

# AMBA<sup>®</sup> Adaptive Traffic Profiles

## Specification



# AMBA Adaptive Traffic Profiles Specification

Copyright © 2017, 2018, 2019 Arm Limited or its affiliates. All rights reserved.

## Release Information

The following changes have been made to this specification.

Change history			
Date	Issue	Confidentiality	Change
15 March 2019	A	Non-Confidential	First release

## Proprietary Notice

This document is NON-CONFIDENTIAL and any use by you is subject to the terms of this notice and the Arm AMBA Specification Licence set out below.

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT.

For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © [2017, 2018, 2019] Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.  
110 Fulbourn Road, Cambridge, England CB1 9NJ.  
LES-PRE-21451

## ARM AMBA SPECIFICATION LICENCE

THIS END USER LICENCE AGREEMENT (“LICENCE”) IS A LEGAL AGREEMENT BETWEEN YOU (EITHER A SINGLE INDIVIDUAL, OR SINGLE LEGAL ENTITY) AND ARM LIMITED (“ARM”) FOR THE USE OF THE RELEVANT AMBA SPECIFICATION ACCOMPANYING THIS LICENCE. ARM IS ONLY WILLING TO LICENSE THE

RELEVANT AMBA SPECIFICATION TO YOU ON CONDITION THAT YOU ACCEPT ALL OF THE TERMS IN THIS LICENCE. BY CLICKING “I AGREE” OR OTHERWISE USING OR COPYING THE RELEVANT AMBA SPECIFICATION YOU INDICATE THAT YOU AGREE TO BE BOUND BY ALL THE TERMS OF THIS LICENCE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENCE, ARM IS UNWILLING TO LICENSE THE RELEVANT AMBA SPECIFICATION TO YOU AND YOU MAY NOT USE OR COPY THE RELEVANT AMBA SPECIFICATION AND YOU SHOULD PROMPTLY RETURN THE RELEVANT AMBA SPECIFICATION TO ARM.

“LICENSEE” means You and your Subsidiaries.

“Subsidiary” means, if You are a single entity, any company the majority of whose voting shares is now or hereafter owned or controlled, directly or indirectly, by You. A company shall be a Subsidiary only for the period during which such control exists.

1. Subject to the provisions of Clauses 2, 3 and 4, Arm hereby grants to LICENSEE a perpetual, non-exclusive, non-transferable, royalty free, worldwide licence to: (i) use and copy the relevant AMBA Specification for the purpose of developing and having developed products that comply with the relevant AMBA Specification; (ii) manufacture and have manufactured products which either: (a) have been created by or for LICENSEE under the licence granted in Clause 1(i); or (b) incorporate a product(s) which has been created by a third party(s) under a licence granted by Arm in Clause 1(i) of such third party’s Arm AMBA Specification Licence; and (iii) offer to sell, sell, supply or otherwise distribute products which have either been (a) created by or for LICENSEE under the licence granted in Clause 1(i); or (b) manufactured by or for LICENSEE under the licence granted in Clause 1(ii).
2. LICENSEE hereby agrees that the licence granted in Clause 1 is subject to the following restrictions: (i) where a product created under Clause 1(i) is an integrated circuit which includes a CPU then either: (a) such CPU shall only be manufactured under licence from Arm; or (b) such CPU is neither substantially compliant with nor marketed as being compliant with the Arm instruction sets licensed by Arm from time to time; (ii) the licences granted in Clause 1(iii) shall not extend to any portion or function of a product that is not itself compliant with part of the relevant AMBA Specification; and (iii) no right is granted to LICENSEE to sublicense the rights granted to LICENSEE under this Agreement.
3. Except as specifically licensed in accordance with Clause 1, LICENSEE acquires no right, title or interest in any Arm technology or any intellectual property embodied therein. In no event shall the licences granted in accordance with Clause 1 be construed as granting LICENSEE, expressly or by implication, estoppel or otherwise, a licence to use any Arm technology except the relevant AMBA Specification.
4. THE RELEVANT AMBA SPECIFICATION IS PROVIDED “AS IS” WITH NO REPRESENTATION OR WARRANTIES EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF SATISFACTORY QUALITY, MERCHANTABILITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE, OR THAT ANY USE OR IMPLEMENTATION OF SUCH ARM TECHNOLOGY WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER INTELLECTUAL PROPERTY RIGHTS.
5. NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS AGREEMENT, TO THE FULLEST EXTENT PERMITTED BY LAW, THE MAXIMUM LIABILITY OF ARM IN AGGREGATE FOR ALL CLAIMS MADE AGAINST ARM, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS AGREEMENT (INCLUDING WITHOUT LIMITATION (I) LICENSEE’S USE OF THE ARM TECHNOLOGY; AND (II) THE IMPLEMENTATION OF THE ARM TECHNOLOGY IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS AGREEMENT) SHALL NOT EXCEED THE FEES PAID (IF ANY) BY LICENSEE TO ARM UNDER THIS AGREEMENT. THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.
6. No licence, express, implied or otherwise, is granted to LICENSEE, under the provisions of Clause 1, to use the Arm tradename, or AMBA trademark in connection with the relevant AMBA Specification or any products based thereon. Nothing in Clause 1 shall be construed as authority for LICENSEE to make any representations on behalf of Arm in respect of the relevant AMBA Specification.
7. This Licence shall remain in force until terminated by you or by Arm. Without prejudice to any of its other rights if LICENSEE is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to You. You may terminate this Licence at any time. Upon expiry or termination of this Licence by You or by Arm LICENSEE shall stop using the relevant AMBA Specification and destroy all copies of the relevant AMBA Specification in your possession together with all documentation and related materials. Upon expiry or termination of this Licence, the provisions of clauses 6 and 7 shall survive.

8. The validity, construction and performance of this Agreement shall be governed by English Law.

**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

## AMBA Adaptive Traffic Profiles Specification

### Preface

About this document .....	x
Intended audience .....	x
Typographic conventions .....	x
Timing diagram conventions .....	x
Additional reading .....	xi
Feedback on this specification .....	xi
Feedback .....	xi

### Chapter 1

#### Introduction

1.1	About the AMBA Adaptive Traffic Profiles .....	1-14
1.2	Use cases .....	1-15
1.2.1	Representative component behavior during simulation .....	1-15
1.2.2	Defined transaction sequence during simulation .....	1-15
1.2.3	Dynamic interface specification for system construction .....	1-15
1.2.4	Checking interface dynamic characteristics .....	1-15
1.2.5	Capturing interface dynamic characteristics .....	1-16
1.2.6	Generating traffic in different environments .....	1-16
1.3	Hierarchy .....	1-17

### Chapter 2

#### Signal values

2.1	General description of signal values .....	2-20
2.2	Components of traffic signals .....	2-21
2.2.1	Read or write transaction .....	2-21
2.2.2	Address .....	2-21
2.2.3	Transaction identifier .....	2-23
2.2.4	Data .....	2-24

<b>Chapter 3</b>	<b>Timing Parameters</b>	
3.1	Primary and secondary timing .....	3-26
3.1.1	Read transaction .....	3-27
3.1.2	Write transaction .....	3-28
3.1.3	Snoop transactions .....	3-29
3.2	Complex transactions .....	3-31
<b>Chapter 4</b>	<b>FIFO timing model</b>	
4.1	Timing model .....	4-36
4.2	Generator Specification .....	4-37
4.2.1	Generator description .....	4-37
4.2.2	Formal Generator description .....	4-37
4.3	FIFO timing points .....	4-41
4.3.1	Normal operation .....	4-41
4.3.2	Reset behavior .....	4-42
4.4	Checker behavior .....	4-43
4.5	Generator characteristics .....	4-44
4.5.1	Initial Peak Rate .....	4-44
4.5.2	Constrained Peak Rate .....	4-44
4.5.3	Average Rate .....	4-44
4.6	Linked traffic profiles .....	4-45
<b>Chapter 5</b>	<b>Event coordination</b>	
5.1	Synchronization between traffic profiles .....	5-48
5.2	Concurrent traffic profile behavior .....	5-49
<b>Chapter 6</b>	<b>Slave traffic profiles</b>	
6.1	Slave traffic profiles .....	6-52
<b>Appendix A</b>	<b>Default signal values</b>	
<b>Appendix B</b>	<b>AXI signal identifiers</b>	
<b>Appendix C</b>	<b>Example FIFO model behaviors</b>	
C.1	FIFO Model .....	C-60
C.2	Display .....	C-61
C.3	CPU .....	C-62
C.4	GPU .....	C-63
C.5	Network interface .....	C-64
<b>Appendix D</b>	<b>Example Waveforms</b>	
D.1	Basic Read with FIFO empty .....	D-66
D.1.1	Configuration .....	D-66
D.1.2	Timing diagram .....	D-66
D.2	Basic Read FIFO full .....	D-67
D.2.1	Configuration .....	D-67
D.2.2	Timing diagram .....	D-67
D.3	Basic Write with the FIFO full .....	D-69
D.3.1	Configuration .....	D-69
D.3.2	Timing diagram .....	D-69
D.4	Basic Write with the FIFO empty .....	D-71
D.4.1	Configuration .....	D-71
D.4.2	Timing diagram .....	D-71
D.5	Read with FIFO underflow .....	D-74
D.5.1	Configuration .....	D-74
D.5.2	Timing diagram .....	D-74
D.6	Write with FIFO overflow .....	D-75

	D.6.1	Configuration .....	D-75
	D.6.2	Timing diagram .....	D-75
D.7		Read with delayed slave .....	D-76
	D.7.1	Configuration .....	D-76
	D.7.2	Timing diagram .....	D-76
D.8		Read gated by Outstanding Transaction limit .....	D-78
	D.8.1	Configuration .....	D-78
	D.8.2	Timing diagram .....	D-78
D.9		Write influenced by slave profile .....	D-80
	D.9.1	Configuration .....	D-80
	D.9.2	Timing diagram .....	D-80

## Appendix E

## Revisions





# Preface

This document describes the AMBA Adaptive Traffic Profiles.

This chapter contains the following sections:

- *About this document on page x*
- *Intended audience on page x.*
- *Typographic conventions on page x.*
- *Additional reading on page xi.*
- *Feedback on this specification on page xi.*

About this document

Intended audience

This specification is written for hardware and software engineers who want to design or debug systems and modules that are compatible with AMBA Adaptive Traffic Profiles .

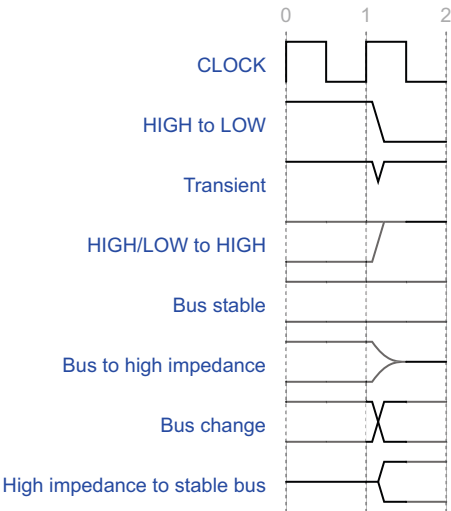
Typographic conventions

Convention	Meaning
<i>italic</i>	Introduces special terminology, denotes cross-references, and citations.
<b>bold</b>	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Used for assembler syntax descriptions, pseudocode, and source code examples. Also used for expersions and equations.
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.

Timing diagram conventions

The figure[Key to timing diagram conventions](#) explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Timing diagrams sometimes show single-bit signals as HIGH and LOW at the same time and they look similar to the bus change that the Key to timing diagram conventions figure shows. If a timing diagram shows a single-bit signal in this way then its value does not affect the accompanying description.

## Additional reading

This section lists relevant documents published by third parties:

*AMBA AXI and ACE Protocol Specification* ARM IHI 0022F

## Feedback on this specification

If you have comments on the content of this specification, send e-mail to [errata@arm.com](mailto:errata@arm.com). Give:

- The title *AMBA Adaptive Traffic Profiles Specification*.
- The number, ARM IHI 0082A.
- The page number(s) that your comments apply.
- A concise explanation of your comments.

## Feedback

Arm welcomes feedback on its documentation.



# Chapter 1

## Introduction

This chapter introduces the AMBA Adaptive Traffic Profiles specification:

- *About the AMBA Adaptive Traffic Profiles* on page 1-14.
- *Use cases* on page 1-15.
- *Hierarchy* on page 1-17.

## 1.1 About the AMBA Adaptive Traffic Profiles

A traffic profile is a definition of the transaction characteristics of an interface. The AMBA Adaptive Traffic Profile (ATP) is a specification of the dynamic characteristics of an interface. AMBA ATP includes information on the types of transactions and the timing characteristics of those transactions.

The primary use of traffic profiles is describing the behavior of master components in a system. Unless the text specifically references the slave component viewpoint, this document describes traffic profiles from the master component point of view. Traffic profiles for slave components are covered in [Chapter 6 Slave traffic profiles](#).

Traffic profiles can be used with various interface protocols. This specification focuses on the use of traffic profiles with an AXI interface.

## 1.2 Use cases

There are many different uses that can be made of traffic profiles. This section illustrates several uses, but it is not exhaustive.

### 1.2.1 Representative component behavior during simulation

Traffic profiles can be used during system simulation to represent the behavior of a component. In this context, the term *traffic profile* is used to describe a simple transaction stimulus generator in the simulation. The simulation uses a traffic profile definition to determine when a particular transaction should be issued.

When simulating a reasonably complex system, it is desirable to observe the interaction between various components. A simple simulation, with only one or two components generating transactions, is not sufficiently close to the final system behavior to obtain meaningful results. However, setting up a simulation with many components, each with highly accurate models, is a difficult task. It can result in long simulation times before the system settles to a state where meaningful results can be obtained.

Traffic profiles can be used to replace RTL or highly accurate models for various components in the simulation. Typically, those traffic profiles that provide background transactions make the simulation result more realistic without adding much complexity.

In a simulation environment there are two key advantages to using a traffic profile:

- A traffic profile-based transaction generator can issue transactions in a similar fashion to a real component and is able to react to the latency of transaction responses.
- A traffic profile is simple and predictable, while being dynamic. In a complex environment, there are many components that are interacting and the behavior of the entire system is difficult to discern. The components being represented by traffic profiles can be easily understood, rather than adding further complications as might happen with a precise model.

### 1.2.2 Defined transaction sequence during simulation

The most common use of a traffic profile is providing a concise abstract definition of the general behavior of an interface. It is also possible for a traffic profile to be used in a more detailed fashion. Traffic profiles can describe a sequence of fully-defined transactions, including address and data values that might be used to program the setup of a device.

### 1.2.3 Dynamic interface specification for system construction

Traffic profiles can be used to describe the required dynamic behavior of an interface. This description can be used during system construction to influence and check the system configuration at the point that components, such as interconnects and memory controller, are being configured.

