Advanced Architectures

A2R

Synchronous Register Bus

Revision: 1.01

Author:	Roger Thorpe
Release Date:	April 2007

Table of Contents

Table of Figures

Table of Tables

1 Introduction

A2R provides an interconnection mechanism between control registers in an ASIC design and any number of control devices; CPUs, debug ports etc. The bus is especially suited for synthesizable designs. It is specifically developed to meet the challenges of long interconnect delays in large System-on-chip designs and can be tailored to match system clock rates.

A2R is a fully synchronous bus and is user configurable to provide a wide variety of performances to best match the system level requirements of a particular implementation. It may be configured for bus width, the addition of special bus fields to transfer custom information and the selection of the arbitration algorithm. The bus may also be segmented to allow concurrent access within segments.

The A2R bus is designed as a utility mechanism that provides debug access and register access to configuration and control modules in a system. These accesses are usually not time-critical and the full performance of the system bus (e.g. A2R bus) is not necessary.

1.1 Principal Features

- 8-bit, 16-bit, or 32-bit data path
- Geographic device selection (no configuration required)
- Clock rate is implementation dependant
- Fully synchronous operation
- Configurable Interconnect types
  - Multiplexor
  - Pseudo tri-state
  - Point-to-Point Ring
  - Hierarchical

1.2 Extensions

- Parity protection on any or all buses
- Additional User-defined bus fields
- OCP Compatibility

1.3 Sample Configurations

Figure 1 shows an example of a hierarchy of A2R buses. The A2R does not use a bridge, per se; rather it uses what is called a gate. In order to access through the gate an master must assert a higher-level request. When this request is granted the master has access to the higher level and can send its access upwards. This mechanism allows the masters in separate segments to access within their segment at any time and only at a higher level through a higher-level arbitration. The A2R vertical segment to the left in the drawing performs arbitration amongst the three gates and to any other top level masters. Once a request is granted the master will send an access to across this bus to another TAP or GATE.

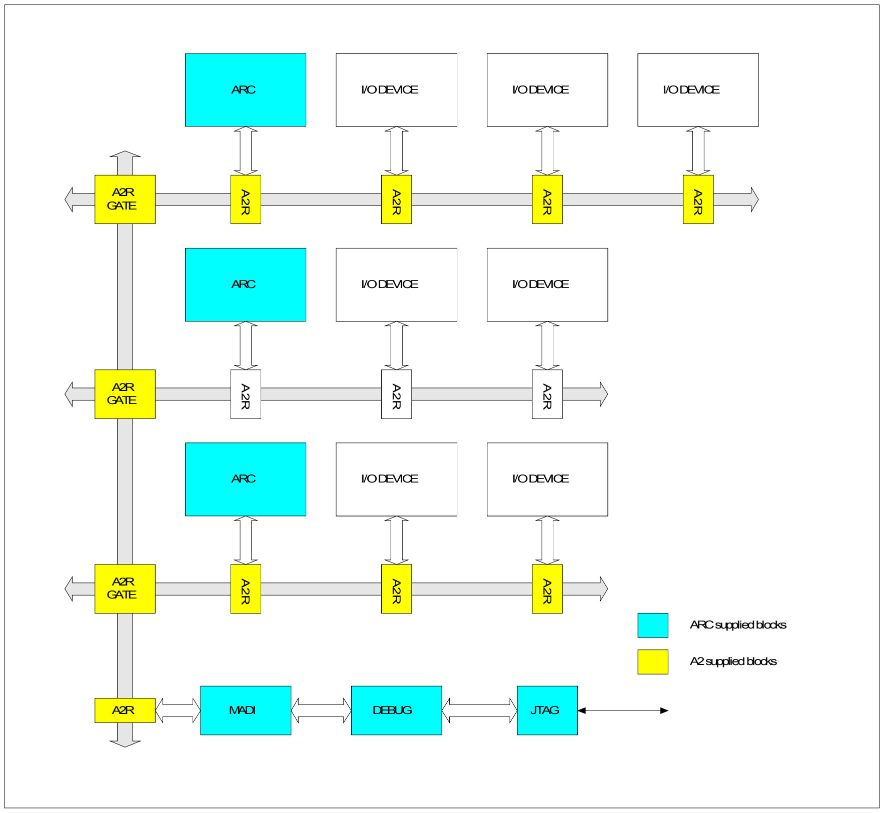

Figure 1.1: Hierarchy example of A2R buses

A request arriving at a destination gate will assert a high priority request so that it gets the next transaction on the downstream segment. In the upper right-hand side of the figure is a gate that only has slave devices below it. This slave gate will only open when an access to the address space below is received. This is used primarily to reduce the loading of a segment and if a serial point-to-point fabric is selected then it will also reduce access times when not accessing the devices in the lower segment.

The A2R can be configured as either a parallel OR fabric or as a point-to-point fabric. The point-to-point fabric has several advantages as all bus links have a single fan-out and fan-in and may also be pipelined. This eases the constraints required for the bus in the physical design process as the bus can be tuned to achieve a specific clock speed rather easily and long lines across a chip can be accommodated. In figure 1.1, for example, the leftmost bus could well be a physically long bus but by pipelining it at one or more A2R GATES it can be broken up into different clock cycles so that timing can be met. As the accesses over this higher level are less frequent then the added access times are not usually an issue.

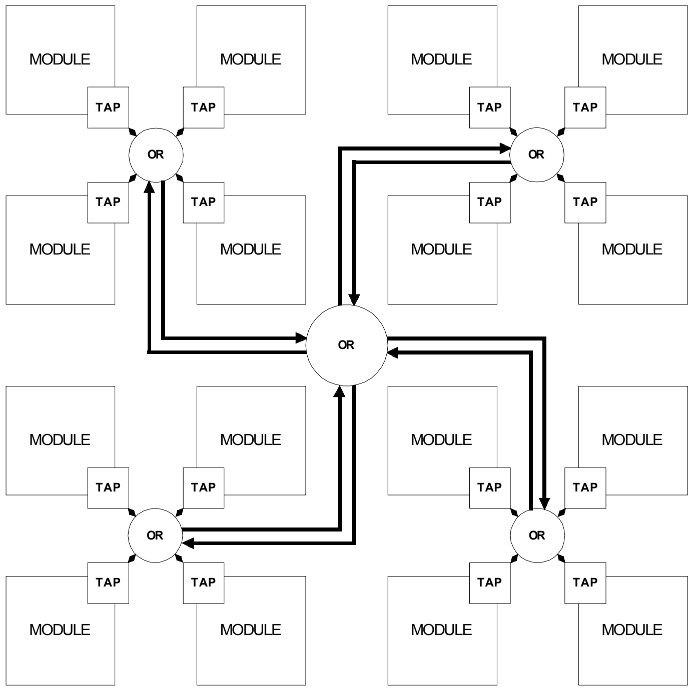

Figure 1.2: Star topology

Figures 1.2 and 1.3 illustrate the difference between star and ring topologies for the system in Figure 1.1. The star topology requires that all signals be sent to a central point to be OR-ed together. They are then driven back to using high drive buffers to all receivers. The Manhattan distance for a signal to travel from one device to another and back is 8 assuming each module is a block. The issue is that there must be a number of large drivers at the top level to buffer the signals back to the receivers. Also there are 8 sets of buses traversing the chip 4 inbound and 4 outbound. A ring topology removes the need for any gates on the top-level, reduces the number of wires by 50% and maintains the round trip Manhattan distance.

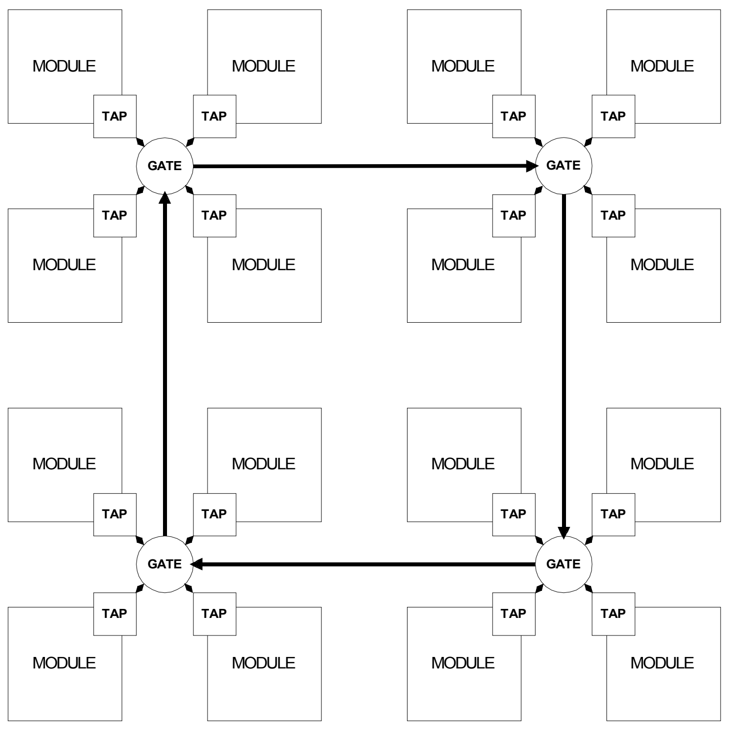

Figure 1.3: Ring Topology

1.4 Bus Interface Logic

The A2R protocols define the method by which bus transactions are performed across the bus/buses. In order to manage these protocols every device requires a bus interface module or TAP.

There are two types of TAPs. Master TAPs can create transactions over the bus for read or write data, these are bus master TAPs. Slave TAPs respond to transactions over the bus/buses by performing the read or write access as requested and returning data or status back to the master.

