Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

Advanced Microcontroller Bus Architecture system

School of Computer, Wuhan University, Wuhan 430072, China
(2013)

Abstract

The role of the arbiter in an AMBA system is to control which master has access to the bus. Every bus master has a two wire REQUEST and GRANT interface to the arbiter and on every cycle the arbiter uses a prioritization scheme to decide which bus master is currently the highest priority master requesting the bus.

A shared bus lock signal, **BLOK**, driven by the currently granted bus master is used to indicate that the current transfer is indivisible from the following transfer and no other master should be granted the bus.

The detail of the priority scheme is not specified and is defined for each application. It is acceptable for the arbiter to use other signals, either AMBA or non-AMBA, to influence the priority scheme that is in use.

Introduction

The bus can be re-arbitrated on every clock cycle. The arbiter samples all the request signals, **AREQx**, on the falling edge of **BCLK** and during the LOW phase of **BCLK** the arbiter asserts the appropriate **AGNTx** signal using the internal priority scheme and the value of **BLOK**.

As the arbitration can change every cycle, it is possible that during an extended transfer, the highest priority bus master may change several times before the transfer eventually completes. The bus master that has **AGNT** asserted when the transfer completes will

During bus master handover the **BLOK** signal is not driven and hence the arbiter must assume that this signal is LOW.

The arbiter must retain a copy of which master is currently granted so it can:

- keep the current bus master granted if **BLOK** is asserted
- determine when the bus master changes, and so determine when there is a cycle of bus master handover.

4.13.3 Timing diagrams

Figure 4-43 shows the arbiter timing parameters.

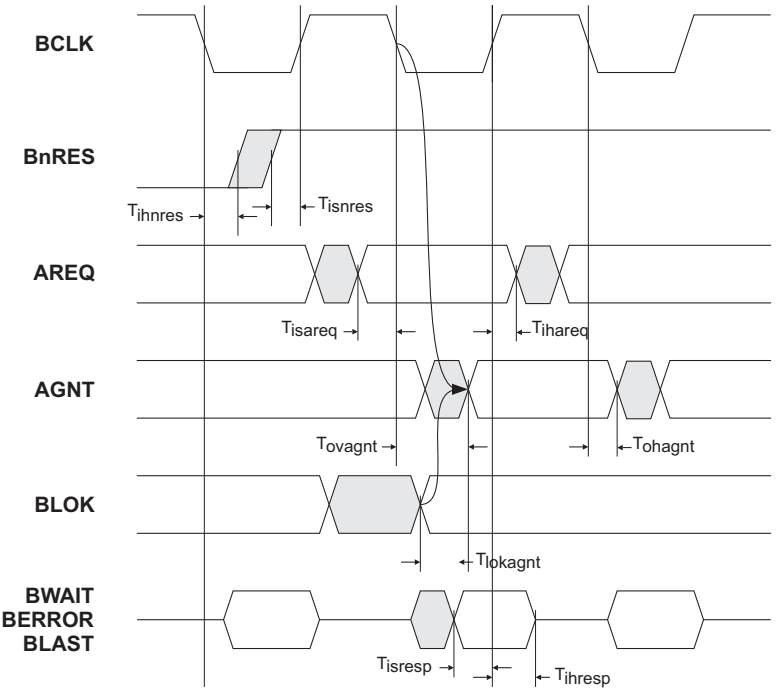

Figure 4-43 ASB arbiter parameters

4.13.4 Timing parameters

The timing parameters related to an ASB arbiter are given in the following tables:

- Table 4-14 is for input signals
- Table 4-15 is for output signals
- Table 4-16 is for combinatorially generated outputs.

Table 4-14 ASB arbiter input parameters

Parameter	Description
T_{clkl}	**BCLK** LOW time
T_{clkh}	**BCLK** HIGH time
T_{isnres}	**BnRES** de-asserted setup to rising **BCLK**
T_{ihnres}	**BnRES** de-asserted hold after falling **BCLK**
T_{isareq}	**AREQ** setup to falling **BCLK**
T_{ihareq}	**AREQ** hold after rising **BCLK**
T_{isresp}	**BWAIT** setup to rising **BCLK**
T_{ihresp}	**BWAIT** hold after rising **BCLK**

Table 4-15 ASB arbiter output parameters

Parameter	Description
T_{ovagnt}	**AGNT** valid after falling **BCLK**
T_{ohagnt}	**AGNT** hold after falling **BCLK**

Table 4-16 ASB arbiter combinatorial parameters

Parameter	Description
$T_{lokagnt}$	Delay from valid **BLOK** to valid **AGNT**

AMBA APB

This chapter introduces the *Advanced Microcontroller Bus Architecture* (AMBA) *Advanced Peripheral Bus* (APB) specification in the following sections:

5.1 About the AMBA APB

The *Advanced Peripheral Bus* (APB) is part of the *Advanced Microcontroller Bus Architecture* (AMBA) hierarchy of buses and is optimized for minimal power consumption and reduced interface complexity.

