EXCALIBUR™

Excalibur

Devices

Hardware Reference Manual
July 2002
Version 3.0

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com

MNL-EPXA10HRM-3.0

About this Manual

This manual provides comprehensive information about the Altera® Excalibur™ devices.

Table 1 shows the manual revision history.

Go to the following sources for more information:

- Refer to the readme file for late-breaking information that is not available in this manual

Table 1. Revision History

Date	Description
January 2001	Initial publication
April 2002	Additional information for EPXA1 and EPXA4 devices, pinout revision, timing diagram revision
July 2002	Final timing numbers for the dual-port SRAM, EBI, and embedded stripe bridges; other minor changes

How to Find Information

- The Adobe Acrobat Find feature allows you to search the contents of a PDF file. Click on the binoculars icon in the top toolbar to open the **Find** dialog box
- Bookmarks serve as an additional table of contents
- Thumbnail icons, which provide miniature previews of each page, provide a link to the pages
- Numerous links, shown in green text, allow you to jump to related information

How to Contact Altera

For the most up-to-date information about Altera products, go to the Altera world-wide web site at http://www.altera.com.

For additional information about Altera products, consult the sources shown in Table 2.

Table 2. How to Contact Altera

Information Type	Access	USA & Canada	All Other Locations
Altera Literature Services	Electronic mail	lit_req@altera.com *(1)*	lit_req@altera.com *(1)*
Non-technical customer service	Telephone hotline	(800) SOS-EPLD	(408) 544-7000 (7:30 a.m. to 5:30 p.m. Pacific Time)
	Fax	(408) 544-7606	(408) 544-7606
Technical support	Telephone hotline	(800) 800-EPLD (6:00 a.m. to 6:00 p.m. Pacific Time)	(408) 544-7000 *(1)* (7:30 a.m. to 5:30 p.m. Pacific Time)
	Fax	(408) 544-6401	(408) 544-6401 *(1)*
	World-wide web support	www.mysupport.altera.com	www.mysupport.altera.com
	FTP site	ftp.altera.com	ftp.altera.com
General product information	Telephone	(408) 544-7104	(408) 544-7104 *(1)*
	World-wide web site	http://www.altera.com	http://www.altera.com

Note:
(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The *Excalibur Devices Hardware Reference Manual* uses the typographic conventions shown in Table 3.

Table 3. Conventions

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\QuartusII** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75 (High-Speed Board Design).*
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <*file name*>, <*project name*>**.pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of Quartus II Help topics are shown in quotation marks. Example: "Configuring an ARM-Based Device with the ByteBlasterMV™ Download Cable."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input`. Active-low signals are denoted by suffix `_n`, e.g., `reset_n`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\quartusII\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c.,...	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
	The hand points to information that requires special attention.
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Notes:

Contents

How to Find Information ..... **iii**
How to Contact Altera ..... **iv**
Typographic Conventions v
Features... 11
General Description 12
Functional Description 15
The Embedded Processor 15
Internal Memory 17
External Memory Controllers 18
Embedded Peripherals 18
Configuration Registers 18
Memory Map 19
Embedded Stripe Phase-Locked Loops (PLLs) 19
UART 20
Timer 20
Watchdog Timer 20
General Purpose I/O Port 21
Interrupt Controller 21
Embedded Stripe-to-PLD Interface Clocks 21
Programmable Logic Architecture 22
Technical Description 22
Embedded Processor 22
Debug Support 25
Trace Support 25
Bus Architecture 27
AHB1 29
AHB2 31
Embedded Stripe Bridges 34
On-Chip SRAM 47
Single-Port SRAM 48
Registers 50
Dual-Port SRAM 51
Registers 66
SDRAM Controller 68
Module I/O 69
Bus Interface 70
Endianness 71
Memory Access State Machine 73
Registers 74

Connecting SDRAM Devices 77
SDRAM Device Configuration 79
Clocking 82
Expansion Bus Interface 83
EBI Operation 83
AHB Slave Interface 85
EBI FIFO Buffers 86
EBI Interface 87
Interface Signals 91
EBI Timing Diagrams 91
EBI Clock 94
Connecting the EBI to LH28F160BE-BTL90 Flash Memory 94
Registers 95
Embedded Peripherals Memory Map 99
Memory Mapping in Boot-From-Flash Mode 99
Memory Map Control Registers 100
UART 102
UART Pins and Signals 103
Reset Behavior 104
Transmitter Operation 104
Receiver Operation 105
Modem Status Lines 105
UART Data Formats 106
Registers 107
Timer 114
Registers 117
Watchdog Timer 119
Interface Signals 120
Counter Operation 120
Trigger Modes 121
Resetting the Watchdog Timer 122
Software Locking 122
Reset 122
Registers 123
Interrupt Controller 124
Interrupt Requests to the Processor 126
Interrupt Priority 126
Indirect Access to FIQ and IRQ 127
Operating Modes 127
Interrupt Signals 130
Registers 130
Clocks 136
External Reference 137
PLLs 138
Clock Control 140
Low-Power Mode 140

Registers 142
PLD Clocks 148
Configuration Logic 149
Boot from Flash 150
Boot from External Configuration Source 151
Registers 153
Reset & Mode Control 155
Types of Reset 155
Sources of Reset 157
Reset Operation 160
Registers 162
IEEE Std. 1149.1 (JTAG) Support 163
JTAG Modes 163
JTAG Boundary-Scan Testing 164
PLD TAP Controller 165
SignalTap Embedded Logic Analyzer 166
Embedded Processor TAP Controller 166
I/O Features 167
I/O Control 167
Registers 167
Dedicated I/O 169
Operating Conditions 171
Electrical Characteristics & Timing Diagrams 174
Embedded Stripe Bridges 174
EBI 181
SDRAM 186
Trace Port 193
UART 194
JTAG 195
Dual-Port SRAM 196
Device Pinouts 199
Packaging 199
Embedded Processor Register Summary 199
Comprehensive Pin & Signal Listing 207
Abbreviations 215
Glossary 217

Notes:

Features...

- Combination of a world-class RISC processor system with industry-leading programmable logic on a single device
- Industry-standard ARM922T™ 32-bit RISC processor core operating at up to 200 MHz
  - ARMv4T instruction set with Thumb® extensions
  - Memory management unit (MMU) included for real-time operating system (RTOS) support
  - Harvard cache architecture with 64-way set associative separate 8-Kbyte instruction and 8-Kbyte data caches
- APEX™ 20KE-like programmable logic architecture ranging from 100,000 to 1,000,000 gates (see Appendix A)
- Advanced bus architecture based on advanced microcontroller bus architecture (AMBA™) high-performance bus (AHB)
- Embedded programmable on-chip peripherals
  - ETM9™ embedded trace module to assist software debugging
  - Flexible interrupt controller
  - Universal asynchronous receiver/transmitter (UART)
  - General-purpose timer
  - Watchdog timer
- Advanced memory support
  - Internal single-port SRAM up to 256 Kbytes
  - Internal dual-port SRAM up to 128 Kbytes
  - Internal SDRAM controller
    - Single data-rate (SDR) and double data-rate (DDR) support
    - Up to 512 Mbytes
    - Data rates to 133 (266) MHz
  - Expansion bus interface (EBI)
    - Compatible with industry-standard flash memory, SRAMs, and peripheral devices
    - Four devices, each up to 32 Mbytes
- PLD configuration/reconfiguration possible via the embedded processor software
- Extended Quartus® II development environment for Excalibur support
  - Integrated hardware and software development environment
  - MegaWizard® Plug-In interface configures the embedded processor, PLD, bus connections, and peripherals
  - C/C++ compiler, source-level debugger, and RTOS support

- Fully configurable memory map
- Extensive embedded system debug facilities
  - SignalTap® embedded logic analyzer
  - ARM® JTAG processor debug support
  - Real-time data/instruction processor trace
  - Background debug monitoring via the IEEE Std. 1149.1 (JTAG) interface
- Multiple and separate clock domains controlled by software-programmable phased-lock loops (PLLs) for embedded processor, SDRAM, and PLD
  - ClockBoost® circuitry provides clock multiplication for the embedded stripe and the PLD
  - ClockLock® circuitry reduces clock delay and skew in the PLD
- Advanced packaging options
- 1.8-V supply voltage, but many I/O standards supported

See Appendix A for a comparison of devices in the Excalibur family.