Some devices attached to the bus will require both an Master and a Slave TAPs. Figure 2 illustrates that Masters that have both Slave and Master TAPs. The Slave interface is so that another Master can set up a Master's registers so that it may carry out transactions autonomously by writing configuration information to I/O registers. The I/O master then transfers data between its I/O interface and system memory through its Master interface.

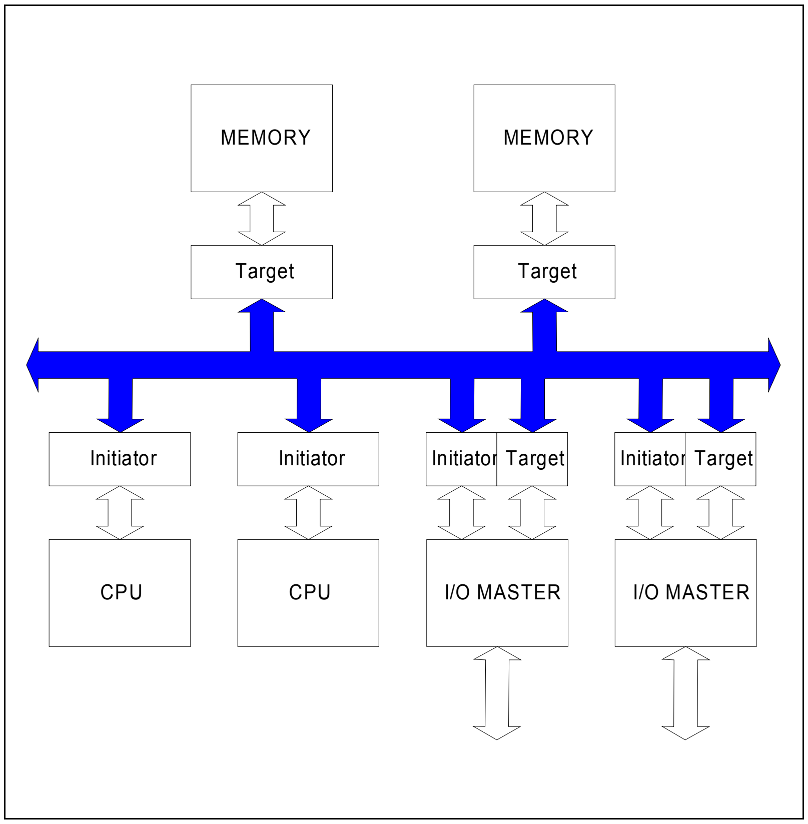

Figure 1.4: Parallel configuration showing Master and Slave Taps

1.4.1 Bus Interconnections

The A2R protocols allow the use of parallel or serial interconnection between modules within a single bus segment. Parallel configurations simply OR together the outputs from each of the slaves and masters and the output of the OR gates drive all the corresponding master and slave inputs.

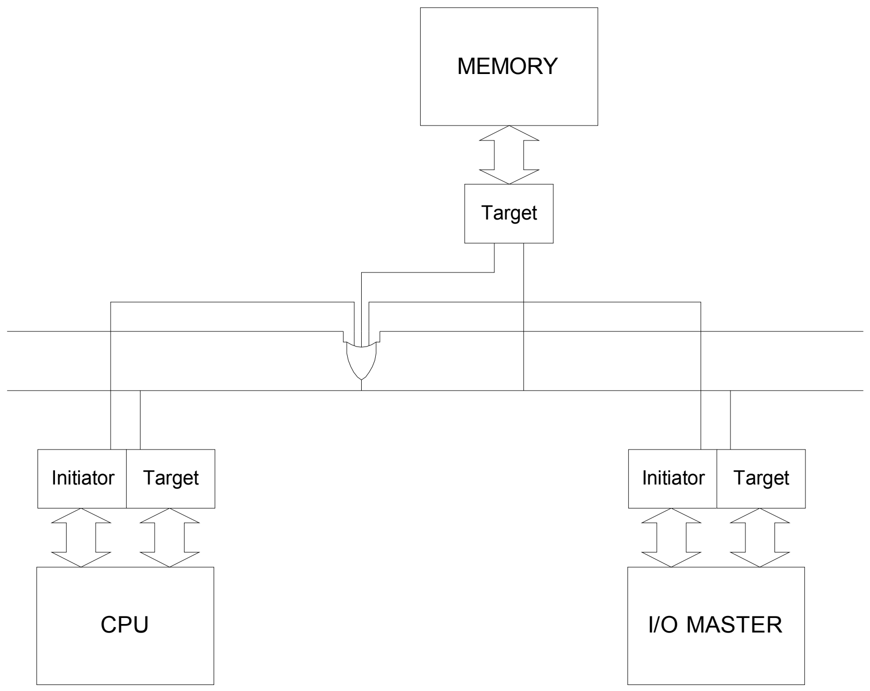

Figure 1.5 Parallel – Central OR gate topology

An alternative to a central OR gate topology is to use a pseudo-three-state topology. In this mode signals are sent on two directions from each TAP and are OR-ed together at each receiver. This has the effect of having a true bi-directional three-state bus using only uni-directional signals. Its advantage is that each section is re-driven at every TAP so that signal runs are typically much shorter and there is no need for a large central driver. The wiring is similar as there are two wires per bit, one for each direction as opposed to a wire to the OR gates and one from the OR gates.

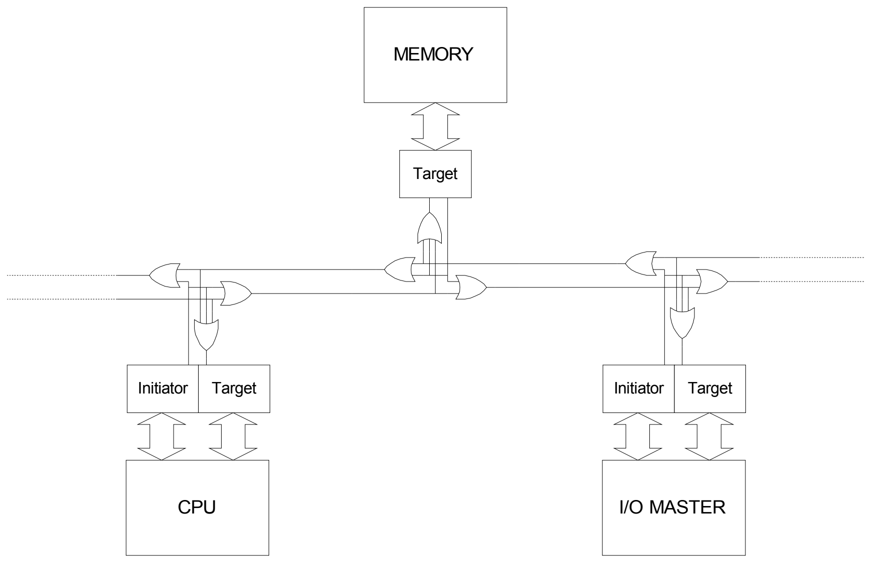

Figure 1.6 Pseudo three-state topology

Serial topologies daisy chain the outputs from one stage to the inputs of the next and so form a ring. The ring can be configured with pipeline stages in any number of stages (masters or slaves). If no registers are configured then an asynchronous loop is created; correct operation is still guaranteed but simulation and synthesis tools will usually issue warnings.

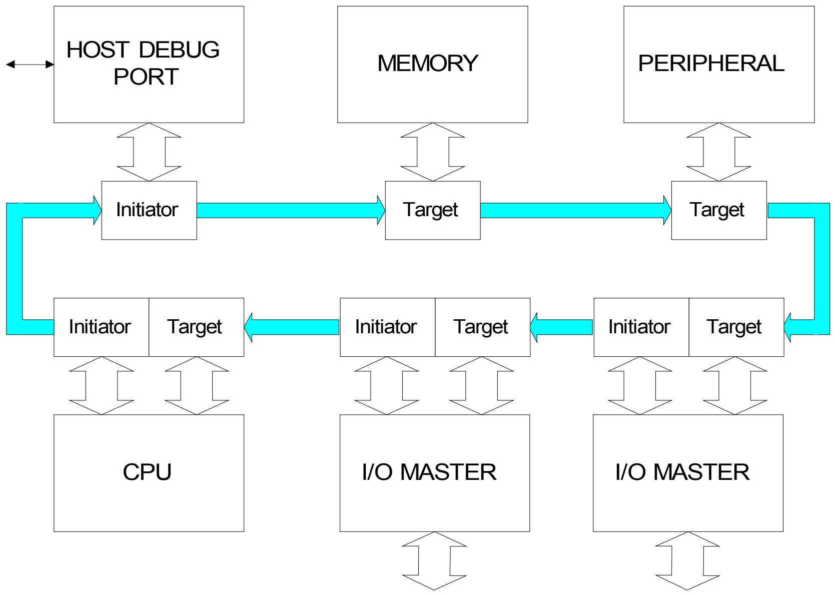

Figure 1.7: Serial Configuration

1.5 Arbitration

The A2R modifies its arbitration scheme depending on the topology used for the interconnection fabric. If a parallel scheme is used then all A2R

TAPs issue requests that go to all other TAPs. Arbitration is then performed locally at each TAP and a TAP to grant decided upon. If a serial scheme is used then a token is passed around the ring.

1.5.1 Parallel Arbitration

The standard arbitration scheme for a parallel topology is 'round robin'. The mechanism is provided as a function in the A2R parameters section and can be modified or replaced at the architect's discretion. From reset, after 15 clocks to ensure quiescence, the A2R designated as TAP-0 is given the grant. The first request on the bus will then take over the grant if multiple requests occur concurrently then they will be resolved in numerical order starting at the TAP after the one holding grant. The numerical order is modulo the number of TAPs so wrapping around from the top back to zero.

When a TAP is granted the bus the former grant holder releases grant and the new bus owner will not reassert a grant until a cycle after it has issued an address phase. This allows for the data phase of a transaction to be in parallel with the arbitration of a subsequent transaction.

