

## Features

- Facilitates Development of Embedded Microcontroller Products with One or More CPUs or Signal Processors
- Minimizes the Silicon Infrastructure Required to Support Efficient On-chip and Off-chip Communication for Both Operation and Manufacturing Test
- Independent of a Specific Technology
- Ensures that Highly Reusable Peripheral and System Macrocells can be Migrated across a Diverse Range of IC Processes and are Appropriate for Full-custom, Standard-cell and Gate Array Technologies
- Encourages Modular System Design to Improve Processor Independence, Providing a Development Roadmap for Advanced Cached CPU Cores and the Development of Peripheral Libraries
- Use of Multiplexed Buses Facilitates Management of Data Buses and Provides Improved Testability

## Description

The system architecture derived from the specification for the Advanced Microcontroller Bus Architecture, AMBA™, defines an on-chip communications standard for designing high-performance 32-bit and 16-bit embedded microcontroller-based systems.

Two distinct buses are defined within the AMBA specification:

- the Advanced System Bus (ASB) for high-performance system modules
- the Advanced Peripheral Bus (APB) for low-power peripherals

ASB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions. The bus also provides the test infrastructure for modular macrocell test and diagnostic access.

APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions.

The ARM® AMBA specification, from which this document is derived, may be consulted under the reference ARM IHI0001D, dated April 1997.

## A Typical AMBA-based Microcontroller

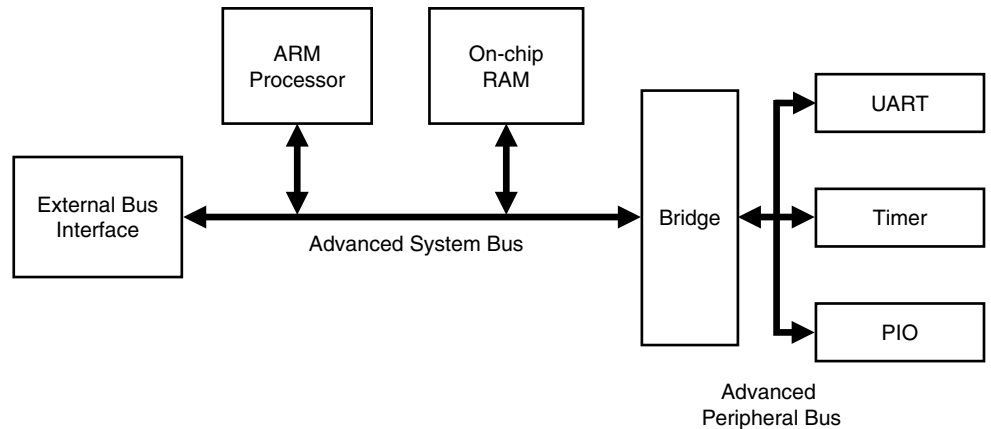
An AMBA-based microcontroller typically consists of a high-performance system backbone bus, able to sustain the external memory bandwidth, on which the CPU and other direct memory access devices reside, plus a bridge to a narrower APB bus on which the lower bandwidth peripheral devices are located.



## ARM7TDMI™ - based Microcontroller

## System Architecture

**Figure 1. A Typical AMBA System**



## Terminology

The following terms are used throughout this specification.

**Bus Cycle** – A bus cycle is a basic unit of one bus clock period and for the purpose of protocol descriptions, is defined from falling-edge to falling-edge transitions. Bus signal timing is referenced to the bus cycle clock.

**Bus Transfer** – A bus transfer is a read or write operation of a data object, which may take one or more bus cycles. The bus transfer is terminated by a “completion” response from the addressed slave.

The transfer sizes supported include byte (8-bit), half-word (16-bit) and word (32-bit) with a reserved code for future extension.

**Bus Master** – A bus master is a system module that is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

**Bus Slave** – A bus slave is a system module that responds to a read or write operation within a given address-space range. The bus slave signals back to the active master the success, failure or extension of the data transfer.

**Bus Arbiter** – The bus arbiter is a module that ensures that only one bus master at a time is allowed to initiate data transfers. The arbitration scheme is not enforced such that “highest priority” or “fair” access algorithms may be implemented, depending on the application requirements.

**Bus Decoder** – The bus decoder is a module that performs the decoding of the transfer addresses and selects slaves appropriately. The bus decoder also ensures that the bus remains operational when no bus transfers are required.

**Burst Operation** – A burst operation is defined as one or more data transactions initiated by a bus master, which have a consistent width of transaction to an incremental region of address space. The increment step per transaction is determined by the width of transfer (byte, half-word or word).

Bursts may be of arbitrary length and can be broken down into smaller packets by bus slaves, which may not be able to accept burst operations, either over a particular address boundary or over a particular burst length.

## Signal Prefixes

This section contains a brief description of the signals associated with an AMBA-based system and a full description of each of the signals may be found in later sections of this document.

The first letter of the signal name is used to indicate the signal connectivity:

- A= unidirectional signal between bus masters and the arbiter
- B = ASB signal
- D = unidirectional signal from the bus decoder
- P = APB signal

## Signal List

**Table 1.** Signal List

Name	Description
AGNTx	Bus Grant. A signal from the bus arbiter to a bus master “x”, which indicates that the bus master will be granted the bus when BWAIT is low. There is an AGNTx signal for each bus master in the system, as well as an associated bus request signal, AREQx.
AREQx	Bus Request. A signal from bus master “x” to the bus arbiter, which indicates that the bus master requires the bus. There is an AREQx signal for each bus master in the system, as well as an associated bus grant signal, AGNTx.
BA[31:0]	Address Bus. The system address bus, which is driven by the active bus master.
BCLK	Bus Clock. This clock times all bus transfers. Both the low phase and high phase of BCLK are used to control transfers on the bus.
BRDATA[31:0]	ASB Read Data Bus. This is the system read data bus. The read data bus is driven by the selected bus slave during read transfers.
BWDATA[31:0]	ASB Write Data Bus. This is the system write data bus. The write data bus is driven by the current bus master during write transfers.
BERROR	Error Response. A transfer error is indicated by the selected bus slave using the BERROR signal. When BERROR is high, a transfer error has occurred; when BERROR is low, then the transfer is successful. This signal is also used in combination with the BLAST signal to indicate a bus retract operation. When no slave is selected, this signal is driven by the bus decoder.
BLAST	Last Response. This signal is driven by the selected bus slave to indicate if the current transfer should be the last of a burst sequence. When BLAST is high, the decoder must allow sufficient time for address decoding. When BLAST is low, the next transfer may continue a burst sequence. This signal is also used in combination with the BERROR signal to indicate a bus retract operation. When no slave is selected, this signal is driven by the bus decoder.
BLOKx	Locked Transfers. An active high signal from the bus master “x” to the arbiter indicates that the current transfer and the next transfer are to be indivisible and no other bus master should be given access to the bus. This signal is used by the bus arbiter. There is a BLOKx signal for each ASB bus master.
BnRES	Reset. The bus reset signal is active low and is used to reset the system and the bus. This is the only active low signal.
BPROT[1:0]	Protection Control. The protection control signals provide additional information about a bus access and are primarily intended for use by a bus decoder when acting as a basic protection unit. The signals indicate if the transfer is an opcode fetch or data access, as well as if the transfer is a supervisor mode access or user mode access. The signals are driven by the active bus master and have the same timing as the address bus.

**Table 1.** Signal List (Continued)

Name	Description
BSIZE[1:0]	Transfer Size. The transfer size signals indicate the size of the transfer, which may be byte, half-word or word. The signals are driven by the active bus master and have the same timing as the address bus.
BTRAN[1:0]	Transfer Type. These signals indicate the type of the next transaction, which may be address-only, non-sequential or sequential. These signals are driven by a bus master when the appropriate AGNTx signal is asserted.
BWAIT	Wait Response. This signal is driven by the selected bus slave to indicate if the current transfer may complete. If BWAIT is high, a further bus cycle is required; if BWAIT is low, then the transfer may complete in the current bus cycle. When no slave is selected, this signal is driven by the bus decoder.
BWRITE	Transfer Direction. When high, this signal indicates a write transfer and when low, a read transfer. This signal is driven by the active bus master and has the same timing as the address bus.
DSELx	Slave Select. A signal from the bus decoder to a bus slave “x”, which indicates that the slave device is selected and a data transfer is required. There is a DSELx signal for each ASB bus slave.
PA[31:0]	APB Address Bus. This is the APB address bus, which may be up to 32 bits wide and is driven by the peripheral bus bridge unit.
PRDATA[31:0]	APB Read Data Bus. The peripheral read data bus is driven by the selected peripheral bus slave during read cycles (when PWRITE is low). This data bus may be up to 32 bits wide.
PWDATA[31:0]	APB Write Data Bus. The peripheral write data bus is driven by the peripheral bus bridge unit during write cycles (when PWRITE is high).
PSELx	APB Select. A signal from the secondary decoder, within the peripheral bus bridge unit, to each peripheral bus slave “x”. This signal indicates that the slave device is selected and a data transfer is required. There is a PSELx signal for each bus slave.
PSTB	APB Strobe. This strobe signal is used to time all accesses on the peripheral bus. The falling edge of PSTB is coincident with the falling edge of BCLK.
PSTB_RISING	This signal can be used as a clock signal to time all write transfers into peripherals. PSTB_RISING is derived from the rising edge of BCLK. PSTB_RISING changes only when a peripheral is accessed.
PWRITE	APB Transfer Direction. When high, this signal indicates an APB write access and when low, a read access.

## Functional Description

This section introduces the AMBA system hierarchy, defining the high-performance ASB and the low-power APB.

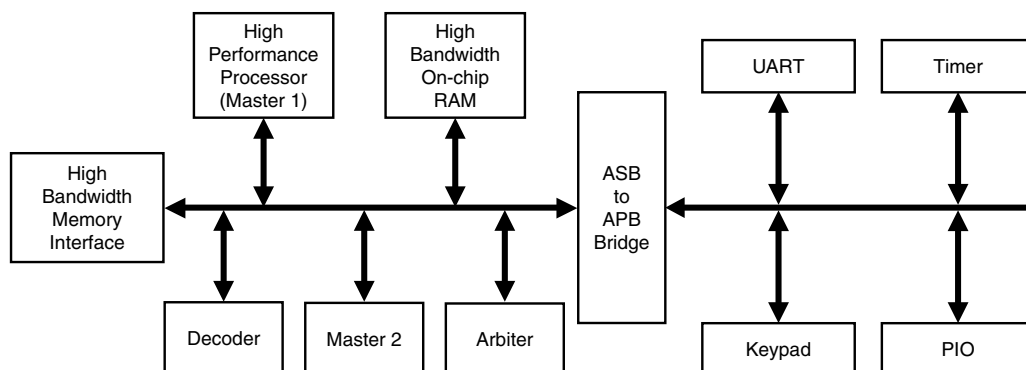
The ASB transfer mechanism is described, starting with a basic outline and then a more detailed explanation of the transfer types and transfer responses. This is followed by details of the arbitration and reset processes. Finally, APB transfers are also described.

## AMBA Hierarchy

A typical AMBA-based microcontroller is shown in Figure 2. The processor, on-chip memory and external bus interface all reside on the high-performance system bus. This bus provides a high bandwidth interface between the elements that are involved in the majority of transfers. Also located on the high-performance ASB is a bridge to the lower bandwidth APB, where most of peripherals in the system reside.

The APB provides the basic peripheral macrocell communications infrastructure as a secondary bus from the higher bandwidth pipelined main system bus. Such peripherals typically have interfaces that are memory-mapped registers, but have no high bandwidth interfaces and are accessed under programmed control.

**Figure 2. A Typical AMBA System**



## ASB and APB

The APB appears as a local secondary bus that is encapsulated as a single ASB slave device. The APB provides a low-power extension to the system bus, which builds on ASB signals directly.

The APB bridge appears as a slave module that handles the bus handshake and control signal retiming on behalf of the local peripheral bus. By defining the APB interface from the starting point of the system bus, the benefits of the system diagnostics and test methodology can be exploited.

## When is ASB Less Appropriate than APB?

Building all peripherals as fully functional ASB modules is feasible but may not always be desirable for the following reasons:

- In designs with a large number of peripheral macrocells, the increased bus loading may increase power dissipation and sacrifice performance.
- Where timing analysis is required, the slowest element on the bus will limit the maximum performance.
- Many simple peripheral macrocells need latched addresses and control signals as opposed to the high-bandwidth macrocells, which benefit from pipelined signaling.
- Many peripheral functions simply require a selection “strobe”, which conveys macrocell selection and read/write bus operation without the requirement to broadcast the high-frequency BCLK signal to every peripheral.
- Typically, macrocells will need some form of clock, which comes from a centralized clock divider source. Low-power designs often benefit from a single programmable prescaler and thus do not need a reference BCLK signal broadcast around the integrated circuit.

## When to Use ASB or APB

A full ASB interface is used for:

- Bus masters
- On-chip memory blocks
- External memory interfaces
- High-bandwidth peripherals with FIFO interfaces
- DMA slave peripherals

A simple APB interface is recommended for:

- Simple register-mapped slave devices
- Very low power interfaces where clocks cannot be globally routed
- Grouping narrow-bus peripherals to avoid loading the system bus

## ASB Components

An ASB system design contains the following components:

**ASB Master** – A bus master is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

The system may contain one or more bus masters. Typically, a system would contain at least the processor, however it would also be common for a DMA (direct memory access) or DSP (digital signal processor) to be included as bus masters.

**ASB Slave** – A bus slave responds to a read or write operation within a given address-space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.

The external memory interface, APB bridge and any internal memory are the most common ASB slaves. Any other peripheral in the system could also be included as an ASB slave; however, low bandwidth peripherals typically reside on the APB.

**ASB Decoder** – The bus decoder performs the decoding of the transfer addresses and selects slaves appropriately. The bus decoder also ensures that the bus remains operational when no bus transfers are required.

A single centralized decoder is required in all ASB implementations.

**ASB Arbiter** – The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. The arbitration scheme is not enforced such that “highest priority” or “fair” access algorithms may be implemented, depending on the application requirements.

An ASB would include only one arbiter, although this would be trivial in single bus master systems.

## APB Components

An APB implementation would contain the following components:

**APB Bridge** – A single bridge is required to convert ASB transfers into a suitable format for the slave devices on the APB. The bridge provides latching of all address, data and control signals, and provides a second level of decoding to generate slave select signals for the APB peripherals.

**APB Slave** – The APB can contain many different APB slaves.

The APB slaves have the following interface specification:

- Address and control valid throughout the access (unpipelined)
- Zero-power interface during non-peripheral bus activity (peripheral bus is static when not in use)
- Timing provided by decode with strobe timing (unclocked interface)
- Write data valid for the whole access (allowing glitch-free transparent latch implementations)

## Advanced System Bus Description

### Introduction

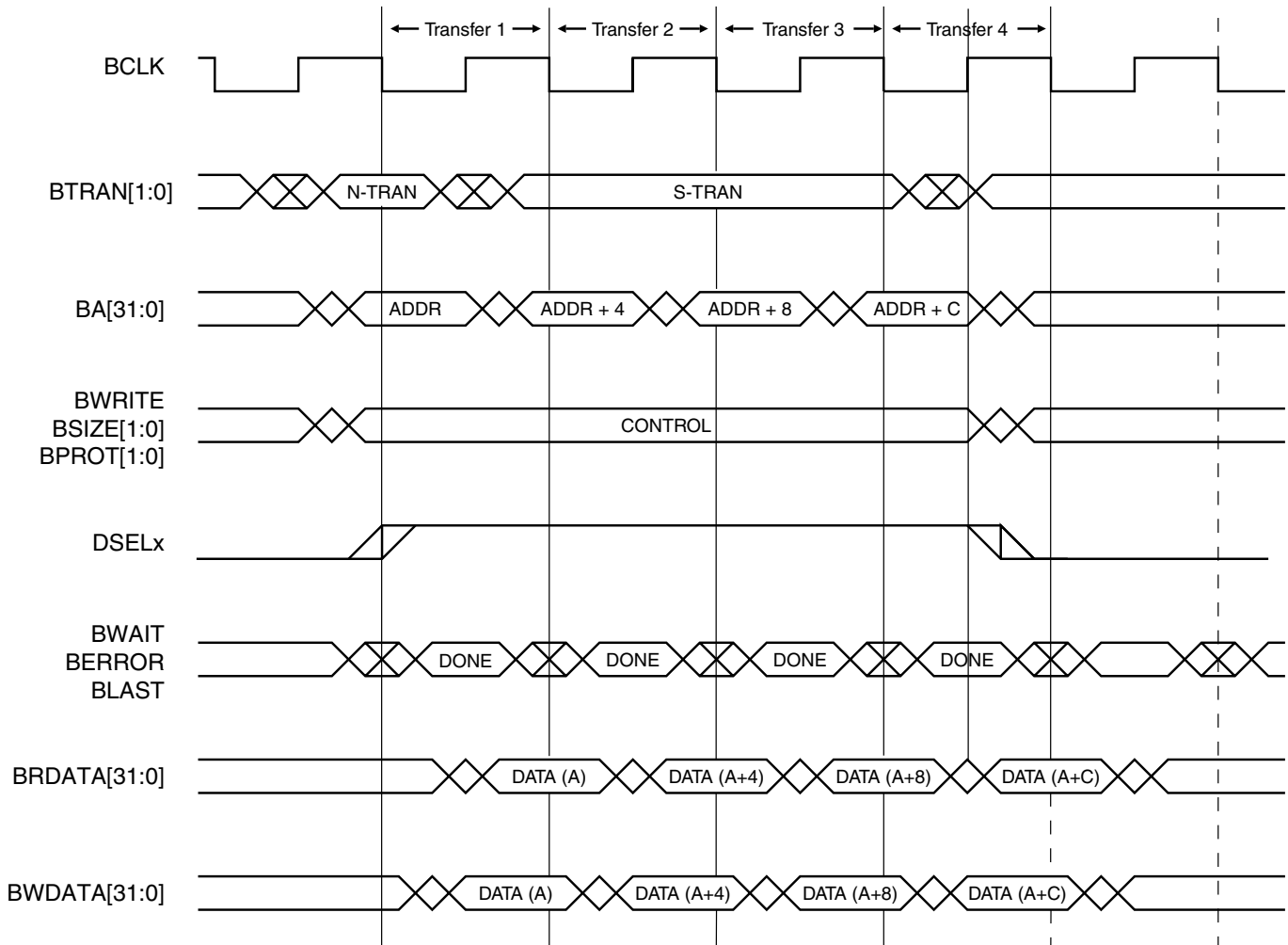
The ASB is a high-performance pipelined bus that supports multiple bus masters.

The basic flow of the bus operation is:

- The arbiter determines which master is granted access to the bus.
- When granted, a master initiates transfers on the bus.
- The decoder uses the high-order address lines to select a bus slave.
- The slave provides a transfer response back to the bus master and data is transferred between the master and slave.
- There are three types of transfer that can occur on the ASB:
- Non-sequential transfers are used for single transfers or for the first transfer of a burst.
- Sequential transfers are used for transfers in a burst. The address of a sequential transfer is always related to the previous transfer.
- Address-only transfers are used when no data movement is required. The three main uses for address-only transfers are:
  1. Idle cycles
  2. Bus master handover cycles
  3. Speculative address decoding without committing to a data transfer

Figure 3 shows the use of non-sequential and sequential transfers to perform a burst transaction. The burst starts with a non-sequential transfer to address A and the following sequential transfers are to successive addresses A + 4, A + 8 and A + 12.

**Figure 3. ASB Transfers**





## ASB Transfers

When a master has been granted the bus, it can perform the following transfers:

- Non-sequential data transfer
- Sequential data transfer
- Address-only transfer

A transfer is defined as starting at the falling edge of BCLK after the previous transfer has completed, as indicated by BWAIT being low, and running until the falling edge of BCLK after a complete transfer response is received, again indicated by BWAIT being low.

The type of transfer that a bus master will perform can be determined by the value on the BTRAN signal at the start of the transfer. During the transfer the BTRAN signal will change to indicate the type of the following transfer.