The latest revision of the APB ensures that all signal transitions are only related to the rising edge of the clock. This improvement means the APB peripherals can be integrated easily into any design flow, with the following advantages:

The AMBA APB should be used to interface to any peripherals which are low-bandwidth and do not require the high performance of a pipelined bus interface.

- performance is improved at high-frequency operation
- performance is independent of the mark-space ratio of the clock
- static timing analysis is simplified by the use of a single clock edge
- no special considerations are required for automatic test insertion
- many *Application-Specific Integrated Circuit* (ASIC) libraries have a better selection of rising edge registers
- easy integration with cycle based simulators.

These changes to the APB also make it simpler to interface it to the new *Advanced High-performance Bus* (AHB).

5.1.1 A typical AMBA-based microcontroller

An AMBA-based microcontroller typically consists of a high-performance system *backbone* bus, able to sustain the external memory bandwidth, on which the CPU and other *Direct Memory Access* (DMA) devices reside, plus a bridge to a narrower APB bus on which the lower bandwidth peripheral devices are located. Figure 5-1 shows the APB in a typical AMBA system.

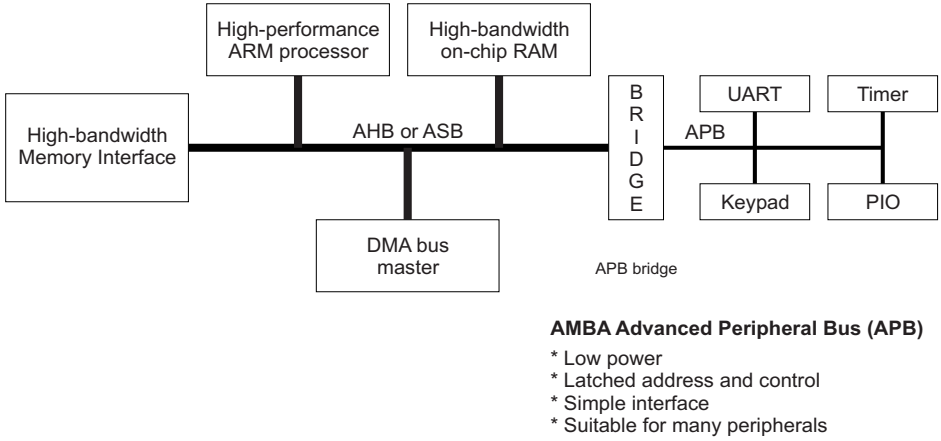

Figure 5-1 The APB in a typical AMBA system

A system bus that includes a *Test Interface Controller* (TIC) allows modular testing of both system bus and APB modules.

5.2 APB specification

The APB specification is described under the following headings:

- *State diagram*
- *Write transfer* on page 5-5
- *Read transfer* on page 5-6.

5.2.1 State diagram

The state diagram, shown in Figure 5-2, can be used to represent the activity of the peripheral bus.

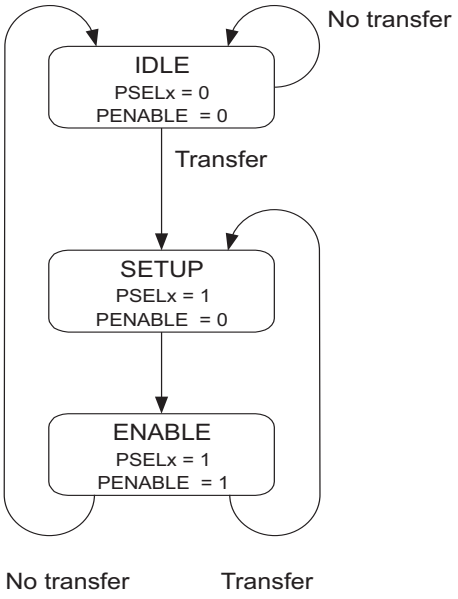

Figure 5-2 State diagram

Operation of the state machine is through the three states described below:

IDLE — The default state for the peripheral bus.

SETUP — When a transfer is required the bus moves into the SETUP state, where the appropriate select signal, **PSELx**, is asserted. The bus only remains in the SETUP state for one clock cycle and will always move to the ENABLE state on the next rising edge of the clock.

ENABLE

In the ENABLE state the enable signal, **PENABLE** is asserted. The address, write and select signals all remain stable during the transition from the SETUP to ENABLE state.

The ENABLE state also only lasts for a single clock cycle and after this state the bus will return to the IDLE state if no further transfers are required. Alternatively, if another transfer is to follow then the bus will move directly to the SETUP state.

It is acceptable for the address, write and select signals to glitch during a transition from the ENABLE to SETUP states.

5.2.2 Write transfer

The basic write transfer is shown in Figure 5-3.