1.5.2 Serial Arbitration

Serial arbitration starts, following reset, in the same manner as for parallel arbitration. The grant is passed around the ring as a token that is captured by the first TAP encountered with an active request. Again the grant is not reinserted into the ring until a clock cycle following an address phase. The grant traverses the ring with the same timing as the rest of the signals on the bus. When a register is encountered all signals are clocked through into the next cycle thus if the topology of the ring has 3 registers then the grant will circulate the ring every 3 cycles if no requests are active.

1.5.3 Gateways

Gateways provide a means of creating a hierarchy of A2R bus segments. They assert their own requirements on the arbitration scheme. The top-most segment always has the highest priority in a system such that an active transaction on the top-level, when it arrives at a gateway and has to arbitrate for access to the lower level will force itself on the lower level. This transaction will assert a request into the lower level and wait for a grant by the normal means. In the meantime if a lower level request tries to go **up** the gateway then that access will be aborted by the gateway asserting "retry". This will cause the requesting device to remove its request and re-arbitrate. Doing so, because of the round-robin scheme, forces the access to retry after the higher level access proceeds. Retry is not seen by the master, the A2R tap performs all the necessary actions.

1.6 Latency and Performance

One of the principal rationales behind the design of the A2R protocols was the desire for simple, straightforward attachment of the bus mechanism to registers and memories in the command and control structure of an ASIC. High performance is NOT an issue, simple and efficient connectivity is.

There are various fabric choices for A2R so that the optimal mechanism is used in each application of A2R. Within blocks where there is a need for multiple A2R taps simple multiplexing is usually optimal. Where there are multiple blocks that hook together at a top-level then a pseudo tri-state or ring topology are more usually optimal. Often a hierarchy should be built so that there is a top-level that interconnects the major system blocks and within each major block there are A2R branches that may use alternative topologies. The gateway mechanism provides a means of bridging between the hierarchical levels.

2 Transactions

2.1 Transaction Phases

Each transaction over the A2R is divisible into a number of phases. Some phases can be overlapped with other phases to achieve the maximum utilization of the available bus bandwidth.

2.1.1 Arbitration

All masters will arbitrate for ownership of the bus in any cycle. Depending on the topology of the bus and the number of pipeline stages configured in to the system this phase may last several clock cycles. When a request is granted all system grants are removed until after the address phase. The bus owner, having been granted the bus, is responsible for re-asserting the grant during the data phase.

2.1.2 Address

During the address phase the owner of the bus will drive the address, command signals. The address phase lasts only a single cycle. The bus may be pipelined and so the address phase will propagate sequentially through the pipeline as a single event. In a ring topology, when the address phase returns to the initiating TAP the address will be removed from the ring.

2.1.3 Data

The data phase is used to drive data onto the bus and is timed to follow in the cycle after the address phase. The command signals are held active to the same levels as those asserted in the address phase for the duration of the data phase. For a write command the write data is asserted by the master until it sees a returning ready from the accessed slave. For a read command the read data is asserted by the slave along with the ready signal and lasts only for a single cycle.

2.2 Alignment

All write data presented on to the bus will be in their relative position and no re-alignment is performed. If a byte has an address L.S.Bs. = 3 it will be transferred on byte lane 3. There are no exceptions. If an Master only supplies data aligned to the least significant end of a word then the TAP interface logic must pre-align the data for the bus.

All read data presented on to the bus will be in their relative position and no re-alignment is performed. If a byte has an address L.S.Bs. = 3 it will be transferred on byte lane 3. There are no exceptions. If an Master requires that data be aligned to the least significant end of a word then the TAP interface logic must perform this function.

2.2.1 Endian-ness

The A2R protocols make no assumptions as to the endian-ness of the data and the alignment and addressing of the bytes within a field. The masters and slaves of a single system must agree on the interpretation of the addresses and the alignment of the bytes within a multi-byte data field.

2.3 Bus Transactions

2.3.1 Write Transactions

Write Transactions begin by the master asserting a master_write signal together with address and write data to the A2R tap. If this cycle is enabled for arbitration, that is, there is a grant signal incoming to the TAP, the master TAP will check against the other requests and grants on the bus to determine the winner. If the master wins then it will initiate an address phase (aphi) and asserting address and write command onto the bus. A cycle later it will remove the address and assert the write data. The master will continue to drive these signals until is receives a ready signal from the accessed slave. After a single cycle of driving the write data the master TAP will assert grant to alow the next transaction to be arbitrated for. This may be accompanied by an error and / or retry signal. If there is an error signal it is reported back to the master and the access terminates. If there is no error but retry is asserted then the master TAP will re-arbitrate for the bus and try again.

Figure 2.1 shows a typical write transaction. The diagram assumes only a single pipeline stage for the bus which reflects all parallel topologies and a zero pipeline ring. Caution must be used for zero-pipeline rings as these cause a giant asynchronous latch to be instantiated. While the logic in the system can handle this situation many tools will complain! If there are pipeline stages in the bus topology the a2r_xxx signals will be delayed by an appropriate number of clocks (the number of stages between the master and the slave for address, command and write data and the number of stages between slave and master around the other side of the ring for read data, ready, error and retry.

In the diagram the solid black upwards arrows represent where the pipeline stages are inserted for the A2R TAP configuration. The diagram is shown with all pipestages included. When a pipestage is excluded the transition will occur within the same cycle. So the timing connector arrows as shown will transition in the same cycle. There are 5 selectable pipestages in the A2R tap creating 32 combinations of timing!

In general a design should be tried with all pipestages included and then removed one by one to see if timing is maintained. If timing fails replace that stage and try the rest.

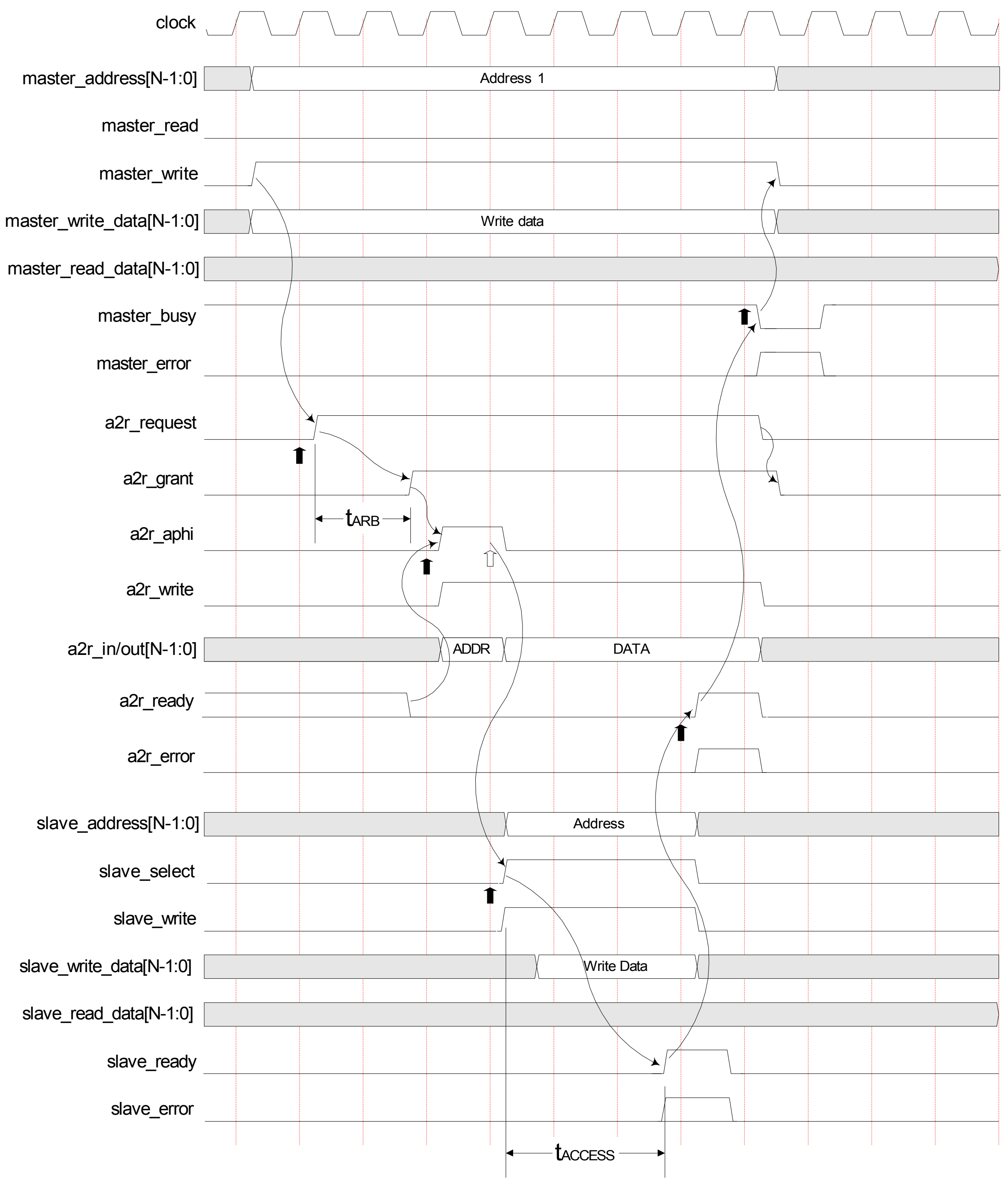

Figure 2.8: Write Transaction

2.3.2 Read Transactions

Read Transactions begin by the master asserting a master_read signal together with the address to the A2R tap. If this cycle is enabled for arbitration, that is, there is a grant signal incoming to the TAP, the master TAP will check against the other requests and grants on the bus to determine the winner. If the master wins then it will initiate an address phase (aphi) and asserting address and read command onto the bus. A cycle later it will remove the address. The master will wait until is receives a ready signal from the accessed slave. After a single cycle of waiting the master TAP will assert grant to allow the next transaction to be arbitrated for. The returning ready signal may be accompanied by an error and / or retry signal. If there is an error signal it is reported back to the master and the access terminates. If there is no error but retry is asserted then the master TAP will re-arbitrate for the bus and try again.