## Non-sequential Transfer

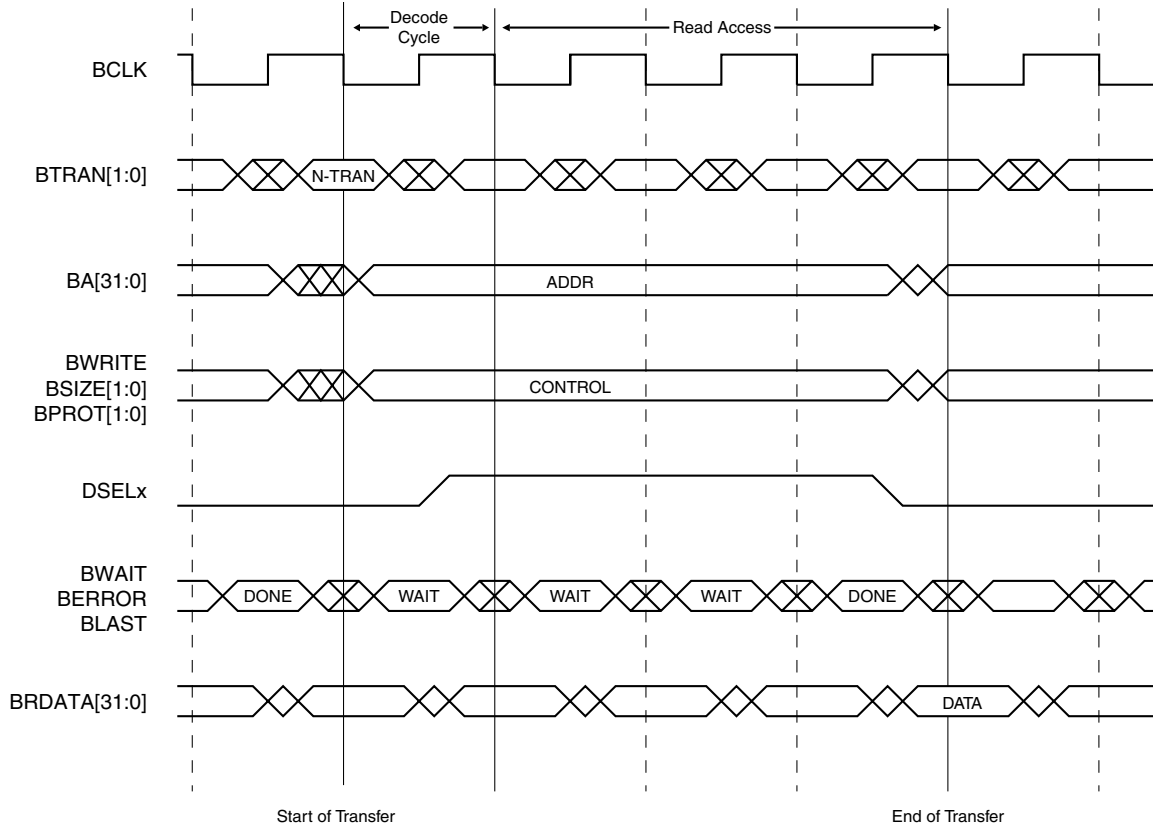
A non-sequential transfer occurs for either a single transfer or the start of a burst of transfers. Figure 4 shows a typical non-sequential read transfer including wait states and decode cycle.

The address and control signals start to change in the BCLK high phase before the transfer starts, but for a non-sequential transfer a valid address may not be available until very late in the BCLK high phase or even until the start of the clock low phase at the beginning of the transfer.

The decoder, which requires a stable address in order to select the correct slave, will automatically insert a wait state in the first cycle of non-sequential transfers. This is referred to as a decode cycle and provides an adequate time for the decoder to examine the high-order address lines and assert the appropriate DSELx during the high phase of the decode cycle. See Figure 4. For the remaining cycles of the transfer, the slave will provide a transfer response and the data exchange will occur between the master and slave.

**Note:** In certain system designs, which are typically those with a low-frequency system clock, the address is valid early enough in the BCLK high phase before the start of the transfer, allowing the decoder to generate a valid DSELx signal before the falling edge of BCLK. Such systems do not require the addition of a decode cycle at the start of the non-sequential transfers as shown in Figure 3. See “Address Decode” on page 14 for a more detailed description of the operation of such a system.

**Figure 4. Non-sequential Transfer with Decode Cycle and Wait States**



The read data bus, BRDATA[31:0], must be valid by the falling edge of BCLK at the end of the transfer. During a write cycle, the bus master is responsible for driving the write data bus, BWDATA[31:0], which it will do from the start of the clock high phase in order that the slave may accept valid data by the falling edge of the clock. During a read cycle, the appropriate slave must drive the data bus, such that it is valid by the end of the high phase.

## Sequential Transfer

A sequential transfer occurs when the address is related to that of the previous transfer. The control information, as indicated by BWRITE, BPROT and BSIZE, will be the same as the previous transfer.

If the sequential transfer follows a non-sequential or another sequential transfer, the address can be calculated by using the previous size and address. For example, a burst of word accesses would be to addresses A, A + 4, A + 8, whereas a burst of half-word accesses would be to addresses A, A + 2, A + 4.

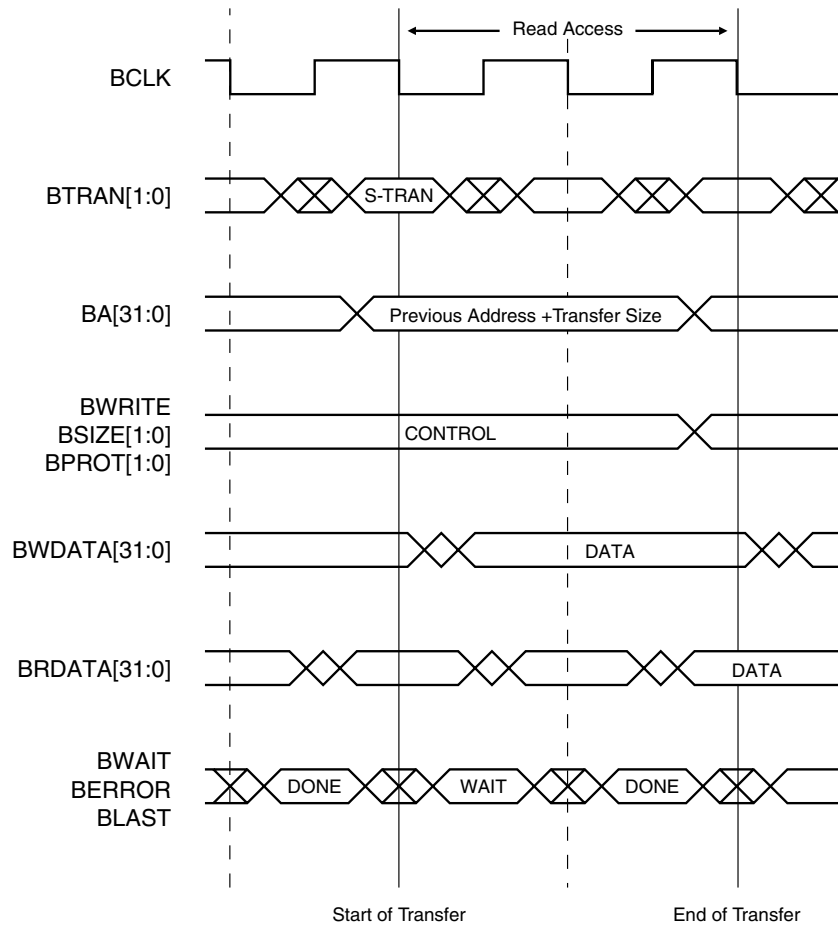
If a sequential transfer follows an address-only cycle, then the address will be the same as that of the address-only cycle. This combination of an address-only followed by sequential allows both a single access using a sequential transfer and a burst of transfers to start with a sequential transfer. An example of the use of an address-only followed by sequential is shown later in Figure 7.

Figure 5 shows a sequential transfer with one wait state. Note that it closely resembles a non-sequential transfer. The main differences are:

- The BTRAN signal indicates a sequential transfer
- Address is always valid in the BCLK high phase at the start of the transfer
- Address is related to the preceding transfer

- Control information remains the same as the preceding transfer

**Figure 5. Sequential Transfer**



## Address-only Transfer

An address-only transfer indicates that no data transaction is required. During an address-only transfer it is possible that the address and control information may also be invalid. The only signals that must be driven to valid levels are:

- BTRAN – To indicate the type of the next transfer
- BLOKx – To allow the arbitration process to continue

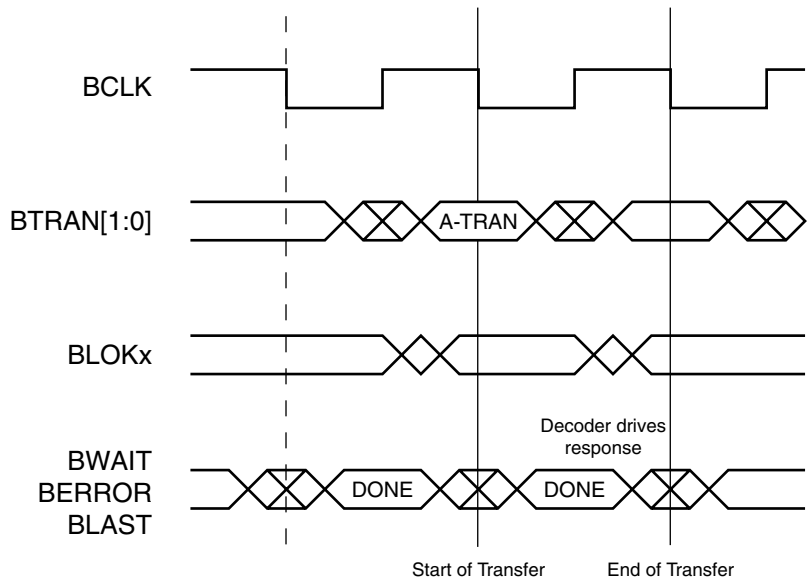
As address-only transactions do not access slaves on the bus, they only require a single cycle and therefore the BWAIT signal will indicate NOT(Wait). This signal is driven by the bus decoder, as no slave will be selected during the address-only cycle. A bus master may perform a number of address-only transfers in succession if it does not require the bus for data transfer.

The address-only transfer can actually be used in a number of different ways:

- As a true idle cycle, when the bus master does not require the bus
- To speculatively broadcast an address for the next transfer, without committing to the transfer
- To provide a turnaround cycle during bus master handover when using tristate buses on ASB

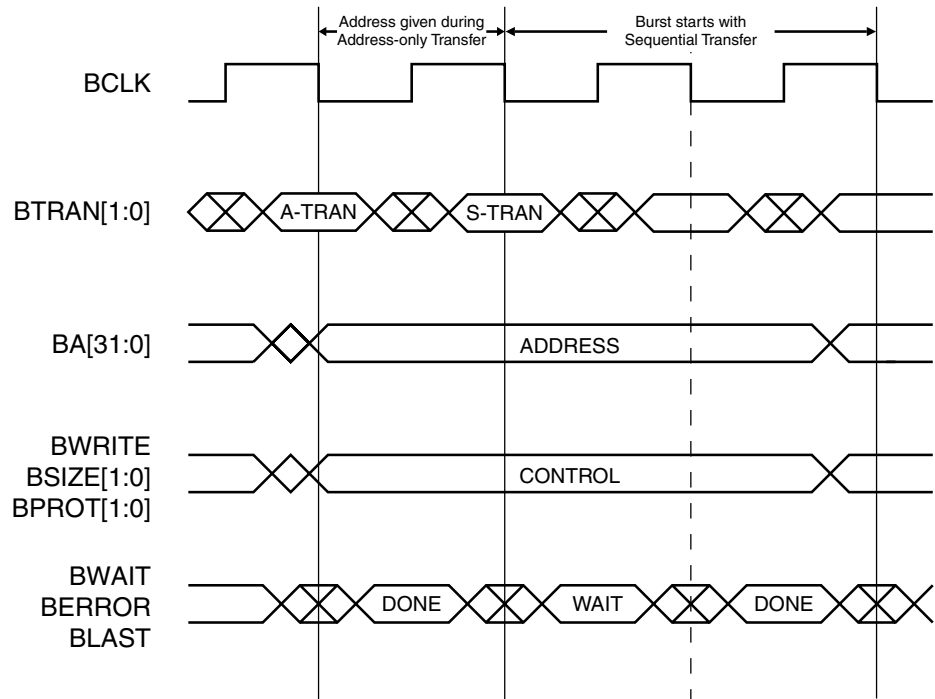
If the address-only transfer is used as a true idle cycle, then the address and control signals are not required to be valid at any point during the transfer. The BLOKx signals are the only exception and must be driven to a valid level during all address-only transfers to allow the arbitration process to continue. See Figure 6.

**Figure 6.** Address-only Transfer



The second use of the address-only transfer is to speculatively broadcast the address for a transfer, without actually committing to the transfer. This allows the address decoding to be performed by the decoder during the address-only cycle. If the bus master then commits to the burst, it is possible to start the burst with a sequential transfer, thus removing the need for an extra decode cycle before the transfer starts. See Figure 7.

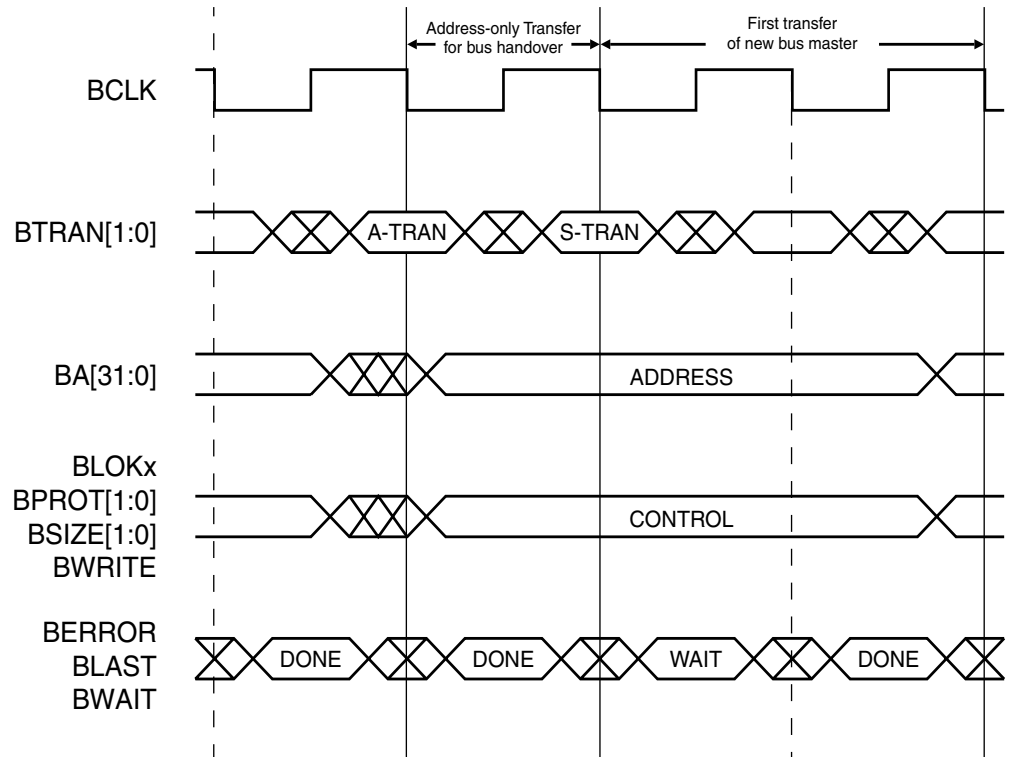
**Figure 7.** Address-only Transfer to Start Burst



The final use of an address-only transfer is to provide a turnaround period during bus master handover. A bus master that becomes granted on the bus must start with an address-only transfer and, in this case, the new bus master does not drive the address and control signals immediately, but provides a phase of turnaround before driving the signals in the low phase of the transfer.

It is important to note that in this case the address and control information will not become valid until the low phase of BCLK. See Figure 8.

**Figure 8. Address-only Transfer for Bus Master Handover**



## Address Decode

In an AMBA system the address decoding is performed by a centralized decoder.

The decoder uses the type of each transfer to determine which of the following functions should be performed:

1. For an address-only transfer, the decoder will respond with a done transfer response and no slaves will be selected. During address-only transfers, the decoder performs an address decode speculatively in case the address-only transfer is followed immediately by a sequential transfer.
2. For non-sequential transfers (or when the previous transfer was terminated with a last transaction response), the decoder will insert a single wait state at the start of the transfer to allow sufficient time for address decoding (although the additional wait state may not be required in all systems).  
The additional wait state inserted by the decoder is referred to as a decode cycle and during the decode cycle no select signals, DSELx, are asserted.  
In the second cycle of the transfer, the decoder will either select the appropriate slave or provide an error transfer response.  
An error response is provided in the following circumstances:

- There are no slaves present at the address of the transfer
  - The transfer is to a protected region of memory
  - The alignment of the transfer is not supported by the memory system. In the more usual case of a valid transfer, the decoder will assert the appropriate slave DSELx signal and allow the selected slave to provide the transfer response for the remaining cycles of the transfer.
3. For sequential transfers, the decoder asserts the appropriate DSELx signal and the selected slave provides the transfer response. It is not necessary for the decoder to decode the address as this will have been performed in a previous non-sequential or address-only transfer.

As the decoder does not perform an address decode on sequential transfers, it is necessary for the slave to provide a last transfer response if a transfer is about to cross a memory boundary. The decoder is also responsible for generating an internal version of the last signal when the decoder detects that a sequential transfer will cross a memory boundary.

The insertion of a decode cycle (as described in step 2) on non-sequential transfers can be used to improve the performance of the system. In a typical design, the time required for address decoding will increase the critical path of an access to a slave and often result in the need for additional wait states. The decoder can be used to reduce this overhead by automatically inserting a decode cycle on non-sequential transfers only, but allowing sequential transfers to complete without additional wait states.

In some systems, typically those with a low clock frequency, additional wait states are not required for address decoding, and in such systems the decoder may be simplified such that both sequential and non-sequential transfers occur without the addition of a decode cycle.

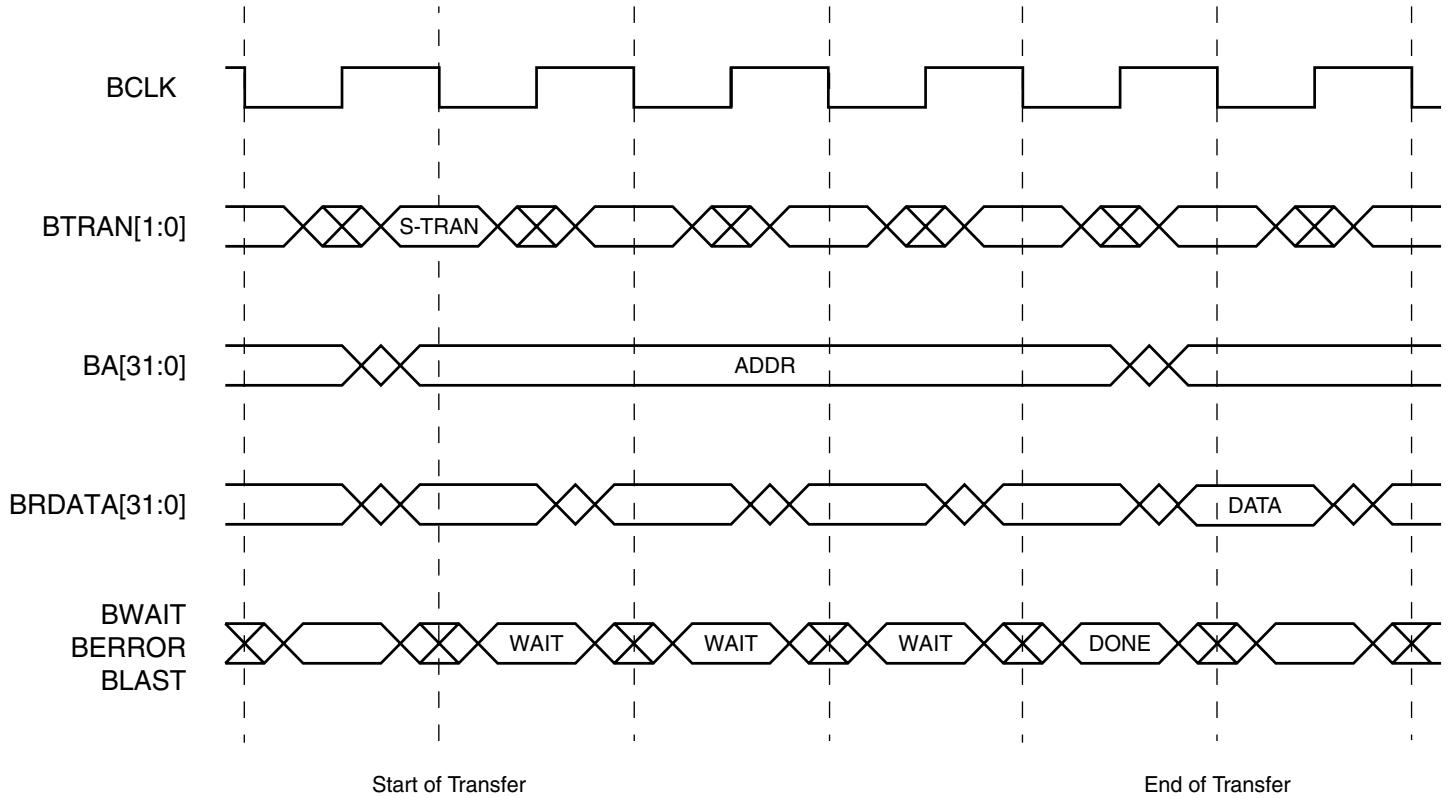
## Transfer Response

For every transfer that is initiated by a bus master, a response must be generated and this is provided either by the decoder or by the selected bus slave. The transfer response is provided using the BWAIT, BERROR and BLAST signals, which are valid during the low phase of the clock.

