MIPS32® microAptiv™ UP Processor Core AHB-Lite Interface

Document Number: MD01082
Revision 01.00
July 30, 2014

Introduction

This document describes the AHB-Lite™ interface, which is present on the MIPS32® microAptiv™ UP processor core. The AHB-Lite interface is described in the *AMBA 3 AHB-Lite Protocol Specification*.

This document contains the following major sections:

- Section 1, "Interface Transactions"
- Section 2, "Clock Ratios"
- Section 3, "Write Buffer"
- Section 4, "Merging Control"

1 Interface Transactions

The microAptiv UP implements 32-bit unidirectional address and data buses: *HADDR[31:0]* for address, *HRData[31:0]* for read operations, and *HWData[31:0]* for write operations. All single timings are related to the rising edge of *HCLK*. The *HCLK* is a reference clock from the gated main clock *SI_ClkIn*, i.e, *HCLK* will be gated off via execution of the WAIT instruction. *HCLK* is also a primary output of the microAptiv UP.

1.1 Basic Transfers

An AHB-Lite basic read and write transfer consists of two phases:

- *Address*: The Address phase lasts for a single *HCLK* cycle unless it is extended by the previous bus transfer.
- *Data*: The Data phase might require several *HCLK* cycles. It uses the *HREADY* signal to control the number of clock cycles required to complete the transfer.

The simplest transfer is one with no wait states, so the transfer consists of one address cycle and one data cycle. Figure 1 shows a simple read transfer, and Figure 2 shows a simple write transfer.

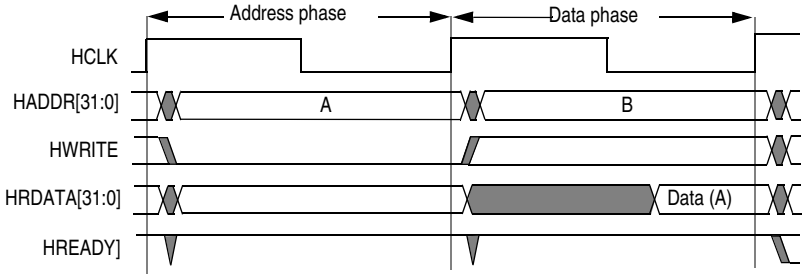

Figure 1 Read Transfer with no Wait States

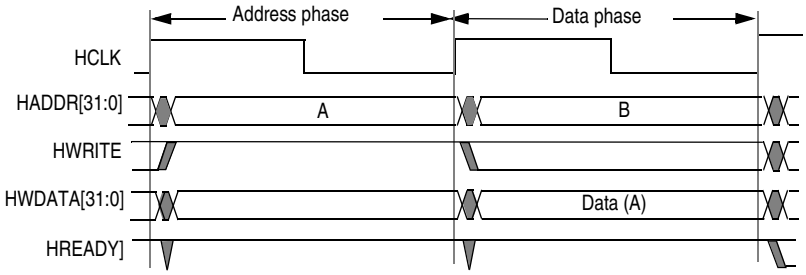

Figure 2 Write Transfer with no Wait States

A slave device can insert wait states into any transfer to enable additional time for completion. Figure 3 shows a read transfer with two wait states.

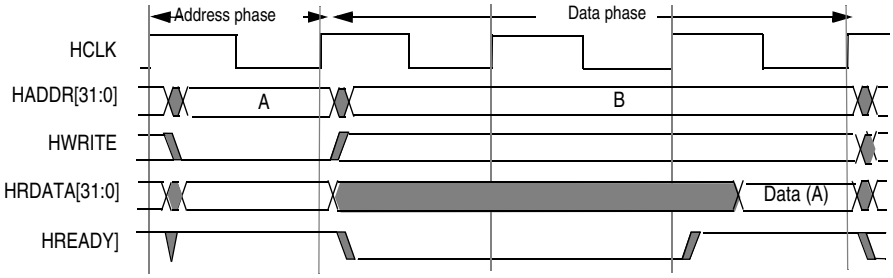

Figure 3 Read Transfer with Two Wait States

Figure 4 shows a write transfer with one wait state.

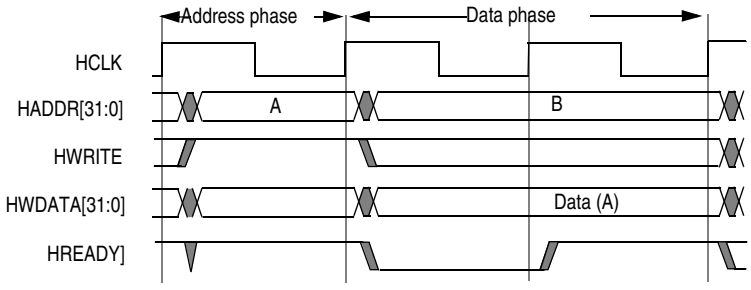

Figure 4 Write Transfer with One Wait State

The address phase of any transfer occurs during the data phase of the previous transfer. This overlapping of address and data is fundamental to the pipelined nature of the bus. When a transfer is extended, it has the side-effect of extending the address phase of the next transfer. For write operations, the master holds *HWDATA* stable throughout

the extended cycles. For read transfers, the slave does not have to provide valid data on *HRDATA* until the transfer is about to complete. Figure 5 shows three transfers to unrelated address, A, B and C with an extended address phase for address C.