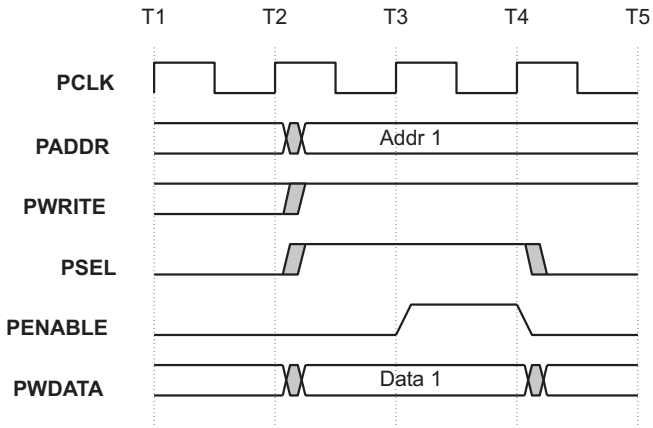

Figure 5-3 Write transfer

The write transfer starts with the address, write data, write signal and select signal all changing after the rising edge of the clock. The first clock cycle of the transfer is called the SETUP cycle. After the following clock edge the enable signal **PENABLE** is asserted, and this indicates that the ENABLE cycle is taking place. The address, data and control signals all remain valid throughout the ENABLE cycle. The transfer completes at the end of this cycle.

The enable signal, **PENABLE**, will be deasserted at the end of the transfer. The select signal will also go LOW, unless the transfer is to be immediately followed by another transfer to the same peripheral.

In order to reduce power consumption the address signal and the write signal will not change after a transfer until the next access occurs.

The protocol only requires a clean transition on the enable signal. It is possible that in the case of back to back transfers the select and write signals may glitch.

5.2.3 Read transfer

Figure 5-4 shows a read transfer.

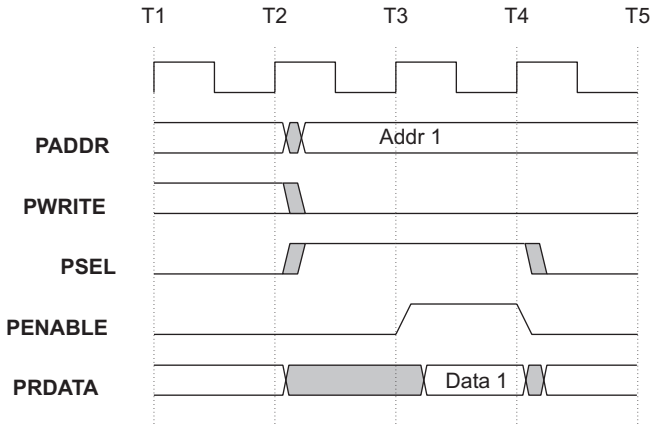

Figure 5-4 Read transfer

The timing of the address, write, select and strobe signals are all the same as for the write transfer. In the case of a read, the slave must provide the data during the ENABLE cycle. The data is sampled on the rising edge of clock at the end of the ENABLE cycle.

5.3 About the APB AMBA components

The following notation is used for the timing parameters:

- T_{is} - input setup time
- T_{ih} - input hold time
- T_{ov} - output valid time
- T_{oh} - output hold time.

5.4 APB bridge

The APB bridge is the only bus master on the AMBA APB. In addition, the APB bridge is also a slave on the higher-level system bus.

5.4.1 Interface diagram

Figure 5-5 shows the APB signal interface of an APB bridge.

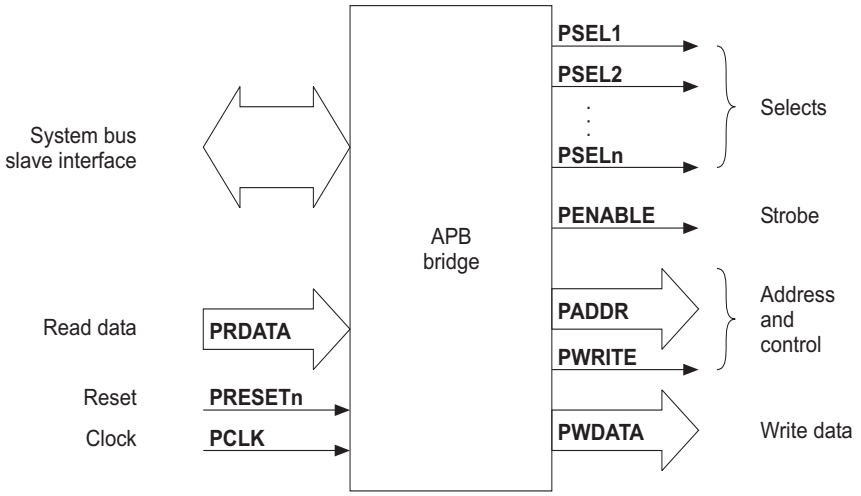

Figure 5-5 APB bridge interface diagram

5.4.2 APB bridge description

The bridge unit converts system bus transfers into APB transfers and performs the following functions:

- Latches the address and holds it valid throughout the transfer.
- Decodes the address and generates a peripheral select, **PSELx**. Only one select signal can be active during a transfer.
- Drives the data onto the APB for a write transfer.
- Drives the APB data onto the system bus for a read transfer.
- Generates a timing strobe, **PENABLE**, for the transfer.