Figure 9 on page 16 shows an example of how the transfer response is used to insert three wait states in order to extend a transfer.

Table 2 on page 16 shows transfer responses that are available.

**Figure 9. Transfer Response**



**Table 2. Available Transfer Responses**

Response	Description
Wait	The transfer must be extended before it can complete.
Done	The transfer has completed successfully.
Error	The transfer has completed, but an error has occurred. The error condition should be signaled to the bus master so that it is aware the transfer was unsuccessful.
Last	The transfer has completed successfully, but the slave is unable to accept further burst transfers or a memory boundary has been reached. This response is identical to done for the bus master, but indicates to the decoder that it must insert a decode cycle at the start of the next transfer.
Retract	The transfer has not yet completed, so the bus master should retry the transfer. The Retract response can be used by a slave to signal to a bus master that the transfer can complete, but this may take a number of bus cycles.
RetNext	Retract on next cycle. The transfer has not yet completed and will not be completed on the next cycle (Retract response). Thus the bus master must retry the transfer.

Using the Retract response prevents the bus from being locked up by a transfer, which may take a long time to complete, and frees the bus for use by a higher priority bus master.

Unlike the other response codes, which take a single cycle, the Retract response is a two-stage response, as shown in Figure 10 on page 17. In the first stage the slave signals to the bus master that a retract is going to take place using the response RetNext, and in the second stage the transfer is completed when the slave provides the Retract response. This two-stage response provides the bus master with adequate warning that



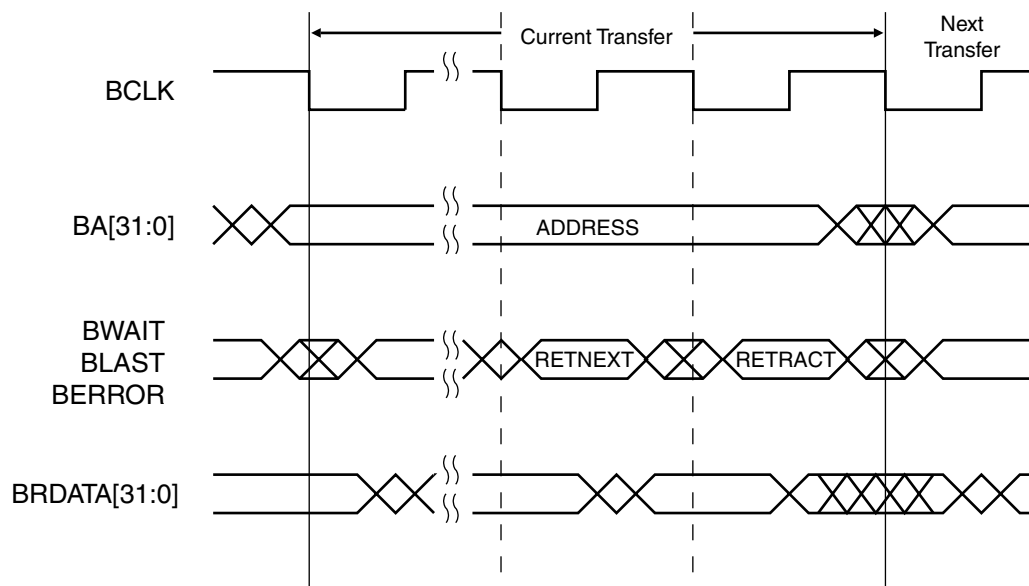
it should not consider the transfer to have completed when Transfer response goes to the Retract response.

All bus masters must support the retract mechanism; however, not all slaves are required to implement the Retract response. Typically, a Retract response would only be provided by a slave that does not have a short guaranteed completion time and, hence, could deadlock the bus for a significant period of time.

For most transfers the response will be provided by the selected bus slave. However, the decoder provides the response when:

- The transfer is address-only
- The transfer is to an area of memory where there are no bus slaves
- An access violation occurs to a protected region of memory

**Figure 10. Retract Operation**



## Multi-master Operation

The AMBA bus specification supports multiple bus masters on the high-performance ASB. A simple two-wire request/grant mechanism is implemented between the arbiter and each bus master. The arbiter ensures that only one bus master is active on the bus and also ensures that when no masters are requesting the bus, a default master is granted.

The specification also supports a lock signal. This allows bus masters to indicate that the current transfer is indivisible from the following transfer and will prevent other bus masters from gaining access to the bus until the locked transfers have completed.

Efficient arbitration is important to reduce “dead-time” between successive masters being active on the bus. The bus protocol supports pipelined arbitration, such that arbitration for the next transfer is performed during the current transfer.

The arbitration protocol is defined, but the prioritization is flexible and left to the application. Every system must also include a default bus master, which is granted the bus when no bus masters are requesting it.

The request signal, AREQx, from each bus master to the arbiter indicates that the bus master requires the bus. The grant signal from the arbiter to the bus master, AGNTx, indicates that the bus master is currently the highest priority master requesting the bus.

The bus master must drive the BTRAN signals during BCLK high when AGNTx indicates that it is currently the highest priority master. In addition, the bus master will become granted when AGNTx indicates that it is currently the highest priority master and BWAIT indicates NOT(Wait) on a rising edge of BCLK.

The lock signals, BLOKx, indicate to the arbiter that the following transfer is indivisible from the current transfer and no other bus master should be given access to the bus.

A bus master must always drive a valid level on its BLOKx signal when granted the bus to ensure the arbitration process can continue, even if the bus master is not performing any transfers.

## Arbiter

The arbiter functions as follows:

1. Bus masters assert AREQx during the high phase of BCLK.
2. The arbiter samples all AREQx signals on the falling edge of BCLK.
3. During the low phase of BCLK the arbiter also samples all the BLOKx signals and then asserts the appropriate AGNTx signal.

If all BLOKx signals indicate that the following transfer is not indivisible from the current transfer, then the arbiter will grant the highest priority bus master.

If a BLOKx signal indicates that the following transfer is indivisible from the current transfer and if the corresponding bus master is currently granted, then the arbiter will keep the same bus master granted.

The arbiter can update the grant signals every bus cycle; however, a new bus master can only become granted and start driving the bus when the current transfer completes, corresponding to the Transfer response indicating Done or Retract. Therefore, it is possible for the potential next bus master to change during waited transfers.

The BLOKx signals are ignored by the arbiter during the single cycle of handover between two different bus masters if the arbiter performs this type of cycle.

If no bus masters are requesting the bus, then the arbiter must grant the default bus master.

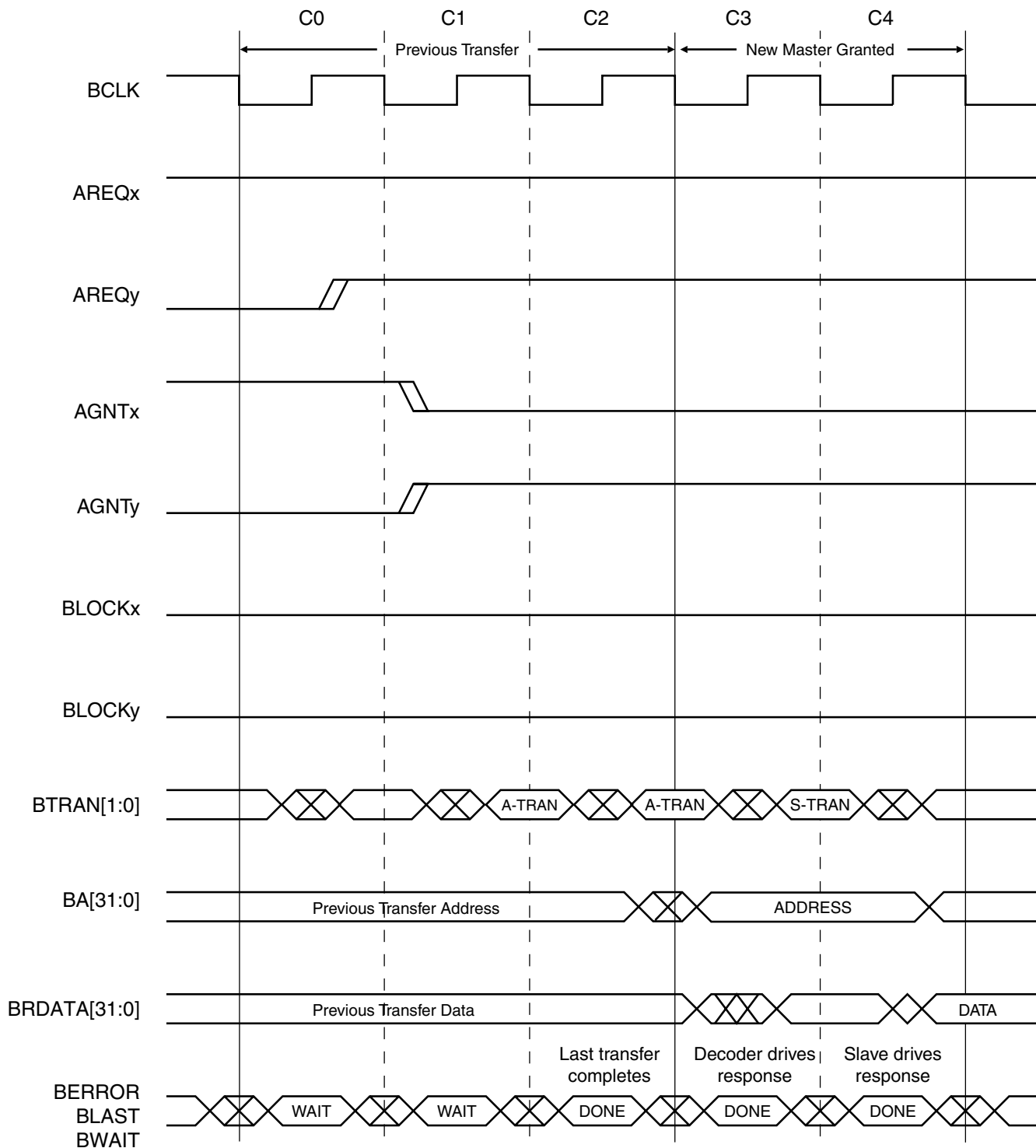
The arbitration protocol is defined, but the prioritization is flexible and left to the application. A simple fixed priority scheme may be used; alternatively, a more complex scheme can be implemented if required by the application.

## Default Bus Master

Every system must be designed with a single default bus master, which will be granted when no other bus master is requesting the bus. The default bus master is responsible for driving the following signals to ensure the bus remains active:

- BTRAN must be driven to indicate address-only transfer
- BLOKx must be driven to indicate NOT LOCKED transfer

**Figure 11.** Bus Master Change from Master X to Higher Priority Master Y



## Reset Operation

The reset signal, BnRES, is active low and may be asserted asynchronously to guarantee the bus is in a safe state. During reset the following actions occur on the bus:

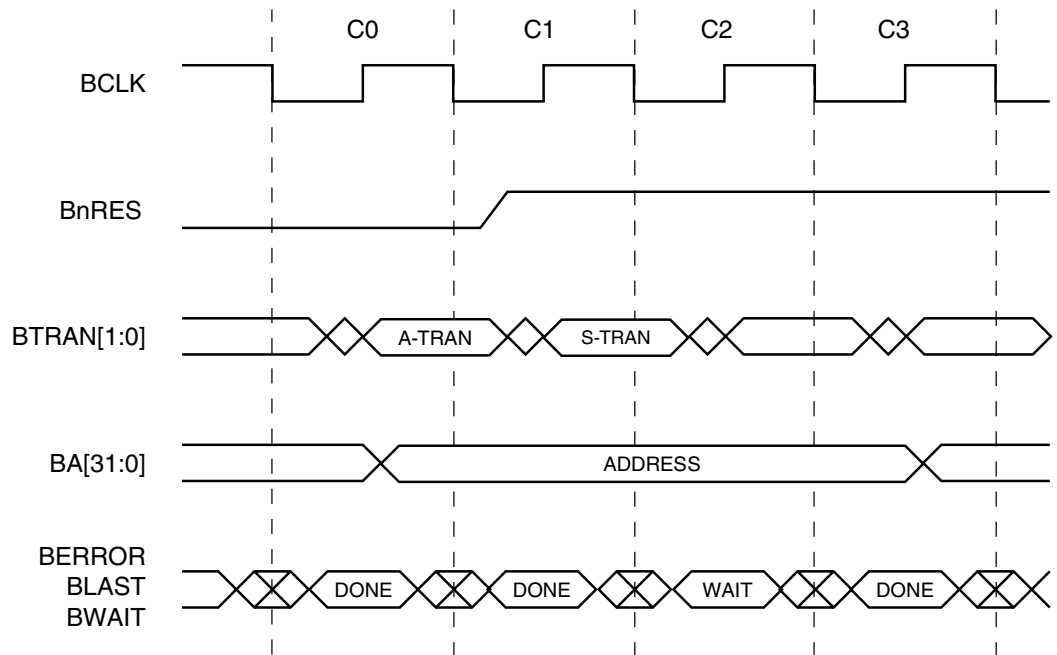
- The arbiter grants the default bus master
- The default bus master must:
  1. Drive BTRAN to indicate address-only transfer
  2. Drive BLOKx to indicate NOT LOCKED transfer to allow arbitration
- The decoder must:
  1. De-assert all slave select signals, DSELx
  2. Provide the appropriate transfer response

## Exit from Reset

Figure 12 below shows an example of the exit from reset sequence.

1. During cycle C1, BnRES is de-asserted during the clock low phase.
2. During the clock high phase of cycle C1, the default bus may drive the BTRAN signal to indicate that it wishes to start a transfer.
3. The transfer will start during cycle C2 and in the example shown, the transfer is waited and continues into cycle C3.

**Figure 12.** Exiting from Reset



## Advanced Peripheral Bus (APB)

The APB provides a low-power extension to the higher performance ASB. APB transfers are non-pipelined with address, PA[31:0], and control, PSELx and PWRITE, information set up before the start of the transfer and held valid throughout the transfer.

Typically not all 32 bits of the address bus are implemented. All peripherals should be aligned to word boundaries and, hence, the lowest two bits of the address are not required. The high-order address lines are not required as the address decode for the peripheral bus is carried out within the bridge unit and signaled to the slaves using the PSELx signals. Of the other address lines, only those required by the slaves in a particular system need to be implemented.

During write accesses the data bus, PWDATA[31:0], is also valid throughout the transfer. For read accesses the APB slave must supply valid data on PRDATA[31:0] before the falling edge of PSTB, which is coincident with the falling edge of BCLK. (Sampling point for the data inside the bridge.)

Figure 13 on page 22 shows the basic APB transfers, which operate as follows:

1. At the start of phase P1, the address, select and write signals all become valid for the access. If the transfer is a write access, then the data will also become valid in phase P1. PSTB\_RISING is driven to low level.

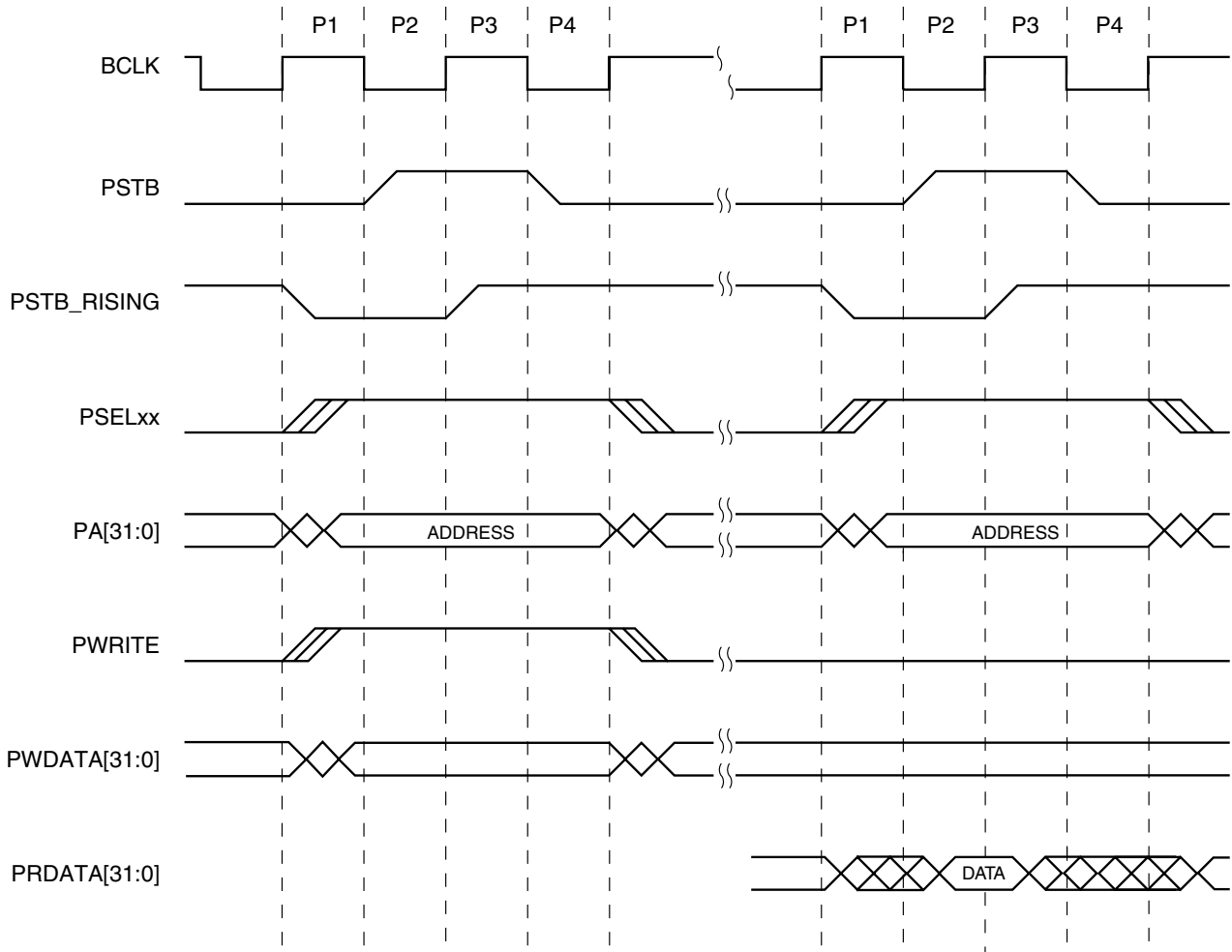
During phase P1, the slave that is selected can decode the address and write signal to generate an internal write strobe qualifier for the register that is about to be accessed.

2. At the start of phase P2, the timing strobe PSTB is asserted. The address and write signal are set up well in advance of the rising edge of PSTB, as is the data in the case of a write access.
3. For a read access, the slave must drive valid data before the start of phase P3. The slave may use PSTB, qualified with PSELx, as the data bus tri-state enable control.
4. At the start of phase P3, the signal PSTB\_RISING is driven high. The rising edge of this signal can be used for write access as a clock signal. Throughout P2 and P3, the address, write signal and the data (in the case of a write access) remain stable and valid.

PSTB can be used in conjunction with PSELx and the appropriate address lines in a number of different ways:

- As a clock enable, for either rising or falling edge triggered registers
  - Directly as a clock, for either rising or falling edge triggered registers
  - As a transparent latch clock enable, using BCLK as the primary clock
  - Directly as a transparent latch clock
5. At the end of the transfer, the address and control information remains stable throughout phase P4, providing guaranteed hold time for all APB slaves.

**Figure 13. APB Access**



## Signal Description

This section provides more detailed information on all the AMBA signals, including their intended use and phase- accurate timing requirements.

## Clock and Reset

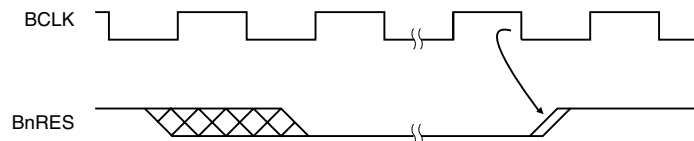
### Clock

BCLK is the primary clock, which is used to time all bus transfers. Both edges of the clock are used.

### Reset

A single active low reset signal, BnRES, is supported, which is used to reset the bus. The reset signal may be asserted low asynchronously during either clock phase, but is always de-asserted during the low phase of BCLK.

**Figure 14. Reset Signal**

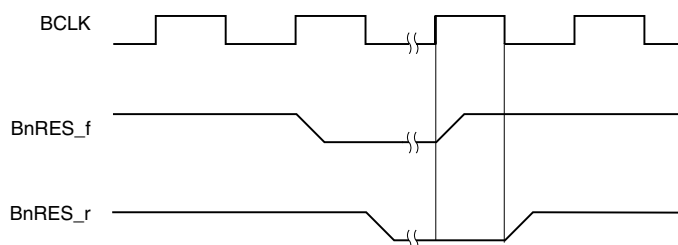


In order to avoid problems with the reset, two reset signals may be created:

1. One for sequential elements with rising edge of BCLK (BnRES\_r)
2. One for sequential elements working with the falling edge of BCLK (BnRES\_f)

Thus the reset signals can be generated with the edges of BCLK. See Figure 15.