General Description

Members of the family of Excalibur embedded processor programmable logic devices (PLDs) combine an unparalleled degree of integration and programmability. The Excalibur family offers an outstanding embedded system development platform, providing a cost-efficient access to leading-edge embedded processors and PLD performance.

The Excalibur family offers a variety of PLD densities and memory sizes to fit a wide range of applications and requirements. The high-performance embedded architecture is ideal for compute-intensive as well as high data-bandwidth applications.

Figure 1 shows the structure of the Excalibur family. The embedded stripe contains the processor core, peripherals, and memory subsystem. The amounts of single- and dual-port memory vary as listed in Appendix A.

Figure 2 on page 14 shows the system architecture of the embedded stripe and the interfaces to the PLD portion of the devices. This architecture promotes maximum integration with minimal system cost and allows the embedded stripe and PLD to be independently optimized for maximum performance.

Figure 1. Excalibur Embedded Processor PLD Architecture

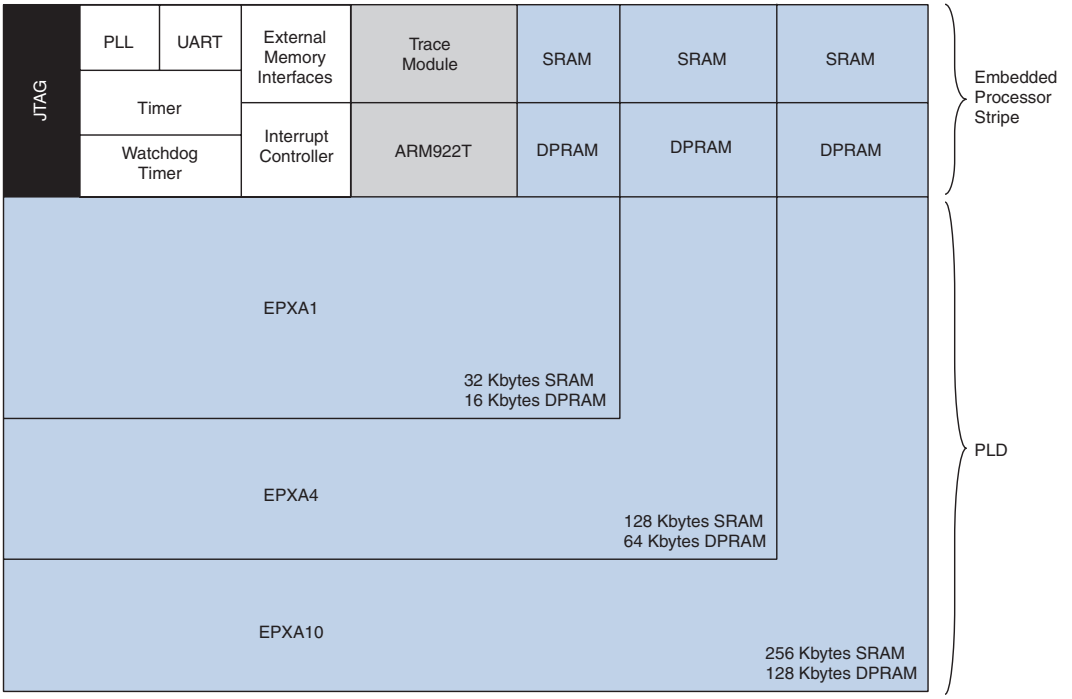
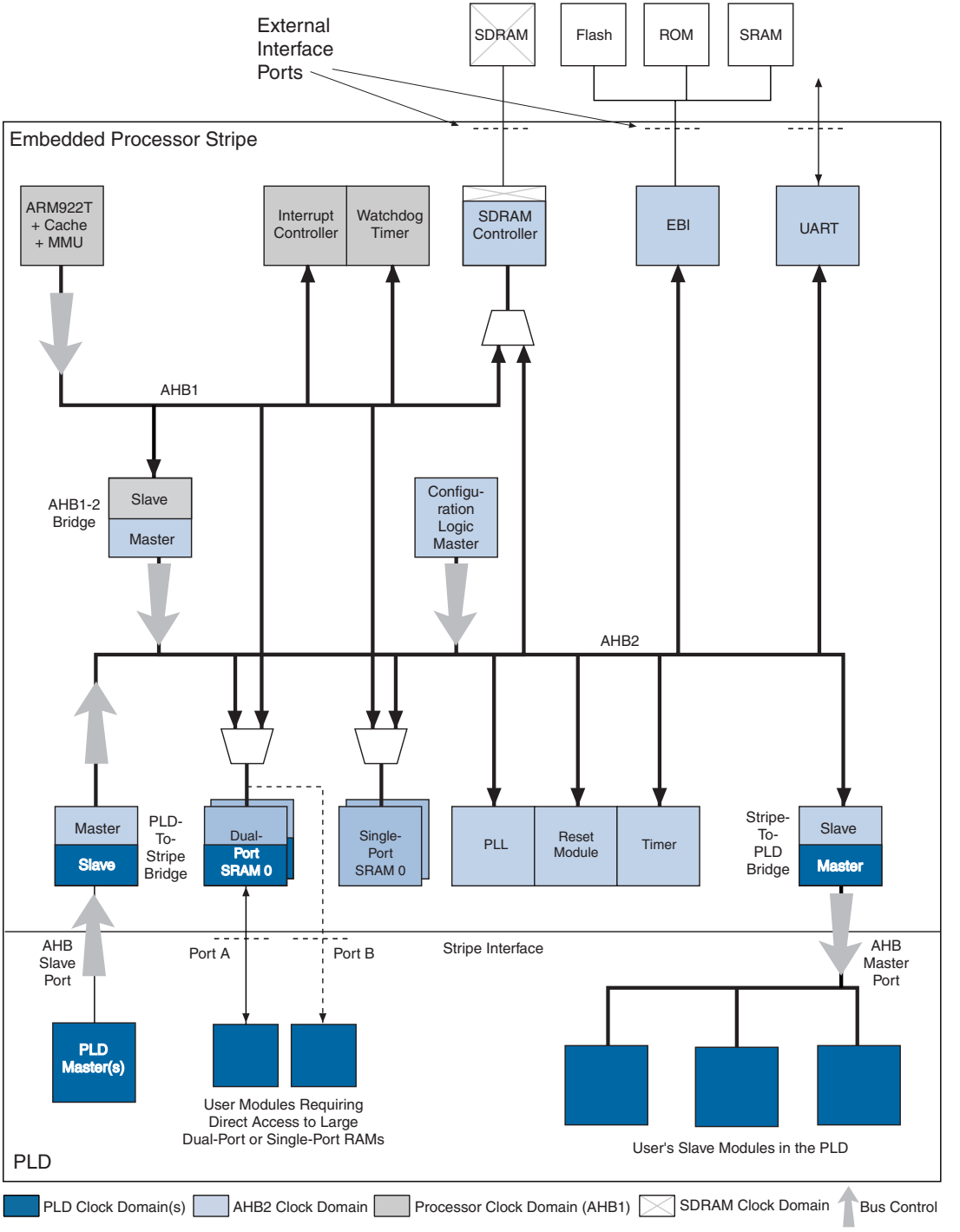


Figure 2. Excalibur Embedded Processor PLD System Architecture



Two AMBA-compliant AHBs ensure that the embedded processor activity is unaffected by peripheral and memory operation. Three bidirectional AHB-to-AHB bridges enable embedded peripherals and PLD-implemented peripherals to exchange data with the embedded processor or with other peripherals. With these interfaces, the performance of the ARM922T is uncompromised, and is equivalent to an ASIC implementation on a 0.18-µm CMOS process.