Figure 2.1 shows a typical read transaction. The diagram assumes only a single pipeline stage for the bus which reflects all parallel topologies and a zero pipeline ring. Caution must be used for zero-pipeline rings as these cause a giant asynchronous latch to be instantiated. While the logic in the system can handle this situation many tools will complain! If there are pipeline stages in the bus topology the a2r_xxx signals will be delayed by an appropriate number of clocks (the number of stages between the master and the slave for address, command and write data and the number of stages between slave and master around the other side of the ring for read data, ready, error and retry.

In the diagram the solid black upwards arrows represent where the pipeline stages are inserted for the A2R TAP configuration. The diagram is shown with all pipestages included. When a pipestage is excluded the transition will occur within the same cycle. So the timing connector arrows as shown will transition in the same cycle. There are 5 selectable pipestages in the A2R tap creating 32 combinations of timing!

In general a design should be tried with all pipestages included and then removed one by one to see if timing is maintained. If timing fails replace that stage and try the rest.

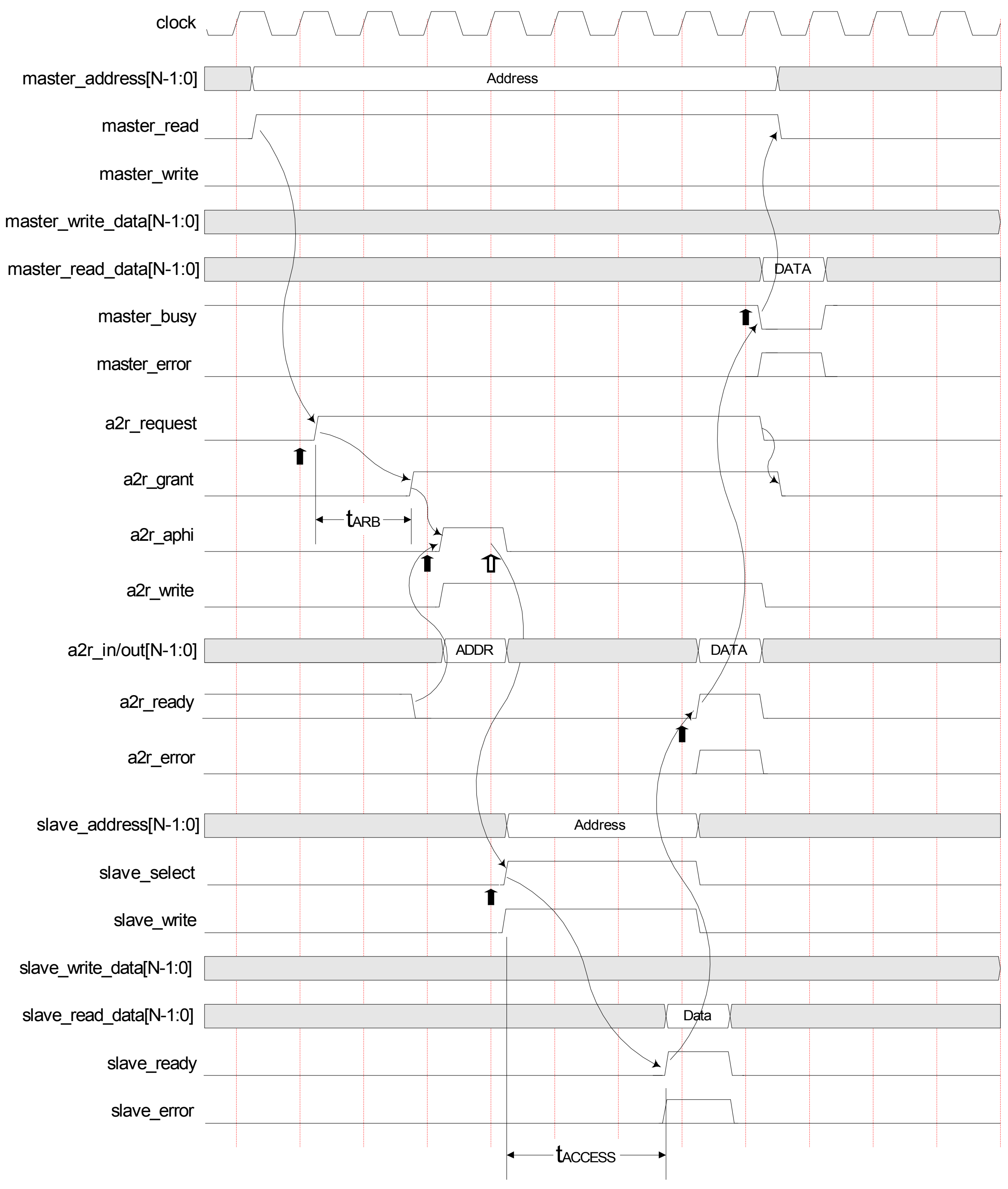

Figure 2.9: Read Transaction

2.4 Pipelined Transactions

If the bus is configured as a ring then the ring may be pipelined to simplify the timing at the system level. Pipelining places registers at strategic points around the ring. Each TAP may be configured as a pipe stage. In this mode the TAP rather than simply re-driving the signals will output the signals from its capture register instead.   This number of pipestages affects the latency and bandwidth of accesses on the ring but does simplify timing.

It is recommended that all TAP initially be made pipestages and the system tested. The system should then also be tested with just one TAP set as a pipestage. This should be done with at least two different TAPs set as the pipeline. After initial place and route the real number of pipestages can be determined and set appropriately. Rarely does the positions of the pipestages need to be changed late in the physical design process but the pipe-enable signals can be taken out to the top level where they may easily be forced high or low to effect a pipelining change.

Figure 2.3 illustrates a single pipestage ring. If more stages are present the multiple cycle delays may be seen between a2r_out and a2r_in depending on where the master and slave of each transaction physically lie relative to each other. The total delay will, of course, be constant for any all transactions.

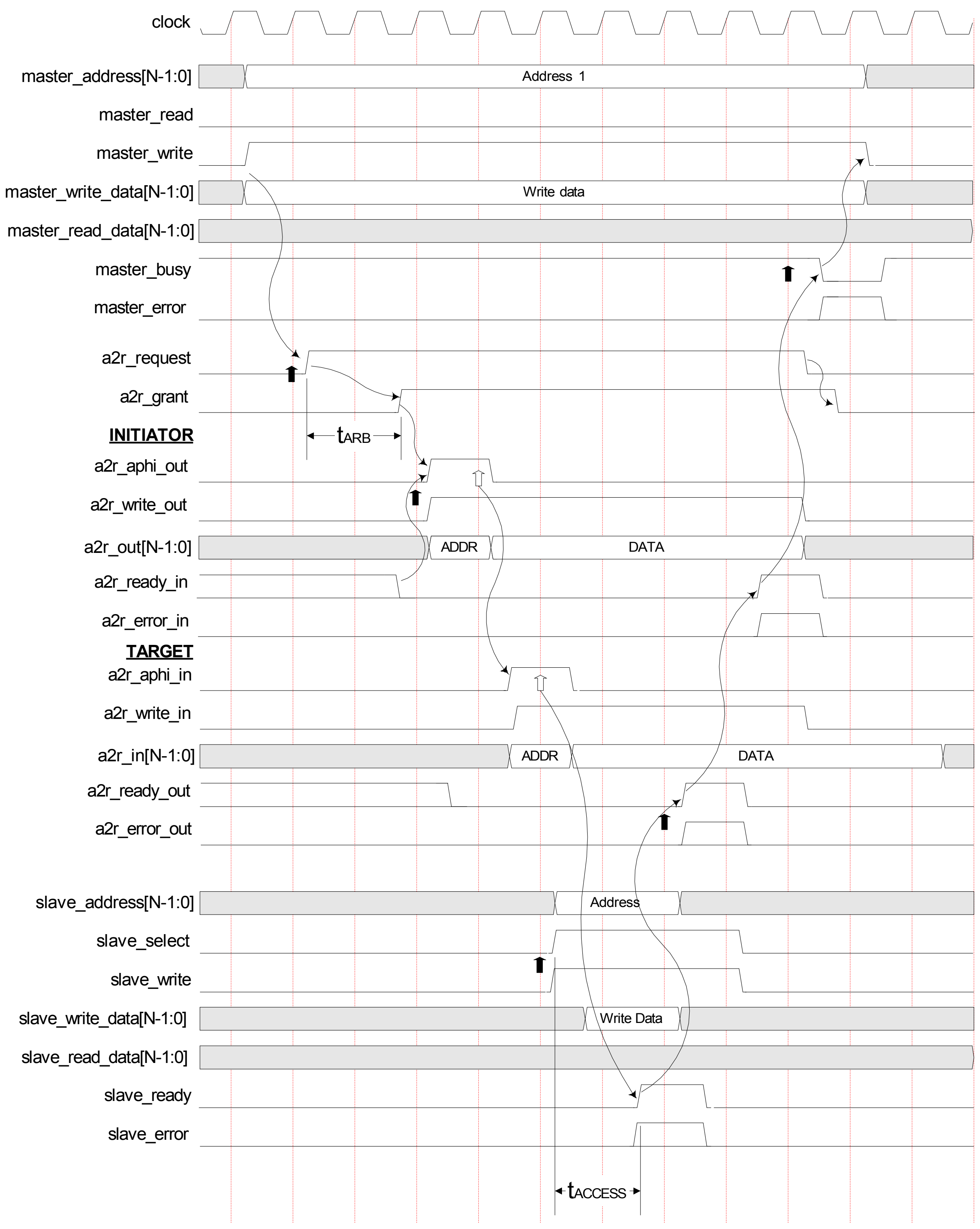

Figure 2.10: Pipelined transaction

3 Signals

This section provides a description of the actual bus signals. These are given in order to better understand the overall operation of the bus.