5.4.3 Timing diagrams

The timing parameters for an APB bridge are shown in Figure 5-6.

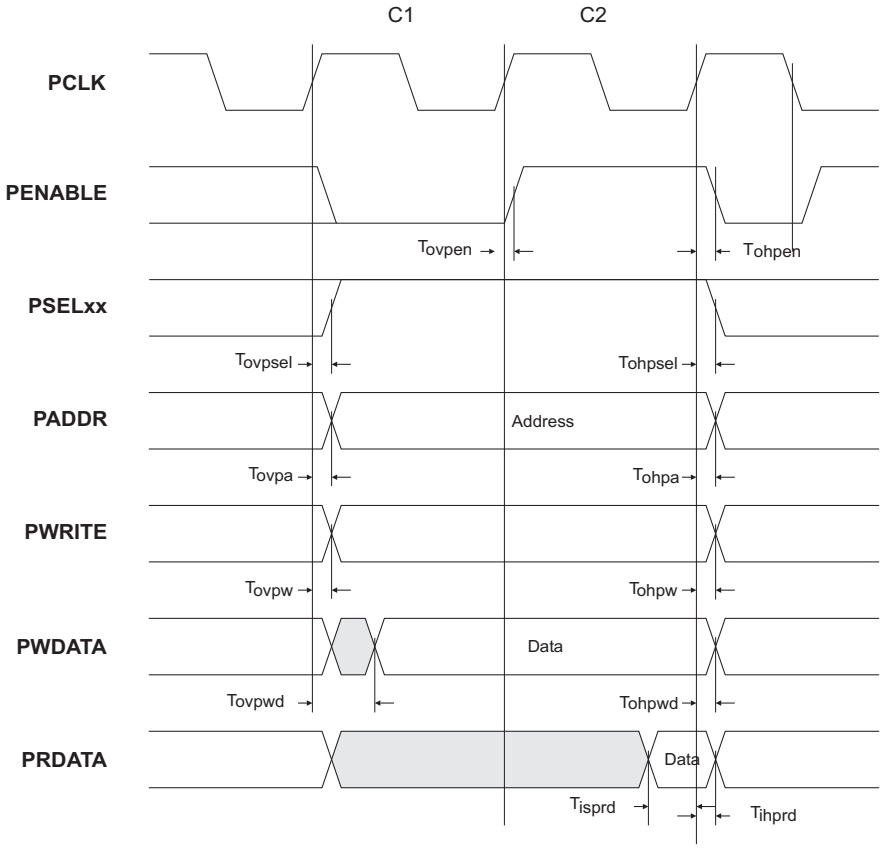

Figure 5-6 APB bridge transfer

5.4.4 Timing parameters

The timing parameters related to an APB bridge are given in Table 5-1 for input signals and Table 5-2 for output signals.

Table 5-1 APB bridge input parameters

Parameter	Description
T_{clkl}	**PCLK** LOW time
T_{clkh}	**PCLK** HIGH time
T_{isnres}	**PRESETn** de-asserted setup to rising **PCLK**
T_{ihnres}	**PRESETn** de-asserted hold after rising **PCLK**
T_{isprd}	For read transfers, **PRDATA** setup to rising **PCLK**
T_{ihprd}	For read transfers, **PRDATA** hold after rising **PCLK**

Table 5-2 APB bridge output parameters

Parameter	Description
T_{ovpen}	**PENABLE** valid after rising **PCLK**
T_{ohpen}	**PENABLE** hold after rising **PCLK**
T_{ovpsel}	**PSEL** valid after rising **PCLK**
T_{ohpsel}	**PSEL** hold after rising **PCLK**
T_{ovpa}	**PADDR** valid after rising **PCLK**
T_{ohpa}	**PADDR** hold after rising **PCLK**
T_{ovpw}	**PWRITE** valid after rising **PCLK**
T_{ohpw}	**PWRITE** hold after rising **PCLK**
T_{ovpwd}	For write transfers, **PWDATA** valid after rising **PCLK**
T_{ohpwd}	For write transfers, **PWDATA** hold after rising **PCLK**

5.5 APB slave

APB slaves have a simple, yet flexible, interface. The exact implementation of the interface will be dependent on the design style employed and many different options are possible.

5.5.1 Interface diagram

Figure 5-7 shows the signal interface of an APB slave.