The Excalibur family is supported by the Altera® Quartus II development system for both PLD logic design and the integration of embedded software. The Quartus II software provides an integrated package for complete hardware logic design, including HDL and schematic design entry, compilation and logic synthesis, full simulation and timing analysis, and programming file generation, as well as hardware logic debug using SignalTap logic analyzer.

With the Quartus II SoftMode™ co-design capability, embedded software development, debugger support, and unified programming file generation can be easily combined from a single integrated design environment (IDE). The Quartus II tools are pre-configured to support embedded software development tools such as the ARM Developer Suite or Red Hat GNUPro Tools for ARM922T processors. The Quartus II software operates on Windows-based PCs, Sun SPARCstations, and HP 9000 Series 700/800 workstations. The Quartus II software provides NativeLink® integration to 3rd-party industry-standard PC and UNIX workstation-based electronic design automation (EDA) tools.

Functional Description

The family of Excalibur embedded processor PLDs has a system architecture (embedded processor bus structure, on-chip memory, and peripherals) that combines the performance advantages of ASIC integration with the flexibility and time-to-market advantages of PLDs.

The Embedded Processor

The ARM922T is a member of the ARM9 family of processor cores. It supports both the 32-bit ARM and 16-bit Thumb instruction sets, allowing users to achieve a balance between high code-density and performance. Its Harvard architecture, implemented using a five-stage pipeline, allows single clock-cycle instruction operation through simultaneous fetch, decode, execute, memory, and write stages.

Independent of PLD configuration, the embedded processor can undertake the following activities:

- Boot from external memory
- Execute embedded software
- Communicate with the external world
- Run a real-time operating system
- Run interactive embedded software debugging sessions
- Configure/reconfigure the PLD
- Detect errors and restart/reboot/reconfigure the entire system as necessary

The PLD can be configured to implement various extensions:

- Additional soft-core peripherals such as a UART, Ethernet MAC, CAN controllers, PCI, or any other IP core
- Peripherals that are bus masters, sharing the embedded stripe on-chip and off-chip memories as well as other PLD peripheral
- Peripherals that are slaves, controlled by the embedded processor
- Peripherals that exchange data using the on-chip dual-port RAM
- High speed data paths under embedded processor control
- Multi-processor systems, using multiple Nios embedded processor solutions
- Additional embedded processor interrupt sources and controls

PLD designers can take full advantage of the extensive range of Altera intellectual property (IP) MegaCore® functions to implement complex SOPC designs in minimal time but with maximum customization.

The bidirectional bridges and dual-port memory interfaces between the embedded stripe and the PLD are synchronous to the clock domain that drives them; however, the embedded processor domain and the PLD domains can be asynchronous. The clock domain for each side of the interfaces can be optimized for performance. The bidirectional bridges handle the resynchronization across the domains and are capable of supporting 32-bit data accesses to the entire 4-Gbyte address range (32-bit address bus).

The SDRAM memory controller PLL allows users to tune the frequency of the system clock to the speed of the external memory implemented in their systems.

Internal Memory

The embedded stripe contains both single-port and dual-port SRAM. There are two blocks of single-port SRAM; both are accessible to the AHB masters via an arbitrated interface within memory. Each block is independently arbitrated, allowing one block to be accessed by one bus master while the other block is accessed by the other bus master.

Up to 256 Kbytes of single-port SRAM are available, as two blocks of 2 × 128 Kbytes. Each single-port SRAM block is byte-addressable. The size of the SRAM blocks depends on the device, as shown in Table 26. Byte, half-word and word accesses are allowed and are enabled by the slave interface. The behavior of byte and half-word reads is controlled by the system endianness.

Table 4 shows the SRAM block sizes for the Excalibur embedded processor PLD devices.

Table 4. SRAM Block Sizes

Device	Block 0 Size (Kbytes)	Block 1 Size (Kbytes)
EPXA1	16	16
EPXA4	64	64
EPXA10	128	128

In addition, there are either one or two blocks of dual-port SRAM in the embedded stripe, depending on the device type. The outputs of the dual-port memories can be registered. One of the ports gives dedicated access to the PLD; the other port can be configured for access by AHB masters or by the PLD. The width of the data port to the PLD is configurable as ×8, ×16, or ×32 bits. For the larger devices, the dual-port SRAM blocks can be combined to form a ×64-bit data-width interface. This allows the designer to build deeper and wider memories and multiplex the data outputs within the stripe.

Table 5 shows the SRAM block sizes for the Excalibur embedded processor PLD devices.

Table 5. Dual-Port SRAM Block Sizes

Device	Block 0 Size (Kbytes)	Block 1 Size (Kbytes)
EPXA1	16	N/A
EPXA4	32	32
EPXA10	64	64

External Memory Controllers

The Excalibur family provides two embedded memory controllers that can be accessed by any of the bus masters: one for external SDRAM, and a second for external flash memory or SRAM.

The SDRAM memory controller supports the following commonly-available memory standards, without the addition of any logic:

- Single-data rate (SDR) 133-MHz data rates
- Double-data rate (DDR) 266-MHz data rates

An embedded stripe PLL supplies the appropriate timing to the SDRAM memory controller subsystem. Users can program the frequency to match the chosen memory components.

The EBI supports the interface to system ROM, allowing external flash memory access and reprogramming. In addition, static RAM and simple peripherals can be connected to this interface externally.

Embedded Peripherals

A single 16-Kbyte memory region in the embedded stripe contains configuration and control registers, plus status and control registers for the embedded peripherals. The region contains the following modules:

- Configuration Registers
- Embedded Stripe PLLs
- UART
- Timer
- Watchdog timer
- General Purpose I/O Port
- Interrupt controller

Configuration Registers

Configuration and control registers for the embedded memories and controllers, as well as the embedded peripherals, are located within one 16-Kbyte memory region. The registers are located at address `7FFFC000H` on reset, but can be relocated at any time. In this document, all references to register addresses are offsets from the base address.

See the section "Embedded Processor Register Summary" on page 199 for a comprehensive list of all registers.

Memory Map

The elements listed in Table 6 are present as memory-mapped slave peripherals in the embedded stripe. The base address and range for each element can be configured. A memory range can be set to any size that is a power of two.

Table 6. Memory Map Peripherals & Elements

Memory Map Element	Range *Note (1)*		
	EPXA1	EPXA4	EPXA10
Registers, including those for embedded stripe peripherals	16 Kbytes		
Internal SRAM0, SRAM1 (total)	64 Kbytes	128 Kbytes	256 Kbytes
Internal dual-port DPSRAM0, DPSRAM1 (total)	16 Kbytes	64 Kbytes	128 Kbytes
EBI0, EBI1, EBI2, EBI3	16 Kbytes to 32 Mbytes		
SDRAM0, SDRAM1	16 Kbytes to 256 Mbytes		
PLD0, PLD1, PLD2, PLD3	16 Kbytes to 2 Gbytes		

Notes:
(1) Range is for each memory map element

Embedded Stripe Phase-Locked Loops (PLLs)

Each member of the device family accommodates PLLs in both the PLD logic and the embedded stripe. Within the PLD, between two and four PLLs are available, as in the comparably-sized APEX 20KE devices. The functionality of the PLLs is identical to APEX 20KE devices, including the ClockBoost and ClockLock capabilities.