The user (designer of a module attaching to the A2R) will design to the master and slave interface signals and their respective timings rather than those described here.

Table 3-1: A2R Bus Signals

BUS SIGNAL	SIZE	DESCRIPTION
a2r_in	width	Multiplexed address and data bus
a2r_aphi_in	1	Address phase indicator
a2r_write_in	1	Write command
a2r_ready_in	1	Ready response from slave
a2r_error_in	1	Error response from slave
a2r_grant_in	num_requestor s	Arbitration grant
a2r_request_in	num_requestor s	Arbitration request
a2r_retry_in	1	EXTENSION Retry response for hierarchies
a2r_out	width	Multiplexed address and data bus
a2r_aphi_out	1	Address phase indicator
a2r_write_out	1	Write command
a2r_ready_out	1	Ready response from slave
a2r_error_out	1	Error response from slave
a2r_grant_out	1	Arbitration grant
a2r_request_out	1	Arbitration request
a2r_retry_out	1	EXTENSION Retry response for hierarchies
clock	1	
reset	1	

3.1 Bus Signal Descriptions

The signals described here apply to either the signal input or output and so each signal is notated as in/out. In the actual bus there are separate signals for in and out for each A2R TAP.

3.1.1 a2r_in/out: Multiplexed Address and Data

The address/data field has a parameterized width (typically 32 or 64 bits). The addresses point to object that are the width of the bus. So if the bus is 8-bits only 256 bytes are addressable. If the bus is 16-bits then 65536 16-bit words are addressable. For a 32-bit bus 2^{32} 4 byte data are addressable.

3.1.2 a2r_aphi_in/out: Address Phase

The address phase signal indicates that the transaction address is currently on the bus.

3.1.3 a2r_write_in/out: Write

The write signal indicates the type of command that the access should perform. If the signal is asserted high then a write is performed otherwise a read is performed.

3.1.4 a2r_ready_in/out: Ready

This signal indicates that the addressed slave has completed its access and if the command is a read that the read data is now available on the bus.

3.1.5 a2r_error_in/out: Error

This signal is sent in concert with the ready signal and indicates that an error occurred during the access at the addressed slave.

3.1.6 a2r_retry_in/out: Retry

This signal is a configurable extension to the base protocol. If the A2R is configured with a hierarchy of bus segments then a mechanism is required to break deadlock situations where two accesses attempt to pass through a gateway from different directions at the same time. The access from the lower level attempting to access a higher level must back-off and try again later. To signal this to an master the retry signal is asserted by the gateway. Note that only TAPs on the lower levels need to be capable of retry. The retry mechanism is independent of the master and slave interfaces and operate totally within the A2R protocol.

All an master will see is an extended access time.

3.1.7 a2r_grant_in/out: Grant

For a parallel arbitration topology there are as many grant signals as there are requestors for each bus segment. This allows each TAP to monitor the arbitration of the other TAPs and establish a local arbitration result. In a serial topology there is only one grant signal in which is the incoming token.

Each TAP outputs one and only one grant out for any topology and the signal represents that this TAP was the last device to be granted the bus.

3.1.8 a2R_request_in/out: Request

As for the grant signals there are many request signals as there are requestors for each bus segment for parallel arbitration topologies. These signals individually identify the request for a TAP. Each TAP receives all the request inputs and uses them together with the grant signals to determine the arbitration winner.

In a serial arbitration topology the request signals are not used outside the individual TAPs and should not be connected.

4 Master Interface

SIGNAL	SIZE	I/O	DESCRIPTION
master_address	width	In	Address of a ‘width’ sized datum
master_write	1	In	Write request/command
master_read	1	Out	Read request/command
master_write_data	width	In	Data to be written to the slave
master_read_data	width	Out	Data read from the slave
master_busy	1	Out	Busy indicator – the transaction is in progress
master_error	1	Out	Error

Table 4-2: Master Interface Signals

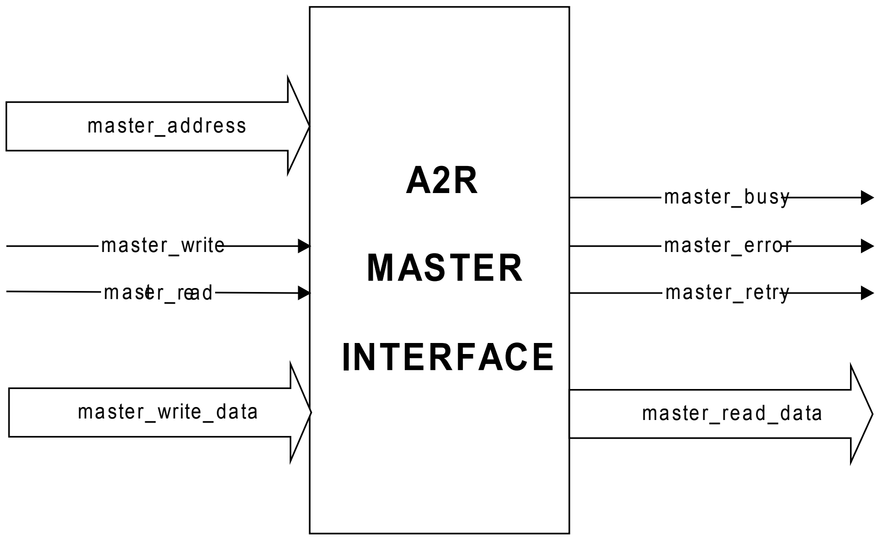

Figure 4.11: Master Interface Diagram

4.1 Write Transactions

For a write operation an master simply asserts the address, write data and the master_write signal to the interface and waits for the master_busy signal to go non-active. This signifies that the slave has finished the access. If at the same time as the busy goes non-active the master_error signal is true then an error occurred at the slave side and the access may not have written the data.

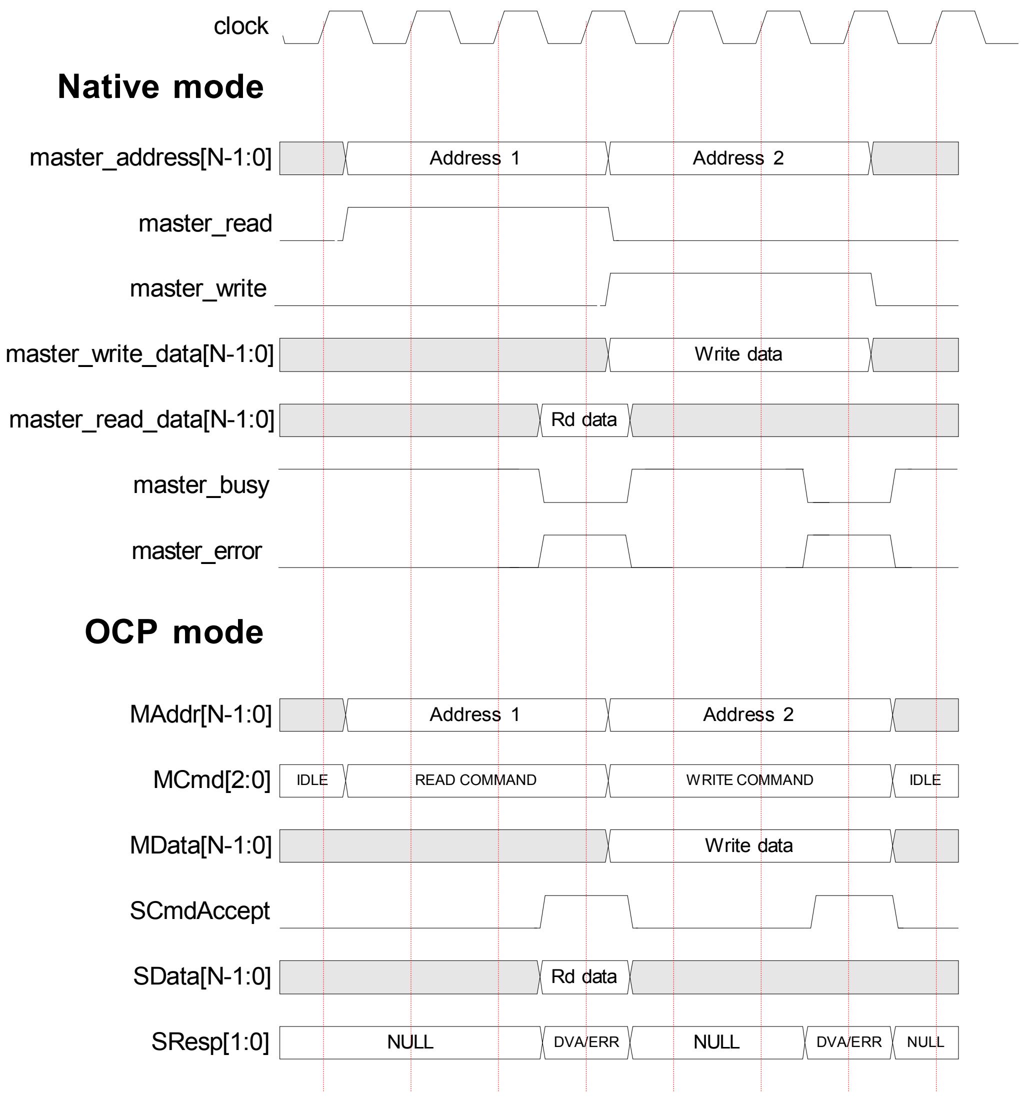

Figure 4.12: Master Write

4.1.1 Read

For a read operation an master asserts the address and the master_read signal to the interface and waits for the master_busy signal to go non-active. This signifies that the slave has finished the access and that the read data is valid. If at the same time as the busy goes non-active the master_error signal is true then an error occurred at the slave side and the access may not have written the data.