For example, a set of traffic profiles can be used to verify that the sum of the bandwidths through a particular point in the interconnect does not exceed the capabilities of that interconnect.

A more complex example is using traffic profiles to determine and configure interconnect parameters, such as data bus width or number of outstanding transactions supported on an interface.

An advantage of using traffic profiles for system construction is that the same traffic profiles can be reused during dynamic simulation to confirm that the requirements have been met.

### 1.2.4 Checking interface dynamic characteristics

A traffic profile can be used by a monitor in a simulation environment to specify the traffic characteristics that should be checked. This can be done in many ways, for example checking that:

- The bandwidth that is obtained is at least as much as specified in the traffic profile.
- The bandwidth on an interface never exceeds a limit that is specified in the traffic profile.
- The latency observed never exceeds a value that is specified in the traffic profile.

### 1.2.5 Capturing interface dynamic characteristics

It is possible to observe the transactions on an interface and derive a traffic profile definition that could be used in a different environment.

Capturing dynamic interface characteristics is a complex process, since there is no formally defined conversion function to translate a sequence of observed transactions to a traffic profile. This conversion is inherently lossy and it is usually not possible to convert from a sequence of transactions to a traffic profile and then precisely regenerate the same transactions.

### 1.2.6 Generating traffic in different environments

An important aspect of traffic profiles is that they allow a defined traffic profile to be replayed in multiple different environments. The portability of traffic profiles ensures that system performance simulation results obtained in one environment can be precisely correlated with results that are obtained in a different environment.

For example, the same stimulus could be generated in all of the following environments:

- Transaction level modeling simulation.
- RTL-based simulation.
- FPGA prototyping.
- Final SoC.

The concise nature of a traffic profile definition means that a low-gate-count hardware component can be used as a traffic generator. This allows a low-gate-count implementation to be used for high-speed hardware emulation or FPGA prototyping. It is also sufficiently low-gate-count that it could be included in a final SoC design for testing or debug purposes.



## 1.3 Hierarchy

A traffic profile is defined as follows:

- A fixed set of control signal values.
- An address sequence.
- A set of *within transaction* timing parameters (intra-transaction).
- A set of parameters to define the *between transaction* timing (inter-transaction).
- Optionally, a single input start event.
- Optionally, a single output event.

Traffic profiles can be combined as follows to represent the behavior of a single agent:

### In a concurrent manner:

Representing different types of transactions from a single component. For example, one traffic profile could be used for read traffic and another is used for write traffic. Alternatively, one traffic profile can be used to represent cacheable memory traffic and another can be used to represent peripheral device accesses.

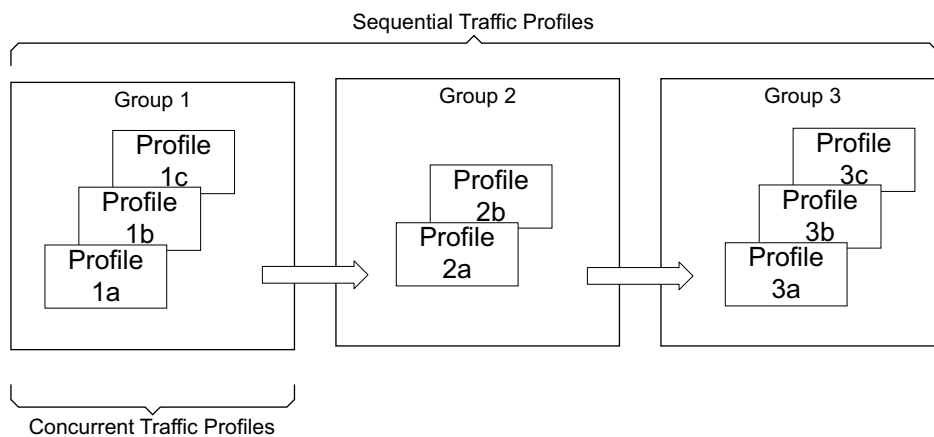
### In a sequential manner:

Where one traffic profile completes before the next in the sequence begins. This can be used to represent the modal behavior of a component. For example, a network interface component accessing header information followed by accesses to payload. Alternatively, a CPU accessing memory with a cold cache, followed by accesses patterns typical of a warm cache.

Concurrent traffic profiles are explained in more detail in [Concurrent traffic profile behavior on page 5-49](#), which describes the expected behavior when concurrent traffic profiles have conflicting requirements.

Sequential traffic profiles are also explained in more detail in [Synchronization between traffic profiles on page 5-48](#), which describes the relationship between the end of one step in the sequence and the beginning of the next.

Figure 1-1 shows the difference between sequential and concurrent traffic profiles.



**Figure 1-1 Concurrent and sequential traffic profiles**

A set of concurrent traffic profiles is called a *Traffic Profile Group*.

A set of sequential traffic profiles is called a *Traffic Profile Sequence*. A Traffic Profile Sequence can consist of a series of individual traffic profiles, a series of traffic profile groups, or a combination of both.

A system simulation, with multiple components, each executing traffic profiles, traffic profile groups, or traffic profile sequences is called a *Scenario* or *System Scenario*.

This specification includes a mechanism to link together traffic profiles, so that progress of one traffic profile can affect the progress of another traffic profile. See [Linked traffic profiles on page 4-45](#) for more details.

## Chapter 2

# Signal values

This chapter describes the signal component of traffic profiles:

- *General description of signal values on page 2-20.*
- *Components of traffic signals on page 2-21.*

## 2.1 General description of signal values

A traffic profile is a set of definitions that includes both:

- Signal values.
- Timing parameters.

This section describes the signal values. Timing parameters, which are handshake signals that control the timing of the transactions, are covered in [Chapter 3 Timing Parameters](#).

It is expected that most of the signals that are associated with a traffic profile have a fixed value that remains constant for the duration of the traffic profile. If the fixed value of a signal is different than the defined default value for that signal, then it must be specified in the traffic profile. If the fixed value is the same as the default value, then it is not required to be specified in the traffic profile, but it is permitted.

For AXI and ACE, see *Default Signal Values* of the *AMBA AXI and ACE Protocol Specification*, for details on the signals associated with the interface and the default value of these signals. A summary of the default values is provided in [Appendix A Default signal values](#).

[Appendix B AXI signal identifiers](#) of this specification gives the enumerations for selected AXI signals that can be used as a standard human-readable set of values for particular control signals.

Information must be provided in the traffic profiles to allow all of the following to be generated:

- Read or Write Transaction.
- Address: **AWADDR** or **ARADDR**.
- Transaction Identifier: **AWID**, **BID**, **ARID**, or **RID**.
- Data: **WDATA** or **RDATA**.

## 2.2 Components of traffic signals

The following must be specified in the traffic profile:

- [Read or write transaction](#).
- [Address](#).
- [Transaction identifier on page 2-23](#).
- [Data on page 2-24](#).

### 2.2.1 Read or write transaction

The direction of the transaction is either:

<b>read</b>	A read transaction, which has all of the data movement towards the master agent. In AXI, the AR and R channels are used.
<b>write</b>	A write transaction, which has all of the data movement away from the master agent. In AXI, the AW, W and B channels are used.

### 2.2.2 Address

The following mechanisms are supported for the generation of address values that are associated with a traffic profile:

<b>sequential</b>	A sequential range of address values is used. A Base and Range is specified. The first transaction uses Base. The next transaction uses an address value that is incremented by the size of the transaction. When a transaction that includes the address $\text{Base} + \text{Range} - 1$ has been used, the next transaction uses the original Base.
<b>twodim</b>	<p>A two-dimensional address pattern is used. The values that are required are:</p> <p><b>Base</b> The address pattern begins at Base and successive transactions use an address that is incremented by the transaction size.</p> <p><b>XRange</b> After a transaction that uses address that is specified by:  <math>\text{Base} + (\text{N} * \text{Stride}) + \text{XRange} - 1</math>  the next transaction uses address that is specified by:  <math>\text{Base} + (\text{N} + 1) * \text{Stride}</math>  This pattern continues until YRange is reached.</p> <p><b>Stride</b> Determines the offset of each new row.</p> <p><b>YRange</b> Indicates the size of the address set being used. If the next calculated address would be larger than or equal to <math>\text{Base} + \text{YRange}</math>, then the address wraps and Base is used.</p> <p><a href="#">Figure 2-1 on page 2-22</a> illustrates the use of these parameters.</p>
<b>random</b>	<p>A random address value is used. A Base and Range is specified. The highest address that can be included in a transaction is:</p> $\text{Base} + \text{Range} - 1$
<b>file</b>	The address value to be used for each transaction is read from a file. The Filename is specified. An optional Base is specified. The values read from the file are added as an offset to the Base. If the Base is not specified it defaults to zero, making the entries in the file absolute addresses.

Use of the random mechanism might not be supported by all traffic profile environments. If the random mechanism is specified, but it is not supported, it is replaced by the sequential mechanism.

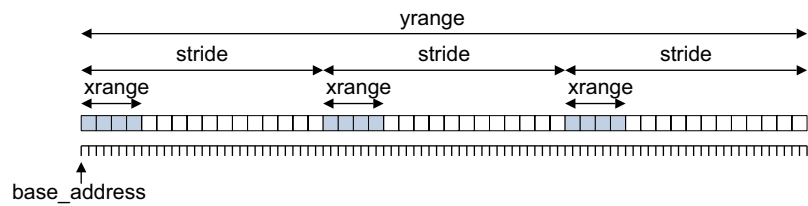


Figure 2-1 Parameters associated with the twodim mechanism

Figure 2-2 is an example of memory accesses using the twodim mechanism that has:

- XRange of 0xC.
- Stride of 0x14.
- Base of 0x2000.
- YRange of 0x3C.

The transaction size is 4B and letters a to i indicate the order of the transaction issue.

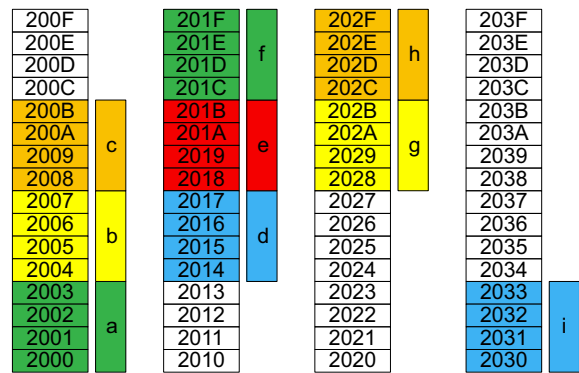


Figure 2-2 Twodim parameters example

The XRange parameter must be an integer multiple of the transaction size. In the example that is shown in Figure 2-2, the YRange could be any value between 0x34 and 0x3C to achieve the same address patterns.

Table 2-1 summarizes the parameters that must be supplied for each type of address pattern.

**Table 2-1 Address pattern parameters**

Mechanism	Required Values
sequential	Base Range
twodim	Base XRange Stride YRange
random	Base Range
file	Base Filename

### 2.2.3 Transaction identifier

There are four supported mechanisms for the generation of transaction ID values. The mechanisms are one of the following:

- fixed** A single ID value is used for all transactions. The value to be used is specified.
- cycle** A range of ID values is cycled through. A lower and upper value is specified. The first transaction uses the lower value. The ID value is incremented for each successive transaction. After the upper value has been used, the next transaction uses the lower value.
- unique** A transaction does not use an ID value that is currently in use by an outstanding transaction. This uses the same approach as the cycle mechanism. If an ID value is in use, then the next available ID value that is not currently in use is chosen. It is required that the number of ID values available between the lower and upper values is at least as large as the maximum number of outstanding transactions, see [Generator Specification on page 4-37](#).  
  
This ensures that it is never necessary to reuse an ID value that is already in use. If a maximum number of outstanding transactions are not specified, it is defined to be the range of ID values available.
- file** The ID value to be used for each transaction is read from a file. The Filename is specified.

Table 2-2 summarizes the parameters to be supplied for each ID mechanism.

**Table 2-2 ID mechanism parameters**

Mechanism	Required Values
fixed	Value
cycle	LowerValue UpperValue
unique	LowerValue UpperValue
file	Filename

2.2.4 Data

The uses for the generation of data values that are associated with a traffic profile depend on the source of the data. For traffic profiles where data is an output, the value provided is used as the data value. For traffic profiles where data is an input, the value provided is used to check the data value observed.

The supported mechanisms for data values are:

- fixed**

A single data value is used for all transactions. The value to be used is specified.
- unknown**

A data output is driven as DON'T CARE. A data input is unchecked.
- cycle**

A range of data values is cycled through. A lower and upper value is specified. The first transaction uses the lower value. The data value is incremented by one for each successive transaction. After the upper value has been used, the next transaction uses the lower value.

This approach can be useful in simulation environments where the actual data value is not important and the use of a simple sequence can simplify the tracking of transactions and debugging of simulations.
- random**

A random data value is used.
- file**

The data value to be used for each transaction is read from a file. The Filename is specified.

Use of the random mechanism might not be supported by all traffic profile environments. If the random mechanism is specified, but it is not supported, it is replaced by use of the cycle mechanism. The mechanism starts with a zero as a lower value and uses an upper value equivalent to the largest data value that can be represented by the transaction size. [Table 2-3](#) shows the cycle mechanism parameters.

Table 2-3 Cycle mechanism parameters

Mechanism	Required Values
fixed	Value
unknown	-
cycle	LowerValue UpperValue
random	-
file	Filename



# Chapter 3

## Timing Parameters

This chapter describes the relationships of the signals that are used to control AMBA traffic:

- *Primary and secondary timing on page 3-26.*
- *Complex transactions on page 3-31.*

### 3.1 Primary and secondary timing

There are two types of timing parameters: primary and secondary.

Primary timing parameters are those parameters that are typically the most important in defining the behavior of a system. For example, two of the primary timing parameters include:

- The time between issuing two transactions, which effectively defines the bandwidth of a traffic profile.
- The initial read latency of a read transaction, which is often a key measurement of system performance.

Secondary timing parameters are those timing parameters that are typically less important for understanding system performance. An example of a secondary timing parameter is the time between data beats of a transaction.

Adaptive traffic profiles provide the ability to dynamically control primary timing parameters. Secondary timing parameters can only be statically controlled and are fixed for a given traffic profile. All secondary timing parameters have a default value and only need to be specified in the traffic profile if different from the default value.

This section of the document defines each of the timing parameters supported. [Chapter 4 FIFO timing model](#) describes how the primary timing parameters can be controlled.

Write transactions use the concept of a transaction start time, which is used because either the address or the write data can act as the start of the transaction.

The default transaction start time is defined as the earlier of either:

- The rising clock edge before address valid is asserted.
- The rising clock edge before write data valid is asserted.