**Figure 15.** Reset Signal – Special Case



During reset, the following actions occur on the bus:

- The arbiter grants the default bus master
- The default bus master must:
  - Drive BTRAN to indicate address-only transfer
  - Drive BLOKx to indicate NOT LOCKED transfer to allow arbitration
- The decoder must:
  - De-assert all slave select signals, DSELx
- Provide the appropriate transfer response

The BnRES signal may be used to reset the bus during time-out conditions.

In the majority of bus masters and slaves the BnRES signal will be used to reset both the bus interface and the main core of the component. However, it is acceptable for some system elements, such as a real-time clock, to use BnRES to only reset the bus interface. Such system elements would typically have a second reset input to allow the component core to be reset at initial power-up and for testing purposes.

## Transfer Type

Before a transfer starts, the bus master indicates the type of the transfer. There are three basic transfer types: address-only, non-sequential and sequential.

Table 3 shows the encoding of the BTRAN[1:0] signal.

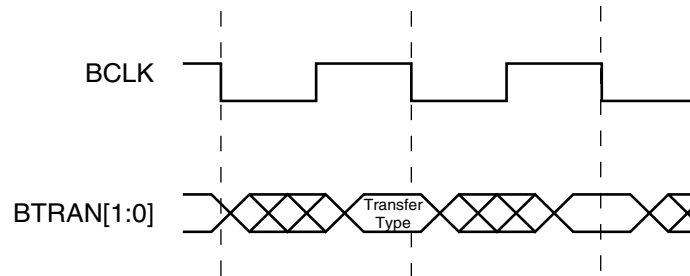
**Table 3.** BTRAN Encoding

BTRAN		Transfer Type	Description
[1]	[0]		
0	0	Address-only	Used when no data movement is required. The three main uses for address-only transfers are for idle cycles; for bus master handover cycles; and for speculative address decoding without committing to a data transfer.
0	1	–	Reserved
1	0	Non-sequential	Used for single transfers or for the first transfer of a burst. The address of the transfer is unrelated to the previous bus access.
1	1	Sequential	Used for successive transfers in a burst. The address of a sequential transfer is always related to the previous transfer.

From Table 3 it can be seen that BTRAN[1] can be used to determine that a data transfer is required next cycle.

The BTRAN signals are driven by a bus master during the high phase of BCLK when the AGNTx input is high. See Figure 16.

**Figure 16.** BTRAN Timing



In a multi-master system, the bus master that drives BTRAN may change during an extended transfer. Therefore, BTRAN must only be considered valid when the previous transfer has completed.



## Address and Control Information

The address and control signals are:

- BA[31:0]
- BWRITE
- BSIZE[1:0]
- BPROT[1:0]

## BA[31:0] – Address Bus

The 32-bit address bus, BA[31:0], provides the address of the transfer. All transfers are memory-mapped and therefore all memory and peripherals within the system must have an address range within which they are accessed. The decoder uses the address bus (usually the higher order bits) to determine which bus slave is to be accessed.

## BWRITE - Transfer Direction

The BWRITE signal is used to indicate the direction of the transfer. When BWRITE is low the transfer is a read access and when BWRITE is high the transfer is a write access.

**Table 4.** BWRITE Encoding

BWRITE	Transfer Direction
0	Read Transfer
1	Write Transfer

## BSIZE[1:0] – Transfer Size

BSIZE[1:0] encodes the size of a transfer. Byte, half-word and word are all defined, with the final encoding being reserved for future use.

**Table 5.** BSIZE Encoding

BSIZE		Transfer Width
[1]	[0]	
0	0	Byte (8 bits)
0	1	Half-word (16 bits)
1	0	Word (32 bits)
1	1	Reserved

When performing transfers that are narrower than the data bus, such as a byte or half-word transfer, the bus master may replicate the data across the bus, making the bus master effectively big-endian. When responding to read cycles, a typical slave will not replicate the data on the bus and therefore it is important that the master is expecting data on the same bus lane as that which the slave is driving.

## BPROT[1:0] – Protection Information

The bus master may use the BPROT signals to provide additional information about the transfer it is performing. This information is primarily intended for use by the decoder when it is acting as a bus protection unit and the majority of bus slaves will not use these signals.

**Table 6.** BPROT Encoding

BPROT		Transfer Privilege
[1]	[0]	
–	0	Opcode Fetch

**Table 6.** BPROT Encoding (Continued)

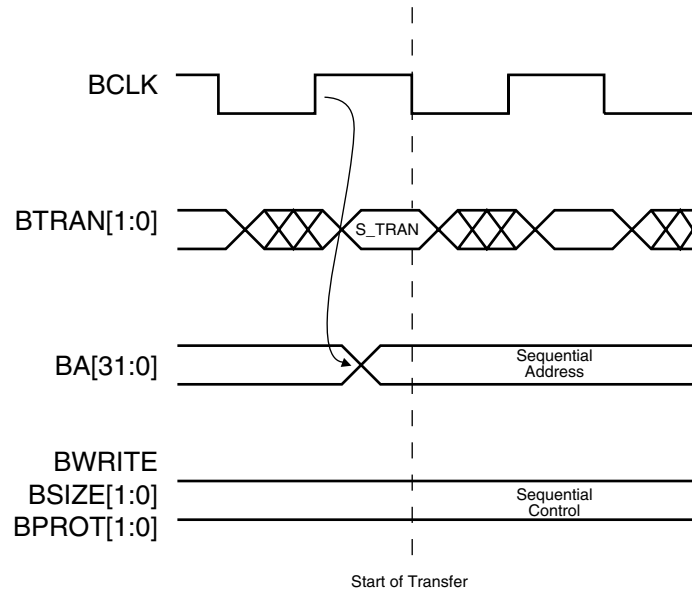
BPROT		Transfer Privilege
–	1	Data Access
0	–	User Access
1	–	Supervisor Access

### Address and Control Signal Timing

The address and control information is generated by the bus master from the rising edge of BCLK. However, the timing of the address and control information is considered separately for non-sequential and sequential transfer types. This is because a bus master will typically have significantly different timing parameters in each case.

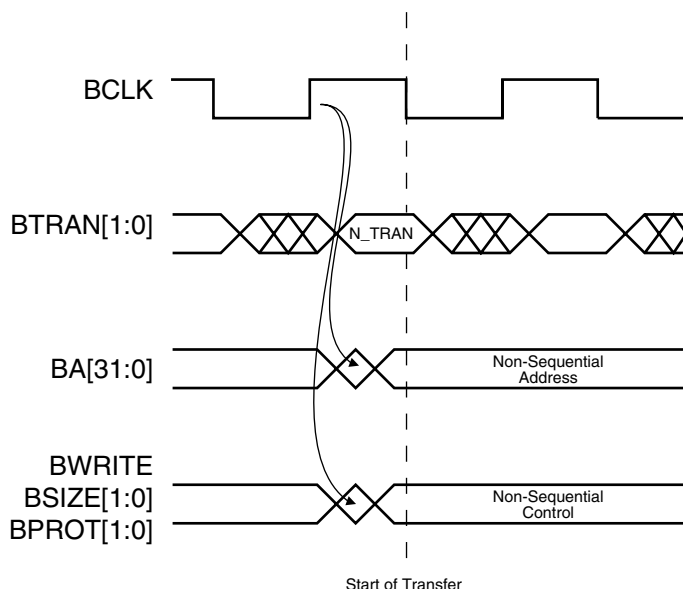
It is a common characteristic that bus masters will have fast address and control output valid timings for sequential transfers, as shown in Figure 17. This is because a bus master is usually able to generate a sequential address well before the start of the transfer and therefore the output valid time from the bus master is mainly dependent on the time required to drive the new value onto the bus.

**Figure 17.** Sequential Address and Control Timing



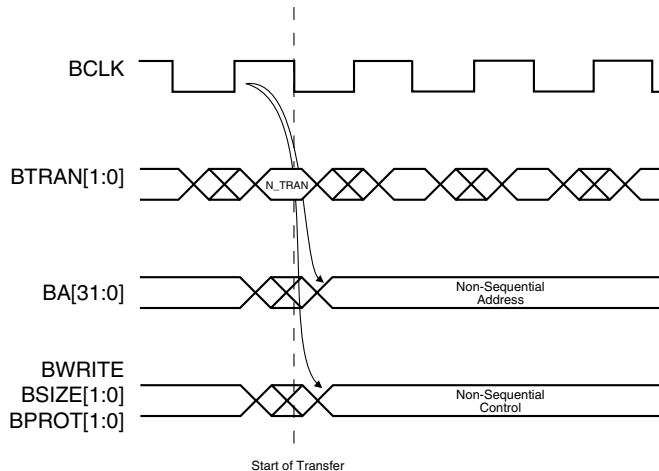
For non-sequential transfers, the bus master will often have significantly slower output valid times for address and control signals, compared to those for sequential transfers. This is shown in Figure 18.

**Figure 18.** Non-sequential Address and Control Timing with Low-frequency Clock



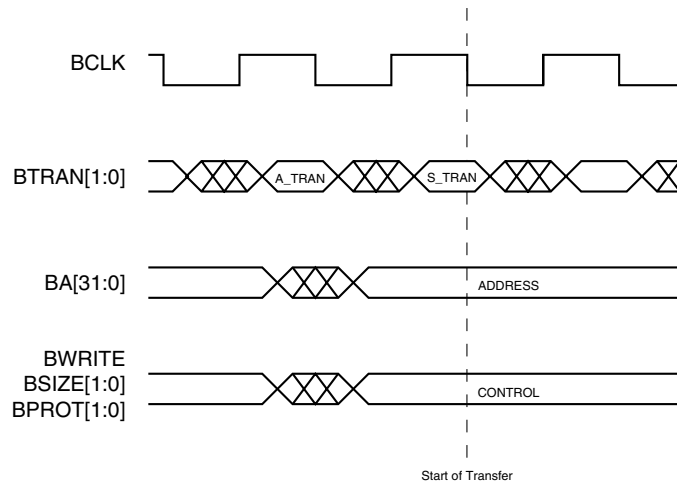
In systems where the clock frequency is approaching the maximum possible, it is common for the address and control output valid time to be greater than a clock phase, thus resulting in the address not becoming valid until the BCLK low phase at the start of the transfer, as shown in Figure 19.

**Figure 19.** Non-sequential Address and Control Timing with High-frequency Clock



For address-only transfers, the address and control information is not valid. In the special case of the address-only followed immediately by a sequential transfer, as shown in Figure 20, the bus master generates the address and control information during the address-only transfer, such that it is valid throughout the BCLK high phase before the start of the sequential transfer.

**Figure 20.** Address-only Followed by Sequential Transfer Address and Control Timing



### Enable of Address and Control Signals

The bus master can start a period of bus ownership with an address-only transfer and the address and control signals may not be valid until the BCLK low phase of the address-only transfer.

For multi-master systems:

- When the master is first granted, the signals are not supposed to be valid until the first low phase of BCLK when the master becomes active on the bus.
- When a bus master is granted the bus, it drives the address and control signals at valid levels during both phases of BCLK.
- The bus master can stop driving valid signals in the last BCLK high phase, when the master loses mastership of the bus.

### Slave Select Signals

Each ASB slave in the system has a DSEL select input signal. This signal indicates that the slave is responsible for supplying a transfer response and that a data transfer is required. The signal name DSELx is used to indicate the DSEL signal to slave “x”.

There is one DSELx signal for each slave on the ASB and these signals are generated by the decoder. Only one DSELx signal will be active during a transfer and there may be cycles when no DSELx signal is active, such as during address-only transfers.

DSELx changes during the BCLK high phase before the start of a transfer and remains valid during the transfer. It will change for the next transfer in the BCLK high phase following a transfer response indicating transfer completed.

When designing a system there are two options for the implementation of an ASB decoder, either with or without decode cycles. This choice is fixed at the design stage and will be based on a timing analysis of the system. In general, a system that is operating up to the maximum speed of the processor will require decode cycles; it is only those systems that operate at a frequency significantly lower than the possible maximum that do not require decode cycles.

The “Decoder with Decode Cycles” and “Decoder without Decode Cycles” cases are discussed separately below.

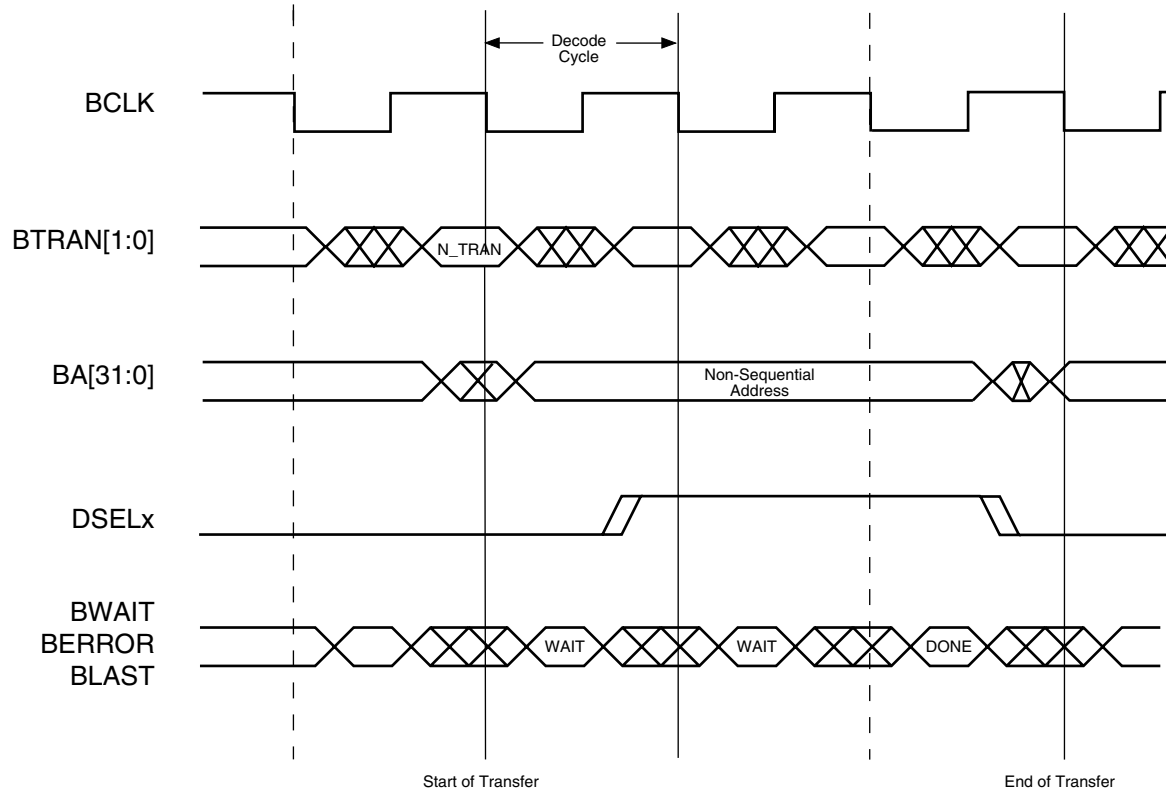
### Decoder with Decode Cycles

In systems with a high clock frequency the critical path to decode the address and select a slave within a single clock phase tends to limit the maximum bus clock speed. In such systems the decoder can be used to automatically insert a wait state, or decode cycle,

at the start of every non-sequential transfer. This implementation allows sequential transfers to continue to operate without the addition of a wait state, as it is known that the address decoding critical path can be avoided on sequential transfers, thus resulting in an overall improvement in bus bandwidth.

For non-sequential transfers, DSELx is asserted in the BCLK high phase during the decode cycle, as shown in Figure 21.

**Figure 21.** Select Signal Timing with Decode Cycle



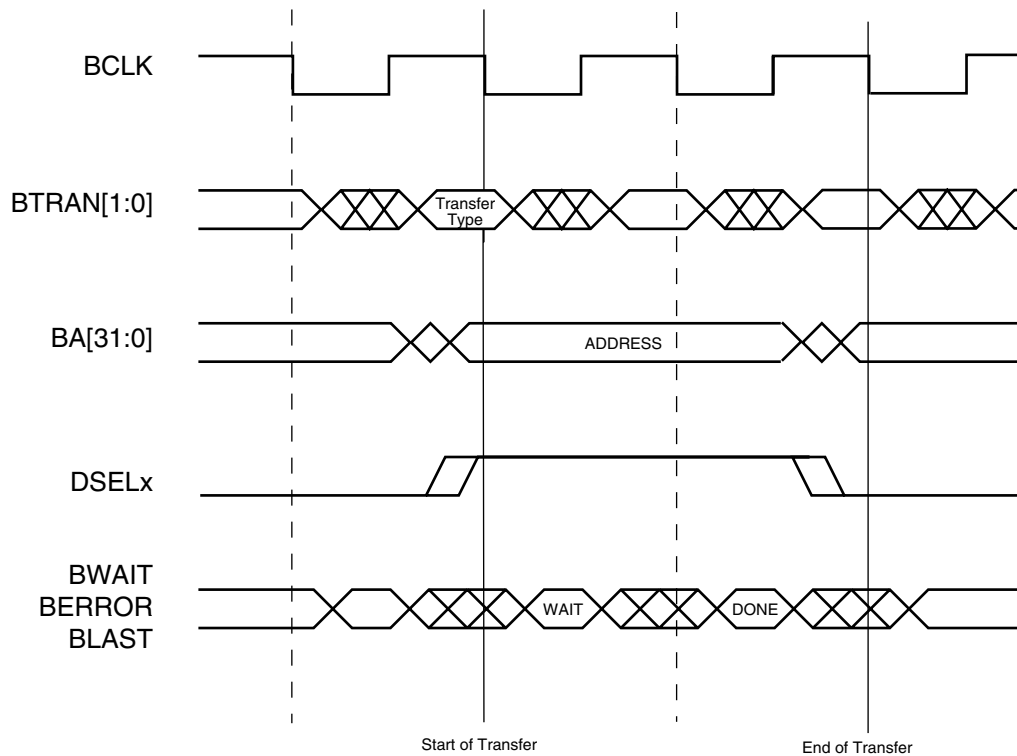
When decode cycles are implemented, the timing of DSELx is dependent only on BTRAN[1:0] and there is no timing dependency on the address and control signals. This is because no DSELx signals are asserted for address-only transfers; for a non-sequential transfer the decode cycle is inserted to provide an entire phase for the address and control information to become valid; and for sequential transfers, the address and control information from the previous cycle is used.

## Decoder without Decode Cycles

In systems with a low clock frequency, the address and control information will be valid in time to decode the address and select a slave within a single clock phase. In such systems a decode cycle is not required.

The select signal becomes valid during the high phase of BCLK before the transfer commences and remains valid throughout the transfer until the high phase of the last cycle. See Figure 22.

**Figure 22.** Select Signal Timing without Decode Cycle



When the decoder does not insert decode cycles, the timing of DSELx becomes dependent on the timing of the address and control signals generated by the currently granted bus master.

## Transfer Response

The transfer response signals are used by slave devices to indicate the status of a transfer.

A valid transfer response must be provided during the low phase of BCLK. Whenever a slave is selected (as indicated by DSELx being asserted), the slave must provide the response. When no slave is selected, for example during an address-only transfer, the response is provided by the decoder.

### BWAIT – Wait Response

BWAIT is used to indicate when a transfer may complete. BWAIT is asserted high when the slave requires extra bus cycles to complete the current transfer. BWAIT low indicates that the transfer may finish. Whether or not the transfer has completed successfully can only be determined by examining the other transfer response signals.

### BERROR – Error Response

An error condition is signalled by the BERROR signal. This may be used to indicate a failed transfer, a transfer to an address where there is no slave device or a protection error.

Many simple bus slaves will not implement error logic and will therefore have a fixed response of BERROR low.

BERROR is also used in conjunction with BLAST to indicate a retract operation. When both these signals are high, this indicates that a bus retract is required.

## **BLAST – Last Response**

BLAST is used to signal if the current transfer must be the last of a burst. This would typically be used to prevent a burst from continuing over a page boundary or other burst length limit.

BLAST is used by the decoder to make sure that the following transfer has the same characteristics as a non-sequential type transfer, rather than a burst transfer. Typically, this involves ensuring there is adequate time to perform a new address decode.

Many bus slave devices will be able to accept any number of burst accesses and these slaves will have a fixed response of BLAST low.

BLAST is also used in conjunction with BERROR to indicate a retract operation. When both these signals are high, this indicates that a bus retract is required.

