_1_Loc:           DHRYSTONE PROGRAM, 1'ST STRING
<0d>        should be:   DHRYSTONE PROGRAM, 1'ST STRING
<0d>Str_2_Loc:           DHRYSTONE PROGRAM, 2'ND STRING
<0d>        should be:   DHRYSTONE PROGRAM, 2'ND STRING
<0d>
<0d>
<0d>_________________________
<0d>Marked software performance monitor results:
<0d>
<0d>Instructions Executed = 305922
<0d>Cycle Count (CCNT) = 384468
<0d>Loads = 51169
<0d>Average cycles per instruction = 1.256752
<0d>
<0d>End of marked software performance monitor results.
<0d>_________________________
<0d>Main Complete, stopping simulation ...
<0d><04>0x04 End Of Simulation message received by TrickBox

Info: /OSCI/SystemC: Simulation stopped by user.
Ending simulation...
AXI logger complete:           pl310_M1_logger
Total cycles:               451266 cycles
Non-reset cycles:           451260 cycles
Number of reads:            3064
Max read latency:           60 cycles happened at cycle number 53245
Ave read throughput:        0 bytes per cycle
Read channel utilization:   1.08385%
Read bus utilization:       11.6613%
Number of writes:           1557
Max write latency:          1681 cycles happened at cycle number 53354
Ave write throughput:       2 bytes per cycle
Write channel utilization: 0.672118%
Write bus utilization:      11.6099%
AXI logger complete:           pl310_M0_logger
Total cycles:               451266 cycles
Non-reset cycles:           451260 cycles
Number of reads:            3074
Max read latency:           54 cycles happened at cycle number 53259
Ave read throughput:        0 bytes per cycle
Read channel utilization:   1.1069%
Read bus utilization:       11.9605%
Number of writes:           1741
Max write latency:          1611 cycles happened at cycle number 53460
Ave write throughput:       2 bytes per cycle
Write channel utilization: 0.683198%
Write bus utilization:      11.6281%
AXI logger complete:           M1_logger
Total cycles:               451266 cycles
Non-reset cycles:           451260 cycles
Number of reads:            3067
Max read latency:           64 cycles happened at cycle number 53243
Ave read throughput:        0 bytes per cycle
Read channel utilization:   1.08585%
Read bus utilization:       12.2402%
Number of writes:           1557
Max write latency:          1684 cycles happened at cycle number 53349
Ave write throughput:       2 bytes per cycle
Write channel utilization: 1.08629%
Write bus utilization:      11.6368%
AXI logger complete:           M0_logger
Total cycles:               451266 cycles
Non-reset cycles:           451260 cycles
Number of reads:            3075
Max read latency:           58 cycles happened at cycle number 53257
Ave read throughput:        0 bytes per cycle
Read channel utilization:   1.10934%
Read bus utilization:       12.2572%
Number of writes:           1741
Max write latency:          1611 cycles happened at cycle number 53458
Ave write throughput:       2 bytes per cycle
Write channel utilization: 0.701591%
Write bus utilization:      11.6447%
AXI logger complete:           trickbox_logger
Total cycles:               451266 cycles
Non-reset cycles:           451263 cycles
Number of reads:            0
Max read latency:           0 cycles happened at cycle number 0
Read channel utilization:   0%
```

```
Read bus utilization:      0%
Number of writes:          1927
Max write latency:         2 cycles happened at cycle number 55039
Ave write throughput:      2 bytes per cycle
Write channel utilization: 0%
Write bus utilization:     0.854047%
AXI logger complete:           ram_logger
Total cycles:              451266 cycles
Non-reset cycles:          451263 cycles
Number of reads:           6137
Max read latency:          6 cycles happened at cycle number 2776
Ave read throughput:       4 bytes per cycle
Read channel utilization:  2.18321%
Read bus utilization:      5.2065%
Number of writes:          1371
Max write latency:         5 cycles happened at cycle number 3040
Ave write throughput:      23 bytes per cycle
Write channel utilization: 0.866679%
Write bus utilization:     1.47431%