If using the default start time, then one or both of the parameters: *transaction start to write address valid* or *transaction start to write data valid*, will have a value of 1. Alternatively, an earlier transaction start time can be defined, so that both of these parameters are greater than 1.

[Table 3-1 on page 3-27](#) gives the definitions of the timing parameters and their characteristics. The table contains the following information:

<b>Name</b>	Short form of the timing parameter. This is the form that is used within a traffic profile to define the parameter.
<b>Description</b>	Description of the timing parameter.
<b>Between or Within</b>	Indicates if the timing parameter defines a parameter within a single transaction or a parameter between two consecutive transactions.
<b>Primary or Secondary</b>	Indicates if the timing parameter is a primary parameter, which controls a key aspect of the traffic profile and must be defined, or a secondary parameter which has a default value and only needs to be defined if different from the default.
<b>Default</b>	A default value can be used instead of specifying the parameter for secondary timing parameters.
<b>Min. Value</b>	The minimum value that is permitted for a parameter.
<b>Owner</b>	The component in a system that controls the timing parameter. Intercon is used as a short form of Interconnect.

Table 3-1 shows the list of timing parameters.

**Table 3-1 Timing parameters**

Name	Description	Default	Min Value	Between or Within	Primary or Secondary	Owner
ARTV	Read address valid to next read address valid	-	1	Between	Primary	Master
ARR	Read address valid to ready	-	0	Within	Primary	Slave
RIV	Read address handshake to first data valid	-	1	Within	Primary	Slave
RBV	Read data handshake to next beat valid	1	1	Within	Secondary	Slave
RBR	Read data valid to same beat ready	0	0	Within	Secondary	Master
RLA	Read last data handshake to acknowledge	1	1	Within	Secondary	Master
AWTV	Write address valid to next write address valid	-	1	Between	Primary	Master
AWV	Transaction start to write address valid	1	0	Within	Secondary	Master
AWR	Write address valid to ready	-	0	Within	Primary	Slave
WIV	Transaction start to first data valid	1	0	Within	Secondary	Master
WBR	Write data valid to same beat ready	0	0	Within	Secondary	Slave
WBV	Write data handshake to next beat valid	1	1	Within	Secondary	Master
BV	Write last data handshake to response valid	-	1	Within	Primary	Slave
BR	Write response valid to ready	0	0	Within	Secondary	Master
BA	Write response handshake to acknowledge	1	1	Within	Secondary	Master
ACTV	Snoop address valid to next address valid	-	1	Between	Primary	Intercon
ACR	Snoop address valid to ready	-	0	Within	Primary	Master
CRV	Snoop address handshake to response valid	-	1	Within	Primary	Master
CRR	Snoop response valid to ready	0	0	Within	Secondary	Intercon
CDIV	Snoop address handshake to first data valid	-	1	Within	Primary	Master
CDBR	Snoop data valid to ready	0	0	Within	Secondary	Intercon
CDBV	Snoop data handshake to next beat valid	1	1	Within	Secondary	Master

### 3.1.1 Read transaction

Figure 3-1 on page 3-28 shows the timing parameters that are associated with a read transaction.

**Table 3-2 Read transaction timing parameters**

Name	Description	Value
ARTV	Read address valid to next read address valid	13
ARR	Read address valid to ready	2
RIV	Read address handshake to first data valid	3
RBR	Read data valid to same beat ready	2
RBV	Read data handshake to next beat valid	2
RLA	Read last data handshake to acknowledge	3

Figure 3-1 on page 3-28 illustrates the interactions that result from the information in Table 3-2

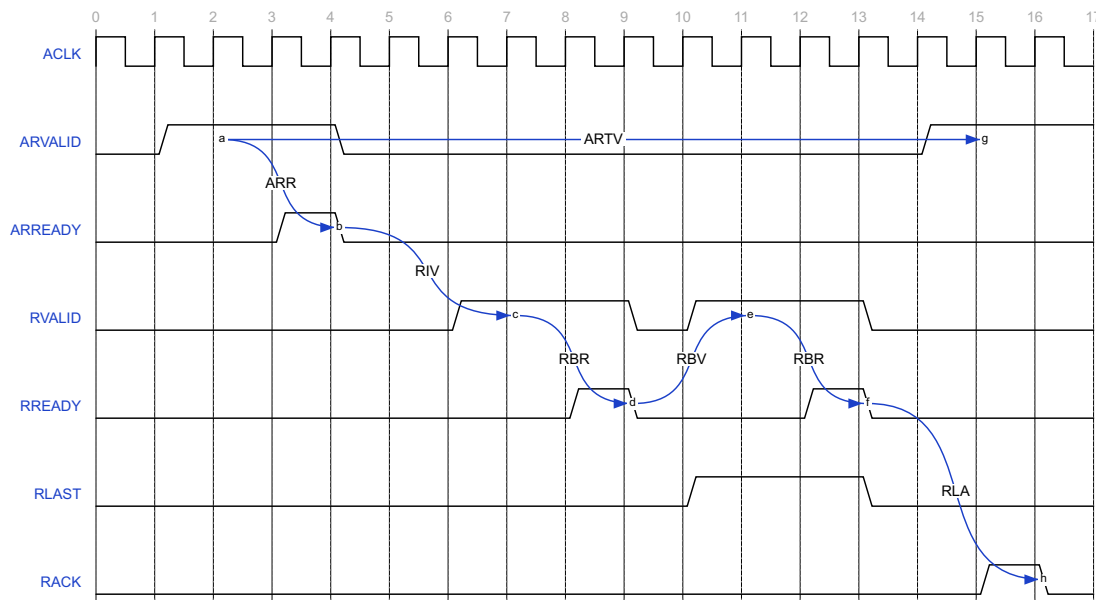


Figure 3-1 Read transaction timing

3.1.2 Write transaction

Table 3-3 shows all the timing parameters that are associated with a write transaction.

Table 3-3 Write transaction timing parameters

Table with 3 columns: Name, Description, and Value. It lists timing parameters for a write transaction such as AWTV, AWV, AWR, WIV, WBR, WBV, BV, BR, and BA.

Figure 3-2 on page 3-29 illustrates the interactions that result from the information in Table 3-3.

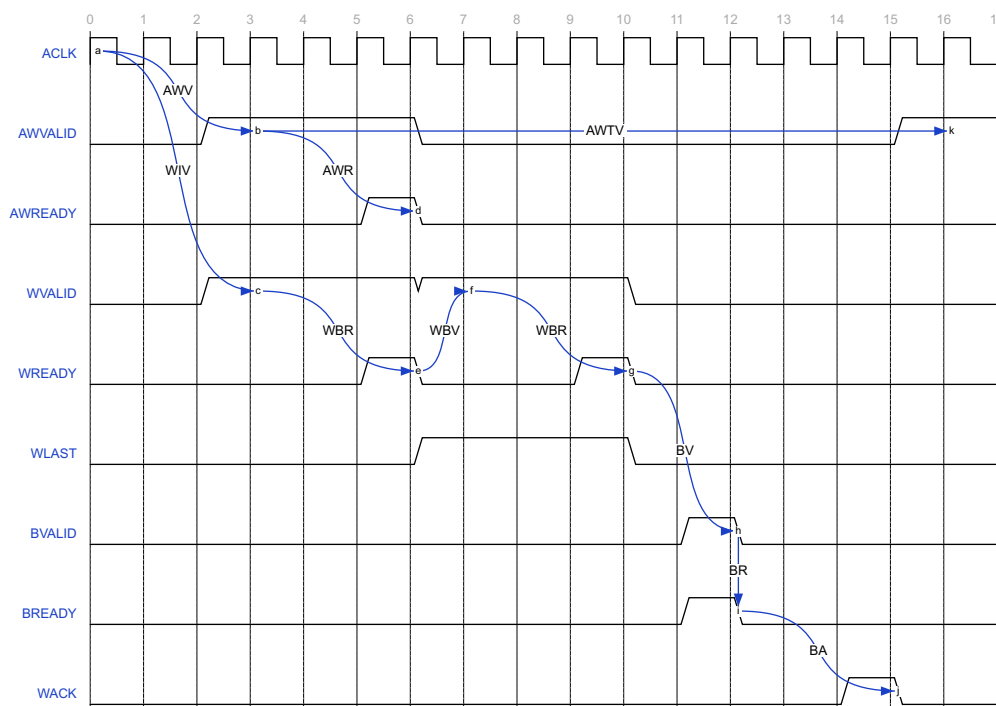


Figure 3-2 Write transaction timing

### 3.1.3 Snoop transactions

Figure 3-3 on page 3-30 shows the timing parameters that are associated with a snoop transaction.

Table 3-4 Snoop transaction parameters

Name	Description	Value
ACTV	Snoop address valid to next address valid	9
ACR	Snoop address valid to ready	1
CRV	Snoop address handshake to response valid	4
CRR	Snoop response valid to ready	0
CDIV	Snoop address handshake to first data valid	3
CDBR	Snoop data valid to ready	2
CDBV	Snoop data handshake to next beat valid	1

Figure 3-3 on page 3-30 illustrates the interactions that result from the information in Table 3-4.

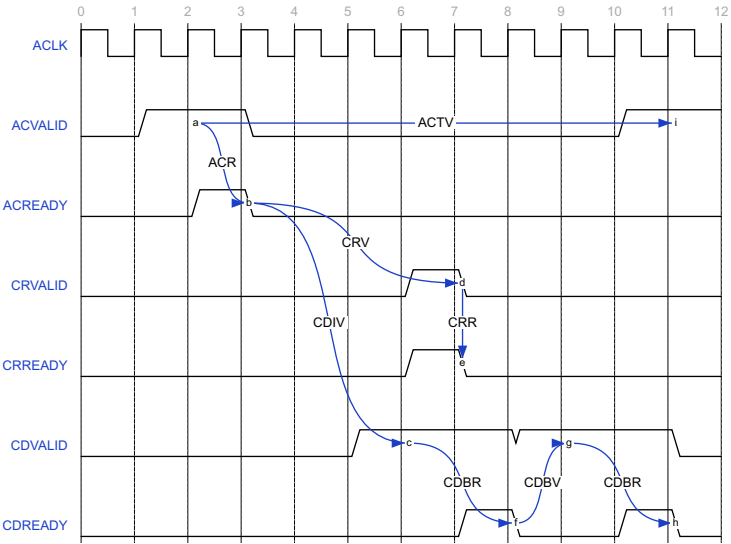


Figure 3-3 Snoop transaction timing

## 3.2 Complex transactions

For most traffic profile uses, the timing parameters for different beats within the same transaction are constant and only a single parameter is used. For more sophisticated uses of timing parameters, the naming convention in [Table 3-5](#) is used to define the timing parameters of different beats within a burst. An example is when timing parameters are used to record observed behavior instead of being used to control generated stimulus

**Table 3-5 Alternate snoop timing parameters**

Description	Read	Write	Snoop
Address or start to first data valid	RIV	WIV	CDIV
First data valid to first data ready	RBR0	WBR0	CDBR0
First data handshake to second data valid	RBV1	WBV1	CDBV1
Second data valid to second data ready	RBR1	WBR1	CDBR1
Second data handshake to third data valid	RBV2	WBV2	CDBV2
Third data valid to third data ready	RBR2	WBR2	CDBR2

The figures in this section give further examples of timing parameter usage for read and write transactions. These diagrams are for illustrative purposes only and provide no additional specification.

[Table 3-6](#) shows the parameters for example read transactions.

**Table 3-6 Example read values**

Transaction	First	Second
ARTV	-	5
ARR	0	1
RIV	2	3
RBV	1	2
RBR	0	1
RLA	1	1

[Figure 3-4 on page 3-32](#) illustrates the interactions that result from the information in [Table 3-4 on page 3-29](#)

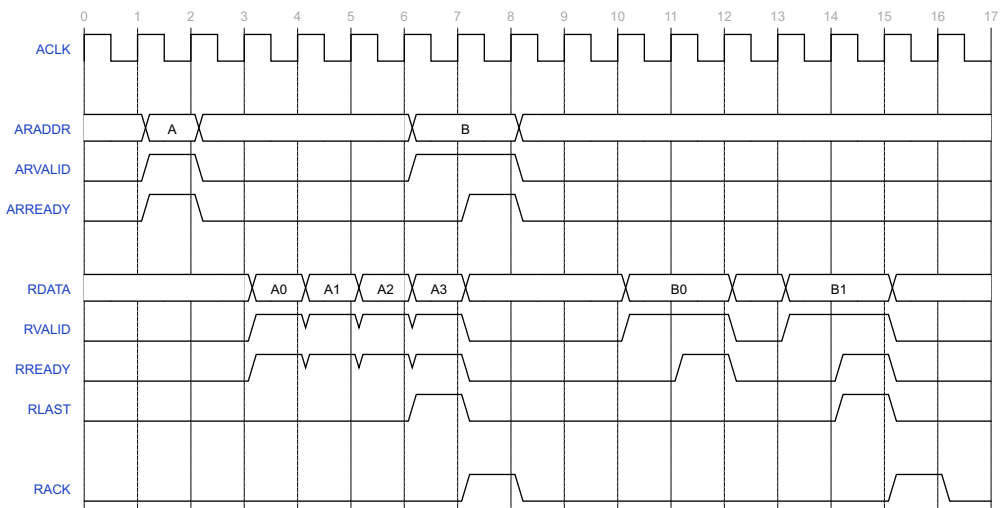


Figure 3-4 Example read timings

Table 3-7 shows the parameters for examples write transactions.

Table 3-7 First write example transaction

Transaction	First	Second
AWTV	-	5
AWV	1	1
AWR	2	1
WIV	2	4
WBR	0	1
WBV	1	2
BV	2	1
BR	1	0
BA	2	1

Figure 3-5 on page 3-33 illustrates the interactions that result from the information in Table 3-7.