## **Bus Retract**

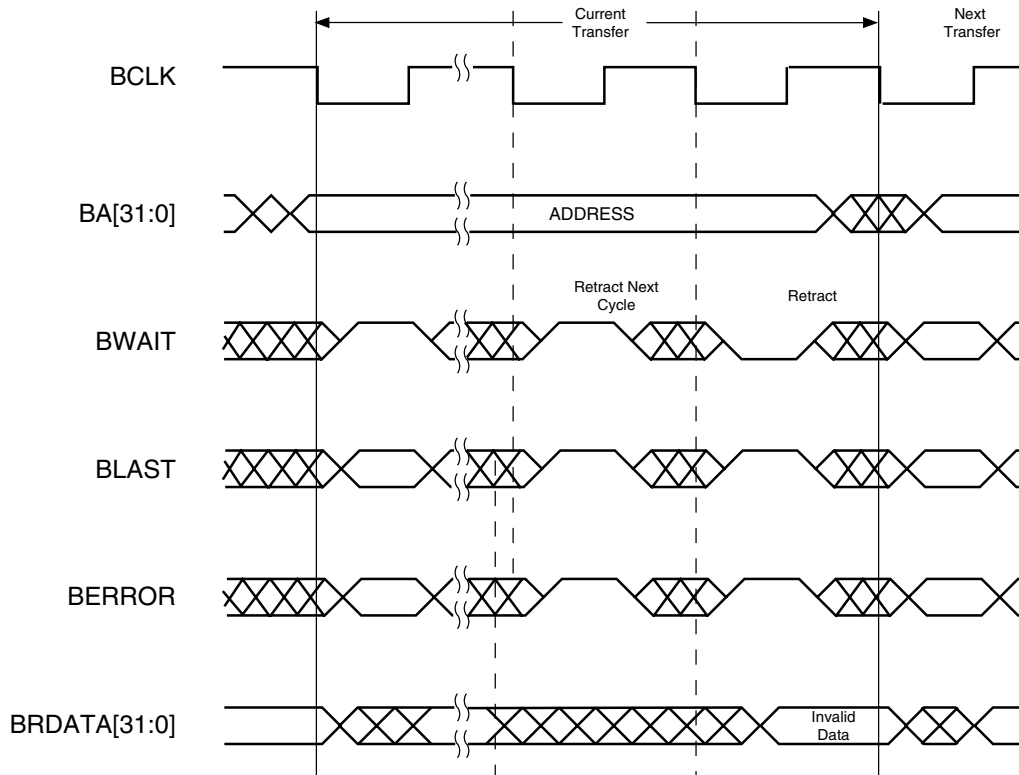
Slaves that cannot guarantee to complete transfers in a small number of wait states can potentially block the bus and stop higher priority transfers from occurring. To prevent such slaves impacting the overall system latency, a retract mechanism is provided, which allows a slave to indicate that a transfer is unable to complete at present, but the operation should be retried until it is able to complete successfully.

A retract is performed in a two-stage process, as shown in Figure 23. First, the slave responds with BWAIT, BLAST and BERROR all high, which indicates that a retract is to occur and the transfer will finish in the next bus cycle.

In the second cycle, the transfer response is BWAIT low, BLAST and BERROR both high. This indicates that the transfer has retracted and the bus is free to be used by higher priority bus masters.

Basic slaves, which have a guaranteed completion time, do not need to support the bus retract mechanism.

**Figure 23. Retract Operation**



**Response Combinations**

Table 7 shows the combinations of the three slave transfer response signals.

**Table 7. Transfer Response Combination**

BWAIT	BLAST	BERROR	Status	Description
0	0	0	DONE	Complete: Transfer Successful
0	0	1	ERROR	Complete: Transfer Error
0	1	0	LAST	Complete: Cannot continue with Burst
0	1	1	RETRACT	Complete: Bus Retract
1	0	0	WAIT	Incomplete: Insert Wait Cycle
1	0	1	–	Reserved
1	1	0	–	Reserved
1	1	1	RETNEXT	Bus Retract Next Cycle

To ensure that the bus remains synchronized, a transfer response must be driven every cycle. During bus transfers, when a slave is selected and its appropriate DSELx signal is asserted, the slave is responsible for driving the transfer response signals.

The bus decoder is responsible for driving the transfer response signals during:

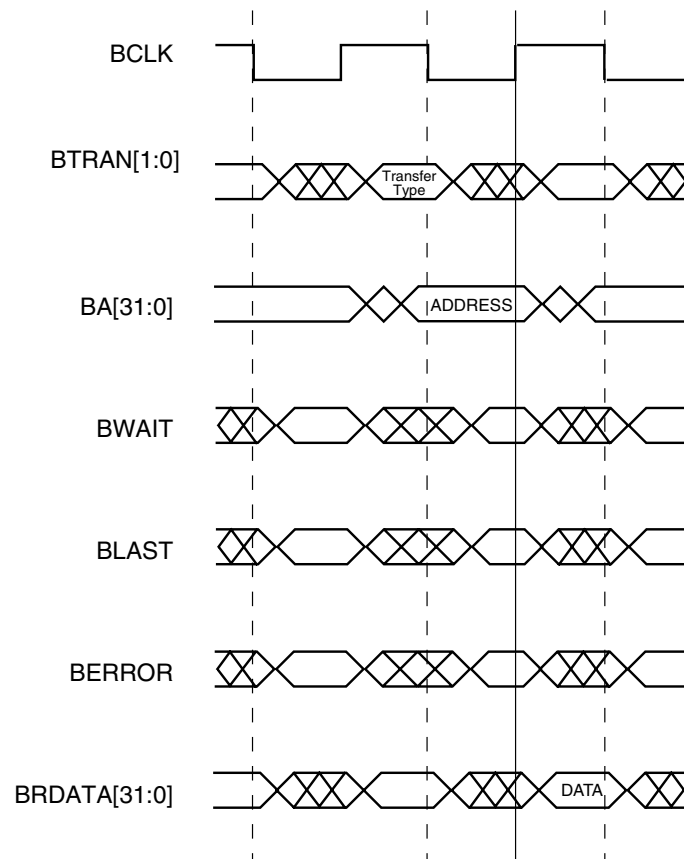
- Address-only transfers
- Decode cycles
- Transfers to an address space where no slave is defined
- Transfers to protected areas, when the access permissions are not met
- Unaligned transfers that are not supported by the memory system



## Transfer Response Timing

The transfer response signals must be set up valid before the rising edge of BCLK.

**Figure 24.** Transfer Response Signal Timing



## Data Bus

The uni-directional data buses, BRDATA[31:0] and BWDATA[31:0], are used to transfer data between bus masters and slaves. The size and direction of the transfer are given by the control signals, as described under “Address and Control Information” on page 25.

The data buses may not be driven at valid value during the first BCLK low phase of a non-sequential transfer.

- During a write transfer, the master drives the data bus BWDATA[31:0].
- During a read transfer, the slave must drive the data bus BRDATA[31:0].

The following diagrams show some examples of how the data buses are driven.

Figure 25 shows the example of a non-sequential write transfer.

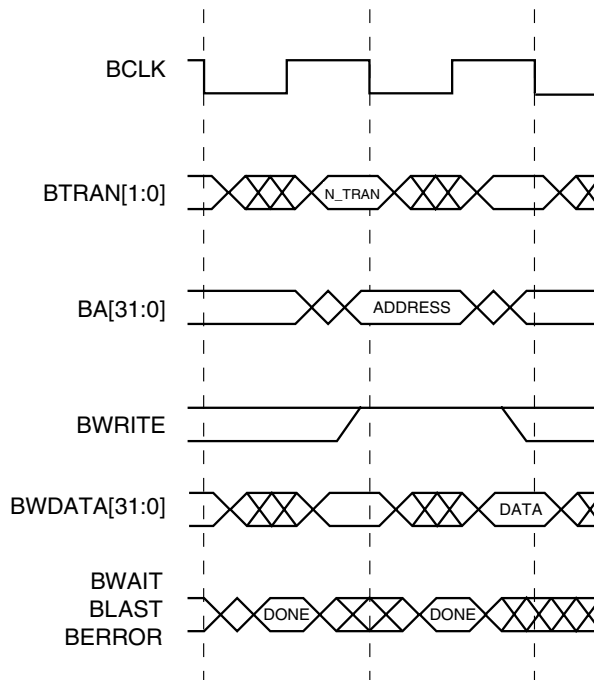
When a write transfer is extended using BWAIT, the data remains valid through the BCLK low phase of the extra cycles that are required to complete the transfer, as shown in Figure 26.

Sequential transfers are shown in Figure 27 on page 35.

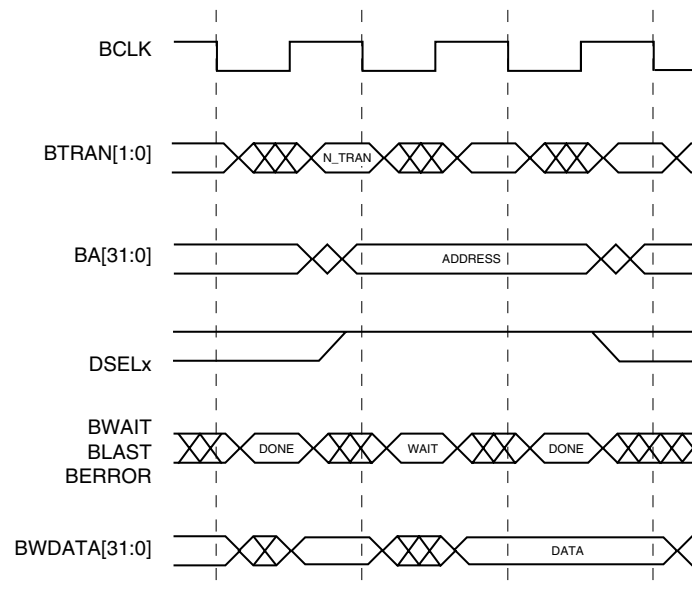
During read cycles, the slave drives the read data bus.

There is no requirement for the slave to drive the read data bus at valid value throughout the transfer. The only requirement is that the data is driven such that it is valid by the end of the last BCLK high phase of the transfer. See Figure 28 on page 36.

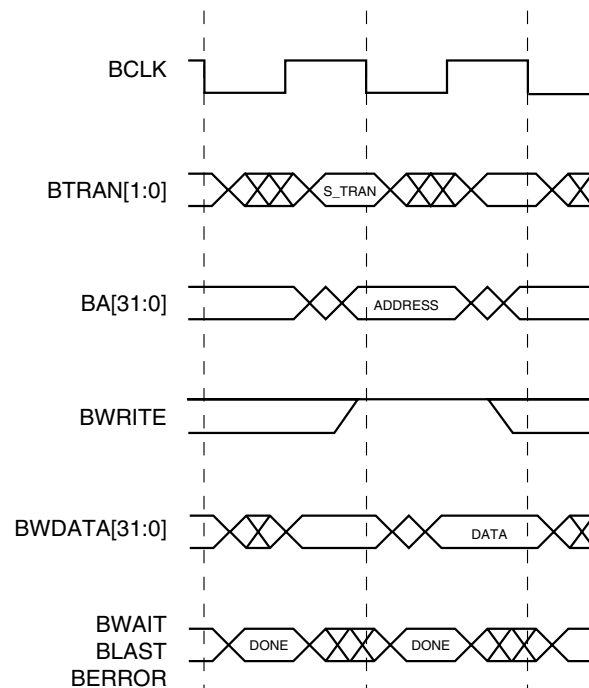
**Figure 25. Non-sequential Write Transfer**



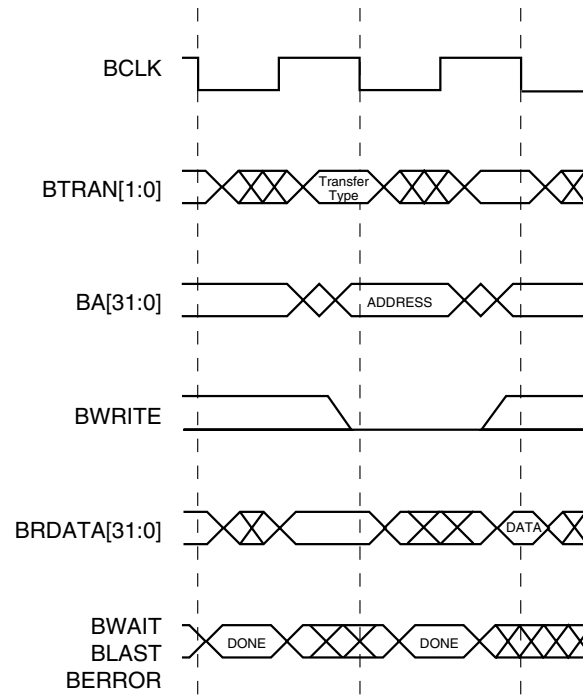
**Figure 26.** Extended Write Transfer



**Figure 27.** Sequential Write Transfer



**Figure 28.** Read Transfer



## Arbitration Signals

### **AREQx – Bus Request (Active high)**

AREQx is the request signal from a master to the arbiter, which indicates that the master requires the bus. Each master has an AREQx signal, which changes during the high phase of BCLK. Refer to Figure 11 on page 19.

### **AGNTx – Bus Grant**

The grant signal from the arbiter to a bus master indicates that the bus master is currently the highest priority master requesting the bus. There is an AGNTx signal for each bus master in the system.

It is important to note that AGNTx does not indicate which master is currently granted the bus. Instead, it shows which master is currently the highest priority and at the completion of a transfer, as indicated by BWAIT low, the master that has AGNTx asserted is granted the bus.

AGNTx is changed by the arbiter during the low phase of BCLK and remains valid through the high phase.

When AGNTx is high, the master must:

- drive the BTRAN signals during BCLK high.
- become granted when transfer is completed.

Refer to Figure 11 on page 19.

### **BLOKx – Bus Lock (Active high)**

The lock signal from a bus master to the arbiter indicates the following transfer is indivisible from the current transfers and no other bus master should be given access to the bus.

A master must always drive a valid level on its BLOKx signal when granted the bus, even if the master is not performing any transfers. This is necessary to ensure the arbitration process can continue.

If all BLOKx are low, the arbiter will grant the highest priority master requesting the bus. See Figure 11 on page 19.

If a BLOKx is high and if the corresponding master is granted, the arbiter will keep the same master granted.

BLOKx are sampled by the arbiter during the low phase of BCLK and it must be valid such that the arbiter can generate valid AGNTx outputs before the rising edge of BCLK. BLOKx are ignored by the arbiter during the bus master handover cycle.

## APB Signal Description

### **PSTB – Peripheral Strobe**

The peripheral strobe signal is used to time all accesses on the peripheral bus. The falling edge of PSTB is aligned with the falling edge of BCLK.

### **PSTB\_RISING – Peripheral Strobe on BCLK Rising Edge**

This signal can be used as a clock signal to time all write accesses on the peripheral bus. The rising edge of PSTB\_RISING is derived from the rising edge of BCLK.

### **PA[31:0] – Peripheral Address Bus**

The peripheral address bus is driven by the peripheral bus bridge unit. A full 32-bit peripheral address is not typically required and only the address lines used by the slave devices would be implemented. APB peripherals are accessed on word boundaries and the lowest two bits of the address bus, PA[1:0], are not usually required.

The peripheral address will become valid before PSTB goes high and remain valid after PSTB goes low.

### PSELx – Peripheral Select (Active high)

A PSELx signal exists for each peripheral bus slave. It is within the peripheral bus bridge unit and indicates that slave device “x” is selected and a data transfer is required.

This signal has the same timing as the peripheral address bus. It will become valid before PSTB goes high and remain valid after PSTB goes low.

### PWRITE – Peripheral Transfer Direction

When high, this signal indicates a write cycle and when low, a read cycle. This signal has the same timing as the peripheral address bus; it will become valid before PSTB goes high and remain valid after PSTB goes low.

### PRDATA[31:0] and PWDATA[31:0] – Peripheral Data Buses

The uni-directional peripheral data bus PWDATA[31:0] is driven by the peripheral bus bridge unit during write cycles (when PWRITE is high). It will become valid before PSTB goes high and remain valid after PSTB goes low.

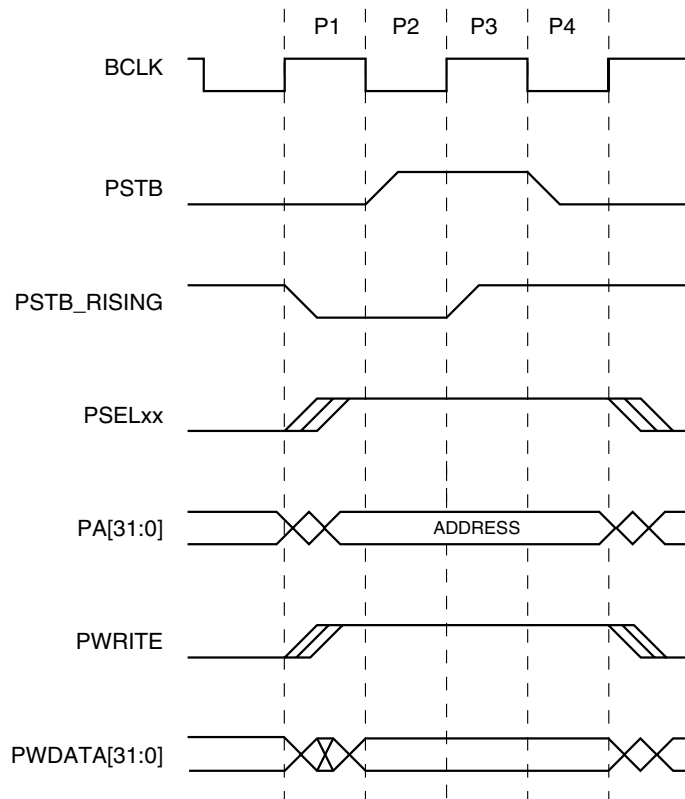
During read cycles (when PWRITE is low), the data bus PRDATA[31:0] is driven by the selected slave and must be valid before the rising edge of BCLK when PSTB is high.

If required, APB peripherals may also make use of common ASB signals, such as BCLK and BnRES.

### APB Write Cycle

The APB write cycle, as shown in Figure 29, has all the APB signals set up before the strobe and held valid after the strobe. The falling edge of the strobe is derived from the falling edge of the main system clock, BCLK.

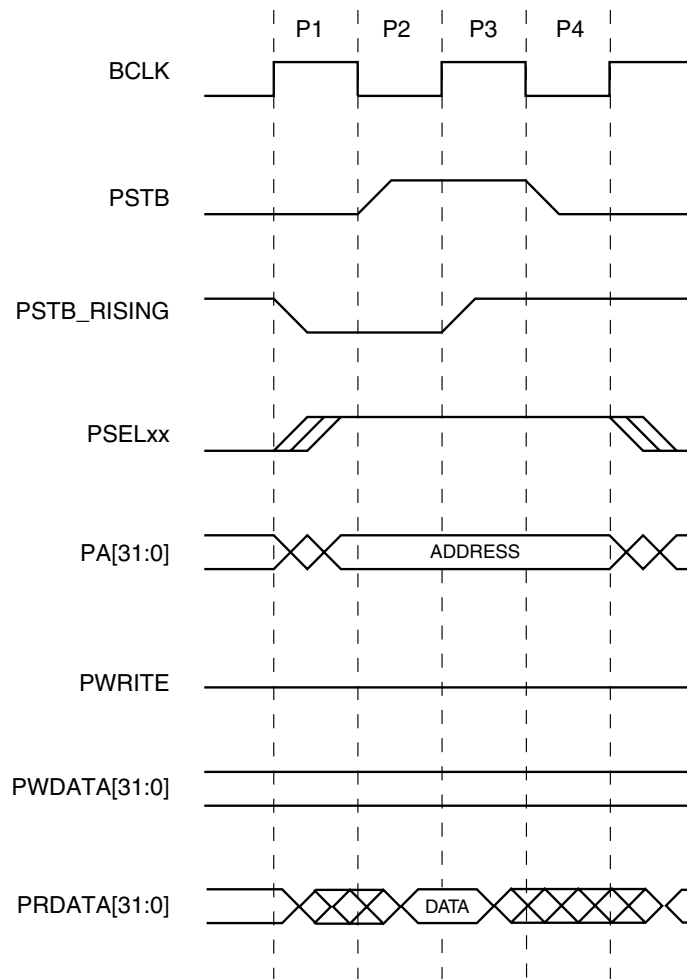
**Figure 29.** APB Write Cycle



## APB Read Cycle

Figure 30 shows an APB read cycle, which has the address and control set up before the strobe and held valid after the strobe. There is no requirement to drive the data at valid value throughout the access, but read data must be valid prior to the falling edge of the strobe, which will be coincident with the falling edge of BCLK.

**Figure 30.** APB Read Cycle



## Atmel AMBA System Architecture

### General Considerations

Note that in an AMBA system, signals:

- must be generated during a phase of BCLK
- must be valid for an edge of BCLK

For the BWAIT (or BLAST or BERROR) signal, for example, the AMBA specification requires that it is valid on the rising edge of the system clock BCLK.

In order to connect the ARM7TDMI core to the ASB, the ARM7TDMI bus interface requires that a latch be used. This is due to the requirement of the ARM core that its wait signal (nWAIT) is valid from the rising edge until the falling edge of the clock MCLK. Nonetheless, the slaves of this AMBA system can provide BWAIT via a D-Flip-Flop clocked on the falling edge of BCLK.