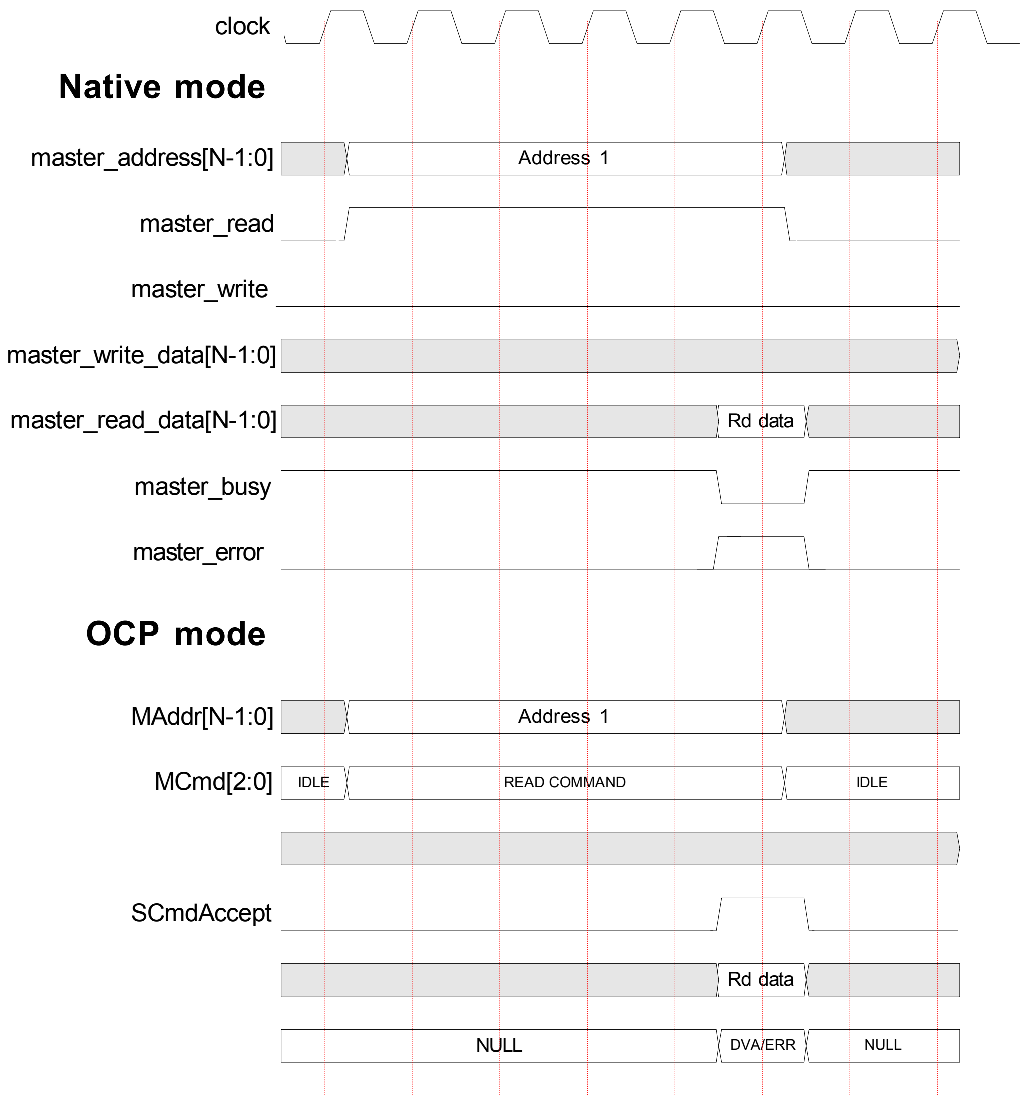

Figure 4.13: Master Read

4.1.2 Back-to-back accesses

The master interface is capable of performing back-to-back accesses as shown below:

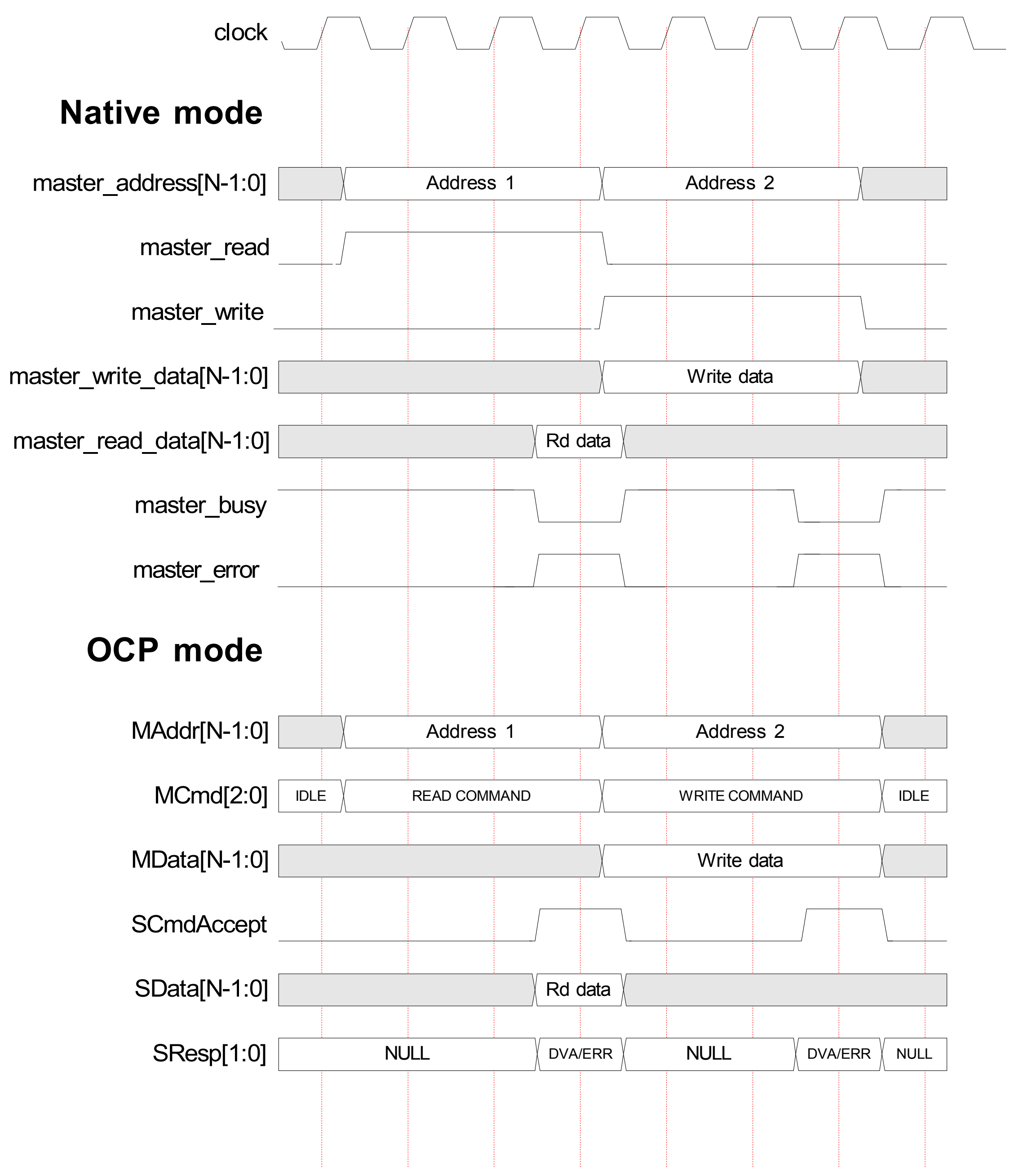

Figure 4.14: Back-to-back transactions

5 Slave Interface

SIGNAL	SIZE	I/O	DESCRIPTION
slave_address	width	Out	Address of a ‘width’ sized datum
slave_select	1	Out	Write request/command
slave_write	1	Out	Read request/command
slave_write_data	width	Out	Data to be written to the slave
slave_read_data	width	In	Data read from the slave
slave_ready	1	In	Busy indicator – the transaction is in progress
slave_error	1	In	Error indicator
slave_retry	1	In	Retry indicator

Table 5-3: Slave Interface Signals

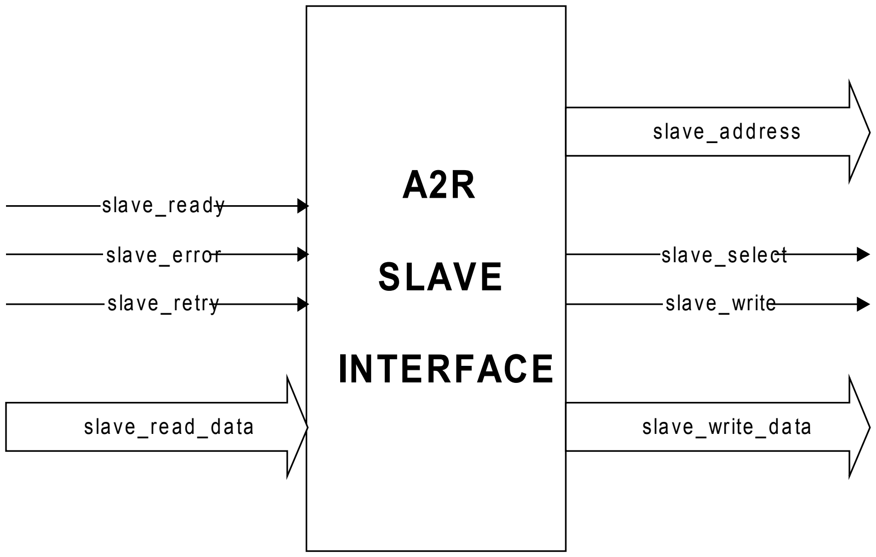

Figure 5.15: A2R Slave Interface Diagram

5.1 Write

For a write operation a slave TAP asserts the address, write data, slave_write and the slave_select signals to the interface and waits for the slave_ready signal to go active. This signifies that the slave has finished the access. If at the same time as the busy goes non-active the slave_error signal is true then an error occurred at the slave side and the access may not have written the data.