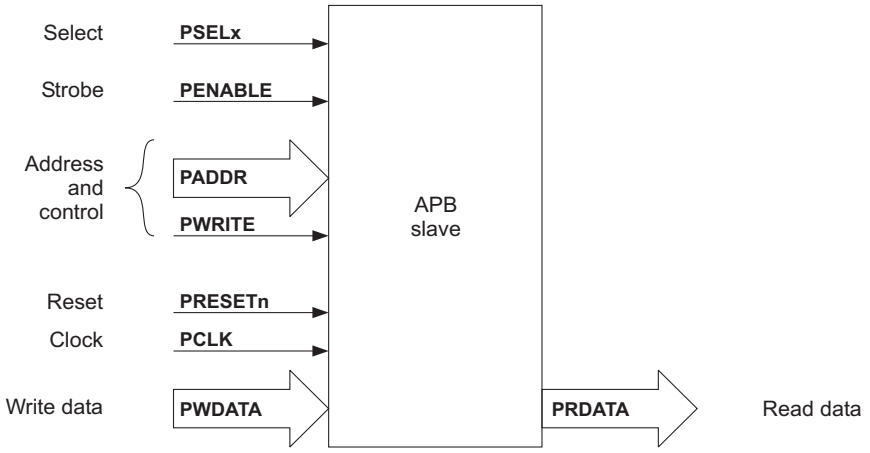

Figure 5-7 APB slave interface description

5.5.2 APB slave description

The APB slave interface is very flexible.

For a write transfer the data can be latched at the following points:

- on either rising edge of **PCLK**, when **PSEL** is HIGH
- on the rising edge of **PENABLE**, when **PSEL** is HIGH.

The select signal **PSELx**, the address **PADDR** and the write signal **PWRITE** can be combined to determine which register should be updated by the write operation.

For read transfers the data can be driven on to the data bus when **PWRITE** is LOW and both **PSELx** and **PENABLE** are HIGH. While **PADDR** is used to determine which register should be read.

5.5.3 Timing diagrams

The timing parameters related to an access to an APB bus slave are shown in Figure 5-8.

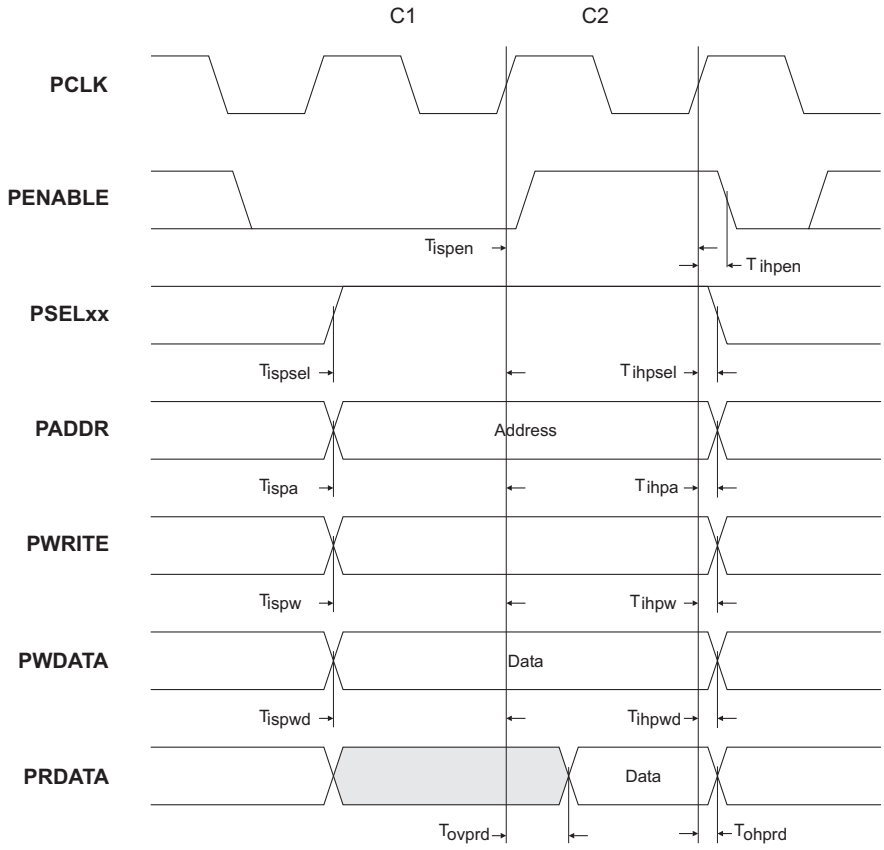

Figure 5-8 APB slave transfer

5.5.4 Timing parameters

The timing parameters related to an APB slave are given in Table 5-3 for input signals and Table 5-4 for output signals.