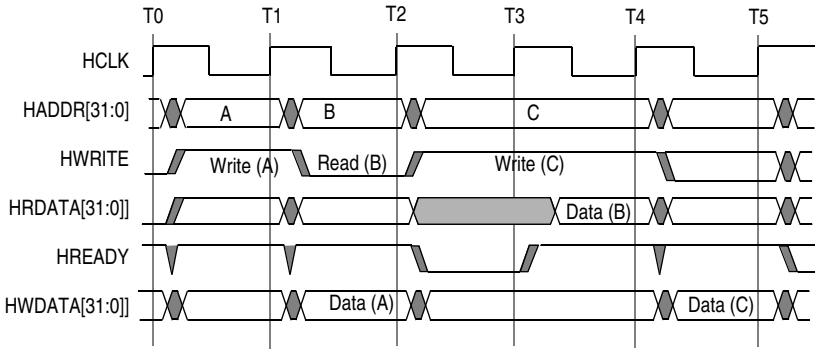

Figure 5 Multiple Transfers

1.2 Transfer Types

The microAptiv UP core implements three of the four types of transfer defined in the AHB-Lite Protocol, namely, IDLE, NONSEQ, and SEQ. The type of transfer is signified by the *HTRANS[1:0]* signal, as shown in Table 1.

Table 1 Transfer Types

Type	HTRANS[1:0]	Description
IDLE	2'b 00	Indicates that no data transfer is required
BUSY	2'b 01	Not supported
NONSEQ	2'b 10	Indicates a single transfer or the first transfer of a burst
SEQ	2'b 11	The remaining transfers in a burst are Sequential and the address is related to the previous transfer

1.3 Transfer Size

The microAptiv UP core has 32-bit wide unidirectional read data and write data bus separately. The *HSIZE[2:0]* signal is used to indicate the size of a data transfer. The microAptiv UP core supports three types of transfer size as shown in Table 2.

Table 2 Transfer Size

Size (Bits)	HSIZE[2:0]	Description
8	3'b 000	Byte Transfer

Table 2 Transfer Size

Size (Bits)	HSIZE[2:0]	Description
16	3'b 001	Halfword Transfer
32	3'b 010	Word Transfer

1.4 Burst Operation

The *HBURST[2:0]* signal is used to indicate the burst type. The microAptiv UP core supports two types of burst operations, SINGLE and WRAP4, as shown in Table 3. A SINGLE burst operation will transfer one word of data. WRAP4 transfers four words of data of the same four-word line, in wrap-around fashion, starting from the addressed word. Figure 4 shows the possible sequence order of the words based on different start address.

Table 3 Burst Operation Types

Burst Operation	HBURST[2:0]	Description
SINGLE	3'b 000	Single Burst
INCR	3'b 001	Not supported
WRAP4	3'b 010	4-beat Wrapping Burst
INCR4	3'b 011	Not supported
WRAP8	3'b 100	Not supported
INCR8	3'b 101	Not supported
WRAP16	3'b 110	Not supported
INCR16	3'b 111	Not supported

Table 4 Sequence Order for 4-beat wrapping burst of word

Start Addr (1st Beat)	*HADDR[3:0]* Sequence			
0	0	4	8	C
4	4	8	C	0
8	8	C	0	4
C	C	0	4	8

Figure 6 shows a write transfer using a four-beat wrapping burst with a wait state added in the first transfer.

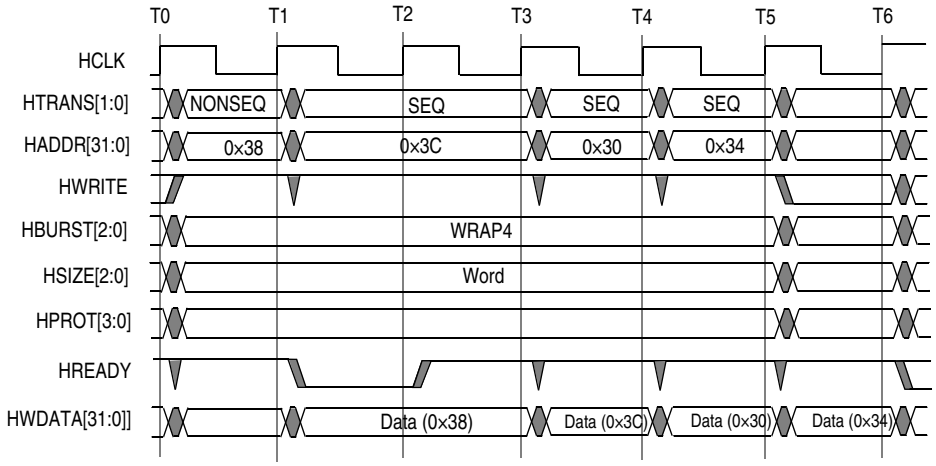

Figure 6 Four-Beat Wrapping Burst of Write Transfer

Figure 7 shows a read transfer using a four-beat wrapping burst, with a wait state added for the first transfer.