The embedded stripe contains two additional PLLs to provide clocks for the embedded stripe AMBA AHBs and the SDRAM memory controller. The PLLs provide the following features:

- Clock generation for the embedded processor and memory subsystem
- A common source for both PLLs
- PLL bypass circuitry, which allows the input reference clock to be routed as the main clock source
- 2.5-V or 3.3-V clock input
- Software control for monitoring status and changing frequency

UART

The universal asynchronous receiver transmitter module (UART) performs serial-to-parallel conversion on data characters received from a peripheral device or modem, and parallel-to-serial conversion on data characters received from the embedded processor. The UART operates in FIFO mode, with the FIFO buffers having a depth of 16 bytes. The CPU can read the status of the UART at any time during operation. The UART reports status information, including the type and condition of the transfer being performed, and any error conditions.

The UART has the following features:

- 5 to 8 data bits, 1 or 2 stop bits
- Even, odd, stick, or no parity
- Programmable baud rate to 230400
- 16-byte receive and transmit FIFO buffers
- False-start bit detection
- Internal diagnostic capabilities
  - Loop-back control for communications-link fault isolation
  - Break insertion and detection in loop-back mode
  - Modem communication support

Timer

The timer is a general purpose dual-channel timer, with the following features:

- 32-bit timer register
- 32-bit clock pre-scaler
- Three operating modes, selectable under register control:
  - Free-running interrupt (heartbeat)
  - Software controlled start/stop (interval timer) with interrupt on limit
  - One-shot interrupt after a programmable delay

Watchdog Timer

The watchdog timer protects the system against software failure, or against severe hardware failures (such as lock-ups), due to power supply problems, for example. It is a one-shot timer, which resets the entire chip when it expires. To prevent the timer from triggering, the peripheral must be regularly written with specific values. The timer period is under software control.

General Purpose I/O Port

The general purpose I/O port is used for signals between the embedded stripe and the PLD. The user accesses these bits through a configuration register. The signals change state synchronously to the AHB2 bus. They are available to the user for a fast access control line to the PLD logic or as a status signal from logic sections within the PLD.

Interrupt Controller

The interrupt controller provides a simple, but flexible, software interface to the interrupt system. The interrupt controller generates two interrupt signals `INT_FIQ_n` and `INT_IRQ_n`, to the embedded processor, using input from up to 17 interrupt sources. The input sources include:

- 10 interrupts from modules within the embedded stripe
- 1 external source
- 6 interrupts from the PLD stripe interface as an interrupt bus (`INT_PLD`).

The six signals from the PLD can be treated in one of three modes.

- As six individual interrupts (the default mode)
- As a single interrupt request, using a six-bit priority value
- As a single interrupt request, using a five-bit priority value together with one individual interrupt

Embedded Stripe-to-PLD Interface Clocks

Up to six clocks can be used in the exchange of data between the embedded stripe and the PLD. Each clock can define a unique clock domain. The clock interfaces include:

- Stripe-to-PLD bridge `MASTER_HCLK`
- PLD-to-stripe bridge `SLAVE_HCLK`
- Dual-port SRAM clocks—up to four

Programmable Logic Architecture

The embedded stripe interfaces with a programmable logic architecture similar to that of an APEX 20KE device. APEX 20KE devices are designed with MultiCore architecture, which combines the strengths of LUT-based and product term- based devices with an enhanced memory structure. LUT-based logic provides optimized performance and efficiency for data-path, register intensive, mathematical, or digital signal processing (DSP) designs. Product-term-based logic is optimized for complex combinatorial paths, such as complex state machines. LUT- and product-term-based logic, combined with memory functions and a wide variety of MegaCore and AMPP functions, make the APEX 20KE-like device architecture uniquely suited for system-on-a-programmable-chip (SOPC) designs.

Applications that historically require a combination of microprocessor, LUT-, product-term-, and memory-based devices can now be integrated into one Excalibur device. The programmable logic of Excalibur devices includes features such as advanced I/O standard support, CAM, multiple global clocks, and ClockLock and ClockBoost clock circuitry.

Technical Description

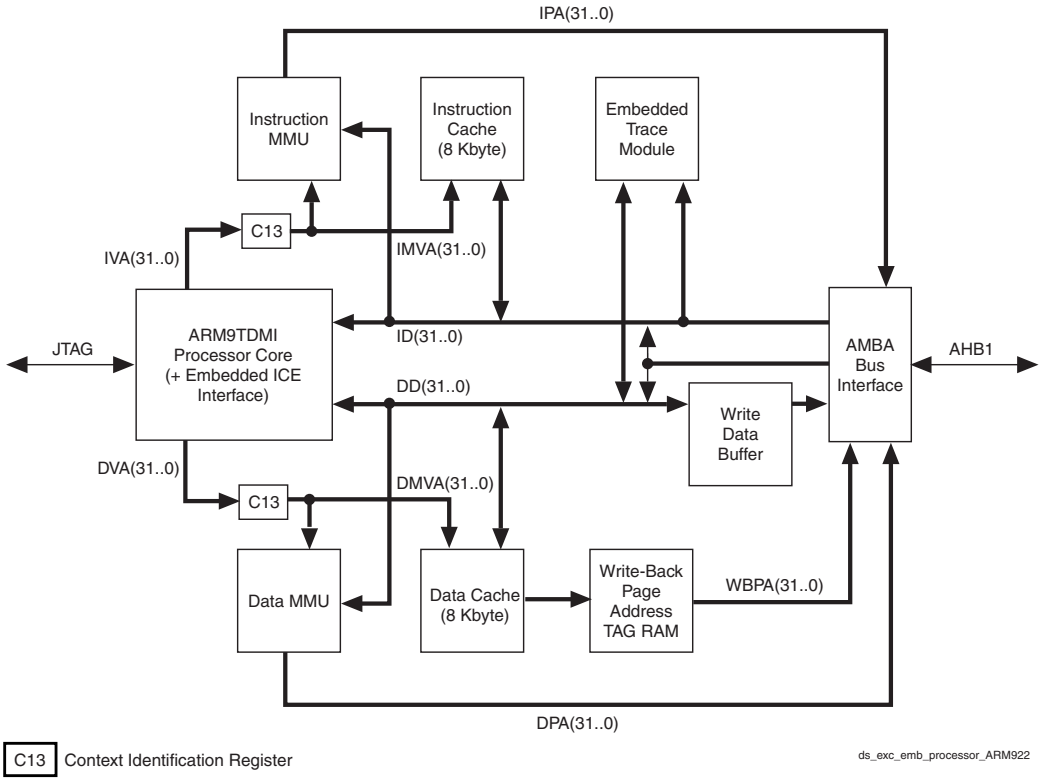
Embedded Processor

The ARM922T is the embedded processor used in the Excalibur family. It supports the 32-bit ARM and 16-bit Thumb instruction sets, which allow users to balance high performance and high code-density. The ARM922T implements an enhanced ARM Architecture V4 MMU, to provide translation and access permission checks for instruction and data addresses.

For detailed information, see the *ARM922T Technical Reference Manual.*

Figure 3 on page 23 shows the internal layout of the ARM922T embedded processor.

Figure 3. ARM922T Embedded Processor Internal Organization



Cache

The embedded processor has separate eight-Kbyte instruction and data caches, each with an eight-word line length. The caches have the following features:

- Virtually-addressed 64-way associative cache
- Write-through and write-back cache operation, selected by memory region
- Eight words per line, with one valid bit and two dirty bits per line, allowing half-word write-backs
- Selectable pseudo-random or round-robin replacement
- Independently-lockable caches, with granularity of 1/64th of cache
- Four-word write buffer, with four addresses

Transfer Types

The embedded processor supports the following AHB transfer types:

- INCR
- INCR4
- INCR8

Locked single-word transfers are used to support swap instructions. INCR4 and INCR8 transfer types are used to support cache operations.