Table 5-3 APB slave input parameters

Parameter	Description
T_{clkl}	**PCLK** LOW time
T_{clkh}	**PCLK** HIGH time
T_{isnres}	**PRESETn** de-asserted setup to rising **PCLK**
T_{ihnres}	**PRESETn** de-asserted hold after falling **PCLK**
T_{ispen}	**PENABLE** setup to rising **PCLK**
T_{ihpen}	**PENABLE** hold after rising **PCLK**
T_{ispsel}	**PSEL** setup to rising **PCLK**
T_{ihpsel}	**PSEL** hold after rising **PCLK**
T_{ispa}	**PADDR** setup to rising **PCLK**
T_{ihpa}	**PADDR** hold after rising **PCLK**
T_{ispw}	**PWRITE** setup to rising **PCLK**
T_{ihpw}	**PWRITE** hold after rising **PCLK**
T_{ispwd}	For write transfers, **PWDATA** setup to rising **PCLK**
T_{ihpwd}	For write transfers, **PWDATA** hold after rising **PCLK**

Table 5-4 APB slave output parameters

Parameter	Description
T_{ovprd}	For read transfers, **PRDATA** valid after rising **PCLK**
T_{ohprd}	For read transfers, **PRDATA** hold after rising **PCLK**

5.6 Interfacing APB to AHB

Interfacing the AMBA APB to the AHB is described in:

- *Read transfers*
- *Write transfers* on page 5-16
- *Back to back transfers* on page 5-18
- *Tristate data bus implementations* on page 5-19.

5.6.1 Read transfers

Figure 5-9 illustrates a read transfer.

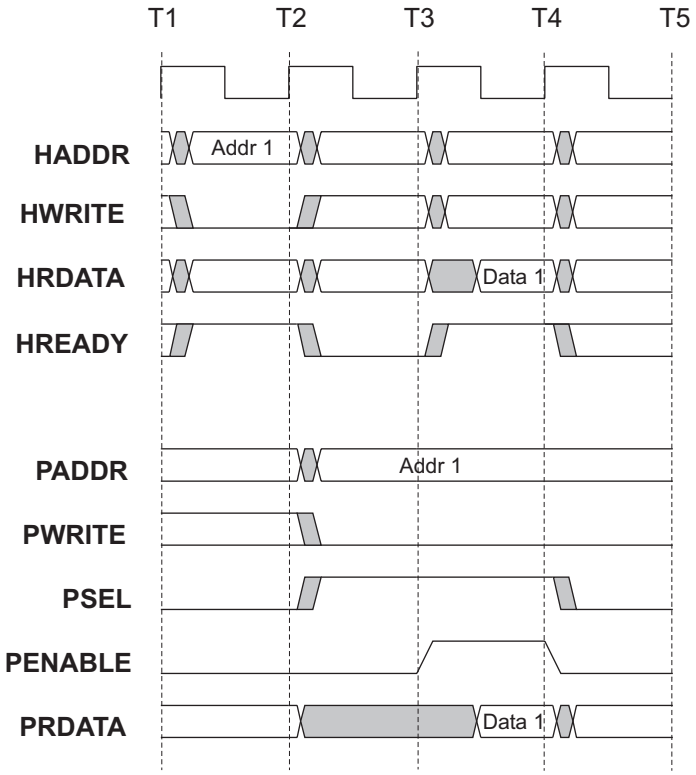

Figure 5-9 Read transfer to AHB

The transfer starts on the AHB at time T1 and the address is sampled by the APB bridge at T2. If the transfer is for the peripheral bus then this address is broadcast and the appropriate peripheral select signal is generated. This first cycle on the peripheral bus is called the SETUP cycle, this is followed by the ENABLE cycle, when the **PENABLE** signal is asserted.

During the ENABLE cycle the peripheral must provide the read data. Normally it will be possible to route this read data directly back to the AHB, where the bus master can sample it on the rising edge of the clock at the end of the ENABLE cycle, which is at time T4 in Figure 5-9.

In very high clock frequency systems it may become necessary for the bridge to register the read data at the end of the ENABLE cycle and then for the bridge to drive this back to the AHB bus master in the following cycle. Although this will require an extra wait state for peripheral bus read transfers, it allows the AHB to run at a higher clock frequency, thus resulting in an overall improvement in system performance. A burst of read transfers is shown in Figure 5-10. All read transfers require a single wait state.