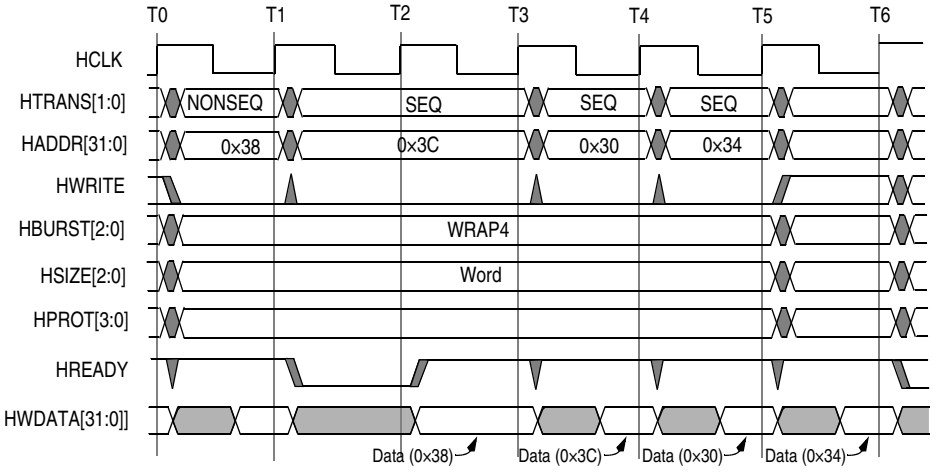

Figure 7 Four-Beat Wrapping Burst of Read Transfer

1.4.1 Early Burst Termination

A burst operation can be terminated by the slave device with an ERROR response on the *HRESP* signal. When a slave device issues an ERROR response, the core discontinues the current transfer, and then issues a new burst transaction or issues single transactions for the remaining incomplete data transfers of the previous burst. Figure 8 shows a waited transfer, with the address changing, followed by an ERROR response from the slave device. The ERROR response to the access of the address 0x24 will raise a bus error exception in the core.

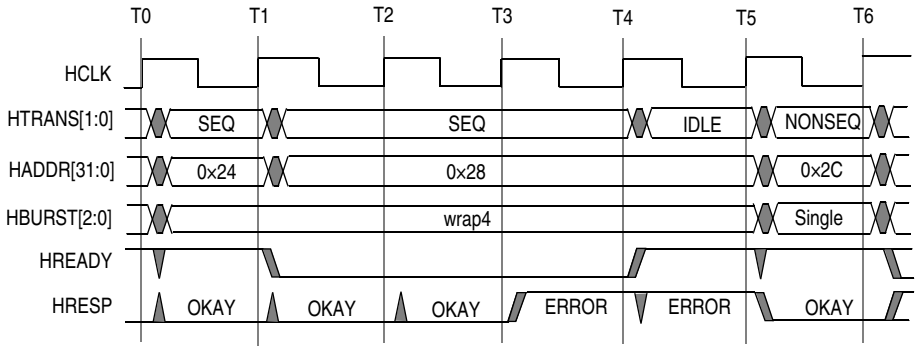

Figure 8 Address Changes During a Waited Transfer After an ERROR

If the ERROR response is the last beat of a burst, and the next terminated access is the first beat of a new burst (single or WRAP), this new burst will be restarted on the bus. An example is shown in Figure 9.

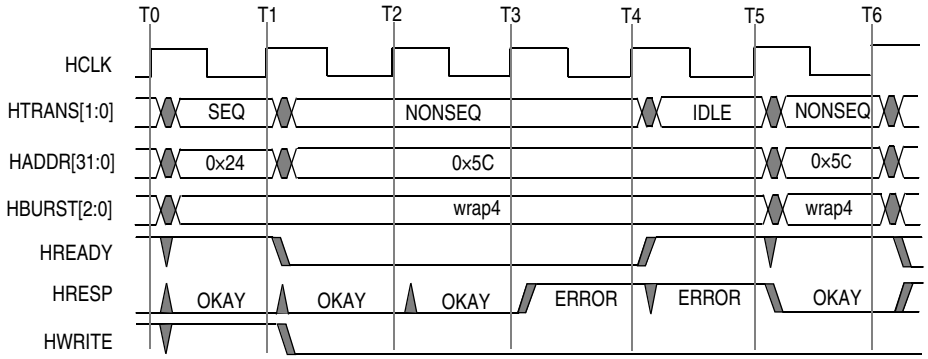

Figure 9 Error Response Terminates the First Beat of a Read burst

1.5 Waited Transfers

This section describes transfer-type changes during wait states and address changes during Waited states.