Atmel peripherals are AMBA-compliant. Whenever possible, they use flip-flops and the majority of the control signals are issued from registers.

The Atmel system uses only multiplexed buses (no tri-state). Nevertheless, any AMBA peripheral can be connected to an Atmel system without any modification (excepted those related to the use of tri-state buses).

### Data Exchanges with Multiple Clocks

The standard practice is to have one clock per APB peripheral in order to minimize power consumption in a design. However, this may have an effect on data exchanges. For example, if a system contains two peripherals, they must:

- have their clocks derived from the same clock
- be working with the same edge of their clock

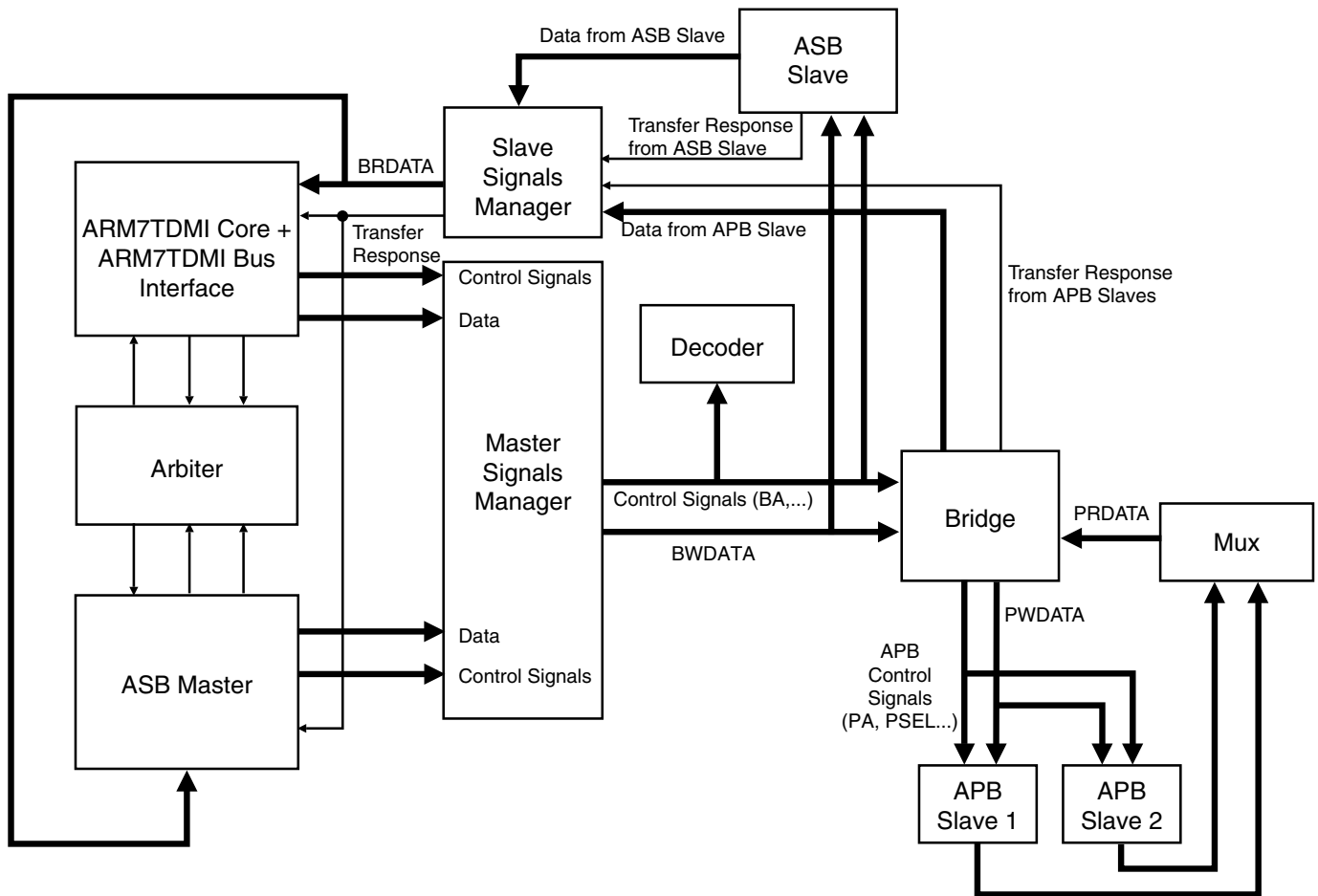
In order to ensure that correct data exchanges occur, two workarounds exist:

1. Balance the two clock trees
2. Sample output data on the opposite edge

Note that the second solution is preferable, especially with respect to place and route guidelines.



**Figure 31.** Example of the System Architecture of an Atmel Microcontroller



## AMBA Components

This section describes each of the elements in an AMBA system and provides the generic timing parameters that are required to analyze an AMBA design.

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

Unless otherwise stated, the timing parameters apply to both the rising and falling edges of the signal.

## ASB Bus Slave

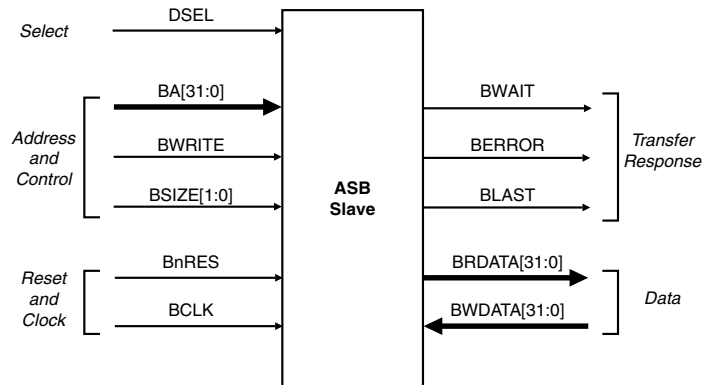
An ASB bus slave responds to transfers initiated by bus masters within the system. The slave uses a DSEL select signal from the decoder to determine when it should respond to a bus transfer. All of the other signals that are required for the transfer, such as the address and control information, will be generated by the bus master.

The decoder greatly simplifies the slave interface and removes the need for the slave to understand the different types of transfer that may occur on the bus.

## Interface Diagram

Figure 32 shows an ASB bus slave interface.

**Figure 32.** ASB Slave Interface



## Bus Slave Interface Description

### Transfer Response

A slave must provide a transfer response in the low phase of BCLK when DSEL is asserted. Using the BWAIT, BERROR and BLAST signals, one of the following responses must be generated.

**Wait** The transfer must be extended before it can complete.

**Done** The transfer has completed successfully.

**Last** The transfer has completed successfully, but the slave is unable to accept further burst transfers or a memory boundary has been reached.

**Error** The transfer has not completed successfully. The error condition will be signaled to the bus master so that it is aware the transfer has not completed correctly.

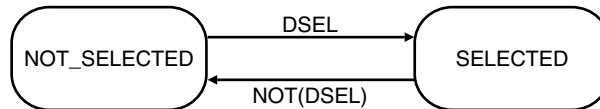
**Retract** The transfer has not yet completed, so the bus master should retry the transfer. The Retract response is used by a slave to prevent the bus from being locked up by a transfer, which may take many cycles to complete.

Many slaves will only use the wait and done responses and in this case, when a transfer response is supplied, both BERROR and BLAST will be low.

## Data

The slave interface is implemented as a simple state machine, operating from the falling edge of the clock to determine when data transfer can occur. During reset, the state machine enters the NOT\_SELECTED state. See Figure 33.

**Figure 33.** ASB Slave Bus Interface State Machine

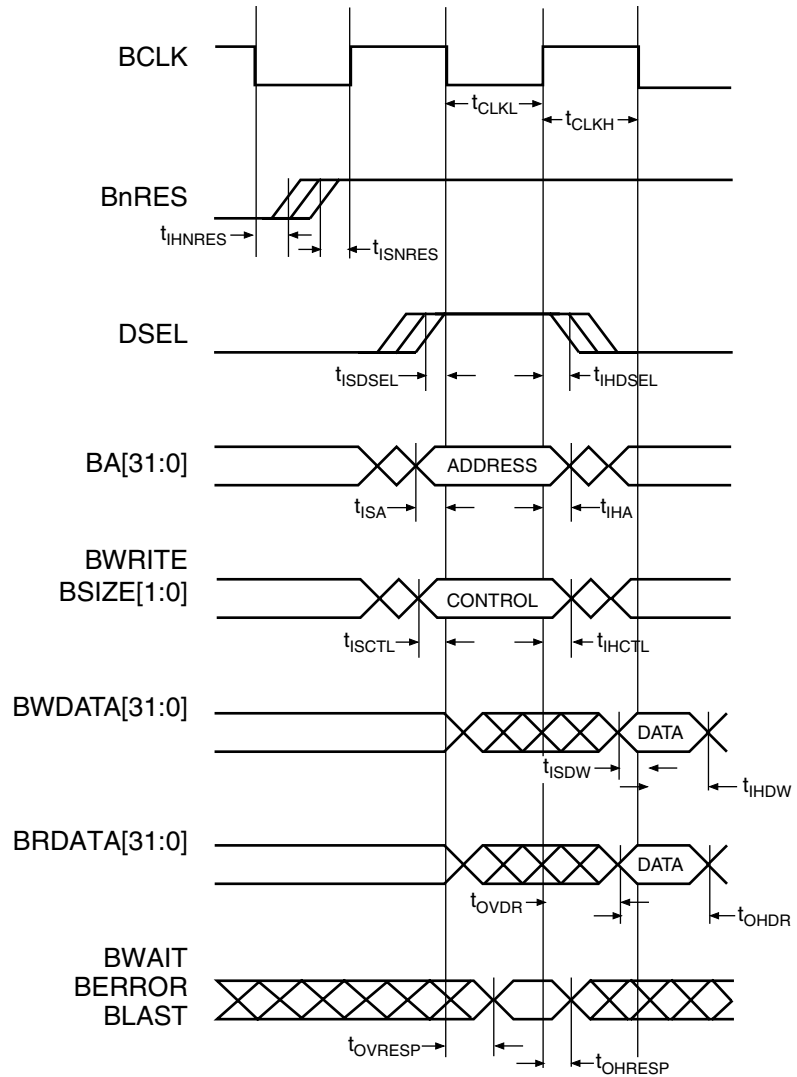


For write transfers, the slave samples the data on the falling edge of the clock when in the SELECTED state. If required, the slave may extend the transfer using the transfer response signals described above.

## Timing Diagrams

The timing parameters related to an access to an ASB bus slave are shown in Figure 34 below.

**Figure 34.** ASB Slave Transfer



## Timing Parameters

The timing parameters related to an ASB bus slave are given in Table 8 and Table 9; Table 8 is for input signals, Table 9 is for output signals.

**Table 8.** ASB Slave Input Parameters

Parameter	Description
$t_{CLKL}$	BCLK low time
$t_{CLKH}$	BCLK high time
$t_{ISNRES}$	BnRES de-asserted setup to rising BCLK
$t_{IHNRRES}$	BnRES de-asserted hold after falling BCLK
$t_{ISDSEL}$	DSEL setup to falling BCLK
$t_{IHDSSEL}$	DSEL hold after rising BCLK
$t_{ISA}$	BA[31:0] setup to falling BCLK
$t_{IHA}$	BA[31:0] hold after rising BCLK
$t_{ISCTL}$	BWRITE and BSIZE[1:0] setup to falling BCLK
$t_{IHCTL}$	BWRITE and BSIZE[1:0] hold after rising BCLK
$t_{ISDW}$	For write transfers, BWDATA[31:0] setup to falling BCLK
$t_{IHDW}$	For write transfers, BWDATA[31:0] hold after falling BCLK

**Table 9.** ASB Slave Output Parameters

Parameter	Description
$t_{OVRESP}$	BWAIT, BERROR and BLAST valid after falling BCLK
$t_{OHRESP}$	BWAIT, BERROR and BLAST hold after rising BCLK
$t_{OVDR}$	For read transfers, BRDATA[31:0] valid after rising BCLK
$t_{OHDR}$	For read transfers, BRDATA[31:0] hold after falling BCLK

It is interesting to note that if the bus slave is designed such that the decoder, address and control signals are all sampled on the falling edge of BCLK, then an entire phase of input hold time is guaranteed by the bus protocol.

It is also possible to ensure an entire phase of hold time is provided on the data bus by inserting an extra wait state into the transfer.

## ASB Bus Master

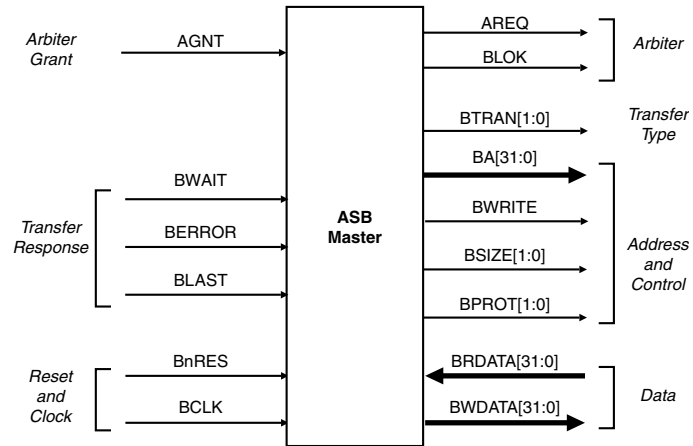
An ASB bus master has the most complex bus interface in an AMBA system. Typically, an AMBA system designer would use pre-designed bus masters and therefore would not need to be concerned with the detail of the bus master interface.

A bus master interface may also include a slave interface, either for test or for programming the operation of the bus master. In such cases a number of the interface signals will become shared between the master interface and slave interface.

## Interface Diagram

The interface diagram of an ASB bus master shows the main signal groups. See Figure 35.

**Figure 35.** ASB Bus Master Interface Diagram



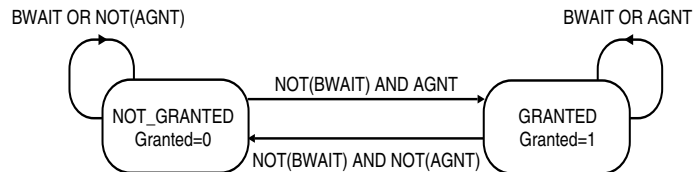
## Bus Master Interface Description

The bus master interface consists of two state machines. The first state machine determines if the master is currently granted the bus and the second, more complex, state machine is used to control the bus interface of the master.

### Granted State Machine

The granted state machine is used to determine whether the bus master has been granted the bus. It runs on the rising edge of BCLK and has only two states, GRANTED and NOT\_GRANTED. The state diagram is shown in Figure 36.

**Figure 36.** Bus Master Granted State Machine



The output from the state machine is the Granted signal, which is used in the main bus master state machine.

It is important to note that the AGNT signal may be asserted for a number of clock cycles, but it is only when AGNT is asserted and BWAIT is low that the bus master actually becomes GRANTED.

An important design consideration is that the state machine may be asynchronously reset into either state, depending on the value of the AGNT signal. During reset, one bus master in the system is set as the default bus master, as indicated by AGNT being asserted during reset, and will be reset into the GRANTED state. All other bus masters will be reset into the NOT\_GRANTED state.

## Bus Interface State Machine

The main bus interface state machine is falling-edge triggered and contains six states. The entire state diagram, as shown in Figure 37 on page 48, is quite complex but can be considered in four quadrants:

No Transfer Request Not Granted	Transfer Request Not Granted
No Transfer Request Granted	Transfer Request Granted

The “Transfer Request, Granted” quadrant contains three states, which handle bus turn-around and the retract operation.

The two internal bus master signals, Granted and Request, control the majority of the transitions around the state diagram. Granted is generated from the simpler state machine described above and Request is generated directly by the bus master. Request is asserted high when the bus master requires a transfer on the bus and is low when the bus master does not need access to the bus.

The only time when a transition around the state diagram is not controlled by Granted and Request is when the bus master is in the ACTIVE state. In this state the transition to the next state is determined by the transfer response that is received. Wait, Done, Last, Error, Retract and RetNext shown in the diagram correspond to the encodings of the transfer response signals. See Table 7 on page 32.

Note that the state diagram assumes that once the bus master has made a request for a transfer, as indicated by Request, then Request will remain asserted until the bus master has performed a transfer.

As the main bus master state machine is operating from the falling edge of the clock, it is necessary to use latched versions of the transfer response signals BWAIT, BERROR and BLAST to control the exit from the ACTIVE state.

The reset conditions are not shown on the state diagram and, in a similar manner to the granted state machine, the main bus master state machine has a complex reset term. If AGNT is asserted during reset, when BnRES is low, the bus master is the default bus master and enters the BUSIDLE state. However, if AGNT is not asserted during reset, then the bus master enters the IDLE state.

**Figure 37. Bus Master Main State Machine**

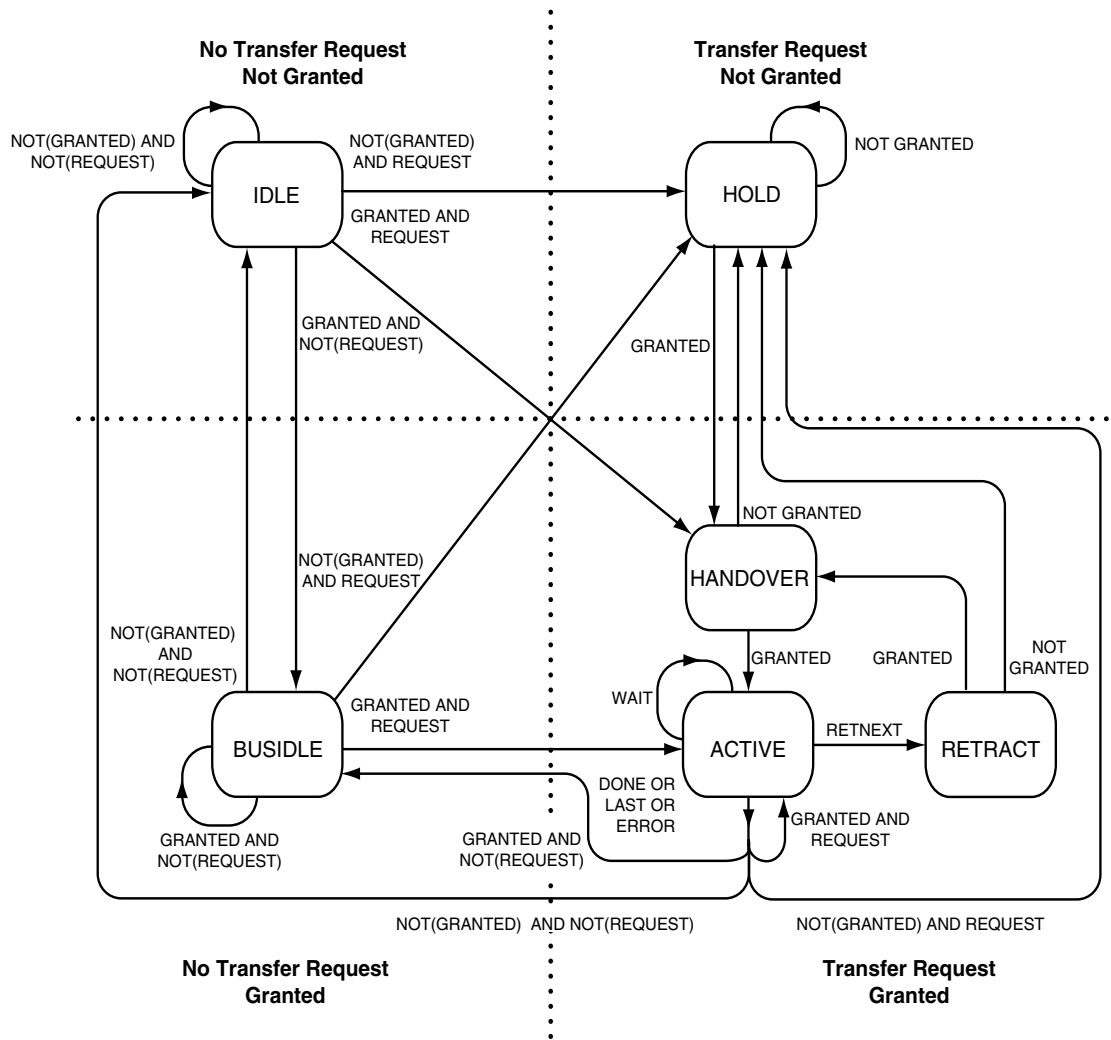




Table 10 indicates the actions that must occur in each state. Master Address Bus Enable is used to control the multiplexer enable of BA[31:0], BWRITE, BSIZE[1:0] and BPROT[1:0]. Master Data Bus Enable may be used to control the multiplexer of BWDATA[31:0].