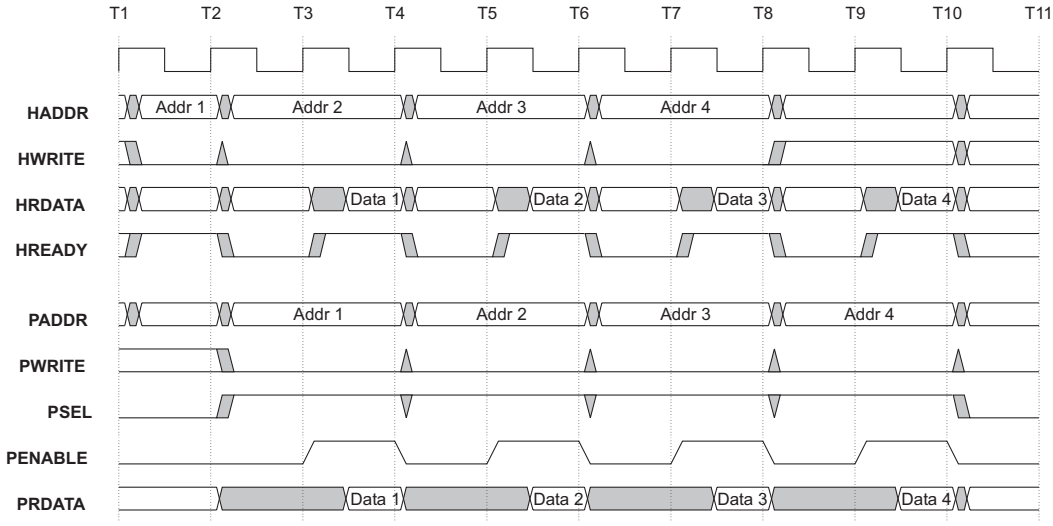

Figure 5-10 Burst of read transfers

5.6.2 Write transfers

Figure 5-11 shows a write transfer.

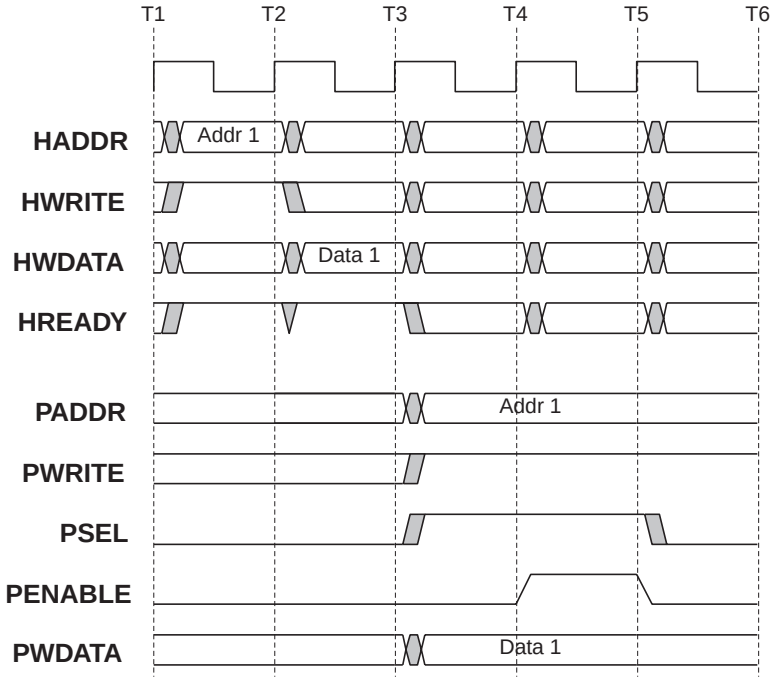

Figure 5-11 Write transfer from AHB

Single write transfers to the APB can occur with zero wait states. The bridge is responsible for sampling the address and data of the transfer and then holding these values for the duration of the write transfer on the APB.

REFERENCES

[1] J. Campbell, "Speaker recognition: a tutorial," *Proc. IEEE*, vol. 85, pp. 1437–1462, Sept. 1997.

[2] D. A. Reynolds, T. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1–3, pp. 19–41, 2000.

[3] D. A. Reynolds, "Comparison of background normalization methods for text-independent speaker verification," in *Proc. Eurospeech*, 1997.

[4] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Trans. Speech Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[5] J. L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 291–298, Apr. 1994.

[6] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1993, pp. 692–695.

[7] K. M. Knill, M. J. F. Gales, and S. J. Young, "Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs," in *Proc. Int. Conf. Spoken Language Processing*, 1996.

[8] D. B. Paul, "An investigation of Gaussian shortlists," in *Proc. Automatic Speech Recognition and Understanding Workshop*, 1999.

[9] T. Watanabe, K. Shinoda, K. Takagi, and K.-I. Iso, "High speed speech recognition using tree-structured probability density function," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1995.

[10] J. Simonin, L. Delphin-Poulat, and G. Damnati, "Gaussian density tree structure in a multi-Gaussian HMM-based speech recognition system,"

[11] T. J. Hanzen and A. K. Halberstadt, "Using aggregation to improve the performance of mixture Gaussian acoustic models," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1998.

[12] M. Padmanabhan, L. R. ahl, and D. Nahamoo, "Partitioning the feature space of a classifier with linear hyperplanes," *IEEE Trans. Speech Audio Processing*, vol. 7, no. 3, pp. 282–288, 1999.
in *Proc. Int. Conf. Spoken Language Processing*, 1998.

[13] R. Auckenthaler and J. Mason, "Gaussian selection applied to text-independent speaker verification," in *Proc. A Speaker Odyssey—Speaker Recognition Workshop*, 2001.

[14] J. McLaughlin, D. Reynolds, and T. Gleason, "A study of computation speed-ups of the GMM-UBM speaker recognition system," in *Proc. Eurospeech*, 1999.