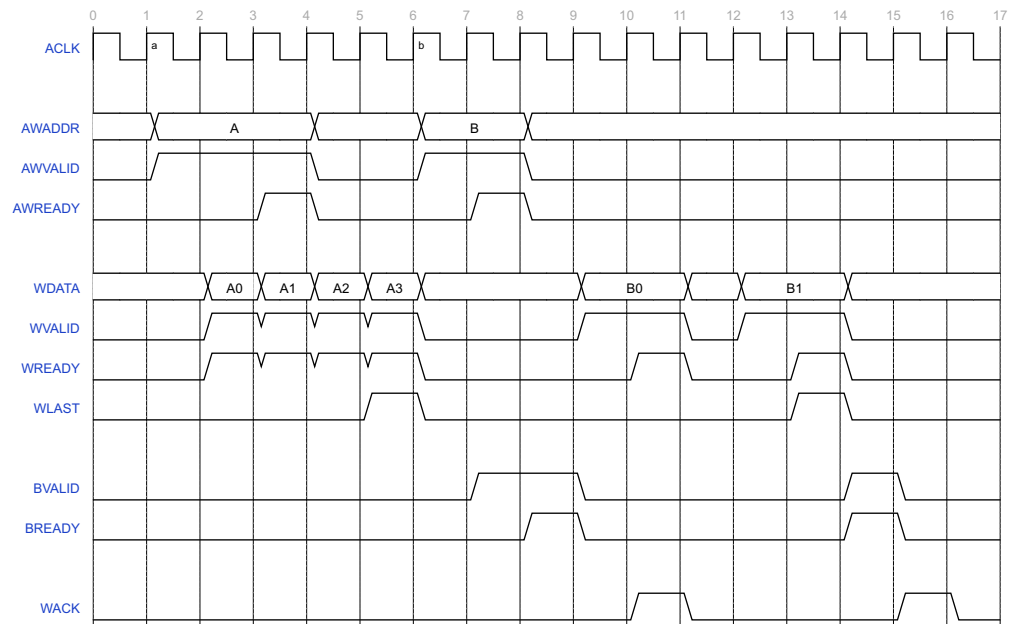


Figure 3-5 Example write timings



# Chapter 4

## FIFO timing model

This chapter describes the FIFO timing model. It contains the following sections:

- *Timing model* on page 4-36.
- *Generator Specification* on page 4-37.
- *FIFO timing points* on page 4-41.
- *Checker behavior* on page 4-43.
- *Generator characteristics* on page 4-44.
- *Linked traffic profiles* on page 4-45.

## 4.1 Timing model

A key part of a traffic profile is the timing between transactions. This information is needed in the generation of transactions and can also be used when monitoring and checking that transactions meet certain criteria.

A timing model should be:

- Easy to use and understand.
- Quick and easy to adapt to the characteristics of different components.
- Sufficiently realistic that performance analysis results are useful.
- Able to give reproducible behavior.

The model that is specified is based on a simple FIFO model.

## 4.2 Generator Specification

The description of a Generator is described in [Generator description](#). The description is in plain language, with a more formal definition in [Formal Generator description](#).

### 4.2.1 Generator description

The behavior of the Generator is:

- The profile models the transaction generation characteristics that would be as seen from a component containing a FIFO.
- When the profile is first triggered, the FIFO is set to either being completely full or completely empty.
- Every cycle the level of the FIFO changes dependent on the constant fill or drain rate:

**Read profile:**

The FIFO empties at this rate.

**Write profile:**

The FIFO fills at this rate.

- A transaction is issued whenever possible, according to the FIFO fill level:

The Generator will issue a new transaction when:

**Read profile:**

A new transaction is issued whenever there is space in the FIFO for the data that will be returned by the transaction. When calculating how much FIFO space is available, the space that is needed for outstanding read transactions must be taken into account.

**Write profile:**

A new transaction is issued whenever there is sufficient data in the FIFO to complete the transaction. When calculating how much data is available, data that is already committed for outstanding writes must be taken into account.

- The FIFO level is changed whenever transaction data is sent or received:

**Read profile:**

The current level of the FIFO is increased whenever read data is received.

**Write profile:**

The current level of the FIFO is reduced whenever write data is sent.

- A warning can be issued if the FIFO overflows or underflows.

### 4.2.2 Formal Generator description

[Table 4-1](#) formally specifies the parameters that control the behavior of the Generator:

**Table 4-1 FIFO parameters**

Name	Parameter	Units	Default	Description
Rate	Rate	Bytes per cycle	-	The rate that the FIFO level will change on each cycle. Recommended to be an integer multiple of $2^{-16}$ .
Full Level	Full	Bytes	-	The maximum number of bytes that the FIFO can contain. This might also be referred to as the FIFO Depth.
Outstanding Transaction Limit	TxnLimit	Integer	1	The maximum number of outstanding transactions that the profile can generate at any point in time.

**Table 4-1 FIFO parameters (continued)**

Name	Parameter	Units	Default	Description
Transaction Size	TxnSize	Bytes	64 B	The number of bytes that are requested by each transaction request.
Data Size	DataSize	Bytes	-	The number of bytes that are transferred for each data beat within a transaction. This is equivalent to the data bus width.
Startup Level (read)	Start	Enum	Empty	The starting level of the FIFO for a read profile.
Startup Level (write)	Start	Enum	Full	The starting level of the FIFO for a write profile.
Priority	Priority	Integer	-	Optional. Priority of traffic from this generator. Higher values have precedence over lower ones.
Frame Size	FrameSize	Bytes	-	Optional. Specifies the number of bytes transferred before this profile triggers an event.
Frame Time	FrameTime	Cycles	-	Optional. Specifies the number of cycles from the start of the profile before it triggers an event.
Clock Frequency	Frequency	MHz.	-	Optional. Interface clock frequency for the profile. Facilitates conversion between time-based and cycle-based representations of the profile.

[Table 4-2](#) shows the internal variables maintained by the Generator.

**Table 4-2 FIFO variables**

Name	Label	Units	Description
Current transactions	CurTxn	Integer	Number of transactions outstanding in the current cycle.
Current level	CurLv1	Integer	The fill level in the current cycle.
Data Pending	DataPend	Bytes	The number of bytes of data that have been committed to be sent or received, but have not yet been sent or received.

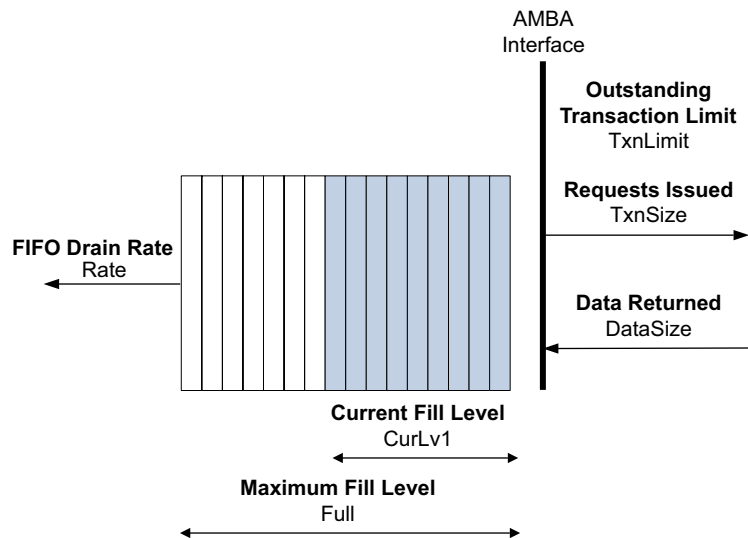
Table 4-3 provides a description of the FIFO behavior, including:

- Details on how the FIFO level should change.
- When transactions should be issued.
- When warnings for overflow or underflow are given.

**Table 4-3 FIFO behavior**

Condition	Read FIFO	Write FIFO
From Trigger, if Start is Empty	$\text{CurLv1} = 0$	$\text{CurLv1} = 0$
From Trigger, if Start is Full	$\text{CurLv1} = \text{Full}$	$\text{CurLv1} = \text{Full}$
Every cycle	$\text{CurLv1} = \text{CurLv1} - \text{Rate}$	$\text{CurLv1} = \text{CurLv1} + \text{Rate}$
If Data issued or returned	$\text{CurLv1} = \text{CurLv1} + \text{DataSize}$ $\text{DataPend} = \text{DataPend} - \text{DataSize}$	$\text{CurLv1} = \text{CurLv1} - \text{DataSize}$ $\text{DataPend} = \text{DataPend} - \text{DataSize}$
Issue transaction if	$(\text{CurLv1} \leq (\text{Full} - \text{DataPend} - \text{TxnSize})) \ \& \ (\text{CurTxn} < \text{TxnLimit})$	$(\text{CurLv1} \geq (\text{DataPend} + \text{TxnSize})) \ \& \ (\text{CurTxn} < \text{TxnLimit})$
If transaction issued	$\text{DataPend} = \text{DataPend} + \text{TxnSize}$	$\text{DataPend} = \text{DataPend} + \text{TxnSize}$
Issue warning after startup if	Condition: Underflow if $\text{CurLv1} < 0$ Action: Issue warning, $\text{CurLv1} = 0$	Condition: Overflow if $\text{CurLv1} > \text{Full}$ Action: Issue warning, $\text{CurLv1} = \text{Full}$
Startup duration, if Start is Empty	$(\text{Full} - \text{TxnSize}) / \text{Rate}$	0
Startup duration, if Start is Full	0	$(\text{Full} - \text{TxnSize}) / \text{Rate}$

Figure 4-1 and Figure 4-2 on page 4-40 give diagrammatic representations of the parameters:



**Figure 4-1 FIFO Read profile**

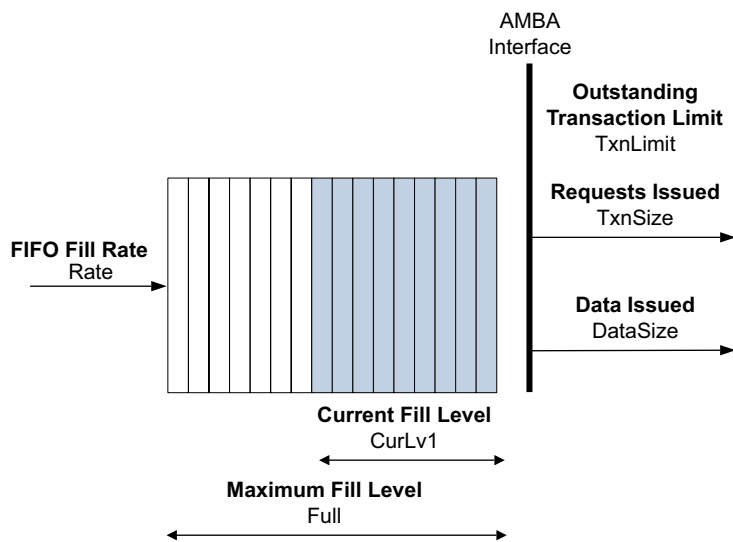


Figure 4-2 FIFO Write profile



## 4.3 FIFO timing points

This section describes the precise timing points that are used for the FIFO calculations.

### 4.3.1 Normal operation

For a Read FIFO on a rising clock edge, the following occur:

- If **RVALID** and **RREADY** are asserted then the FIFO fills by data beat size.
- The FIFO empties by an amount that is determined by Rate and is affected:
  - If the FIFO value calculated during the previous clock period indicates that there is enough data for the FIFO to drain by the amount that is determined by Rate, then this drain occurs.
  - If there is insufficient data for the full amount that is determined by Rate then the FIFO just empties by the amount of data that is available.

Filling and emptying the FIFO are independent.

The amount that the FIFO empties does not take into consideration any data that is returned at the end of the current cycle.

The fill that is caused by **RVALID** and **RREADY** being asserted will always occur because the transaction will only have been issued if there was sufficient space to receive the data. Additional draining of the FIFO makes no difference.

If the FIFO underflows in the same cycle that **RVALID** and **RREADY** are asserted the resultant FIFO level will be one data beat. However, an underflow condition should be signaled.

After the rising clock, the new FIFO value is calculated. This is used to determine if there is sufficient space for a new read transaction to be issued, **ARVALID** asserted, in the same cycle.

For a Write FIFO, on a rising clock edge:

- If **WVALID** and **WREADY** are asserted then the FIFO empties by data beat size.
- The FIFO fills by amount that is determined by Rate, the factors are:
  - If the FIFO value that is calculated during the previous clock period indicates that there is space, then the FIFO fills by the amount Rate.
  - If there is insufficient space for the full amount determined by Rate then the FIFO just fills by the amount of space that is available.

Filling and emptying the FIFO are independent.

The amount that the FIFO fills by does not take in to consideration any data that is sent in the current cycle.

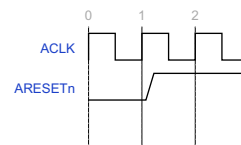
The drain that is caused by **WVALID** and **WREADY** being asserted will always occur because the transaction will only have been issued if there was sufficient data to be sent. Additional filling of the FIFO makes no difference.

If the FIFO over fills in the same cycle that **WVALID** and **WREADY** are asserted the resulting FIFO level will be one data beat less than Full. However, an overflow condition should be signaled.

After the rising clock, the new FIFO value is calculated. This is used to determine if there is sufficient data for a new write transaction to be issued, **AWVALID** asserted, in the same cycle.

### 4.3.2 Reset behavior

Reset behavior is as follows:



**Figure 4-3 Reset timing diagram**

For a read FIFO, on clock edge 2:

- A full read FIFO will drain by an amount that is determined by Rate.
- An empty read FIFO would attempt to drain, but would underflow.  
No underflow warning is issued until after the Startup duration
- The first transaction can be issued in clock cycle 2 to 3.

For a write FIFO, on clock edge 2:

- An empty write FIFO will fill by an amount that is determined by Rate.
- A full write FIFO would attempt to fill, but would overflow.  
No overflow warning is issued until after the Startup duration.
- The first transaction can be issued in clock cycle 2 to 3.

## 4.4 Checker behavior

Checker behavior is almost identical to that of a Generator. However, a checker does not generate transactions.

Optionally, a Checker can report the following parameters:

- |               |   |
|---------------|---|
| <b>MaxLvl</b> | The maximum fill level that is achieved during an analysis. This only has meaning for a profile that had an initial value of Empty. |
| <b>MinLvl</b> | The minimum fill level that is achieved during an analysis. This only has meaning for a profile that had an initial value of Full.  |

## 4.5 Generator characteristics

This section provides an overview of Generator behavior. It does not provide any additional traffic profile specification.

There are typically three phases of operation that will be observed from a Generator:

- *Initial Peak Rate.*
- *Constrained Peak Rate.*
- *Average Rate.*

### 4.5.1 Initial Peak Rate

The *Initial Peak Rate* occurs when the Generator is first triggered. The rate is not constrained by the amount of data in the FIFO or the maximum number of outstanding transactions limit. The Generator is expected to issue a single transaction every cycle with TxnSize bytes per cycle.

The minimum duration of this phase is:

- TxnLimit cycles, if the limit to the number of transactions outstanding becomes the constraint:
  - The duration of the phase is extended if transactions complete before the end of this phase.
  - The *Constrained Peak Rate* phase then follows.
- Approximately (Full / TxnSize) cycles, if the amount of data in the FIFO becomes the constraint:
  - The duration of the phase is extended as the natural filling/emptying continues, which can be calculated to give a more precise duration of this phase.
  - The *Average Rate* phase then follows.

The *Initial Peak Rate* phase will typically last slightly longer than the minimum.