1.5.1 IDLE Transfer

During a waited transfer, the master is permitted to change the transfer type from IDLE to NONSEQ and also change the address.

Figure 10 shows a waited transfer for a SINGLE burst, with a transfer-type change from IDLE to NONSEQ.

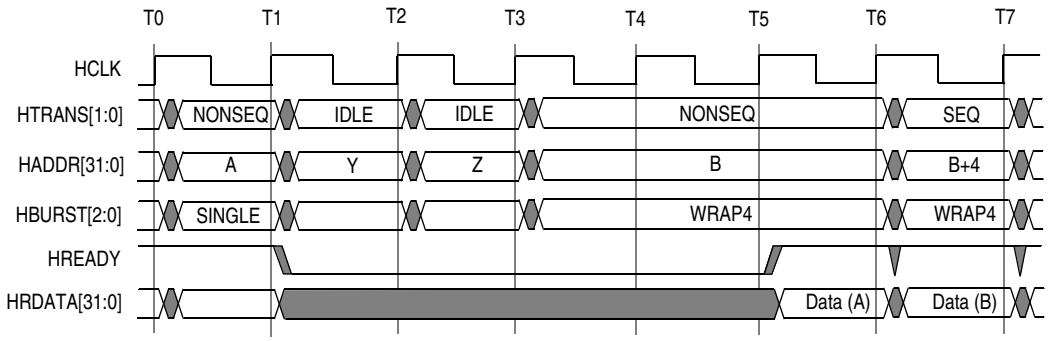

Figure 10 Waited Transfer, IDLE to NONSEQ

1.5.2 Protection Control

The bus interface unit of the microAptiv UP core does not generate all protection information on the *HPROT[3:0]* signal. The upper three bits of *HRPOT* defaults to 3'b001, while the lsb is used to distinguish between an opcode fetch and a data access, as shown in Table 5.

Table 5 Protection Control

HPROT[3:0]	Description
4'b 0010	Opcode Fetch
4'b 0011	Data Access

1.6 Locked Transfers

In the microAptiv UP core, *HMASTLOCK* is used to lock access of a RMW sequence raised by an atomic instruction accessing uncached space. Figure 11 shows a locked transfer.

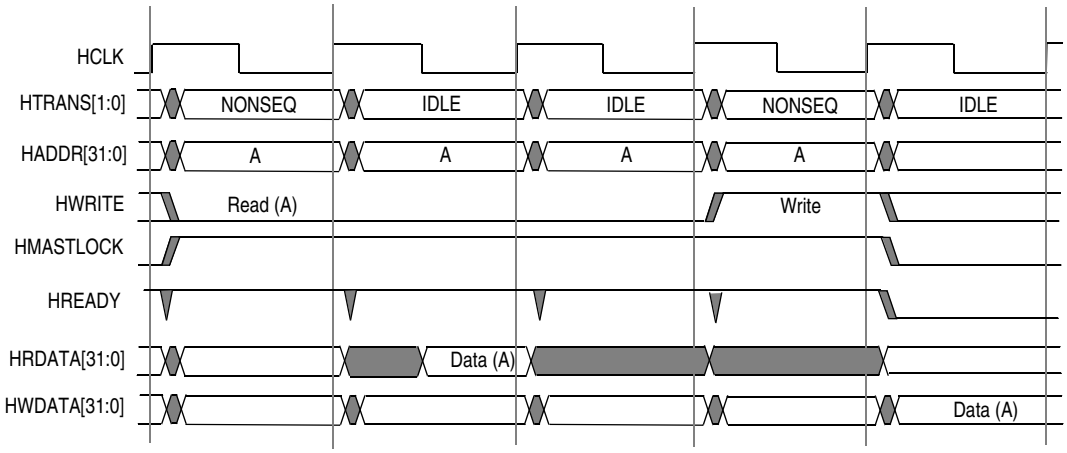

Figure 11 Locked Transfer

Between the locked read and write transactions, there must be IDLE cycles inserted to ensure there is enough time for data calculation and data transmission. Other transactions are not allowed to be inserted during these IDLE cycles. *HMASTLOCK* will be held until the write transaction is completed on the bus. The lock sequence would be terminated when the Read part of the RMW sequence gets an ERROR response as shown in Figure 12. The *HMASTLOCK* would be de-asserted right after the second phase of the ERROR response of the READ.