Internal Coprocessors

The embedded processor contains two internal coprocessors:

- CP14—For debug control
- CP15—For memory system control

There is no provision for users to connect additional coprocessors directly to the core. For further details on CP14 or CP15, refer to the *ARM922T Technical Reference Manual*.

Clocking Modes

The ARM922T embedded processor has two functional clock inputs:

- BCLK
- FCLK

Both clock inputs are directly connected to the AHB1 clock. The ARM922T implements fast bus mode. The user cannot modify these parameters.

For further details about clocking modes and the embedded processor clock inputs, see the *ARM922T Technical Reference Manual*.

Endianness

The ARM9T processor supports big- or little-endian data, but defaults to little-endian. The endianness of the embedded stripe is set by writing to coprocessor register CP15; see the *ARM922T Technical Reference Manual* for details.

Debug Support

The ARM922T processor core includes embedded in-circuit emulator (ICE) logic to provide software debug support. EmbeddedICE provides breakpoint and watchpoint registers that are capable of halting the processor in response to specific events. Access to the embeddedICE registers is via JTAG scan chains.

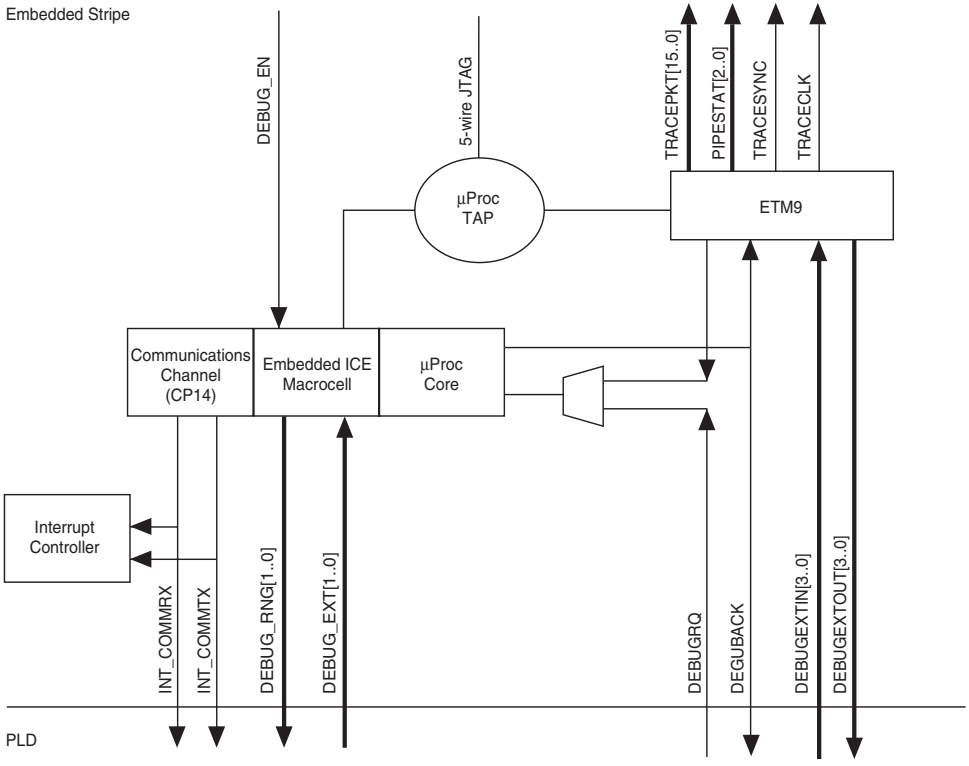
In addition to the breakpoint and watchpoint registers, the embeddedICE logic contains a communication unit, which allows software running on the embedded processor to communicate with a host via JTAG. The embedded processor accesses these registers through MRC and MCR instructions to coprocessor CP14. Host software is required to translate the data. To manage communications between the embedded processor and the control register status bits effectively, `COMMRX` and `COMMTX` are routed from CP14 to the interrupt controller and to the PLD as `INT_COMMRX` and `INT_COMMTX`, respectively.

Trace Support

In EPXA10 and EPXA4 devices, the ARM9 Embedded Trace Macrocell (ETM9) provides additional capabilities for observing embedded stripe operations in real-time and at full operational bus speeds. The ETM9 monitors the address, data and control signals and reports compressed information off-chip to the trace port interface. The ETM9 used in EPXA10 devices is version 1 and in EPXA4 devices is version 2a.

Figure 4 shows how the ETM9 connects directly to the trace interface on the embedded processor. Third-party development tools are required to interpret the information.

Figure 4. ETM Connections to the Embedded Processor



The ETM9 can be enabled and disabled, and the trace port width is determined by PORTSIZE. The maximum size of the trace port is 16 bits. When the debug session starts, the debug tools control ETMEN and PORTSIZE by programming the ETM control register.

External Inputs and Outputs

The ETM9 provides four input and four output signals to the PLD for control and operation by trace debug tools. The four inputs, DEBUG_EXTIN[3..0], are routed from the PLD and provide coarse enable/disable tracing control as trigger or enable signals. Signals are synchronized to the ETM9 clock by two registers.

Four outputs, DEBUG_EXTOUT[3..0], are routed to the PLD. Each is controlled by an ETM9 event, which can be programmed through the trace debug tools. The trace debug tools and the user logic are responsible for these signals.

Table 7 lists the debug signals, which are routed between the embedded processor and the PLD.

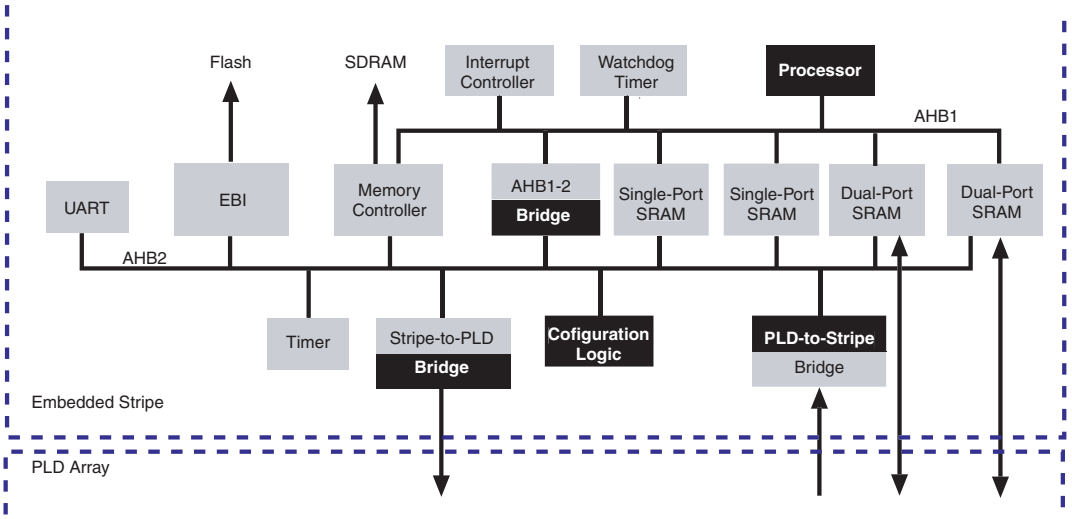
Table 7. Debug Signals

Signal	Source	Description
DEBUG_EN	Input	When high, allows the embedded processor to enter debug mode. When low, prevents the embedded processor from entering debug mode and enables the hardware trigger on the watchdog
DEBUG_RQ	PLD	Forces the embedded processor into debug mode
DEBUG_EXT0, DEBUG_EXT1	PLD	These inputs are matched by the breakpoint/watchpoint unit in the embedded processor, and matches are reported on the DEBUG_RNGx signals
DEBUG_ACK	Stripe	Indicates when the embedded processor is stopped in debug mode. This can be used to stop devices within the PLD when the embedded processor hits a breakpoint
DEBUG_RNG0, DEBUG_RNG1	Stripe	Indicates when the breakpoint/watchpoint unit has found a match (whether enabled or not). The signal is valid for at least one PLD clock cycle
DEBUG_EXTIN[3..0]	PLD	Trace port input signals
DEBUG_EXTOUT[3..0]	Stripe	Trace port output signals

Bus Architecture

The bus architecture of the Excalibur family conforms to AHB specifications, as detailed in the *AMBA Specification, Revision 2.0*. Figure 5 illustrates the Excalibur embedded processor stripe.