$
```

The simulation results show that the application ran successfully, and standard output includes cycle count and performance data for each bus.

Next steps

- View waveforms data. The waveform file is output to `systems/cyclemodel/system/arm_cm_NIC400.vcd`.
- Evaluate performance data. See *2.4 Setting performance monitoring behavior* on page 2-22, and *2.6 AXI bus logging* on page 2-25.
- Develop and run a different application. See *2.5 Developing and running custom applications* on page 2-24.

2.4 Setting performance monitoring behavior

The Arm Cortex-A5 DesignStart Cycle Model FVP includes the source code for the Dhrystone application.

To enable the capture of PMU events in the Cortex-A5, the Dhrystone application provided in the Cortex-A5 Cycle Model FVP uses the function calls `start_marker()` and `stop_marker()`.

These functions are defined in `software/dhrystone/dhry_1.c`, and declared in `software/dhrystone/pmu-and-marker.h`:

- `start_marker()` enables the two PMU counters in the Cortex-A5 processor, and configures them to count the PMU events:
  - `Instructions Executed`
  - `Loads`

  The clock counter is also enabled.
- `stop_marker()` stops the PMU counters and prints their values.

See the *Cortex®-A5 Technical Reference Manual* (DDI0433) for details about PMU events.

Prerequisites

- Before recompiling the Dhrystone application, refer to *1.3 Prerequisites* on page 1-12 for supported compiler versions.
- Make a backup copy of the existing Dhrystone application before making changes to it.

Set markers for the software region of interest

The Dhrystone application provided in the Cortex-A5 Cycle Model FVP enables PMU events for the duration of the `main()` function in `dhry_1.c`. To mark alternate sections of code for PMU monitoring, move the calls to `start_marker()` and `stop_marker()`.

`start_marker()` resets the value of the counters. This allows restarting the same counters during a single application run.

Specify events of interest

Hardware events that are supported by the Cortex-A5 processor are uniquely identified by their Event Number as defined in the *Cortex®-A5 Technical Reference Manual* (DDI0433).

The example Dhrystone application in the Cortex-A5 Cycle Model FVP captures the following PMU events:

- Number of completed instructions (Instructions retired)
- Number of instructions loaded (Loads)

To set the counters to monitor different events, update the second parameters of the `pmn_config()` calls in `software/dhrystone/pmu-and-marker.c`. Specify the Event Numbers of the events of interest.

Recompile the application

When your changes are complete, recompile the Dhrystone application:

1. Set up Arm Compiler 6 (armclang). The following example command uses the bash shell:

   ```
   > export PATH=installation directory/ARMCompiler6.12/bin:$PATH
   ```

2. Clean the build area:

   ```
   > make cleanall
   ```

3. Build the application:

   ```
   > make all
   ```

Note

A warning similar to `'extern' variable has an initializer` may occur. This is expected and does not impact the application.

The `.axf` and `.hex` files required to simulate the Dhrystone application are created.

2.5 Developing and running custom applications

You can simulate applications that you create on the Fast Model and Cycle Model representations of the Arm Cortex-A5 DesignStart Cycle Model FVP systems.

Prerequisites

Before developing new applications, refer to *1.3 Prerequisites* on page 1-12 for supported compiler versions.

Requirement for simulation exit

The Dhrystone application, included by default with the DesignStart Cycle Model FVP system, includes the code required to exit out of the simulation.

If you are writing your own application, your code must also end the system simulation. This is done by sending the sequence `sendchar(4)`to the TrickBox. For example:

```
void simulation_exit()
{
    sendchar(4);
}
```

Application format and location requirements

Create a new directory for the application under the directory `Cortex-A5_DesignStart_CM_FVP_1.0/software`. The name of the directory and the name of the `.axf` file must match; for example, `software/my_app/my_app.axf`.

Applications for the Fast Models FVP system:

- Must be compiled as a `.axf` file. This is the output of the Arm Compiler 6 compilation process.
- Place the `.axf` file in the `software/my_app` directory.

Applications for the Cycle Models FVP system must be in the form of hexadecimal files with the names `image_mem64_lo.hex` and `image_mem64_hi.hex`. To create these files:

1. Compile the application. This generates the `.axf` file.
2. Change to `software/common`.
3. Run the `create_dat_file.sh` script on the `.axf` file:

   ```
   ./create_dat_file.sh path_to_axf_file
   ```

4. The `create_dat_file.sh` script places the `.hex` files in this directory automatically. If you are using a different method, place the `.hex` files in the `software/my_app` directory.

Simulating with a custom application

To enable simulation with an application you write:

1. Ensure the application files meet the requirements for format and location described in this section.
2. Simulate the application using the `run.sh` script with the option `-a app_name`. For example:

   ```
   ./run.sh -a my_app
   ```

2.6 AXI bus logging

AXI bus logging is enabled on the Arm Cortex-A5 DesignStart Cycle Model FVP.

Logged connections

AXI bus logging is supported for all Cycle Model FVP AXI channels:

- M0_logger - Cortex-A5 Master port 0 to PL310 Slave port 0
- M1_logger - Cortex-A5 Master port 1 to PL310 Slave port 1
- pl310_M0_logger - PL310 Master port 0 to NIC400 Slave port 2
- pl310_M1_logger - PL310 Master port 1 to NIC-400 Slave port 3
- ram_logger - NIC-400 to BP140 Memory
- trickbox_logger - NIC-400 to BP140 Trickbox

Logging output

There are two kinds of logging output: the transaction log files, and the summary of performance measures sent to standard output.

During system simulation for performance analysis, the Cycle Model FVP writes transaction data to the AXI bus logs in the directory `Cortex-A5_DesignStart_CM_FVP_1.0/systems/cyclemodel/system`. Data in the logs is overwritten at each simulation.

When simulate ends, the Cycle Model FVP writes a summary of the AXI bus transactions to standard output. The summary includes:

- Total number of cycles
- Total number of non-reset cycles
- Total number of Reads
- Maximum read latency
- Average read throughput
- Read channel utilization as a percentage
- Read bus utilization as a percentage
- Total number of Writes
- Maximum write latency
- Average Write throughput as a percentage
- Write channel utilization as a percentage
- Write bus utilization as a percentage

Chapter 3
DesignStart Cycle Model FVP system details

This section provides configuration details for the components included in the Arm Cortex-A5 DesignStart Cycle Model FVP.

It contains the following sections:

3.1 Configuration reference

For reference, this section describes the specifications of each component of the DesignStart Cycle Model FVP.

Cortex®-A5 processor configuration

The Cortex-A5 processor Cycle Model is based on r0p1 RLT. It is configured with:

- Four cores
- 64 KB instruction cache
- 64 KB data cache
- Floating-point unit (FPU)
- Neon technology
- Advanced Configuration and Power Interface (ACPI)
- Two AXI Master ports
- Number of interrupts set to 224

PL310 component configuration

The PL310 Cycle Model is based on r3p3 RTL.

NIC-400 configuration

The CoreLink NIC-400 Network Interconnect Cycle Model is based on r1p2 RTL.

3.2 Memory map

The Cortex-A5 DesignStart Cycle Model FVP memory map implements a subset of the RTL memory map.

Table 3-1 Memory map

Base address	Size	Component
0x0000_0000	64KB	SRAM0 for ROM
0x1C00_0000	64KB	A5 Peripherals
0x1A20_0000	64KB	UART0