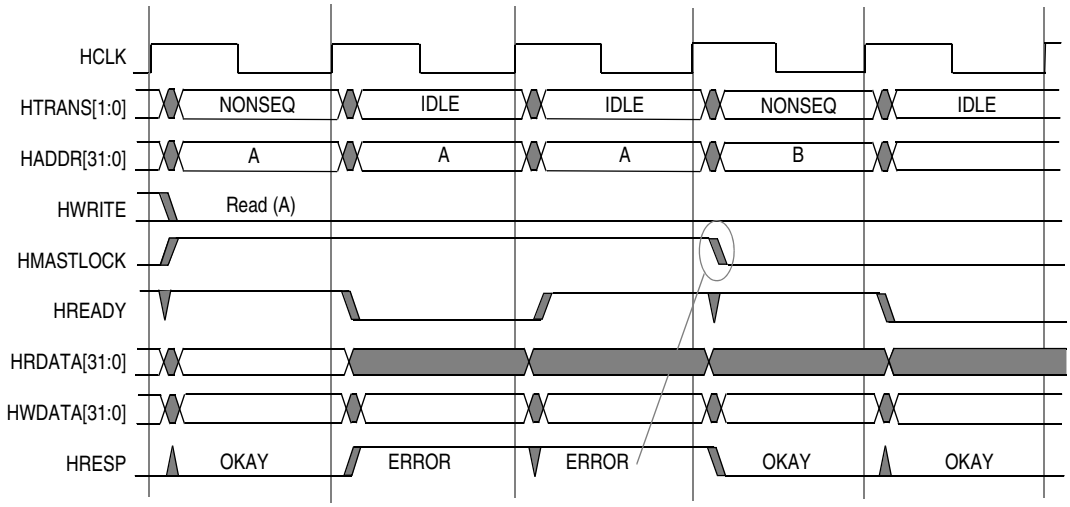

Figure 12 HMASTLOCK was deasserted by the ERROR response of Read Sequence

2 Clock Ratios

The AHB interface of the microAptiv UP core can be connected to a device that is running at a lower clock rate. The core I/O registers are clocked by the primary core clock, but can be selectively enabled so that outputs are driven and

inputs are sampled in the appropriate cycles for communicating with a lower speed device. For devices running at the same core frequency, the core registers will be enabled every cycle.

3 Write Buffer

The write buffer is organized as two, 16-byte buffers. Each buffer contains data from a single 16-byte aligned block of memory. One buffer contains the data currently being transferred on the external interface, while the other buffer contains accumulating data from the core.

Data from the accumulation buffer is transferred to the AHB-Lite bus under one of the following conditions:

- When a store is attempted by the core to a 16-byte block that is different from the block that is currently being accumulated.
- **SYNC** instruction. The **CACHE** instruction also performs an implicit **SYNC**.
- Store to a valid word in the buffer if merging is disabled.
- Any store to uncached memory.
- A load to the line being merged.
- A complete 16-byte line has been gathered for a burst write and the bus is idle.

Note that if a transfer is forced, and the data in the external interface buffer has not been written out to memory, the core is stalled until the memory write completes. After completion of the memory write, accumulated buffer data can be written to the external interface buffer.

4 Merging Control

All microAptiv UP cores implement two, 16-byte collapsing write buffers that allow byte, halfword, tri-byte, or word writes from the core to be accumulated in the buffer into a 16-byte value, before bursting the data out onto the bus in word format. This buffer also gathers dirty cache lines during an eviction. Note that writes to uncached areas are never merged.

Merging can be disabled. When merging is disabled, the buffer will still attempt to gather an entire 16-byte line to generate a bursted write. If a store is attempted to a word that is already valid in the write buffer, the buffer will be flushed, and the two stores will not merge.

The merging option is selected by the *SI_MergeMode[1:0]* input.

5 AHB-Lite Signal Descriptions

Table 6 describes the AHB-Lite interface signals.

Table 6 AHB-Lite Signal Descriptions

Signal Name	Type	Description
SI_SimpleBE[1:0]	SIn	The state of these signals can constrain the core to only generate certain byte enables on AHB-Lite interface transactions. This eases connection to some existing bus standards. Value of *SI_SimpleBE[0]* is visible in the $Config0_{SB}$ register field. **SI_SimpleBE[1:0] / Byte Enable Mode** 00_2 — All BEs allowed 01_2 — Naturally aligned bytes, half-words, and words only 10_2 — Reserved 11_2 — Reserved
SI_GPCInt	Out	For Virtualization module, same as *SI_PCInt* in Guest context.
HADDR[31:0]	Out	The 32-bit system address bus
HBURST[2:0]	Out	The burst type indicates if the transfer is a single transfer or forms part of a burst. Fixed length bursts of 4, 8, and 16 beats are spec'ed but not all are supported. The burst can be incrementing or wrapping. Only Single or WRAP4 are supported in the microAptiv UP core.
HCLK	Out	The bus clock times all bus transfers. All signal timings are related to the rising edge of *HCLK*. It is a reference clock from the gated main clock *SI_ClkIn*.
HGID[7:0]	Out	GuestID associated with the memory access except for write-backs. *HGID* will be cleared to 0 during write-back operations.
HMASTLOCK	Out	When HIGH, this signal indicates that the current transfer is part of a locked sequence. It has the same timing as the address and control signals. In the microAptiv UP core when atomic instruction access uncached space through AHB-Lite, assert *HMASTLOCK* until the atomic write transaction is broadcast on the AHB-Lite bus. Typically the locked transfer is used to maintain the integrity of a semaphore.
HPROT[3:0]	Out	The protection control signals provide additional information about a bus access and are primarily intended for use by any module that wants to implement some level of protection. *HPROT[3:1]* is set default to b001 because the microAptiv UP core can not provide the accuracy of all protection information. But *HPROT[0]* is used to distinguish between Opcode fetch and Data access. *HPROT*=4'b0010 for instruction fetches; 4'b0011 for data loads and stores
HRDATA[31:0]	In	Read Data
HREADY	In	When HIGH, the HREADY signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESETn	Out	The bus reset signal is active LOW and resets the system and the bus. This is the only active LOW AHB-Lite signal.
HRESP	In	The transfer response. When LOW, the HRESP signal indicates that the transfer status is OKAY. When HIGH, the HRESP signal indicates that the transfer status is ERROR.