Figure 5. Excalibur Embedded Processor Stripe



The Excalibur embedded processor PLD uses two AHBs as its communication medium, AHB1 and AHB2. Each bus has 32-bit address, read, and write data buses, which maximize access to the on-chip and off-chip memory resources, as well as to the shared peripherals.

The only bus master on AHB1 is the embedded processor. Processor-specific slaves, such as the interrupt controller, are local to AHB1. Embedded stripe memory resources such as the on-chip SRAM are also local to AHB1, which allows the embedded processor fast access to the memory.

Any transaction which, after decoding, is not intended for a peripheral on AHB1 is then routed to the AHB1-2 bridge. The AHB1-2 bridge is a slave on AHB1, which gives the embedded processor access to AHB2.

There are three bus masters on AHB2:

- **AHB1-2 bridge**—processes transactions that originate from the embedded processor on AHB1 whose destination is either on AHB2 or the PLD
- **Configuration logic**—provides configuration information to the PLD and stripe memory elements
- **PLD-to-stripe bridge**—provides masters in the PLD access to slaves in the embedded stripe.

The configuration logic is both a master and a slave on AHB2.

A priority arbitration scheme grants access to the AHB2 bus masters. AHB2 has several local slaves, such as the embedded stripe memory resources, the UART, and the stripe-to-PLD bridge, which gives AHB2 masters access to the PLD.

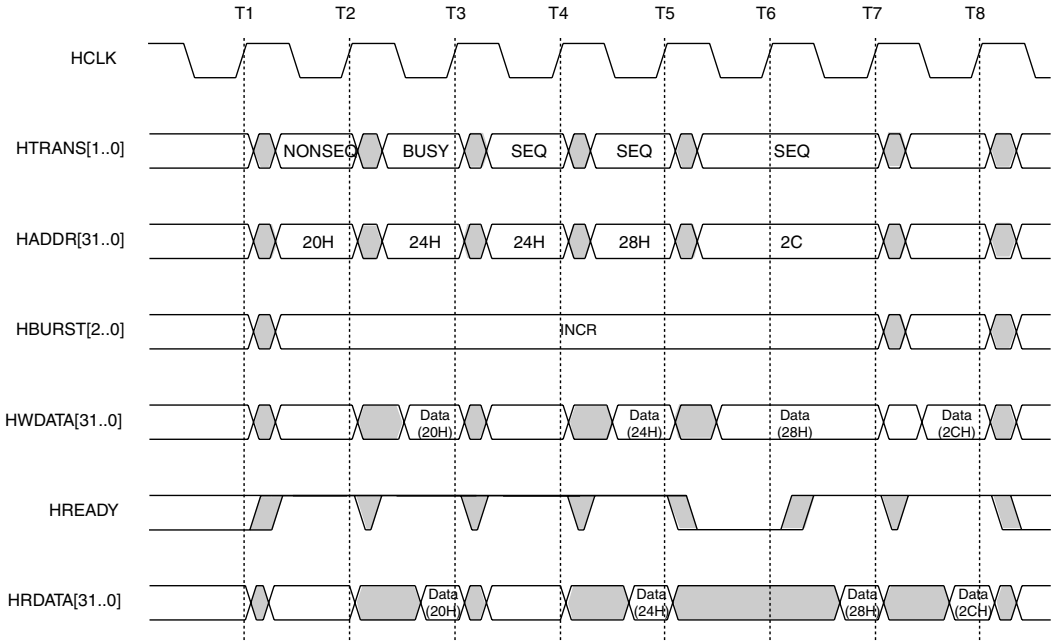
Apart from the dual-port RAM interface, which is a standard interface, the embedded stripe and the bridge interfaces to the PLD use the AHB bus standard. See the section "Embedded Stripe Bridges" on page 34 for details.

The embedded stripe supports all AMBA AHB protocols, including the following:

- Incremental bursts of 4,8,16 and unspecified length
- Wrapping bursts of length 4, 8 and 16
- Early burst termination
- `SPLIT` response on AHB2 (EBI only)
- Locked transfers

Figure 6 shows typical AHB transaction waveforms.

Figure 6. AMBA AHB Typical Transaction Waveforms

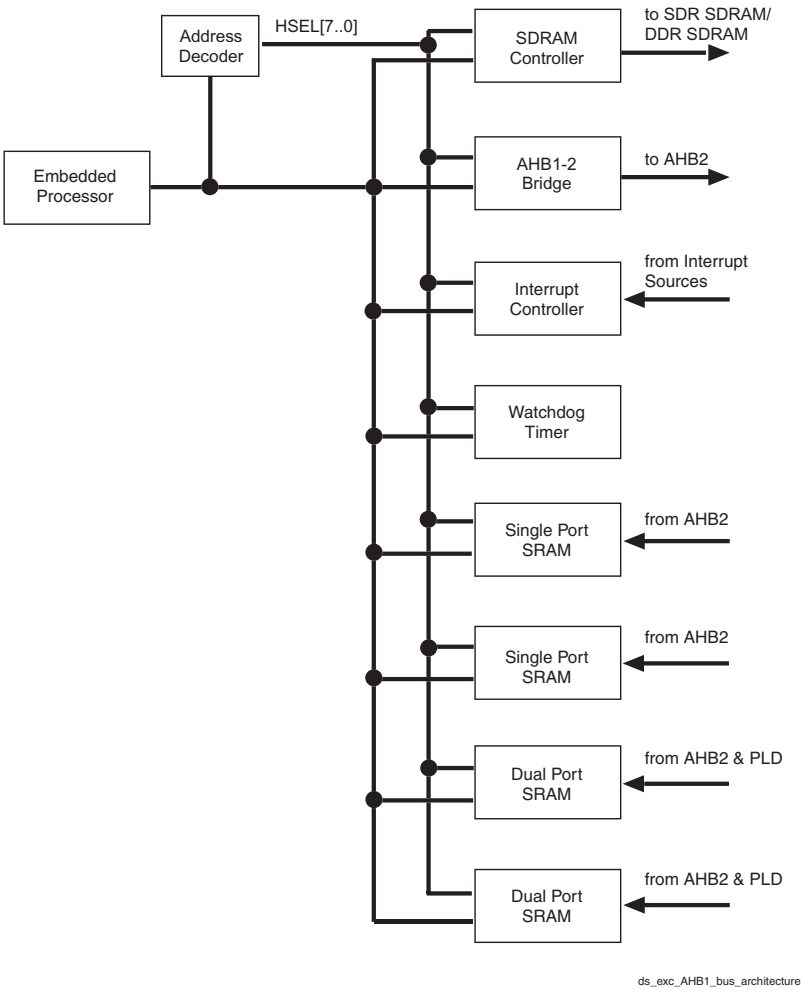


AHB1

The embedded processor is the sole bus master on AHB1. It is wait-stated by the appropriate interface if it tries to access a busy resource.