### 4.5.2 Constrained Peak Rate

In the *Constrained Peak Rate* phase, the issue of transactions is constrained by the limit to the maximum number of outstanding transactions. The peak rate is determined by the latency of the transactions that have been issued. A new transaction is issued when an earlier transaction is fully completed. During this phase, the bandwidth rate is controlled by the system latency, not by the characteristics of the Generator.

The *Constrained Peak Rate* phase continues until:

- The FIFO approaches full, for a read profile.
- The FIFO approaches empty, for a write profile.

When the Rate specified is greater than the bandwidth that is achieved in the system, some components will operate in the *Constrained Peak Rate* phase for most of the time.

### 4.5.3 Average Rate

During the *Average Rate* phase, the issue of transactions is determined by the average rate of filling or emptying the FIFO. The transaction issue rate is constant and is controlled by the Generator parameters. However, the data return rate is controlled by the system that the component is operating in.

## 4.6 Linked traffic profiles

The specification includes a mechanism to link together traffic profiles, so that the progress of one traffic profile can affect the progress of another traffic profile. Typically, this is used to link a read traffic profile and a write traffic profile. Linking can be used to represent the behavior of a component that requires both read and write transactions to make progress to make overall component progress.

When two traffic profiles are linked the behavior is:

- For a read profile linked to a write profile:
  - When the read FIFO is completely empty, then the fill of the write FIFO does not occur.
  - When the write FIFO is completely full, then the drain of the read FIFO does not occur.
- For two read profiles that are linked:
  - When either read FIFO is completely empty, the drain of the other read FIFO does not occur.
- For two write profiles that are linked:
  - When either write FIFO is completely full, the fill of the other write FIFO does not occur.



# Chapter 5

## Event coordination

This chapter describes the Event Coordination mechanism:

- *Synchronization between traffic profiles on page 5-48.*
- *Concurrent traffic profile behavior on page 5-49.*

## 5.1 Synchronization between traffic profiles

An Event Coordination mechanism is used to synchronize the execution of traffic profiles. The profiles might be on different components or on the same component.

Each Event has two fields:

- Address.
- Event Identifier

The Address that is associated with an Event ensures that both agents in a traffic profile are using the same address ranges. Common addresses help when analyzing system behavior that includes the use of caches. Different addresses can be used for different producer or consumer groups, which allows agents within the same group to interact, while agents in a different group remain independent.

The Event Identifier has the following characteristics:

- The Event Identifier is understood by both the sender and receiver to indicate the action that is required.
- A component that is receiving an Event can use the Event Identifier to determine the action to perform.
- A component that is sending an Event determines the Event Identifier to send.
- It is typical that sending an Event will be done at the beginning or completion of a traffic profile.
- An Event Identifier does not need to be associated with a particular component or a particular traffic profile.

A component that is receiving an Event is not required to make use of either the Address or the Event Identifier.

A component that is sending an Event is not required to make use of either the Address or the Event Identifier. When not used, the Address is set to zero and the Event Identifier is set to zero.

An Event that is sent at the start of a traffic profile is issued in the same cycle as the first transaction of the traffic profile.

An Event that is sent at end of a traffic profile is issued in the cycle after all transactions of the associated traffic profile have completed. It is not dependent on the sending of a RACK or WACK completion acknowledge signaling.

Components can support both input Events and output Events. Examples of combinations of input and output Events are:

### **Input event only**

Used by a component that is performing some traffic, based on Events generated by other components in the system. The component does not then trigger other activity.

### **Input and output event**

Allows the elastic behavior of the system to be modeled, where the system latency is changing the time that is required for the traffic profile to execute.

### **Output event only**

Used only by components that begin at the start of simulation.

A traffic profile can be considered complete under any of the following conditions:

- Upon receipt of an input Event.
- After a predetermined number of bytes have been transferred, as specified by the FrameSize parameter.
- After a predetermined number of cycles, as specified by the FrameTime parameter.
- When the end of the file is reached in traffic profiles using the file mechanism to generate address, ID or data.
- When the end of the address range is reached in traffic profiles using the sequential or twodim mechanisms.



## 5.2 Concurrent traffic profile behavior

This section describes the required behavior when multiple traffic profiles are running concurrently on a single component.

In many circumstances it is possible for multiple traffic profiles to operate concurrently, with no interaction between them. The signal characteristics, timing parameters, and co-ordination Events of the profiles can be independent.

An interaction between traffic profiles occurs when two or more traffic profiles require the same physical resource in the same cycle. For example, when two traffic profiles wish to issue a transaction on the same address channel at the same time.

The following rules apply:

- When one traffic profile already has use of a physical resource from a previous cycle and it is required by the protocol to maintain the use of that resource, it continues to use the resource and other traffic profiles are delayed.
- When two or more traffic profiles attempt to begin using a physical resource in the same cycle, then a simple priority mechanism is used. Each concurrent traffic profile is given a *Priority* parameter and the highest priority profile will gain use of the resource.

If a *Priority* parameter is not specified in a traffic profile, then any tool, model, or other environment that is using the traffic profile is permitted to make its own assignment of priorities. It is recommended that the environment report the priorities that have been assigned to each profile. This allows an identical transaction sequence to be generated in different environments by making use of the reported priority assignments.



## Chapter 6

# Slave traffic profiles

The slave traffic profiles are described:

- [Slave traffic profiles on page 6-52.](#)

## 6.1 Slave traffic profiles

Most uses of traffic profiles will be to define the behavior of master components within a system. For completeness, a simple slave traffic profile is also defined.

The features of a slave traffic profile are:

- Defined as Read, Write, or Combined.
- Address timing is determined by specifying the number of transactions that the slave can accept.
- Response timing is determined by a data rate parameter.
- Only OKAY responses are supported.

Table 6-1 shows the parameters that control the dynamic behavior of the slave interface.

**Table 6-1 Dynamic slave behavior parameters**

Name	Parameter	Units	Default	Description
Rate	Rate	Bytes per cycle	-	This is the rate that the slave can process data. Recommended to be an integer multiple of $2^{-16}$ .
Transaction Limit	TxnLimit	Integer	1	This is the limit to the number of outstanding transactions that the slave can accept.
Transaction Size	TxnSize	Bytes	64	Determines the granularity of transactions that are processed.

The *Rate* parameter is used to determine the number of cycles from the address being accepted to the first data beat of read data or the write response.

**Read data** The Rate parameter determines the number of cycles that are required to obtain the entire transaction's data and the first beat of read data is made available after this time. The timing of subsequent cycles of read data is determined by the RBV, read data handshake to next beat valid, parameter:  
 $RIV = \text{RoundUp} (\text{TxnSize} / \text{Rate})$

**Write data** The Rate parameter is used to determine the number of cycles that are required to process the entire transaction's data and the write response is given after this time:  
 $BV = \text{RoundUp} (\text{TxnSize} / \text{Rate})$

The TxnSize parameter is used to determine the granularity of transactions that are processed. For example, if a transaction request is for 8 bytes and the TxnSize parameter is 64 bytes, then a response will only be given after the processing time for 64 bytes.

If the size of the received transaction is greater than TxnSize, then the processing time is determined by rounding up the actual transaction size to the next TxnSize boundary.

In both read and write cases, the processing time is considered separately from the transfer of data across the interface. For read transactions the data transfer only commences once data for the entire transaction is available. For write transactions the processing time only commences once data for the entire transaction has been accepted.

When multiple transactions are in progress at the same time, the processing time for a subsequent transaction starts immediately after the processing time for the previous transaction. It is not dependent on the time that is taken to transfer data across the interface.

The TxnLimit parameter is used to determine the address valid to ready timing parameter. If a slave component has accepted fewer transactions than defined by TxnLimit, then a transaction will be accepted. The acceptance is indicated by the assertion of the Ready signal in the same cycle that the address becomes valid. This corresponds to an ARR or AWR value of zero.

When a slave component has the same number of outstanding transactions as defined by the `TxnLimit` parameter and another address is presented to the slave, then the Ready signal is asserted the cycle after an earlier transaction completes. The completion of an earlier transaction is determined by the following:

- The cycle that the last beat of read data is transferred, as indicated by **RVALID**, **RLAST** and **RREADY** being asserted.
- The cycle that the write response is given, as indicated by **BVALID**, and **BREADY** being asserted.

Table 6-2 summarizes the timing parameters that are defined for a slave traffic profile.

**Table 6-2 Slave traffic timing parameters**

Description	Name	Default	Min Value	Primary/ Secondary
Read address valid to ready	ARR	-	0	Primary
Read address handshake to first data valid	RIV	-	1	Primary
Read data handshake to next beat valid	RBV	1	1	Secondary
Write address valid to ready	AWR	-	0	Primary
Write data valid to same beat ready	WBR	0	0	Secondary
Write last data handshake to response valid	BV	-	1	Primary

The secondary timing parameters can use the default values or can be defined in the traffic profile.



# Appendix A

## Default signal values

This section list the default signals values for common AXI signals. See the AXI specification for a full list of signals and default values.

**Table A-1 Default signal values**

Signal	Description	Default
ARADDR	Address	-
ARBURST	Burst type	0b01, INCR
ARCACHE	Cache type	0b0000
ARID	Address ID	All zeros
ARLEN	Burst length	All zeros, Length 1
ARLOCK	Lock type	All zeros, Normal access
ARPROT	Protection type	-
ARQOS	QoS value	0b0000
ARREGION	Region	All zeros
ARSIZE	Burst size	Data bus width
AWADDR	Address	-
AWBURST	Burst type	0b01, INCR
AWCACHE	Cache type	0b0000

**Table A-1 Default signal values (continued)**

<b>Signal</b>	<b>Description</b>	<b>Default</b>
<b>AWID</b>	Address ID	All zeros
<b>AWLEN</b>	Burst length	All zeros, Length 1
<b>AWLOCK</b>	Lock type	All zeros, Normal access
<b>AWPROT</b>	Protection type	-
<b>AWQOS</b>	QoS value	0b0000
<b>AWREGION</b>	Region	All zeros
<b>AWSIZE</b>	Burst size	Data bus width
<b>BRESP</b>	Response	0b00, OKAY
<b>RDATA</b>	Data	-
<b>RRESP</b>	Response	0b00, OKAY
<b>WDATA</b>	Data	-
<b>WSTRB</b>	Write strobes	All ones



# Appendix B

## AXI signal identifiers

This section lists the identifiers to be used for various AXI control signals.

Table B-1 AXI control signals

Signal	Value	Identifiers
AxBURST	0	SIZE_1
	1	SIZE_2
	2	SIZE_4
	3	SIZE_8
	4	SIZE_16
	5	SIZE_32
	6	SIZE_64
	7	SIZE_128
AxBURST	0	BURST_FIXED
	1	BURST_INCR
	2	BURST_WRAP
AxBURST	0	LOCK_NORMAL
	1	LOCK_EXCLUSIVE

Table B-1 AXI control signals (continued)

Signal	Value	Identifiers
<b>AxPROT</b>	0	PROT_D_S_UP
	1	PROT_D_S_P
	2	PROT_D_NS_UP
	3	PROT_D_NS_P
	4	PROT_I_S_UP
	5	PROT_I_S_P
	6	PROT_I_NS_UP
	7	PROT_I_NS_P
<b>BRESP</b>	0	RESP_OKAY
	1	RESP_EXOKAY
	2	RESP_SLVERR
	3	RESP_DECERR
<b>RRESP</b>	0	RESP_OKAY
	1	RESP_EXOKAY
	2	RESP_SLVERR
	3	RESP_DECERR

## Appendix C

# Example FIFO model behaviors

This appendix describes a number of ways that a FIFO timing model can be used to model the behavior of different types of components that can be found in a typical SoC. It also describes how the parameters of the model can be changed to cause different behaviors.

## C.1 FIFO Model

This appendix is informative and the statements that are made about typical or likely behavior will not apply to all situations.

For all the components described in this section it is expected that the following two parameters will be constant:

### Transaction Size

Transaction size is likely to be a fixed parameter, with an expectation that a 64-byte transaction size will be used in many components, since this works well with coherence protocols and memory controller burst sizes.

### Data Bus Width

Different components are expected to support different data bus width options. It is expected that most components will have a default data bus width and a system architect can choose to experiment with options above and below that width.

## C.2 Display

To model a component such as a Display Controller, the profile parameters would be:

**Drain Rate** Set to the natural rate of the Display, which is typically calculated using screen dimension, bits per pixel and frame rate. It is expected that this is calculated in MB/s and then converted to the Bytes/Cycle metric.

**FIFO Depth** Set to the size of the buffer located inside the Display. System architects can choose to vary this parameter during a performance simulation.

**Transaction Limit**

Typically the maximum number of outstanding transactions is fixed for a given design. A component will need to ensure that it supports a sufficient number of outstanding transactions to achieve the throughput required under average latency conditions. Short periods of high latency can be accommodated by the depth of the buffer.

**Start Level** Empty. A display is expected to be triggered ahead of the time it needs to send the first data to the display.

A system architect can choose to vary the FIFO Depth parameter. Increasing the depth increases the gate count (hence power) of the display, but has the advantage that the display becomes more tolerant to other traffic in the system.

For any given system there is likely to be a minimum buffer depth that is needed to guarantee correct operation.

## C.3 CPU

A CPU will typically have several different modal behaviors.

When operating from a warm cache, it will typically have a low `Rate` parameter and therefore the outstanding transaction limit and `FIFO Depth` will be somewhat irrelevant.

When operating from a cold cache or executing software that has a high miss rate, it will typically have a higher `Rate` parameter. In this case, the outstanding transaction limit will be important to determining the bandwidth. Some software algorithms, such as pointer chasing, will have very few outstanding transactions. Software algorithms with more parallelism will have a higher outstanding transaction limit.

The `FIFO Depth` parameter can be used in CPU profiles if a check is needed that the CPU does not experience long periods without sufficient bandwidth. However, it is expected that in many cases of CPU profiles the `FIFO Depth` parameter is set high enough to not influence the traffic profile.

## C.4 GPU

For GPU traffic profiles, the Rate of the GPU will be set to a higher value than can be achieved in the system. This ensures that the traffic profile will continue to generate transactions, since it will never achieve its desired rate. Because Rate is set to a high level, the traffic profile will appear that it is always operating in an overrun or underrun condition. This is entirely acceptable and it is expected that the reporting of warnings from the component would be disabled.

The use of the FIFO Depth parameter for GPUs is similar to how it can be used in CPU profiles. It can be used if a check is needed that the component does not experience long periods without sufficient bandwidth. However, it is expected that in many cases of GPU profiles the FIFO Depth parameter is set very high.

A GPU traffic profile might contain two timing models, one that is used for generation and one that is used for checking. For example, a GPU can use a generator that requests a very high bandwidth, but also contains a checker that checks for a moderate bandwidth. This means that the generator can be programmed to behave as close to the real component as possible, while the checker is only indicating an error when some minimum bandwidth is threshold is not achieved.