[15] S. van Vuuren and H. Hermansky, "On the importance of components of the modulation spectrum of speaker verification," in *Proc. Int. Conf. Spoken Language Processing*, 1998.

[16] B. L. Pellom and J. H. L. Hansen, "An efficient scoring algorithm for Gaussian mixture model based speaker identification," *IEEE Signal Processing Lett.*, vol. 5, no. 11, pp. 281–284, 1998.

[17] J. Oglesby and J. S. Mason, "Optimization of neural models for speaker identification," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1990, pp. 261–264.

[18] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network—hidden Markov model hybrid," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

[19] H. Bourlard and C. J. Wellekins, "Links between Markov models and multilayer perceptrons," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 1167–1178, Dec. 1990.

[20] J. Navrátil, U. V. Chaudhari, and G. N. Ramaswamy, "Speaker verification using target and background dependent linear transforms and multi-system fusion," in *Proc. Eurospeech*, 2001.

[21] L. P. Heck, Y. Konig, M. K. Sonmez, and M. Weintraub, "Robustness to telephone handset distortion in speaker recognition by discriminative feature design," *Speech Commun.*, vol. 31, pp. 181–192, 2000.

[22] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc.*, vol. 39, pp. 1–38, 1977.

[23] K. Shinoda and C. H. Lee, "A structural Bayes approach to speaker adaptation," *IEEE Trans. Speech Audio Processing*, vol. 9, no. 3, pp. 276–287, 2001.

[24] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* New York: Academic, 1990.

[25] J. C. Junqua, *Robust Speech Recogntion in Embedded Systems and PC*

[26] U. V. Chaudhari, J. Navrátil, S. H. Maes, and R. A. Gopinath, "Transformation enhanced multi-grained modeling for text-independent speaker recognition," in *Proc. Int. Conf. Spoken Language Processing*, 2000.

[27] Q. Lin, E.-E. Jan, C. W. Che, D.-S. Yuk, and J. Flanagan, "Selective use of the speech spectrum and a VQGMM method for speaker identification," in *Proc. Int. Conf. Spoken Language Processing*, 1996.

[28] S. Raudys, *Statistical and Neural Classifiers: An Integrated Approach to Design.* New York: Springer, 2001.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986, pp. 318–364.

[30] [Online] Available: http://www.nist.gov/speech/tests/spk/index.htm.

[31] J. Pelecanos and S. Sridharan, "Feature warping for robust speaker verification," in *Proc. A Speaker Odyssey—Speaker Recognition Workshop*, 2001.

[32] B. Xiang, U. V. Chaudhari, J. Navrátil, N. Ramaswamy, and R. A. Gopinath, "Short-time Gaussianization for robust speaker verification," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 2002.

[33] G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds, "The NIST speaker recognition evaluation—overview, methodology, systems, results, perspective," *Speech Communication*, vol. 31, pp. 225–254, 2000.

Bing Xiang (M'03) was born in 1973 in China. He received the B.S. degree in radio and electronics and M.E. degree in signal and information processing from Peking University in 1995 and 1998, respectively. In January, 2003, he received the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY.

From 1995 to 1998, he worked on speaker recognition and auditory modeling in National Laboratory on Machine Perception, Peking University. Then he entered Cornell University and worked on speaker recognition and speech recognition in DISCOVER Lab as a Research Assistant. He also worked in the Human Language Technology Department of IBM Thomas J. Watson Research Center as a summer intern in both 2000 and 2001. He was a selected remote member of the SuperSID Group in the 2002 Johns Hopkins CLSP summer workshop in which he worked on speaker verification with high-lelvel information. In January, 2003, he joined the Speech and Language Processing Department of BBN Technologies where he is presently a Senior Staff Consultant-Technology. His research interests include large vocabulary speech recognition, speaker recognition, speech synthesis, keyword spotting, neural networks and statistical pattern recognition.

Toby Berger (S'60–M'66–SM'74–F'78) was born in New York, NY, on September 4, 1940. He received the B.E. degree in electrical engineering from Yale University, New Haven, CT in 1962, and the M.S. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA in 1964 and 1966, respectively.

From 1962 to 1968 he was a Senior Scientist at Raytheon Company, Wayland, MA, specializing in communication theory, information theory, and coherent signal processing. In 1968 he joined the faculty of Cornell University, Ithaca, NY where he is presently the Irwin and Joan Jacobs Professor of Engineering. His research interests include information theory, random fields, communication networks, wireless communications, video compression, voice and signature compression and verification, neuroinformation theory, quantum information theory, and coherent signal processing. He is the author/co-author of Rate Distortion Theory: A Mathematical Basis for Data Compression, Digital Compression for Multimedia: Principles and Standards, and Information Measures for Discrete Random Fields.

Dr. Berger has served as editor-in-chief of the IEEE TRANSACTIONS ON INFORMATION THEORY and as president of the IEEE Information Theory