Table 6 AHB-Lite Signal Descriptions (Continued)

Signal Name	Type	Description
HSIZE[2:0]	Out	Indicates the size of the transfer, that is typically byte, halfword, or word. The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
HTRANS[1:0]	Out	Indicates the transfer type of the current transfer. This can be: • IDLE • BUSY—Not implemented in M14Kc • NONSEQUENTIAL • SEQUENTIAL
HWDATA[31:0]	Out	The write data bus transfers data from the master to the slaves during write operations.
HWRITE	Out	Indicates the transfer direction. When HIGH this signal indicates a write transfer and when LOW a read transfer.
SI_AHBStb	In	Enable AHB input and output interface signals to be registered and to suppress HCLK high pulses so that the AHB bus can run at a lower clock ratio than the CPU core clock. This signal is registered by the core prior to use.
SRAM-style Interface: Refer to Chapter 3, "SRAM-Style Interface" on page 31 for more details.		
The SRAM-style interface allows simple connection to fast, tightly-coupled memory devices.It can be configured with independent interfaces for Instruction and Data, or a Unified interface. Signals related to the I-side interface are prefixed with "*IS_*"; signals related to the D-side interface are prefixed with "*DS_*". When the Unified interface is used, then most D-side signals are obsoleted, since they have an I-side equivalent; only the write data bus, *DS_WData*, continues to be used from the D-side.		
DS_Abort	Out	Request for transaction (read, write or sync) to be aborted, if possible. It is optional whether the external logic uses this signal or not, although using it may reduce interrupt latency. Completion of any transaction (aborted or not) is always communicated through *DS_Stall*. Whether the transaction was in fact aborted is signalled using *DS_AbortAck*. *DS_Abort* is asserted through (and including) the cycle where *DS_Stall* is deasserted.
DS_Addr[31:2]	Out	Address of transaction. When *DS_Sync* is asserted high, *DS_Addr[10:6]* holds the "sync type" (the "stype" field of the SYNC instruction).
DS_Atomic	Out	Asserted when a read-modify-write transaction is due to Atomic instruction,
DS_BE[3:0]	Out	Byte enable signals for transaction. *DS_BE[3]* enables byte lane corresponding to bits *31:24*. *DS_BE[2]* enables byte lane corresponding to bits *23:16*. *DS_BE[1]* enables byte lane corresponding to bits *15:8*. *DS_BE[0]* enables byte lane corresponding to bits *7:0*.
DS_CCA[2:0]	Out	Provides the cache coherence attribute (CCA) for the current data request. The core will limit this value to either 2 (uncacheable) or 3 (cacheable)
DS_DTLBAbort	Out	Asserted when a data RPU TLB exception is detected. May be used by external logic to cancel the current transaction. This signal is asserted one cycle after the transaction begins; the external logic must stall transactions by one cycle. *DS_DTLBAbort* is asserted through (and including) the cycle where *DS_Stall* is deasserted.
DS_EjtBreakMPUEn	Out	One or more EJTAG data breakpoints, imprecise breakpoints, or MPU regions are enabled.