## C.5 Network interface

A traffic profile can provide a reasonably accurate representation of something like a network interface. The Rate parameter can be set to the natural bandwidth of the network interface and the FIFO Depth parameter can be set to the buffer size within the network interface IP.



## Appendix D

# Example Waveforms

This Appendix provides a number of waveforms that are generated from some example traffic profiles. These waveforms can be used to verify that the implementation of a traffic profile generator matches the expected cycle-by-cycle behavior in this specification.

Each example given is intended to illustrate one or two particular aspects of the behavior of a traffic profile.

All examples in this section are based on a data bus width of 128 bits, which means that DataSize is 16B.

## D.1 Basic Read with FIFO empty

This example describes:

- A basic read profile with the FIFO level starting empty.
- Filling at the max rate, then running at the FIFO drain rate.

### D.1.1 Configuration

Table D-1 Basic Read, empty FIFO profile

Name	Parameter	Value
FIFO Startup Level	Start	Empty
FIFO Depth	Full	64
Outstanding Transaction Limit	TxnLimit	30
Drain Rate	Rate	4
Transaction Size	TxnSize	16
Address Pattern	-	sequential
Starting Address	Base	0x8000
Range	Range	0x200
ID Type	-	fixed
ID Value	Value	0

### D.1.2 Timing diagram

The following items are interesting to note:

- First read request is at clock cycle 2.
- FIFO is filled at clock cycle 6.
- The behavior repeats on every 4<sup>th</sup> cycle, starting at clock cycle 8.

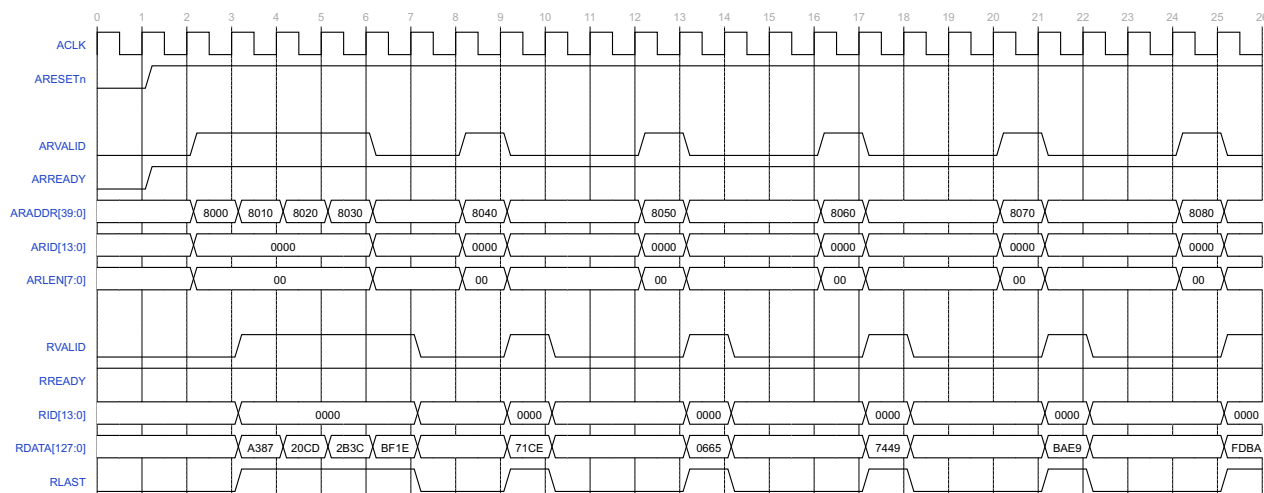


Figure D-1 Basic Read, FIFO empty

## D.2 Basic Read FIFO full

This example describes:

- Basic read with the FIFO starting full.
- Drain rate not a multiple of transaction size.
- Shows pause at the start of the sequence while FIFO drains.

### D.2.1 Configuration

Table D-2 Basic Read, full FIFO profile

Name	Parameter	Value
FIFO Startup Level	Start	Full
FIFO Depth	Full	64
Outstanding Transaction Limit	TxnLimit	30
Drain Rate	Rate	11
Transaction Size	TxnSize	32
Address Pattern	-	sequential
Starting Address	Base	0x9000
Range	Range	0x420
ID Type	-	fixed
ID Value	Value	0
Frame Size	FrameSize	384

### D.2.2 Timing diagram

The following items are interesting to note:

- Pause of 3 clock cycles at the start, to allow enough space in the FIFO to issue a transaction at clock cycle 4.
- From clock cycle 4, the behavior repeats every 3<sup>rd</sup> cycle.

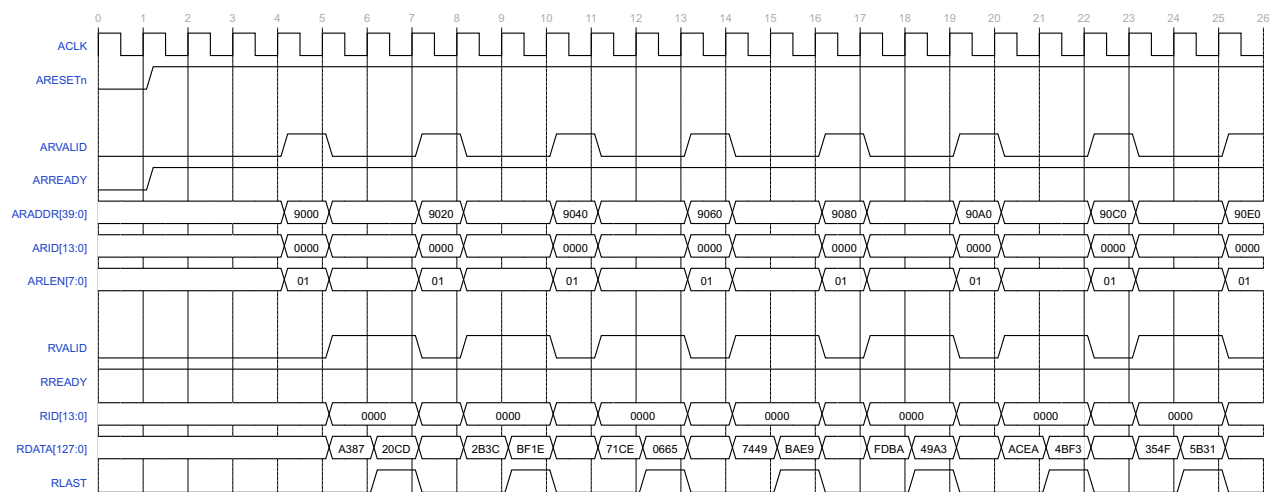


Figure D-2 Basic Read, FIFO full (1 of 3)

Appendix D Example Waveforms  
D.2 Basic Read FIFO full

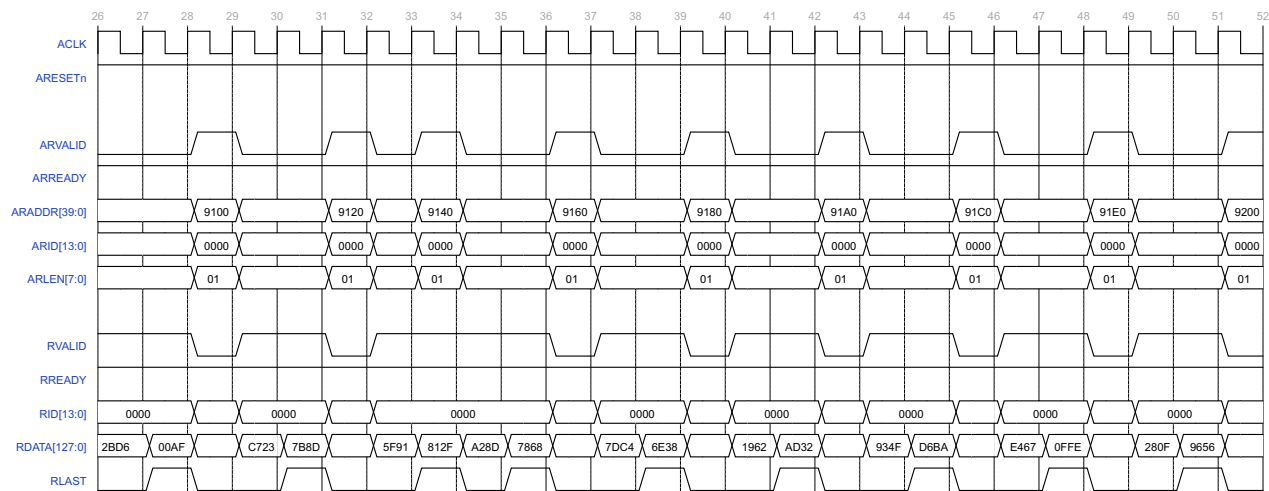


Figure D-3 Basic Read, FIFO full (2 of 3)

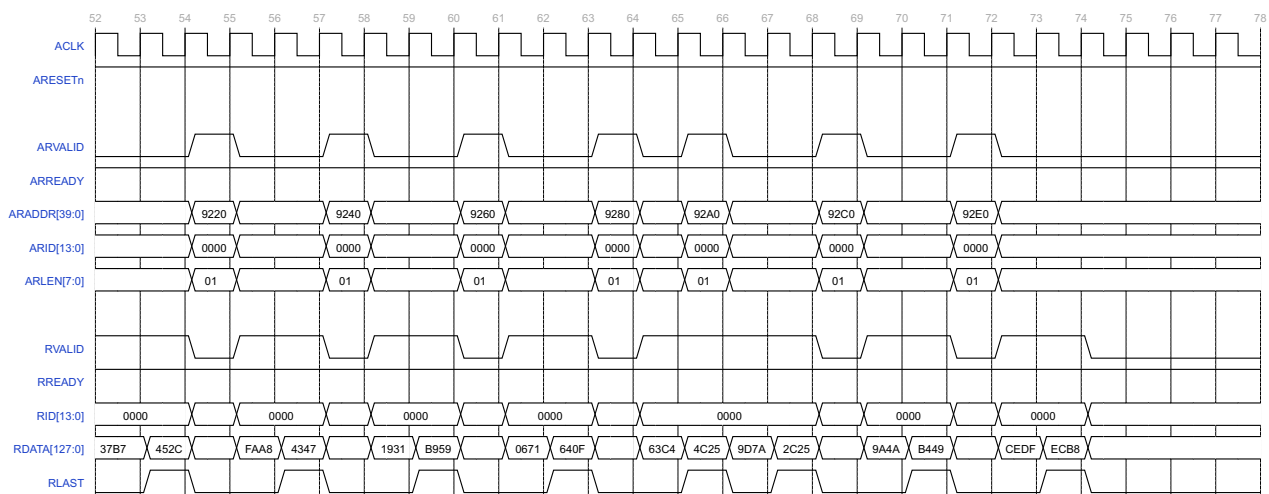


Figure D-4 Basic Read, FIFO full (3 of 3)

## D.3 Basic Write with the FIFO full

This example describes:

- Basic Write with the FIFO starting full.
- Shows address wrap.
- Executes at full rate until FIFO almost empty, then runs at fill rate.

### D.3.1 Configuration

**Table D-3 Basic Write, full FIFO profile**

Name	Parameter	Value
FIFO Startup Level	Start	Full
FIFO Depth	Full	64
Outstanding Transaction Limit	TxnLimit	30
Fill Rate	Rate	4
Transaction Size	TxnSize	16
Address Pattern	-	sequential
Starting Address	Base	0x800
Range	Range	0x90
ID Type	-	fixed
ID Value	Value	0

### D.3.2 Timing diagram

The following items are interesting to note:

- First write request issued at clock cycle 2.
- From clock cycle 7 the behavior repeats on every 4<sup>th</sup> cycle.

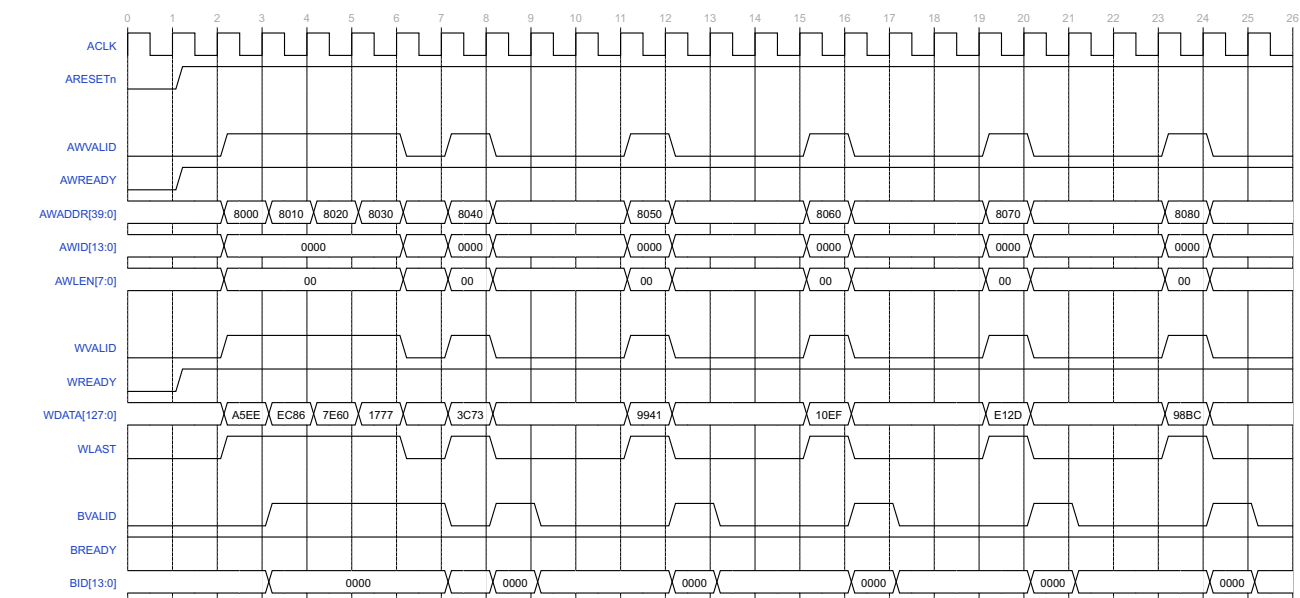


Figure D-5 Basic Write, FIFO full

## D.4 Basic Write with the FIFO empty

This example describes:

- Basic Write with the FIFO starting empty.
- Shows address wrap.
- Executes at full rate until FIFO almost empty, then runs at fill rate.