**Table 10.** State Description

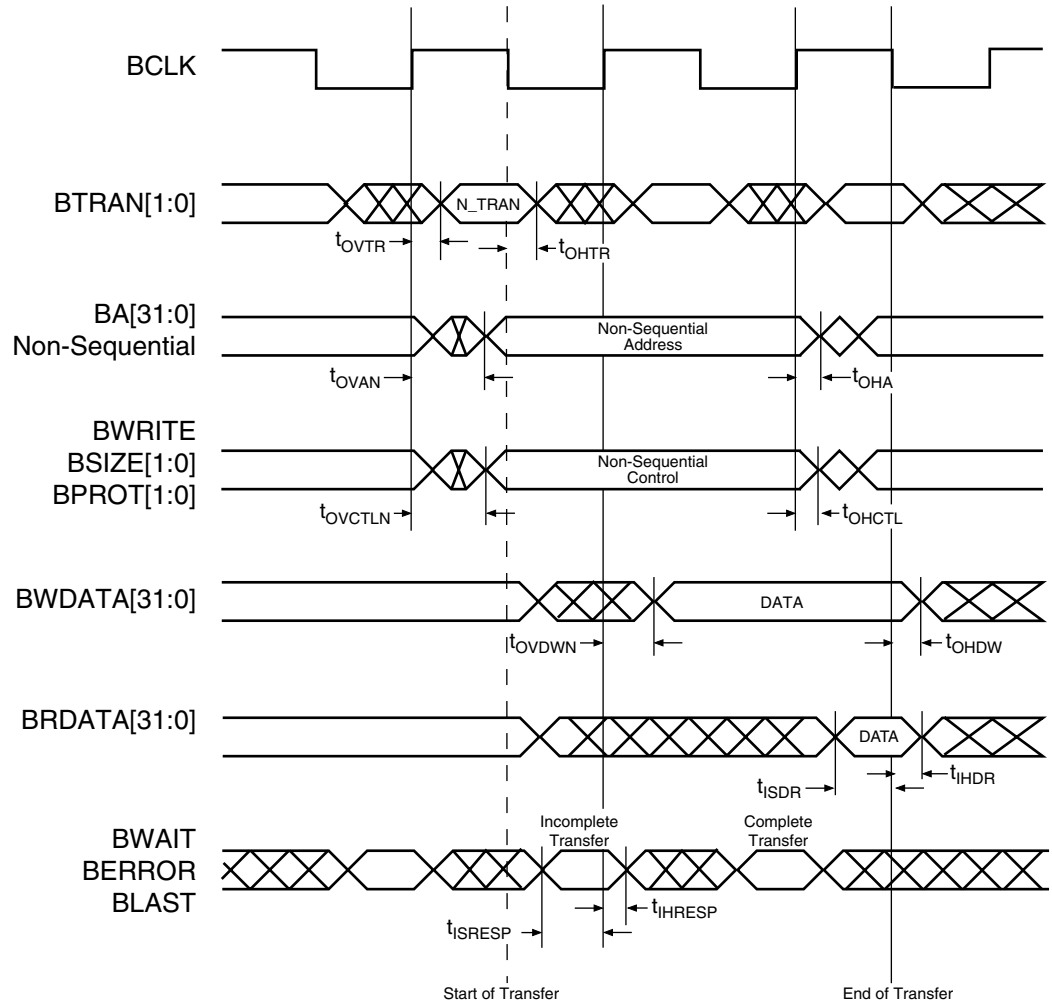
Name	Description	Actions
Idle	The master does not require the bus and is not granted.	Internal BTRAN is address-only Master Clock is enabled Master Address Bus is not valid Master Data Bus is not valid
Busidle	The master does not require the bus, but has been granted anyway.	Internal BTRAN as indicated by master Master Clock is enabled Master Address Bus Enable is generated from Granted signal Master Data Bus is not valid
Hold	The master requires the bus, but has not been granted.	Internal BTRAN is address-only Master Clock is disabled Master Address Bus is tri-state Master Data Bus is not valid
Handover	This state provides bus turnaround when changing between different bus masters.	Internal BTRAN is sequential Master Clock is disabled Master Address Bus Enable is generated from Granted signal Master Data Bus is not valid
Active	Active state when data transfers occur. Exiting this state is dependent on the transfer response.	Internal BTRAN as indicated by master Master Clock Enable is derived from BWAIT Master Address Bus Enable is generated from Granted signal Master Data Bus Enable is enabled if a write transaction
Retract	Retract state, where the rest of the elements in the system see the transfer finish, but the bus master is not advanced.	Internal BTRAN is address-only Master Clock is disabled Master Address Bus Enable is generated from Granted signal Master Data Bus Enable is enabled if a write transaction

## Bus Master Timing Diagrams

The following diagrams show the timing parameters related to an ASB bus master operating in an AMBA system.

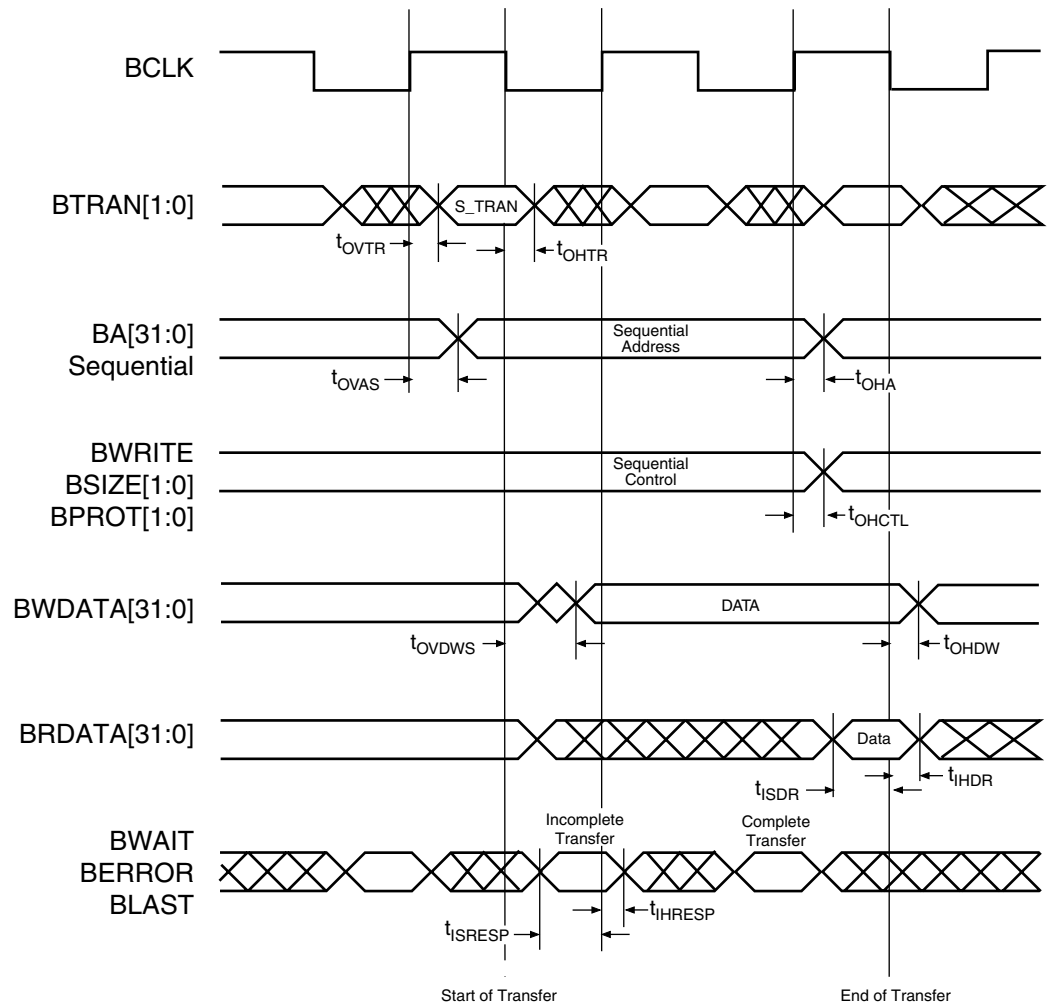
Figure 38 below and Figure 39 on page 51 show the parameters related to non-sequential and sequential transfers respectively, while Figure 41 on page 52 shows the reset signal and arbitration timing.

**Figure 38.** ASB Bus Master Non-sequential Transfer



For the non-sequential transfer shown above, the address and control signals become valid in the BCLK high phase before the start of the transfer. An important feature of the AMBA protocol is to allow for poor output valid times on non-sequential transfers, which is provided through the automatic insertion of a wait state at the start of every non-sequential transfer by the decoder (decode cycle).

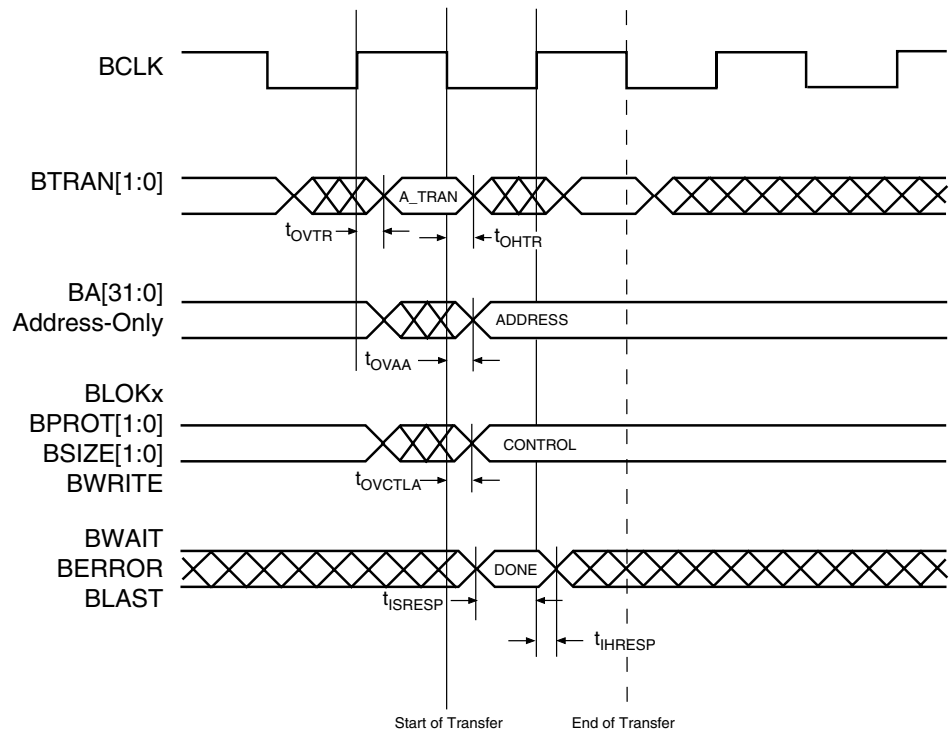
**Figure 39. ASB Bus Master Sequential Transfer**



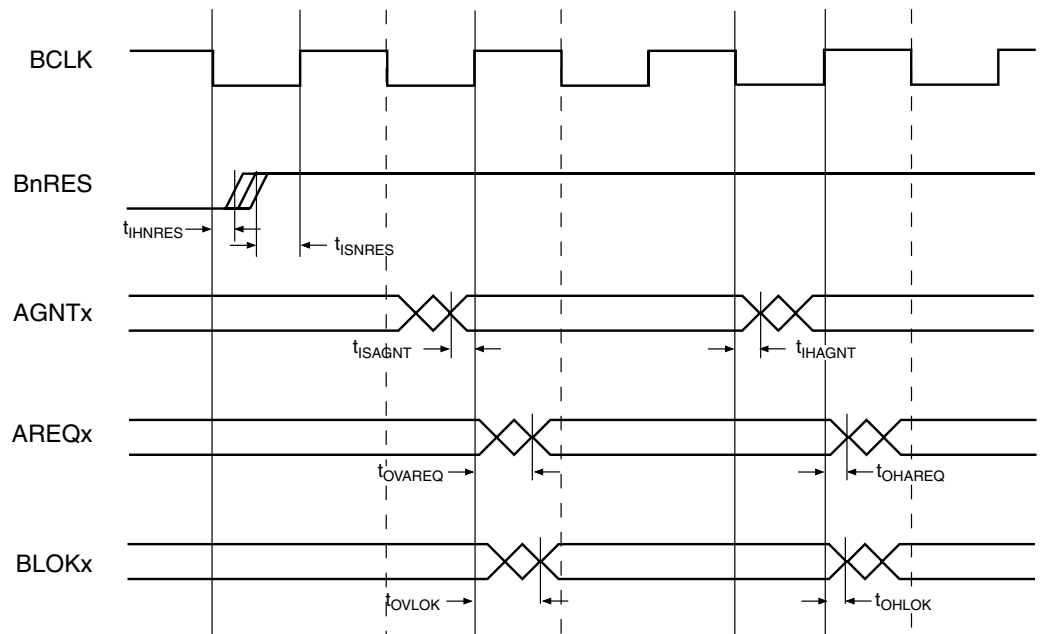
A sequential transfer has different timing parameters for the address and control signal valid times. In a typical bus master, the output valid times for sequential transfers will be far better than for non-sequential transfers. The output hold times for address, control and data are identical and independent of the transfer type.

For an address-only transfer, the address and control signals may be valid in the clock high phase before the start of the transfer (see Figure 40 on page 52), or in the case of bus master handover, may only be valid during the clock low phase of the transfer itself. The address and control valid timing parameters are only relevant when the address-only transfer is followed immediately by a sequential transfer and, in this case, the address and control signals must be driven such that they are valid during the low phase of the address-only transfer, which in turn means they are valid throughout the clock high phase that precedes the sequential transfer.

**Figure 40.** ASB Master Address-only transfer



**Figure 41.** ASB Bus Master Arbitration and Reset Signals



The BnRES signal may be asserted asynchronously, and so there is no setup and hold parameter relating to the assertion of the signal. The AREQ signal, which is an output from the bus master, changes during the high clock phase and the AGNT signal, which is returned from the arbiter, changes during the low clock phase.

## Timing Parameters

The timing parameters related to an ASB bus master operating in an AMBA system are also shown in textual form in the following two tables: Table 11 details the input signals; while Table 12 details output signals.

**Table 11.** Bus Master Input Timing Parameters

Parameter	Description
$t_{CLKL}$	BCLK low time
$t_{CLKLH}$	BCLK high time
$t_{ISNRES}$	BnRES de-asserted setup to rising BCLK
$t_{ISHNRES}$	BnRES de-asserted hold after falling BCLK
$t_{ISRESP}$	BWAIT, BERROR and BLAST setup to rising BCLK
$t_{IHRESP}$	BWAIT, BERROR and BLAST hold after rising BCLK
$t_{ISDR}$	For read transfers, BRDATA[31:0] setup to falling BCLK
$t_{IHDR}$	For read transfers, BRDATA[31:0] hold after falling BCLK
$t_{ISAGNT}$	AGNT setup to rising BCLK
$t_{IHAGNT}$	AGNT hold after falling BCLK

**Table 12.** Bus Master Output Timing Parameters

Parameter	Description
$t_{OVTR}$	BTRAN valid after rising BCLK
$t_{OHTR}$	BTRAN hold after falling BCLK
$t_{OVAN}$	For non-sequential transfers, BA[31:0] valid after rising BCLK
$t_{OVAN}$	For sequential transfers, BA[31:0] valid after rising BCLK
$t_{OVAA}$	For address-only transfers, BA[31:0] valid after falling BCLK
$t_{OHA}$	BA[31:0] hold after rising BCLK
$t_{OVCTLN}$	For non-sequential transfers, BWRITE, BSIZE[1:0] and BPROT[1:0] valid after rising BCLK
$t_{OVCTLA}$	For address-only transfers, BWRITE, BSIZE[1:0] and BPROT[1:0] valid after falling BCLK
$t_{OHCTL}$	BWRITE, BSIZE[1:0] and BPROT[1:0] hold after rising BCLK
$t_{OVDWN}$	For non-sequential write transfers, BRDATA[31:0] valid after rising BCLK
$t_{OVDWS}$	For sequential write transfers, BRDATA[31:0] valid after falling BCLK
$t_{OHDW}$	For write transfers, BWDATA[31:0] hold after falling BCLK
$t_{OVLOK}$	BLOK valid after rising BCLK
$t_{OHLOK}$	BLOK hold after rising BCLK
$t_{OVAREQ}$	AREQ valid after rising BCLK
$t_{OHAREQ}$	AREQ hold after rising BCLK

## ASB Decoder

The decoder in an AMBA system is used to perform a centralized address decoding function, which gives two main advantages:

1. Improves the portability of peripherals by making them independent of the system memory map
2. Simplifies the design of bus slaves by centralizing the address decoding and bus control functions

The three main tasks of the decoder are:

1. Address decoder
2. Default transfer response
3. Protection unit

An ASB decoder generates a select signal for each slave on the ASB bus and, under certain circumstances, will not select any slaves and provide the transaction response itself.

The decoder greatly simplifies the slave interface and removes the need for the slave to understand the different types of transfer that may occur on the bus.

An important feature of the decoder is that it is able to improve the performance of a system by providing decode cycles for address decoding. As the decoder is able to recognize if the transfer is sequential or non-sequential, it is a simple task for the decoder to only add a decode cycle when required.

The decoder actually helps to significantly improve the system performance. In a non-AMBA system the critical path of, for example, a read transfer would be as follows:

1. Address out from master
2. Address decode to select slave
3. Data out and response from slave back to bus master

However, in an AMBA system it is possible to remove the middle stage whenever the bus master is performing a sequential transfer, because it is known that the slave that is selected will be the same as the previous transfer. The decoder can use this fact to improve the system performance by only inserting a wait state for address decoding when needed, which is for non-sequential transfers. This is known as inserting a decode cycle.

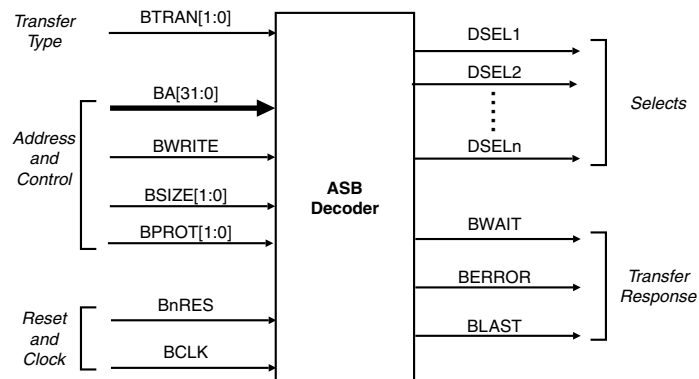
In designs where the clock frequency is low enough that an additional wait state is not required for address decoding, then the role of the decoder is simplified.

The decoder is also used to provide a number of bus maintenance functions. First, the decoder can act as a simple protection unit, which can issue an error response to a bus master that attempts to access an illegal or protected area of the memory map. The decoder also provides a transfer response during address-only transfers, when no slave is selected.

## Interface Diagram

Figure 42 shows an ASB decoder.

**Figure 42.** ASB Decoder Interface Diagram



## Decoder Description

There are two possible implementations of the decoder, depending of the performance requirements of the system design. The normal implementation of a decoder will include the insertion of decode cycles on non-sequential transfers and the breakup of burst transfers over memory boundaries. However, in some system designs, typically with a low clock frequency, the decode cycle will not be required and, hence, a simpler decoder may be implemented.

### With Decode Cycles

The decoder is implemented as a state machine, which operates from the falling edge of the clock and has four states. During reset, the state machine should enter the ADDRONLY state.

Transitions around the state machine are controlled by the transfer type for the next transfer, the transfer response from the current transfer and two internal decoder signals, DecLast and DecError. The Wait, Done, Last, Error, Retract and RetNext shown on the state diagram correspond to the encodings of the transfer response signals.

DecLast is generated by the decoder when it detects that a sequential transfer is about to cross a memory boundary and is used in combination with the external BLAST signal to force the address to be decoded, even on sequential transfers.

DecError is another decoder internal signal and is generated when the decoder detects that:

- There are no slaves present at the address of the transfer.
- The transfer is to a protected region of memory.
- The alignment of the transfer is not supported by the memory system.

The decoder performs the following functions.