Figure 7 on page 30 shows the structure of AHB1.

Figure 7. AHB1 Architecture



The EPXA1 device has only one dual-port SRAM block.

Clock Domain

AHB1 is clocked by a dedicated PLL, which maximizes the achievable performance from the embedded processor core. The clock frequency is selected by writing to registers within the clock module. For more information about selecting clock frequency, see the section "PLLs" on page 138.

Address Decoder

The address decoder routes transactions addressed to the following slave peripherals on AHB1:

- SDRAM memory controller
- On-chip SRAM, both single- and dual-port
- Interrupt controller
- Watchdog timer

These resources have configurable base addresses, set by on-chip registers located in the mode control module on AHB2.

Bus transactions that are not recognized by the address decoder as being for AHB1 resources are sent to the AHB1-2 bridge.

Any bus transactions occurring to an overlapping memory region produce indeterminate results.

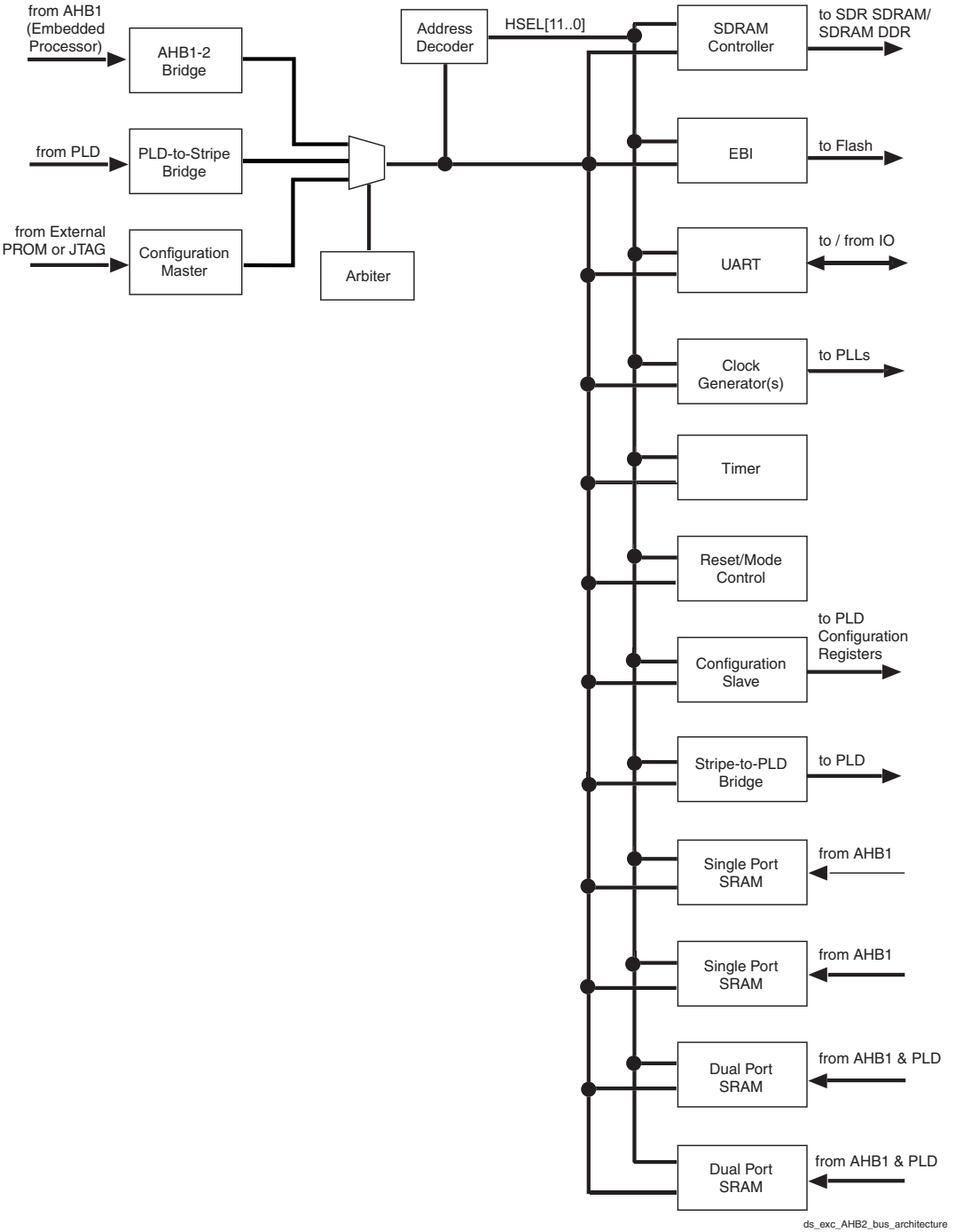
AHB2

The AHB2 has three masters:

- AHB1-2 bridge
- Configuration logic (the configuration logic exists as both a bus master and slave)
- PLD-to-stripe bridge

Figure 8 on page 32 shows the structure of AHB2.

Figure 8. AHB2 Architecture



The EPXA1 device has only one dual-port SRAM block.

AHB2 Arbiter

The AHB2 arbiter determines which AHB2 master is granted the bus, using a round-robin scheme to arbitrate contending bus requests.

Split Transactions

AHB2 supports split transactions from the EBI. Split transactions allow other masters to access the buses while a high-latency slave access, such as reading flash memory, is in progress. Split transactions are a means of improving system performance in the presence of slow peripherals.

If a transfer is likely to take a large number of cycles to perform, the EBI can issue a split response. On receiving a split response, the arbiter de-asserts the bus grant to the requesting master, and masks further requests from that master until the EBI removes the split. This denies the master access to the bus until the EBI is ready to complete the transfer. Other masters, however, are free to use the bus. For example, if the embedded processor sends a transaction to the EBI, the AHB1-2 bridge receives the split response from the EBI. At that point the PLD-to-stripe bridge can gain access to AHB2 (for example, if a master in the PLD sends a transaction targeted to a slave on AHB2). See "Expansion Bus Interface" on page 83 for additional details on split transactions.

Locked Transactions

AHB2 allows locked transfers. Masters request locked access to memory by asserting `MASTER_HLOCK` at the same time as `MASTER_HBUSREQ`. In this situation, the bus remains granted to the master until `MASTER_HLOCK` is de-asserted.

Clock Domain

The AHB1 clock is divided by 2 to derive the AHB1 clock. For more information about selecting clock frequencies, see the section "PLLs" on page 138.

The first read or write access to the embedded stripe bridges should be performed one clock cycle after reset to ensure valid data. Subsequent accesses do not require a delay.

Address Decoder

The address decoder routes transactions addressed to the following devices:

- SDRAM memory controller
- EBI, including the flash memory interface
- On-chip SRAM, both single- and dual-port
- UART
- Embedded stripe bridges
- Timer
- Reset/mode control
- Configuration logic (slave port)
- Stripe-to-PLD bridge (see the section "Embedded Stripe Bridges" below)

All AHB2 resources have configurable base addresses which are set by on-chip registers located within the reset and mode control module. See "Memory Map" on page 19 for details.

Any bus transactions occurring to an overlapping memory region produce indeterminate results.

Bus transactions that are not recognized by the address decoder as being for AHB2 resources are sent to the default slave resource.

Default Slave

The default slave device is part of the address decoder. Transactions for which the addresses are invalid are routed to the default slave device, which provides a two-cycle `ERROR` response to all signals (`SEQ` or `NONSEQ`).