Table 6 AHB-Lite Signal Descriptions (Continued)

Signal Name	Type	Description
DS_EjtBreakMPU	Out	Asserted when an EJTAG data break, imprecise break, or protected exception is detected. May be used by external logic to cancel the current transaction. This signal is asserted one cycle after the transaction begins; the external logic must stall transactions by one cycle if *DS_EjtBreakMPUEn* indicates that a break may occur. *DS_EjtBreakMPU* is asserted through (and including) the cycle where *DS_Stall* is deasserted. This signal may be superseded by other higher priority exceptions. In this scenario, a breakpoint exception may not occur, but the system may need to cancel the current transaction nonetheless.
DS_First	Out	Asserted if a new request (Read, Write, or Sync) might be starting this cycle. This will always be asserted during the first cycle of a new request and will never be asserted while a request is stalled. This signal is used to qualify *DS_Abort* -> *DS_Abort* should be ignored when *DS_First* is asserted.
DS_GuestID[7:0]	Out	GuestID associated with memory access.
DS_Lock	Out	Asserted when a read transaction is due to an LL (load linked) instruction.
DS_ParityEn	Out	Indicates Parity is enabled.
DS_Read	Out	Read strobe.
DS_Sync	Out	Sync strobe.
DS_Unlock	Out	Asserted when a write transaction is due to an SC (store conditional) instruction.
DS_WbCtl	Out	Write buffer control. This signal is asserted when the microAptiv UP core can guarantee that no D-side read transaction will be started in the current clock cycle. For the purpose of generating this signal, if there is a pending transaction, the microAptiv UP core assumes that it will end in this cycle, in order to determine whether a new read transaction might be started or not. Unlike *DS_Read*, there is no asynchronous path from *DS_Stall* or any other input signal to *DS_WbCtl*. Also, it is an earlier signal than *DS_Read*. It is intended to be used by an external agent to control flushing of a write buffer (if a write buffer is present).
DS_WData[31:0]	Out	Write data as defined by *DS_BE[3:0]*/*IS_BE[3:0]*. Used for both D-side and I-side transactions.
DS_WPar[3:0]	Out	Parity Bits for Write operation. Only valid when parity is implemented.
DS_Write	Out	Write strobe.
DS_AbortAck	In	Valid in the cycle terminating the transaction (*DS_Stall* deasserted). Asserted high if transaction was aborted. If no abort was requested (*DS_Abort* is low), and *DS_AbortAck* is asserted high in the cycle terminating the transaction, a bus error exception is taken.
DS_Error	In	Valid in the cycle terminating the transaction (*DS_Stall* deasserted). Asserted high if transaction caused an error. Causes bus error exception to be taken by the core.
DS_ParPresent	In	Indicates that Parity is present.
DS_RBE[3:0]	In	Byte enable signals for *DS_RData[31:0]*. *DS_RBE[3]* enables byte lane corresponding to *DS_RData[31:24]*. *DS_RBE[2]* enables byte lane corresponding to *DS_RData[23:16]*. *DS_RBE[1]* enables byte lane corresponding to *DS_RData[15:8]*. *DS_RBE[0]* enables byte lane corresponding to *DS_RData[7:0]*.

Table 6 AHB-Lite Signal Descriptions (Continued)

Signal Name	Type	Description
DS_RData[31:0]	In	Read data.
DS_Redir	In	Valid in the cycle terminating the transaction (*DS_Stall* deasserted). Asserted high if transaction must be redirected to In-side.
DS_RPar[3:0]	In	Read Parity bits. Ignored when parity is not implemented.
DS_Stall	In	Indicates that the transaction is not ready to be completed.
DS_UnlockAck	In	Valid in the cycle terminating the transaction (*DS_Stall* deasserted). Result of *DS_Unlock* operation. Should be asserted high if system holds a lock on the address used for the write transaction (SC).
IS_Abort	Out	Request for transaction to be aborted, if possible. It is optional whether the external logic uses this signal or not, although using it may reduce interrupt latency. Completion of any transaction (aborted or not) is always communicated through *IS_Stall*. Whether the transaction was in fact aborted is signalled using *IS_AbortAck*. *IS_Abort* is asserted through (and including) the cycle where *IS_Stall* is deasserted. In certain cases, a new request can start in this cycle, while *IS_Abort* is still asserted. The new request should not be aborted. The *IS_First* signal will be asserted in that cycle and can be used to determine whether the abort is applicable to the current request.
IS_Addr[31:2]	Out	Address of transaction. When *IS_Sync* is asserted high, *IS_Addr[10:6]* holds the “sync type” (the “stype” field of SYNC instruction).
IS_Atomic	Out	Asserted when a read-modify-write transaction is due to Atomic instruction,
IS_BE[3:0]	Out	Byte enable signals for transaction. *IS_BE[3]* enables byte lane corresponding to bits *31:24*. *IS_BE[2]* enables byte lane corresponding to bits *23:16*. *IS_BE[1]* enables byte lane corresponding to bits *15:8*. *IS_BE[0]* enables byte lane corresponding to bits *7:0*.
IS_CCA[2:0]	Out	Provides the cache coherence attribute (CCA) for the current fetch request. The core will limit this value to either 2 (uncacheable) or 3 (cacheable)
IS_EjtBreakMPUEn	Out	One or more EJTAG instruction breakpoints or MPU regions are enabled. This signal is also asserted for the Unified Interface when one or more data breakpoints are enabled.
IS_EjtBreakMPU	Out	Asserted when an instruction break or protected exception is detected. Also asserted for the Unified Interface when a data break is detected. May be used by external logic to cancel the current transaction. External logic may determine whether this is an instruction break or a data break based on *IS_Instr*. This signal is asserted one cycle after the transaction start, so when precise breaks are required, the external logic must stall transactions by one cycle if *IS_EjtBreakMPUEn* indicates that a break may occur. *IS_EjtBreakMPU* is asserted through (and including) the cycle where *IS_Stall* is deasserted.
IS_First	Out	Asserted if a new request (Read, Write, or Sync) might be starting this cycle. This will always be asserted during the first cycle of a new request and will never be asserted while a request is stalled. This signal is used to qualify *IS_Abort* -> *IS_Abort* should be ignored when *IS_First* is asserted.
IS_GuestID[7:0]	Out	GuestID associated with memory access.
IS_Instr	Out	Indicates instruction fetch when high, or redirected data read/write when low.