In the ADDRONLY state:

- Speculatively decodes the address
- Provides a done transfer response during the BCLK low phase

DSELx is asserted during the BCLK high phase if the transfer type for the next transfer is S-TRAN and the address is valid

In the DECODE state:

- Decodes the address
- Provides a WAIT transfer response during the BCLK low phase

- DSELx is asserted during the BCLK high phase if the address is valid

In the SLAVESEL state:

- The transfer response is driven by the selected slave
- DSELx remains asserted while the transfer is waited or the next transfer is sequential and no last condition is detected

In the ERROR state:

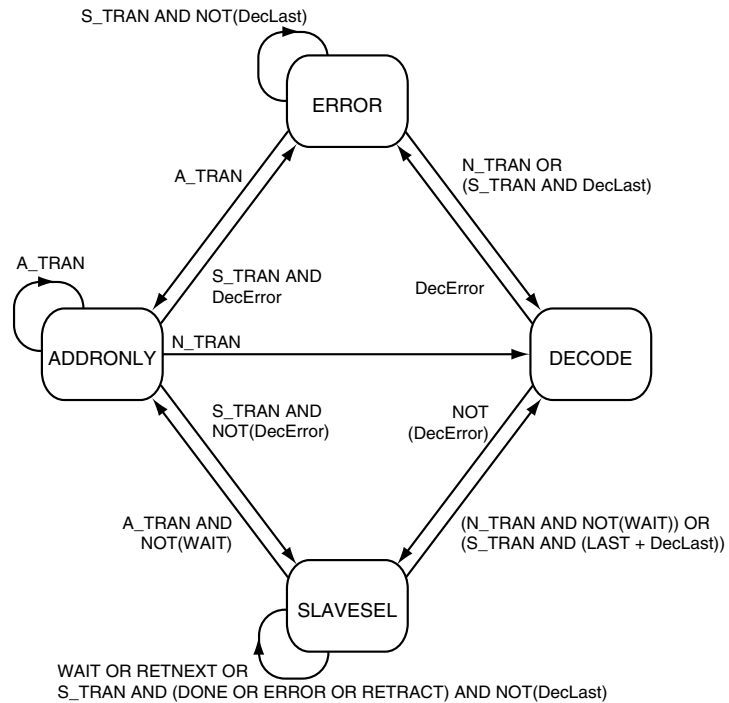
- Provides an error transfer response during the BCLK low phase



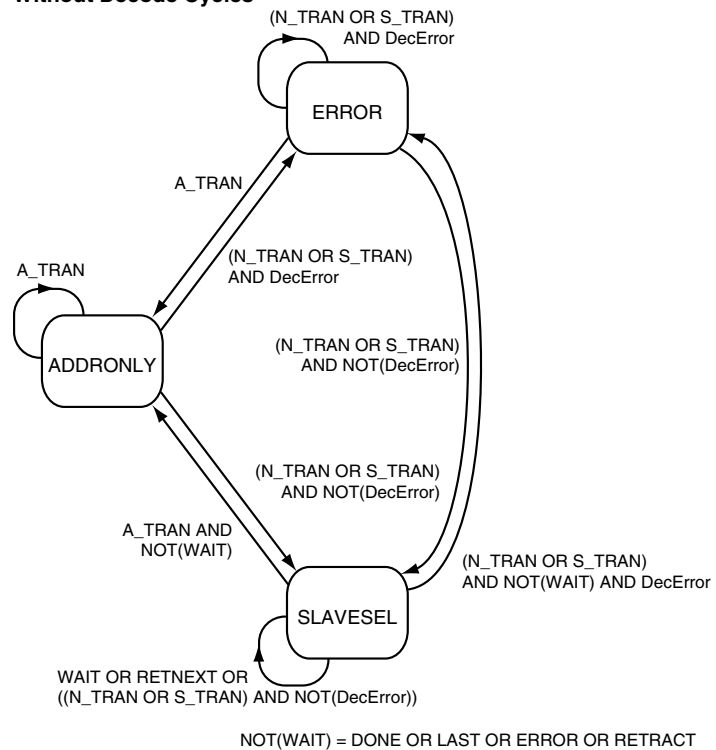
## Without Decode Cycles

A decoder that does not implement decode has the DECODE state removed; thus simplifying the state diagram, as shown in Figure 43.

**Figure 43. Decoder State Machine**



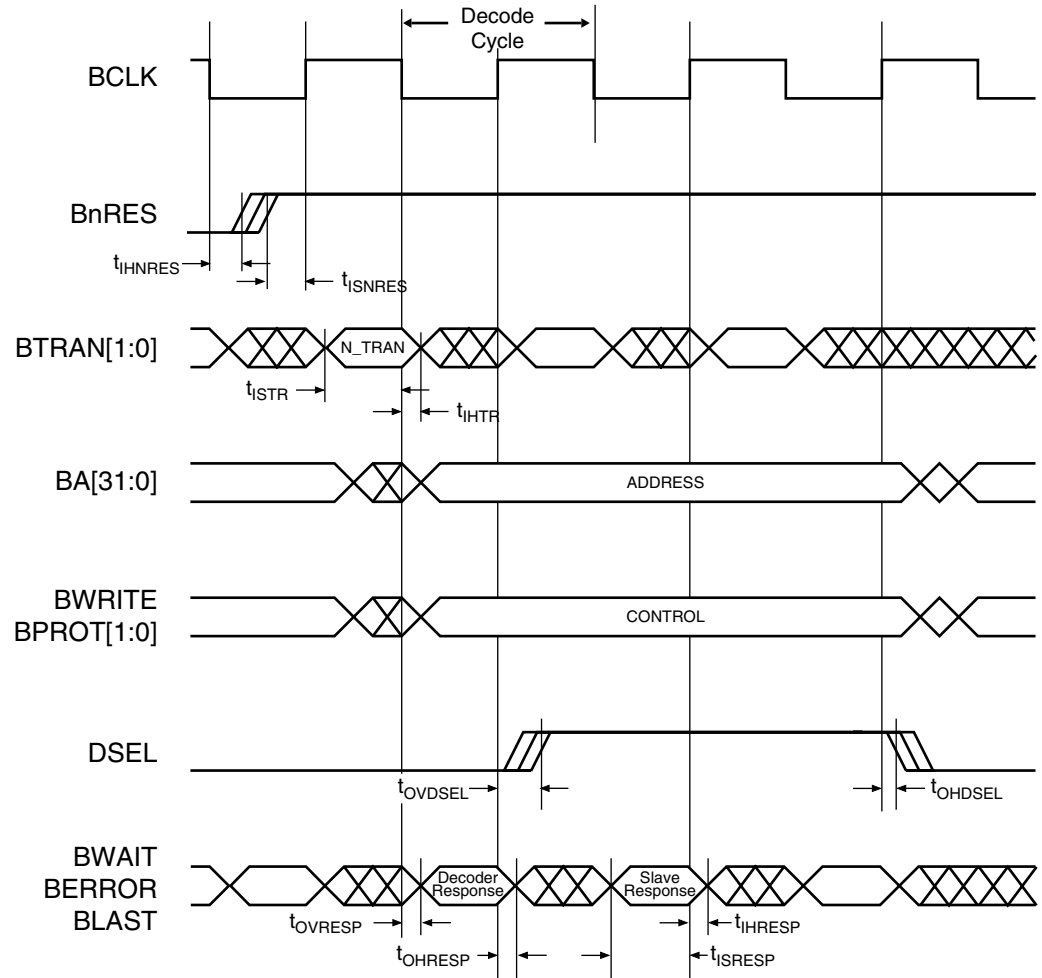
## Without Decode Cycles



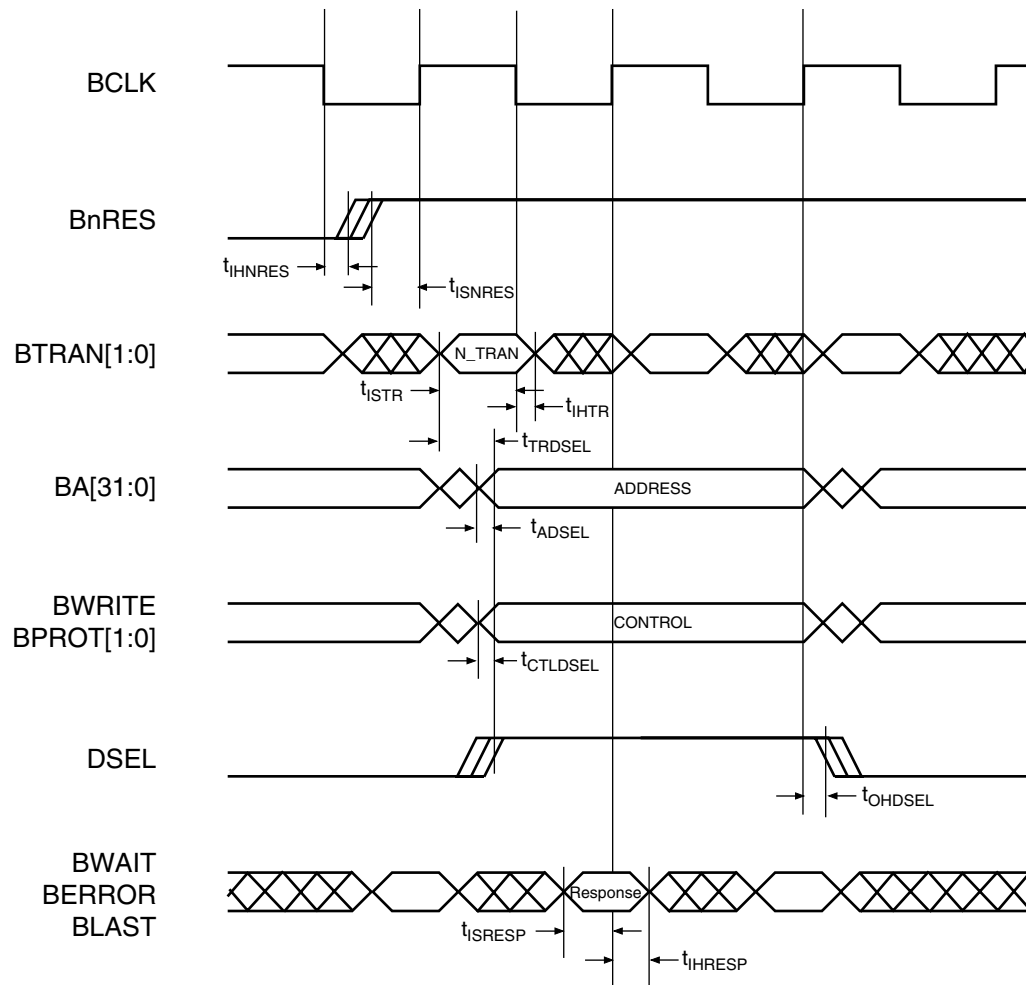
## Timing Diagrams

The timing parameters for an ASB decoder with decode cycles are shown in Figure 44. The parameters for a decoder without decode cycles are shown in Figure 45 on page 58. The main difference between the two diagrams is that when decode cycles are not inserted, then the timing of the DSEL signal becomes dependent on the address and control signal timing.

**Figure 44.** ASB Decoder with Decode Cycles



**Figure 45.** ASB Decoder without Decode Cycles



## Timing Parameters

The timing parameters related to an ASB decoder are given in the following tables; Table 13 is for input signals, Table 14 is for output signals and Table 15 is for combinatorially generated outputs.

**Table 13.** ASB Decoder Input Parameters

Parameter	Description
$t_{CLKL}$	BCLK low time
$t_{CLKH}$	BCLK high time
$t_{ISNRES}$	BnRES de-asserted setup to rising BCLK
$t_{IHNRRES}$	BnRES de-asserted hold after falling BCLK
$t_{ISTR}$	BTRAN setup to falling BCLK
$t_{IHTR}$	BTRAN hold after falling BCLK
$t_{ISRESP}$	BWAIT, BERROR and BLAST setup to rising BCLK
$t_{IHRESP}$	BWAIT, BERROR and BLAST hold after rising BCLK

**Table 14.** ASB Decoder Output Parameters

Parameter	Description
$t_{OVRESP}$	BWAIT, BERROR and BLAST valid after falling BCLK
$t_{OHRESP}$	BWAIT, BERROR and BLAST hold after rising BCLK
$t_{OVDSEL}$	DSEL valid after rising BCLK
$t_{OHDSEL}$	DSEL hold after rising BCLK

**Table 15.** ASB Decoder Combinatorial Parameters

Parameter	Description
$t_{TRDSEL}$	Delay from valid BTRAN to valid DSEL
$t_{ADSEL}$	Delay from valid BA to valid DSEL
$t_{CTLDSEL}$	Delay from valid BWRITE and BPROT[1:0] to valid DSEL

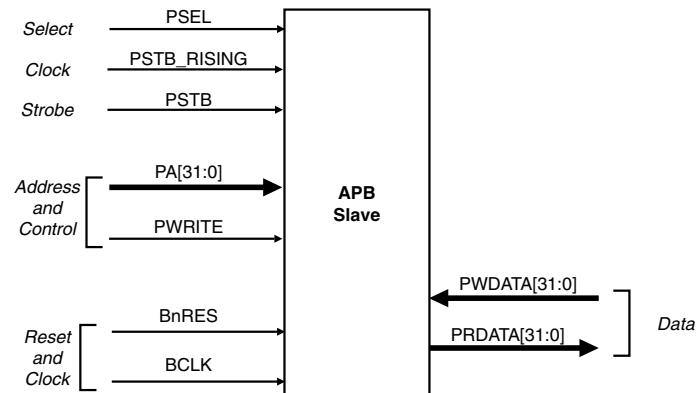
## APB Slave

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

## Interface Diagram

Figure 46 shows the signal interface of an APB slave.

**Figure 46.** APB Slave Interface Description



## APB Slave Description

The APB slave interface is very flexible.

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

- On the rising edge of PSTB
- On the rising edge of BCLK, when PSTB is high
- On the rising edge of PSTB\_RISING
- On the falling edge of PSTB

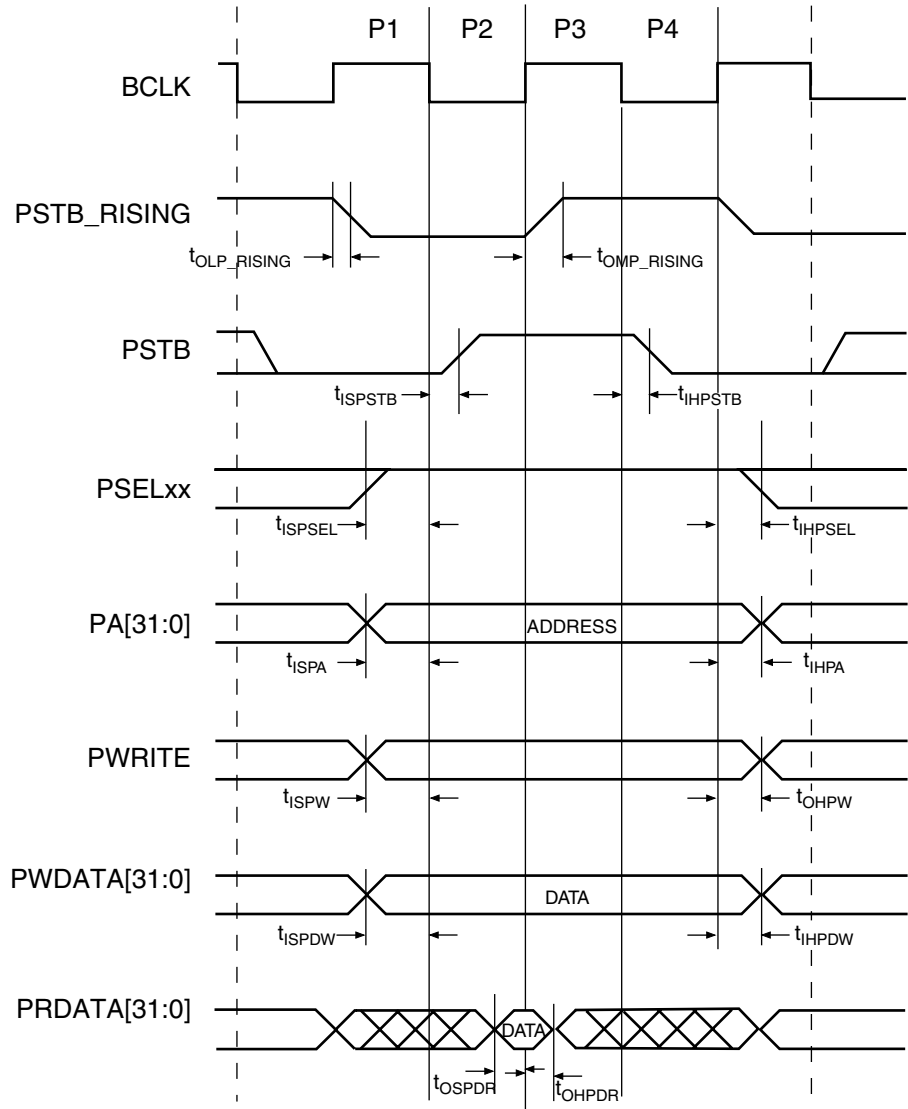
The select signal (PSELx), the address (PA) and the write signal (PWRITE) can be combined to determine which register should be updated by the write operation.

For read transfers, the data can be driven onto the data bus when PWRITE is low and both PSELx and PSTB are high, while PA is used to determine which register should be read.

## Timing Diagrams

The timing parameters related to an access to an APB bus slave are shown in Figure 47.

**Figure 47.** APB Slave Transfer



## Timing Parameters

The timing parameters related to an APB slave are given in the following tables; Table 16 is for input signals and Table 17 is for output signals.

**Table 16.** APB Slave Input Parameters

Parameter	Description
$t_{CLKL}$	BCLK low time
$t_{CLKH}$	BCLK high time
$t_{ISNRES}$	BnRES de-asserted setup to rising BCLK
$t_{IHNRRES}$	BnRES de-asserted hold after falling BCLK
$t_{OLP\_RISING}$	PTSB_Rising low after rising BCLK
$t_{OHP\_RISING}$	PTSB_Rising high after rising BCLK
$t_{ISPSTB}$	PSTB setup to falling BCLK
$t_{IHPSTB}$	PSTB hold after rising BCLK
$t_{ISPSEL}$	PSEL setup to falling BCLK
$t_{IHPSEL}$	PSEL hold after rising BCLK
$t_{ISPA}$	PA setup to falling BCLK
$t_{IHPA}$	PA hold after rising BCLK
$t_{ISPW}$	PWRITE setup to falling BCLK
$t_{IHPW}$	PWRITE hold after rising BCLK
$t_{ISPDW}$	For write transfers, PWDATA setup to falling BCLK
$t_{IHPDW}$	For write transfers, PWDATA hold after rising BCLK

**Table 17.** APB Slave Output Parameters

Parameter	Description
$t_{OSPDR}$	For read transfers, PRDATA setup to rising BCLK
$t_{OHPDR}$	For read transfers, PRDATA hold after rising BCLK



## Atmel Headquarters

### *Corporate Headquarters*

2325 Orchard Parkway  
San Jose, CA 95131  
TEL 1(408) 441-0311  
FAX 1(408) 487-2600

### *Europe*

Atmel SarL  
Route des Arsenaux 41  
Casa Postale 80  
CH-1705 Fribourg  
Switzerland  
TEL (41) 26-426-5555  
FAX (41) 26-426-5500

### *Asia*

Atmel Asia, Ltd.  
Room 1219  
Chinachem Golden Plaza  
77 Mody Road Tsimhatsui  
East Kowloon  
Hong Kong  
TEL (852) 2721-9778  
FAX (852) 2722-1369

### *Japan*

Atmel Japan K.K.  
9F, Tonetsu Shinkawa Bldg.  
1-24-8 Shinkawa  
Chuo-ku, Tokyo 104-0033  
Japan  
TEL (81) 3-3523-3551  
FAX (81) 3-3523-7581

## Atmel Operations

### *Memory*

Atmel Corporate  
2325 Orchard Parkway  
San Jose, CA 95131  
TEL 1(408) 436-4270  
FAX 1(408) 436-4314

### *Microcontrollers*

Atmel Corporate  
2325 Orchard Parkway  
San Jose, CA 95131  
TEL 1(408) 436-4270  
FAX 1(408) 436-4314

### *Atmel Nantes*

La Chantrerie  
BP 70602  
44306 Nantes Cedex 3, France  
TEL (33) 2-40-18-18-18  
FAX (33) 2-40-18-19-60

### *ASIC/ASSP/Smart Cards*

Atmel Rousset  
Zone Industrielle  
13106 Rousset Cedex, France  
TEL (33) 4-42-53-60-00  
FAX (33) 4-42-53-60-01

Atmel Colorado Springs  
1150 East Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906  
TEL 1(719) 576-3300  
FAX 1(719) 540-1759

Atmel Smart Card ICs  
Scottish Enterprise Technology Park  
Maxwell Building  
East Kilbride G75 0QR, Scotland  
TEL (44) 1355-803-000  
FAX (44) 1355-242-743

### *RF/Automotive*

Atmel Heilbronn  
Theresienstrasse 2  
Postfach 3535  
74025 Heilbronn, Germany  
TEL (49) 71-31-67-0  
FAX (49) 71-31-67-2340

Atmel Colorado Springs  
1150 East Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906  
TEL 1(719) 576-3300  
FAX 1(719) 540-1759

### *Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom*

Atmel Grenoble  
Avenue de Rochepleine  
BP 123  
38521 Saint-Egreve Cedex, France  
TEL (33) 4-76-58-30-00  
FAX (33) 4-76-58-34-80



### © Atmel Corporation 2002.

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

ATMEL® is the registered trademark of Atmel.

ARM® and ARM Powered® are the registered trademarks of ARM Limited. AMBA™ and ARM7TDMI™ are the trademarks of ARM Limited. Other terms and product names may be the trademarks of others.

### *e-mail*

[literature@atmel.com](mailto:literature@atmel.com)

### *Web Site*

<http://www.atmel.com>



Printed on recycled paper.