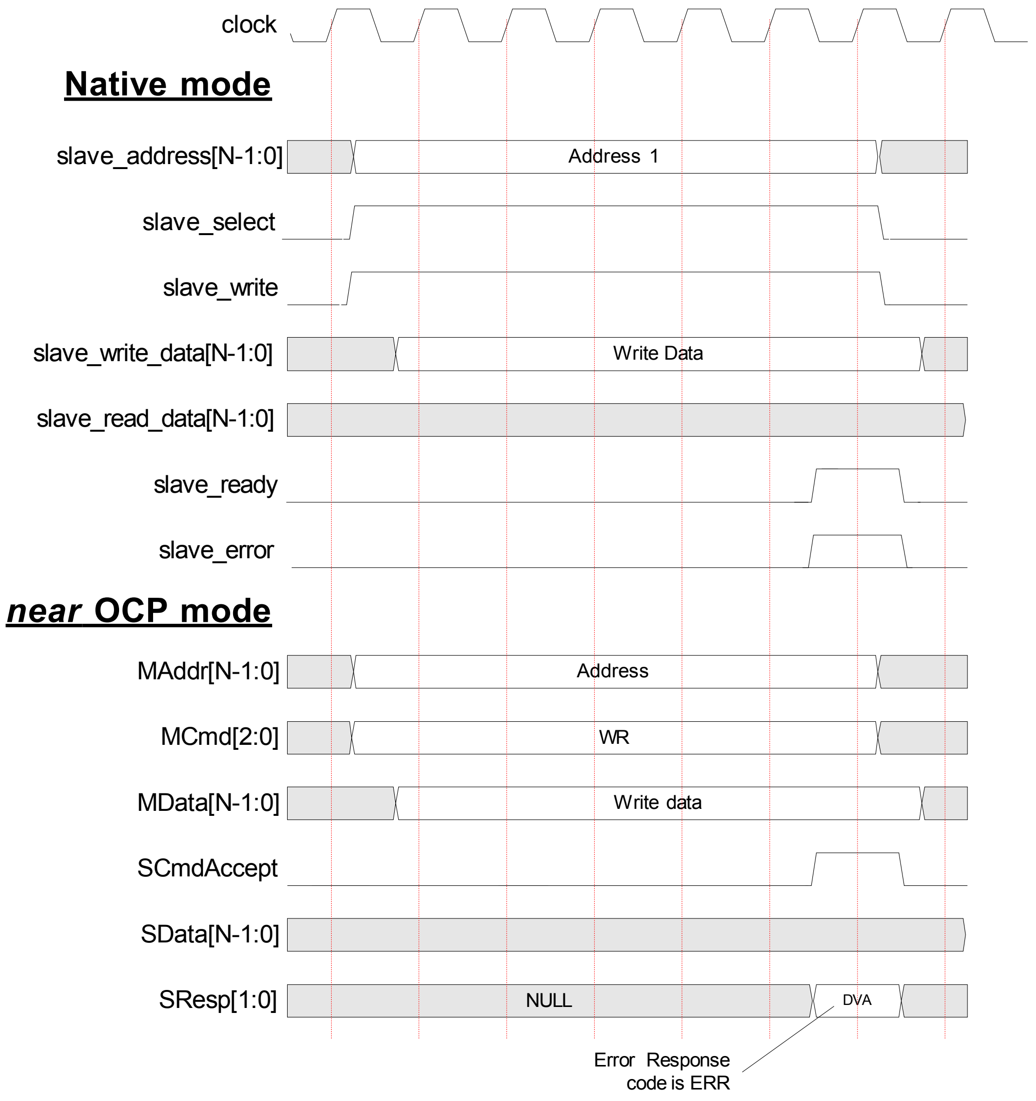

Figure 5.16: Slave Write

5.2 Read

For a read operation a slave TAP asserts the address, write data and the slave_select signals to the interface and waits for the slave_ready signal to go active. This signifies that the slave has finished the access and that the read data is present on the slave_read_data lines. If at the same time as the busy goes non-active the slave_error signal is true then an error occurred at the slave side and the access may not have written the data.