### D.4.1 Configuration

**Table D-4 Basic Write, full FIFO profile**

Name	Parameter	Value
FIFO Startup Level	Start	Empty
FIFO Depth	Full	64
Outstanding Transaction Limit	TxnLimit	30
Fill Rate	Rate	9
Transaction Size	TxnSize	32
Address Pattern	-	sequential
Starting Address	Base	0x9000
Range	Range	0x90
ID Type	-	fixed
ID Value	Value	5

### D.4.2 Timing diagram

The following items are interesting to note:

- First write request issued at clock cycle 5.
- From clock cycle 9 the behavior repeats on every 7<sup>th</sup> cycle.

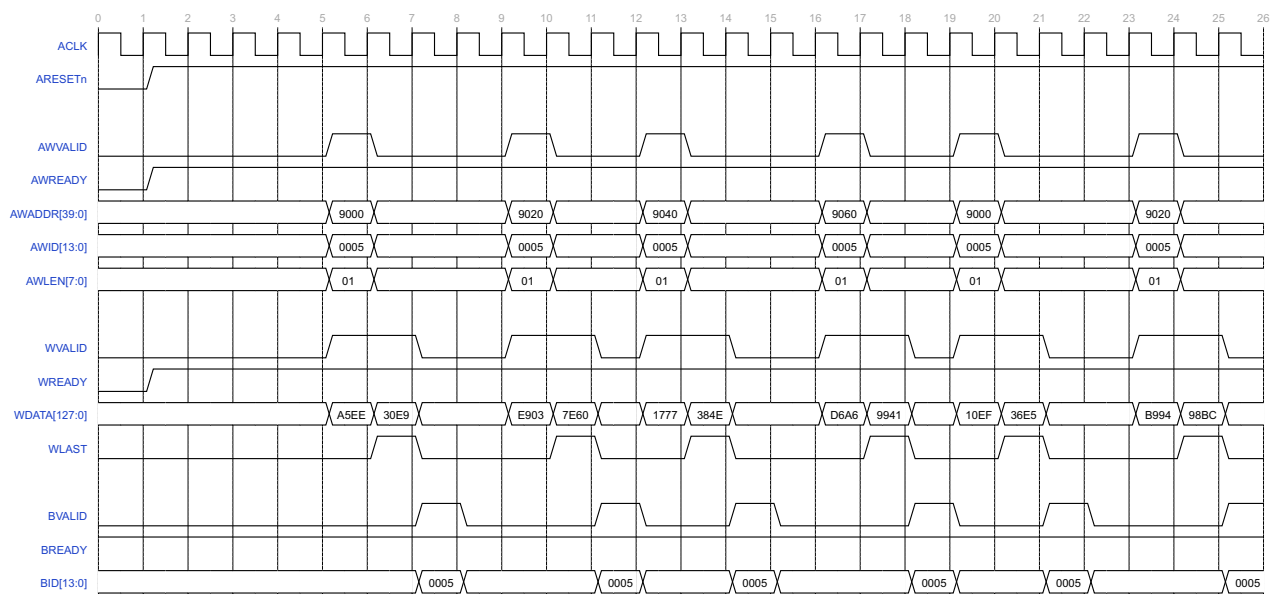


Figure D-6 Basic Write, FIFO empty (1 of 3)

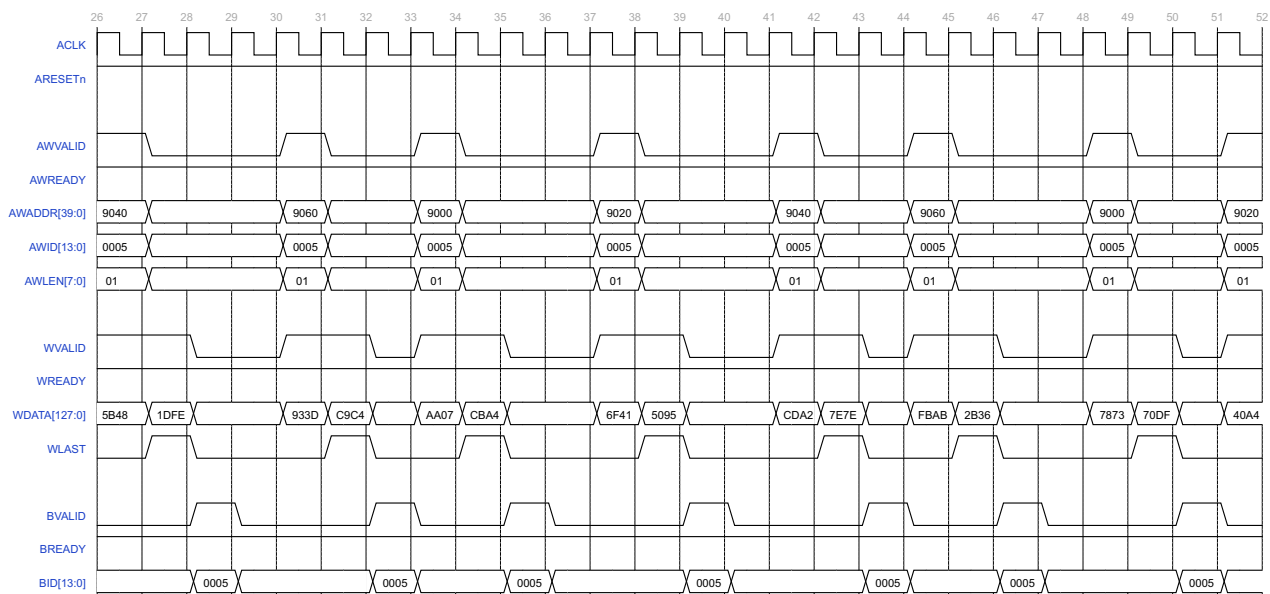


Figure D-7 Basic Write, FIFO empty (2 of 3)



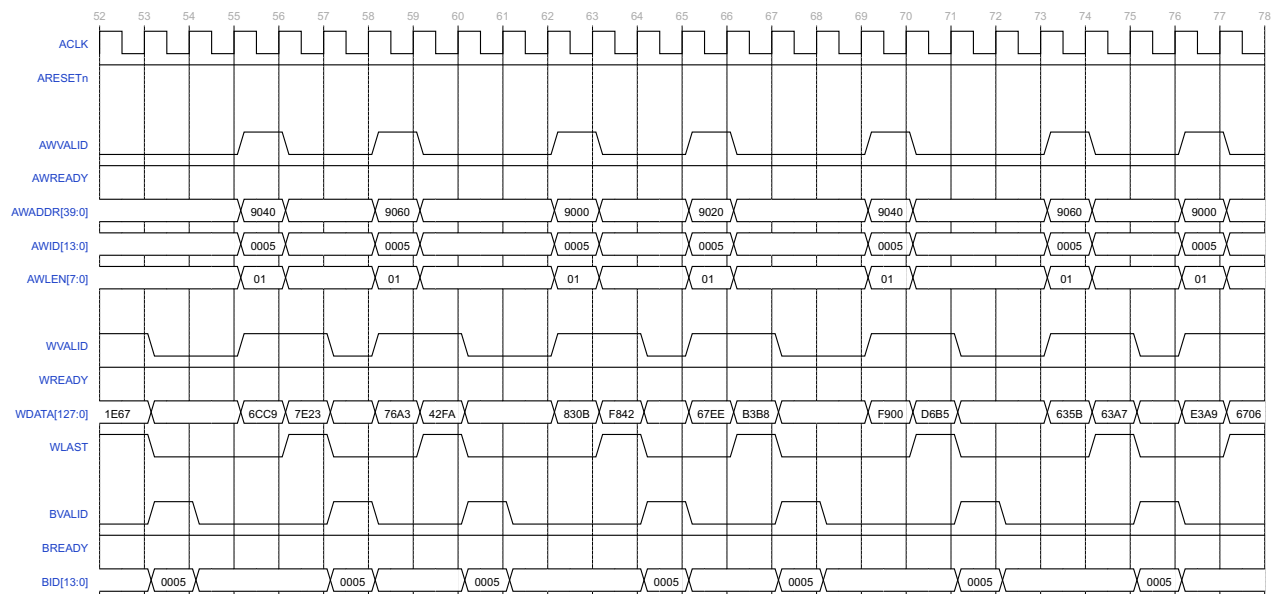


Figure D-8 Basic Write, FIFO empty (3 of 3)

## D.5 Read with FIFO underflow

This example describes FIFO underflow multiple times when applying the rate beyond the startup phase.

### D.5.1 Configuration

Table D-5 Read with FIFO underflow profile

Name	Parameter	Value
FIFO Startup Level	Start	Full
FIFO Depth	Full	64
Outstanding Transaction Limit	TxnLimit	30
Drain Rate	Rate	19
Transaction Size	TxnSize	16
Address Pattern	-	sequential
Starting Address	Base	0x9000
Range	Range	0x200
ID Type	-	cycle
Cycle ID Lower Value	LowerValue	0
Cycle ID Upper Value	UpperValue	4

### D.5.2 Timing diagram

The following items are interesting to note:

- The FIFO is empty from clock cycle 7 and triggers an underflow message every cycle since the drain rate is greater than the transaction size.
- A cyclic ID scheme is used.

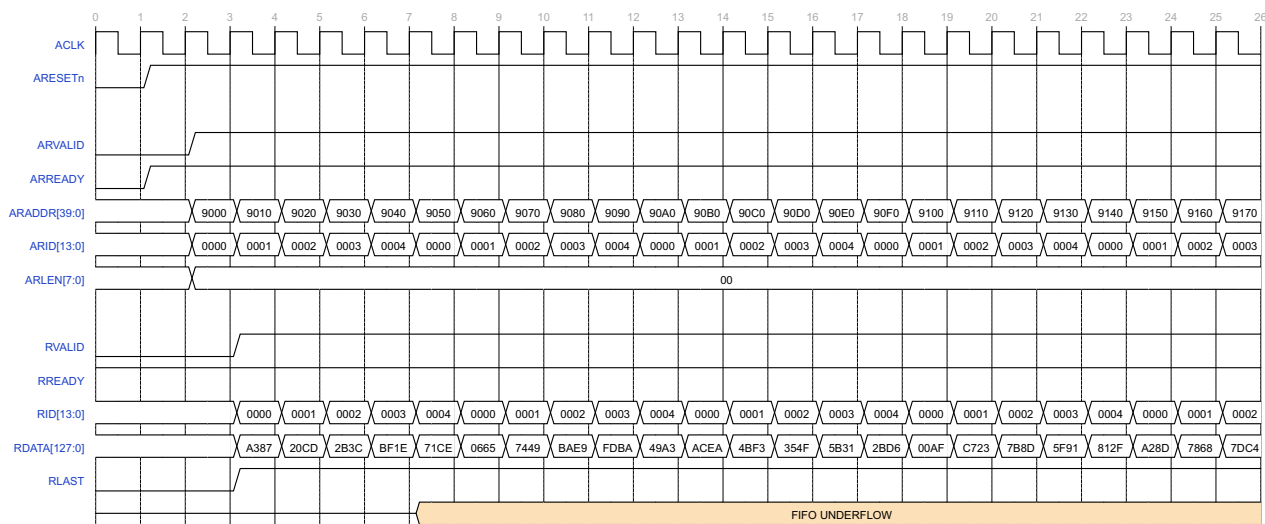


Figure D-9 Read with FIFO underflow

## D.6 Write with FIFO overflow

This example describes how the FIFO saturates and reaches a steady state when the fill rate is larger than the transaction size.

### D.6.1 Configuration

Table D-6 Write with FIFO overflow profile

Name	Parameter	Value
FIFO Startup Level	Start	Empty
FIFO Depth	Full	64
Outstanding Transaction Limit	TxnLimit	32
Fill Rate	Rate	18
Transaction Size	TxnSize	32
Address Pattern	-	sequential
Starting Address	Base	0x9000
Range	Range	0x200
ID Type	-	cycle
ID Value	Value	0

### D.6.2 Timing diagram

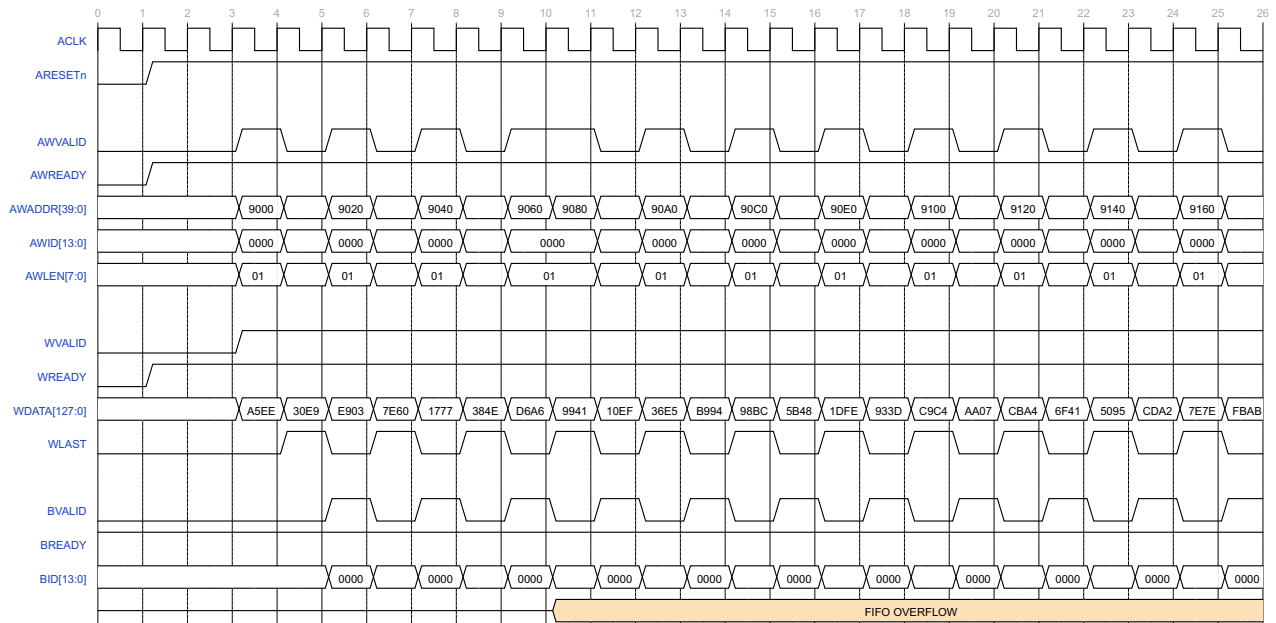


Figure D-10 Write with FIFO overflow profile

D.7Read with delayed slave

- This example demonstrates:
- How read latency affects the behavior of a master using a profile.
  - The use of a two-dimensional striding address pattern.