Table 6 AHB-Lite Signal Descriptions (Continued)

Signal Name	Type	Description
IS_ITLBAbort	Out	Asserted when an instruction RPU TLB exception is detected and also asserted for the Unified Interface when a data RPU TLB exception is detected. May be used by external logic to cancel the current transaction. External logic may determine whether this is an instruction or a data RPU TLB exception based on *IS_Instr*. This signal is asserted one cycle after the transaction starts, so when precise breaks are required, the external logic must stall transactions by one cycle. *IS_ITLBAbort* is asserted through (and including) the cycle where *IS_Stall* is deasserted.
IS_LinkOffset[1:0]	Out	Offset used to determine the link return fetch address relative to the previous fetch address. This signal is only valid when *IS_WasCall* is asserted in the same cycle. The link return address is relative to the fetch immediately prior to the one in which *IS_WasCall* is asserted. Refer Table 3.2 in Section 3.1.8 "External Call Indication" for detail description.
IS_Lock	Out	Asserted when a read transaction is due to a redirected LL (load linked) instruction,
IS_ParityEn	Out	Indicate Parity is enabled, based on the value of the ErrCtl.PE (CP0) bit.
IS_Read	Out	Read strobe.
IS_Sync	Out	Sync strobe.
IS_Unlock	Out	Asserted when a write transaction is due to a redirected SC (store conditional) instruction.
IS_UnlockAll	Out	Asserted for one clock cycle when an ERET instruction is executed.
IS_WasCall	Out	Indicates that a recent fetch was for a control transfer instruction that saves a return address in a GPR (JAL, JALR, JALX, BGEZAL, BGEZALL, BLTZAL, BLTZALL). This indication and a corresponding offset may enable external logic to maintain a buffer of instructions at the return address.
IS_WbCtl	Out	Write buffer control. This signal is asserted when the microAptiv UP core can guarantee that no I-side read transaction will be started in the current clock cycle. For the purpose of generating this signal, if there is a pending transaction, the microAptiv UP core assumes that it will end in this cycle, in order to determine whether a new read transaction might be started or not. Unlike *IS_Read*, there is no asynchronous path from *IS_Stall* or any other input signal to *IS_WbCtl*. Also, it is an earlier signal than *IS_Read*. It is intended to be used by an external agent to control flushing of a write buffer (if a write buffer is present).
IS_Write	Out	Write strobe. Only asserted due to a redirected data write.
IS_AbortAck	In	Valid in the cycle terminating the transaction (*IS_Stall* deasserted). Asserted high if transaction was aborted. If no abort was requested (*IS_Abort* is low), and *IS_AbortAck* is asserted high in the cycle terminating the transaction, a bus error exception is taken.
IS_Error	In	Valid in the cycle terminating the transaction (*IS_Stall* deasserted). Asserted high if transaction caused an error. Causes bus error exception to be taken by the core.
IS_ParPresent	In	Indicates that Parity is present.

Table 6 AHB-Lite Signal Descriptions (Continued)

Signal Name	Type	Description
IS_RBE[3:0]	In	Byte enable signals for *IS_RData[31:0]*. *IS_RBE[3]* enables byte lane corresponding to *IS_RData[31:24]*. *IS_RBE[2]* enables byte lane corresponding to *IS_RData[23:16]*. *IS_RBE[1]* enables byte lane corresponding to *IS_RData[15:8]*. *IS_RBE[0]* enables byte lane corresponding to *IS_RData[7:0]*.
IS_RData[31:0]	In	Read data.
IS_RPar[3:0]	In	Read Parity bits. Ignored when parity is not implemented.
IS_Stall	In	Indicates that the transaction is not ready to be completed.
IS_UnlockAck	In	Valid in the cycle terminating the transaction (*IS_Stall* deasserted). Result of *IS_Unlock* operation. Should be asserted high if system holds a lock on the address used for the redirected write transaction (SC).

Public. This publication contains proprietary information which is subject to change without notice and is supplied ‘as is’, without any warranty of any kind.

Template: nDb1.03, Built with tags: 2B