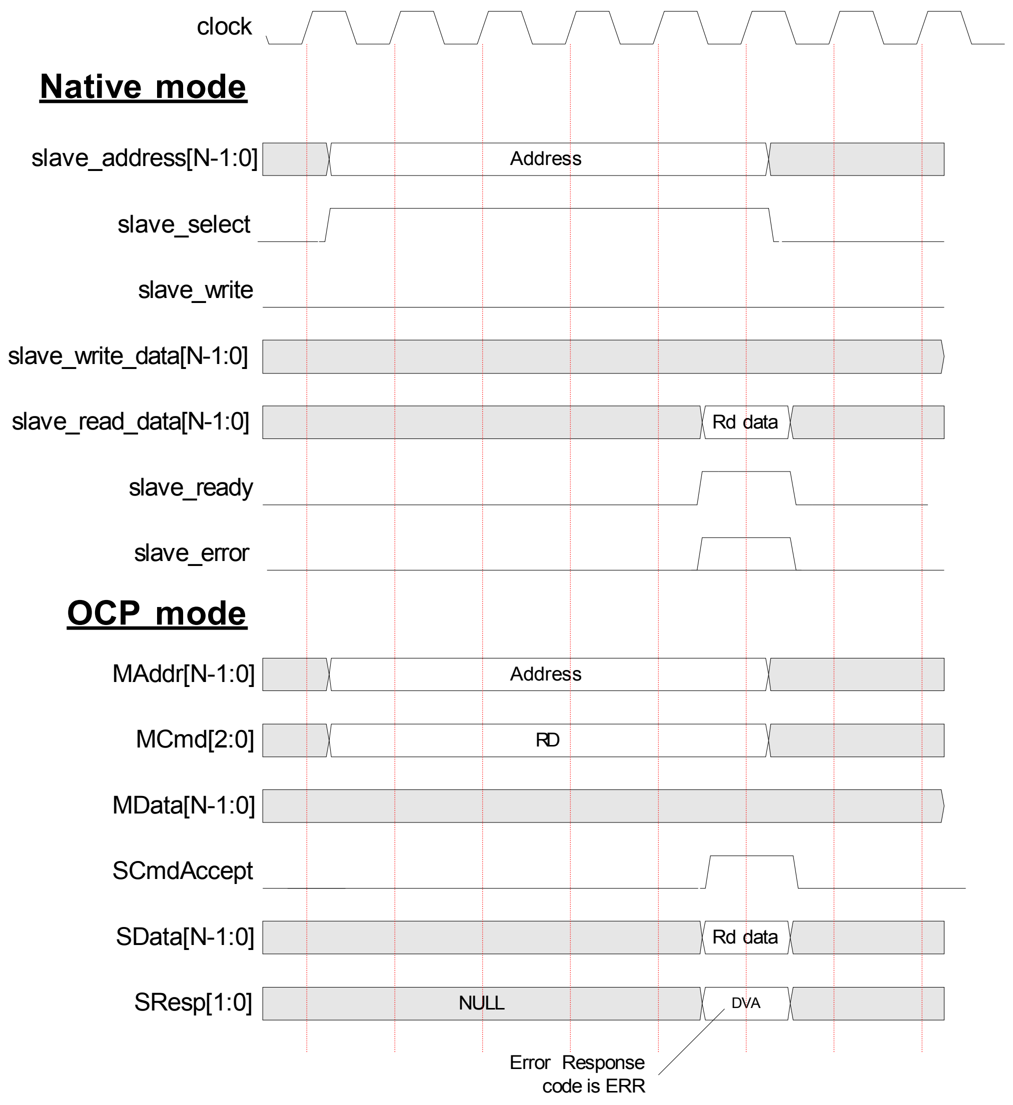

Figure 5.17: Slave Read

5.3 Back-to-back accesses

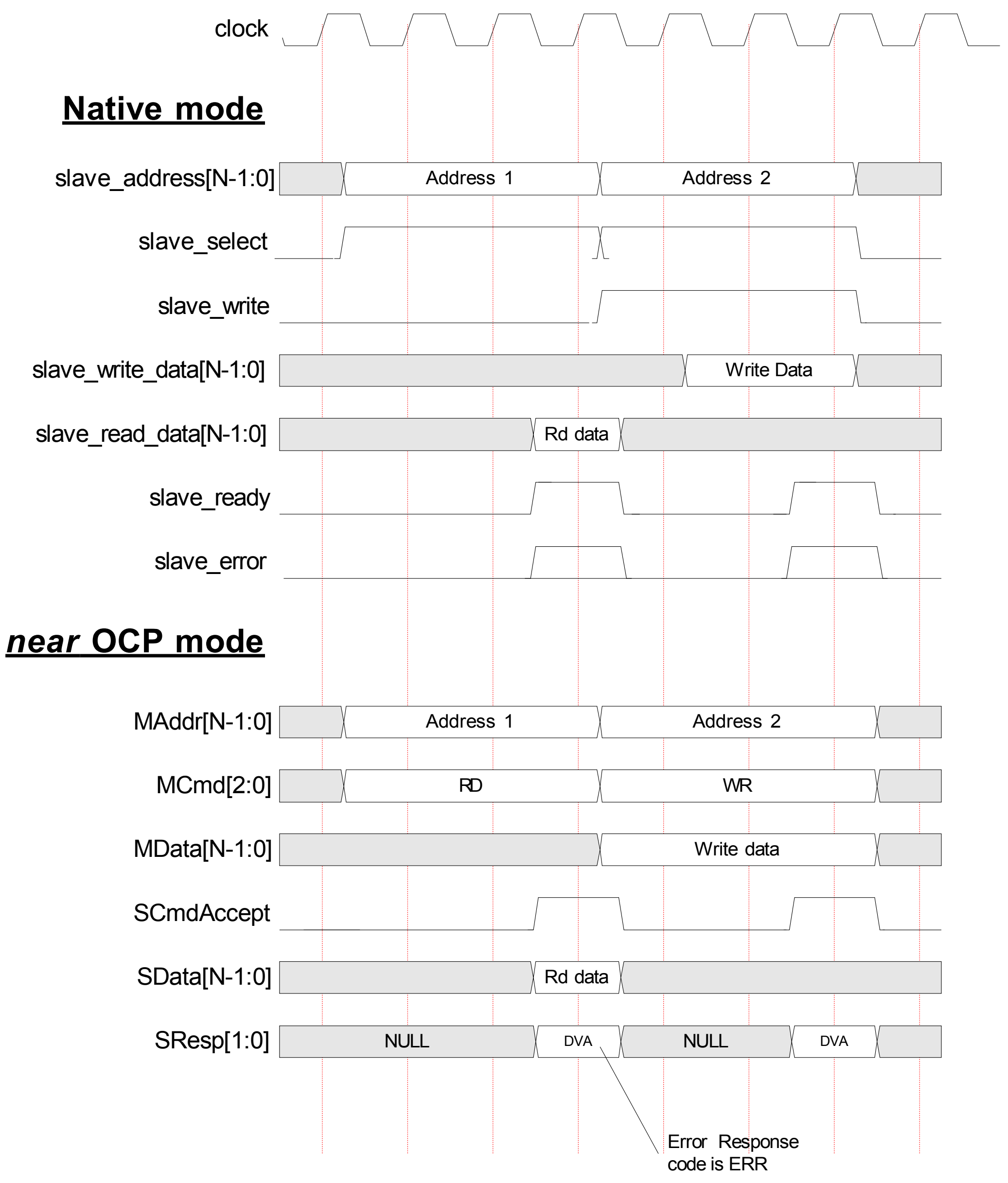

Figure 5.18 Slave Back-to-back accesses

5.4 Full OCP Compliance Writes

The delayed write data due to the multiplexing arrangement of the A2R prevents true OCP compliance. Full compliance can be achieved with the addition of the following circuit which creates the following timing.

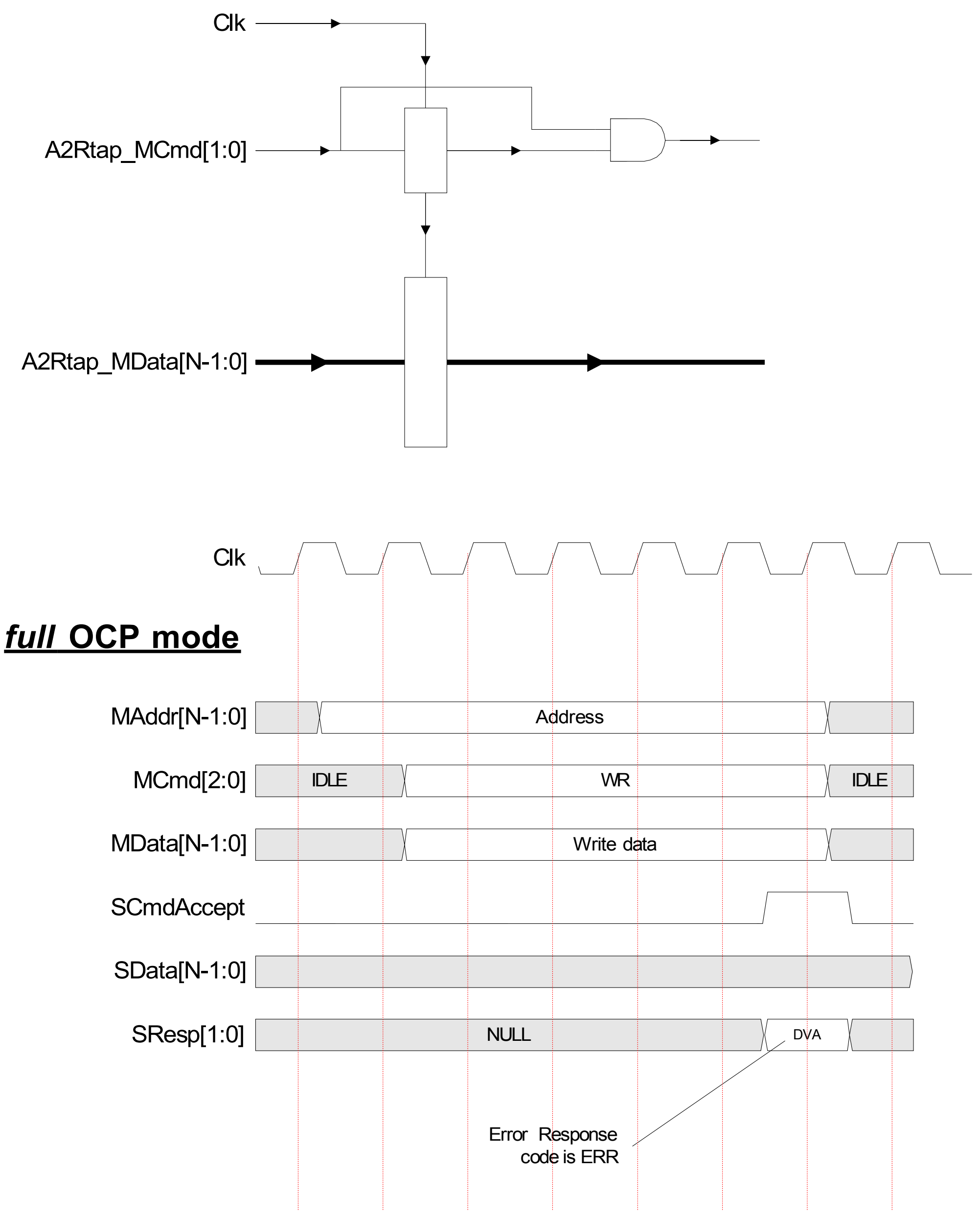

Figure 5.19: Full OCP writes

6 Configuration

Configuration of the A2R is static and controlled through a parameter file. The A2R is not intended for dynamic systems that can have unknown devices attached to them like a PCI bus.

The static configuration of the bus is controlled through a parameter file that controls the following features:

- Bus Width
    - 8, 16, 32 or 64 bits recommended
    - parity may be added but must be at the least significant end so as to not upset address decoding
- Number of Masters
- Number of Slaves
- Pipeline Stages on a ring
    - Set by a signal to each TAP. Should be set after physical layout on the top layer of a design.
- Pipeline Stages within the TAPs
    - Set by parameters to each TAP. The timing diagrams illustrate where the registers are placed. If a low clock rate is used zero pipelining is recommended.

6.1 Identity

Every module attached to the bus is given an identity to distinguish it uniquely on the bus from other modules. Certain modules will have Slave and Master interfaces, usually intelligent I/O devices and bridges.

Each Slave and Master is given a unique identifier that is used by the Addressing scheme. The unique code is hardwired to each TAP.

Each TAP receives a hardwired binary value that uniquely identifies it to the rest of the system.

6.2 Address Mapping

The mapping of address space to slaves is static. The A2R parameter file provides an A2R_num_slaves value that determines the maximum number of slaves in the entire system, which is the sum of slaves in all segments. This number then determines the number upper address bits required to address each slave. The remaining address bits are available for addressing within each slave. Note that each slave is given en equal amount of internal address space. For example, in a 32-bit wide system, if 17 slaves are configured then 5 uppermost address bits are used to address the slaves and each slave has 27 bits are available to each slave.

6.3 Topology

The A2R_OR_fabric for each bus section allows each section to be set as a parallel or serial topology. If set then a parallel fabric is used and the interconnect can use either the A2B_OR_fabric or A2_pseudo_tristate as the modules that interconnect the TAPs. If the A2B_OR_fabric is unset then the topology is assumed to be a ring and the TAPs will connect directly in daisy-chain fashion.

The arbitration scheme will change automatically based on the A2B_OR_fabric also.

7 Bridges & Multi-segment Buses

The A2R easily supports bridges from one A2R bus to another   The A2R protocol is designed to support the direct Transaction of signals from segment to segment without any major translation hardware. The retry signal must be included (by a `define) to avoid deadlock situations. The standard A2R enforces that all A2R segments be of equal width.

Gateways can also be used to split a bus into separate segments to gain even higher bandwidth.

7.1 Gateways

The A2R protocols are not affected by connecting buses together. Gateways are, in fact, simply cross connected TAPs between to different A2R sections. The only addition is simple logic to detect that both sides are requesting at the same time and back-off the lower level bus by asserting retry.

Addressing is handled in the TAPs by designating the TAP connected as a gateway through a parameter. This causes the address decode for this TAP to be subtractive and will pass all unused addresses through to the other side. To do this the TAP detects the number of local slaves and the number of address bits required to address them. It will then inspect its upper bits and if they are no the same as its own will pass the access through the gateway.