D.7.1Configuration

Table D-7 Read with delayed slave profile

Name	Parameter	Value
FIFO Startup Level	Start	Empty
FIFO Depth	Full	128
Outstanding Transaction Limit	TxnLimit	30
Drain Rate	Rate	12
Transaction Size	TxnSize	32
Address Pattern	-	twodim
Starting Address	Base	0x8000
X Range	XRange	0x80
Y Range	YRange	0x800
Stride	Stride	0x200
ID Type	-	fixed
ID Value	Value	0

D.7.2Timing diagram

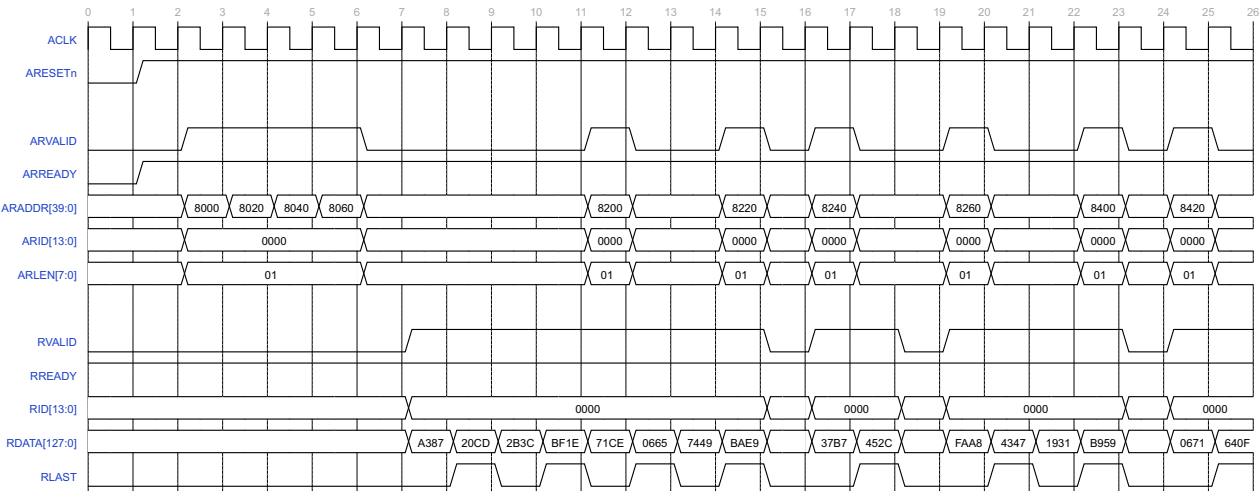


Figure D-11 Read with delayed slave (1 of 2)

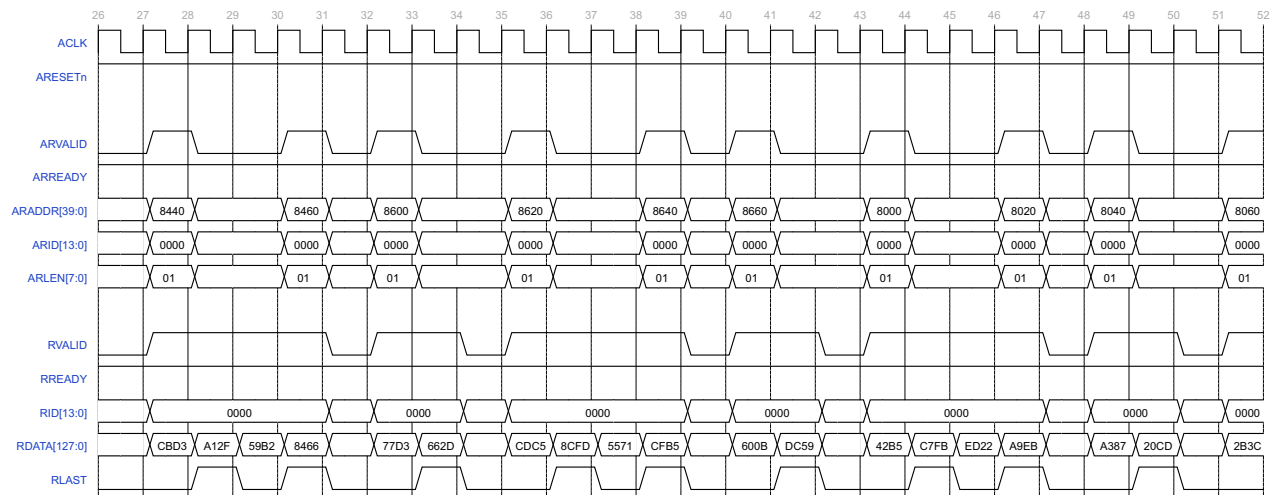


Figure D-12 Read with delayed slave (2 of 2)

## D.8 Read gated by Outstanding Transaction limit

This example describes the effect of Outstanding Transaction limit on issuing new transactions.

### D.8.1 Configuration

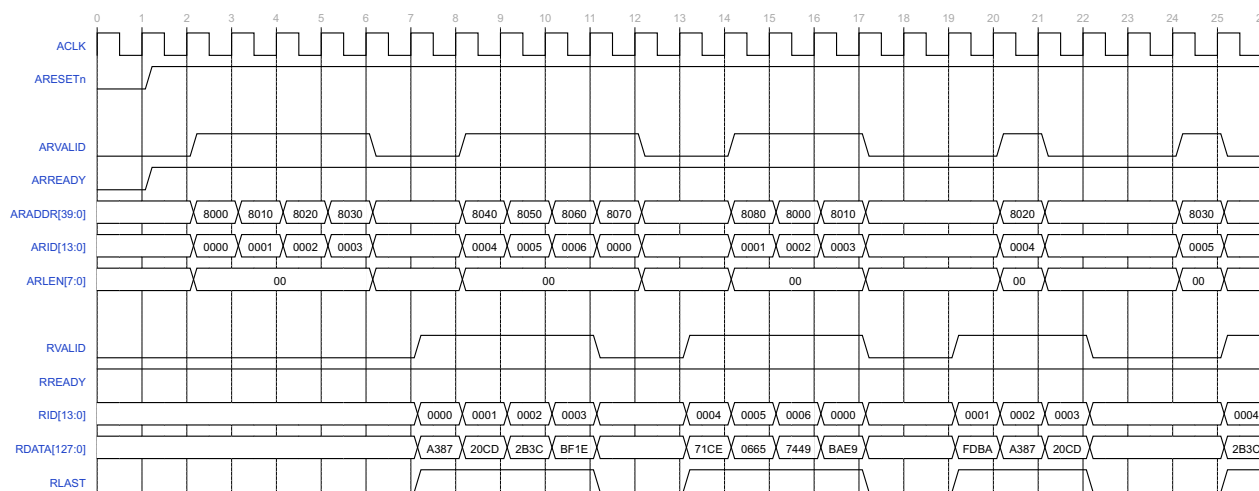
### Table D-8 Read gated by Outstanding Transaction limit profile

Name	Parameter	Value
FIFO Startup Level	Start	Empty
FIFO Depth	Full	144
Outstanding Transaction Limit	TxnLimit	4
Drain Rate	Rate	4
Transaction Size	TxnSize	16
Address Pattern	-	sequential
Starting Address	Base	0x8000
Range	Range	0x90
ID Type	-	cycle
Cycle Lower Value	LowerValue	0
Cycle Upper Value	UpperValue	6
Frame Size	FrameSize	256

### D.8.2 Timing diagram

The following items are interesting to note:

- In clock cycle 6 the master reaches the outstanding transactions limit, so does not issue any more requests.
- In clock cycle 7 the slave responds, its profile has a fixed read latency (RIV) of 5 cycles.
- In clock cycle 8 the master can issue a request because a transaction completed in the previous cycle.



**Figure D-13 Read gated by Outstanding Transaction limit (1 of 2)**

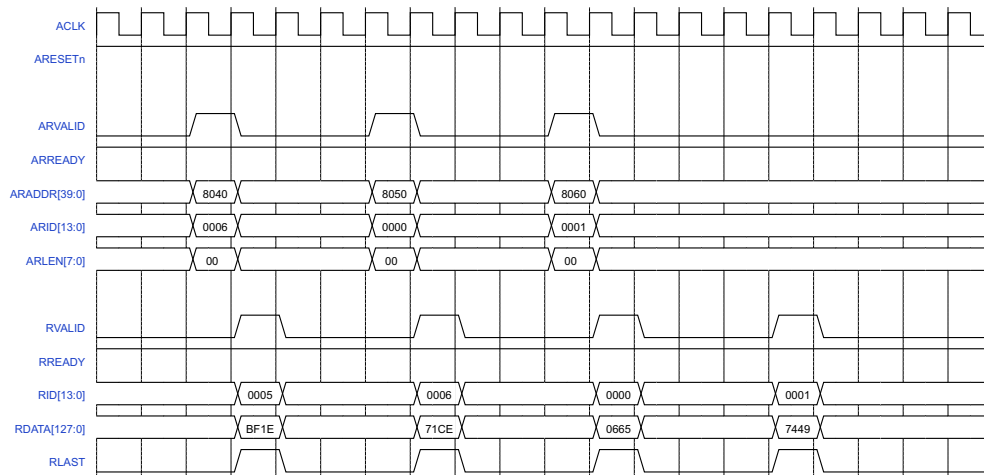


Figure D-14 Read gated by Outstanding Transaction limit (2 of 2)

## D.9 Write influenced by slave profile

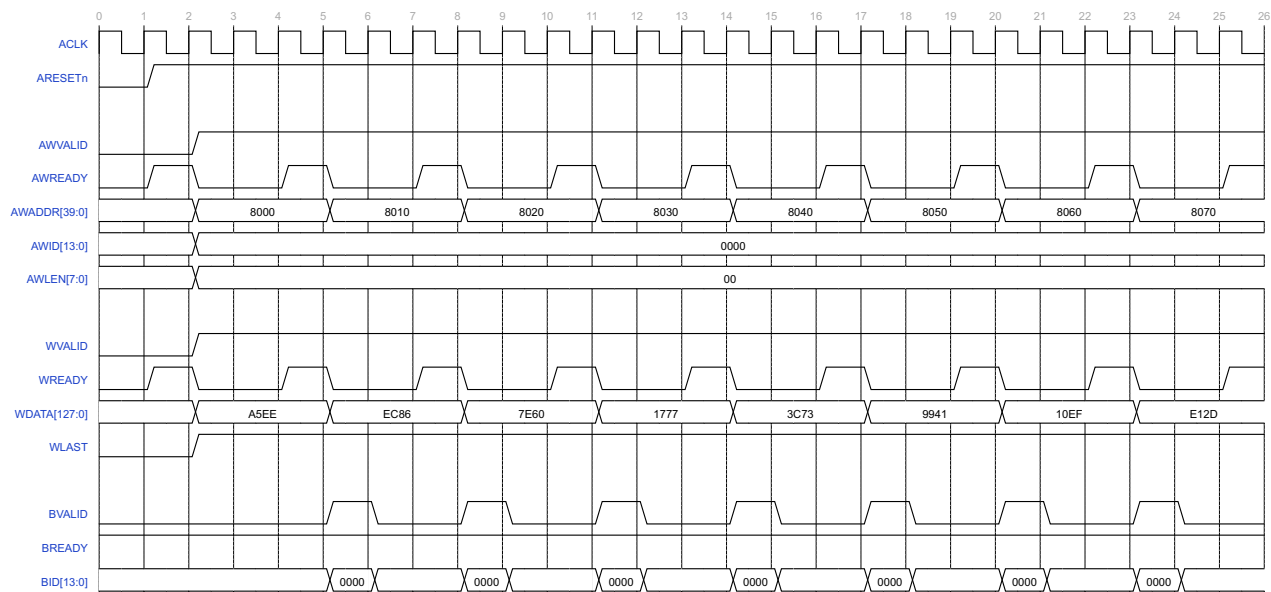
This example describes the effect on the master traffic profile of a slave deasserting **AWREADY** and **WREADY**.

### D.9.1 Configuration

Table D-9 Write influenced by slave profile

Name	Parameter	Value
FIFO Startup Level	Start	Full
FIFO Depth	Full	64
Outstanding Transaction Limit	TxnLimit	30
Fill Rate	Rate	4
Transaction Size	TxnSize	16
Address Pattern	-	sequential
Starting Address	Base	0x8000
Range	Range	0x80
ID Type	-	fixed
ID Value	Value	0

### D.9.2 Timing diagram





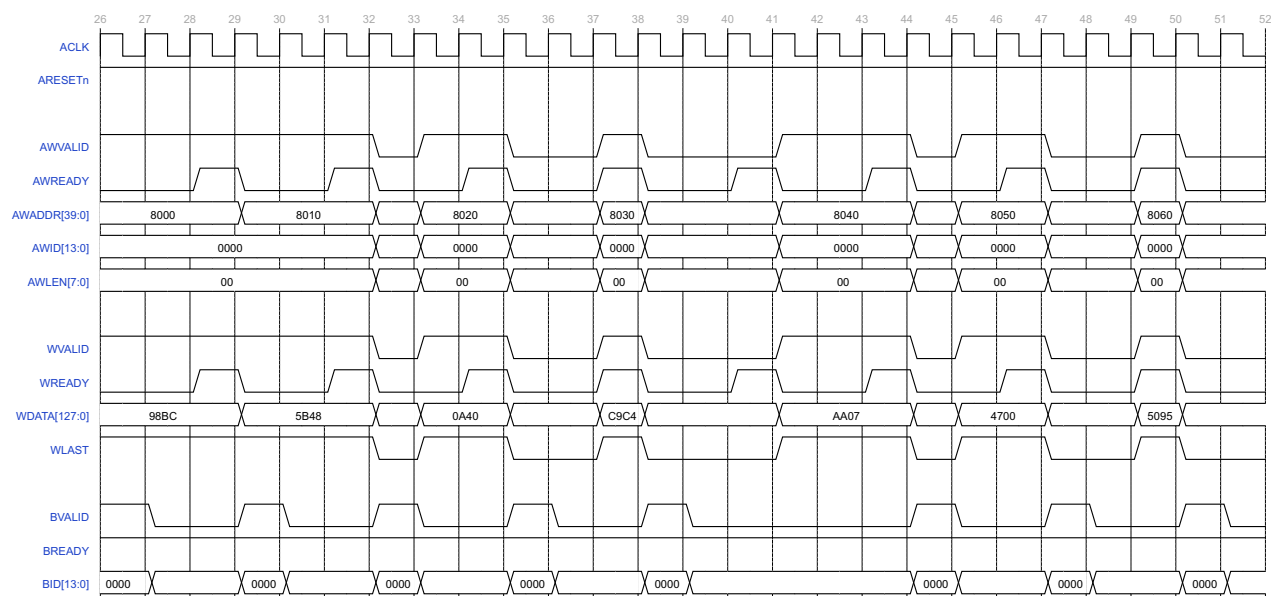


Figure D-16 Write influenced by slave profile (2 of 2)



# Appendix E

## Revisions

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

Table E-1 Issue A	
Change	Location
First release of Version A	–