Embedded Stripe Bridges

There are three AHB bridges in the embedded stripe:

- AHB1-2
- PLD-to-stripe
- Stripe-to-PLD

AHB1-2 Bridge

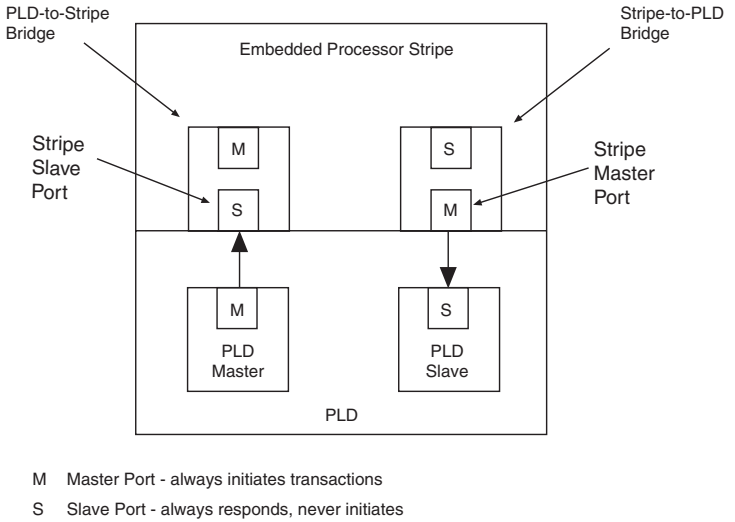
AHB1-2 bridge is the interface that provides the embedded processor, the sole master on AHB1, access to AHB2.

Because AHB1 and AHB2 are synchronous to each other, synchronization logic is not required in the AHB1-2 bridge

PLD-to-Stripe and Stripe-to-PLD Bridges

The PLD-to-stripe and stripe-to-PLD bridges provide the interface between the PLD and the embedded processor stripe. They provide AHB master and slave interfaces to the PLD. The embedded stripe bridges are functionally similar, with minor differences in the decoding scheme and bus structure. The block diagram in Figure 9 highlights the access ports of the bridges.

Figure 9. PLD-to-Stripe and Stripe-to-PLD Bridges



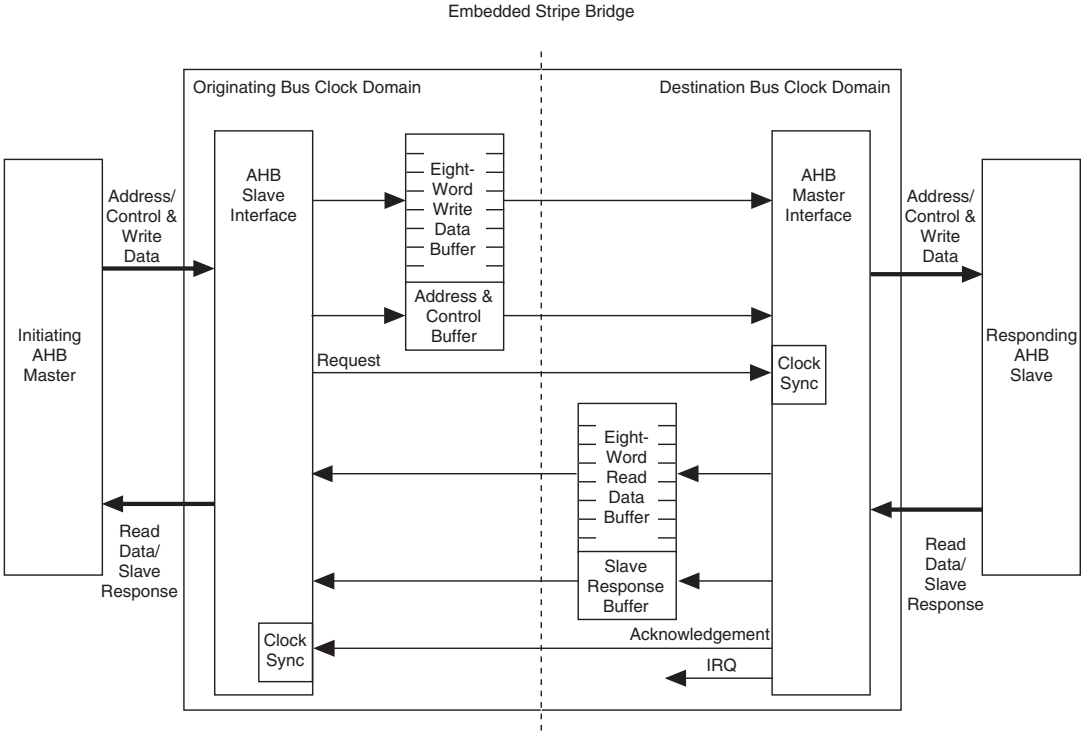
The PLD-to-stripe bridge allows masters in the PLD to access resources in the embedded stripe (that is, SDRAM, EBI, etc.). It is accessed via the slave port on the embedded stripe and appears to the user as a conventional AMBA AHB slave.

The stripe-to-PLD bridge allows bus masters in the stripe to access any slaves in the PLD. It provides a master port on the stripe and appears to the user as a conventional AMBA AHB master.

The PLD-to-stripe bridge and stripe-to-PLD bridge include synchronization logic, allowing the master and slave interfaces to reside in a different clock domain from the AHB2 clock domain.

Figure 10 shows the major functional blocks of each AHB bridge. The originating bus of a transaction is connected to the bridge's slave port. The bridge's master port is connected to the destination bus.

Figure 10. Functional Blocks of an AHB Bridge



AHB Bridge Operation

This section describes how the bridges process write and read transactions.

Read/Write Transactions

For a read or a write transaction, the AHB slave interface passes the address and control information from the initiating master, plus a transaction processing request, to the AHB master interface. The AHB master interface then synchronizes the transaction to its clock domain. The AHB slave interface inserts wait states if the AHB master interface is not granted access to the destination bus. When it is granted the bus, the AHB master presents the transaction to the slave on the destination bus, which accepts the transaction and issues a response according to the AHB protocol. The AHB master interface passes back the response from the slave to the AHB slave interface, which resynchronizes the response to its clock domain and sends an acknowledgement indicating that the bridge is ready to process another transaction. The initiating master on the originating bus then samples the response from the AHB slave interface and responds accordingly.

Write-Posting

To increase bridge throughput, the embedded stripe bridges support write posting. Write posting allows an initiating master to post its transaction to the bridge and not have to wait for a response from the slave. Subsequently, the initiating master samples the response to the transaction from the AHB slave interface and not from the slave itself.

For a posted-write transaction, the AHB slave interface copies the address and control information into the address and control buffer, and sends a request to the AHB master interface. While the request is outstanding, the AHB slave interface continues to accept bursts of data while there is room in its buffer. When the buffer becomes full, the AHB slave inserts wait states until the buffer is ready to accept further data. Upon receiving the request, the AHB master interface requests the bus, synchronizes the transaction to the master domain, and then regenerates the burst's address and control information on the destination bus. The AHB master interface samples the response from the responding slave on the destination bus, and if an `ERROR` response occurs, it generates an interrupt. The bridge status registers hold information on the transaction that caused the error, to allow the system to take appropriate action to reinstate the integrity of the system.

The `NW` bit in the appropriate bridge control register controls write posting. See "Embedded Stripe Registers" on page 43 for more information.

Read Pre-Fetching

To increase bridge throughput, the embedded stripe bridges support read prefetching. Read pre-fetching occurs in situations where the AHB master interface cannot determine the exact amount of data in a burst (i.e., an unspecified length burst is under way). The AHB master interface continues to fill the read buffer until it is full, anticipating that the data will be needed. When a new transaction begins, the data is no longer valid.

Prefetching should not be used where reads have side effects.

The NP bit in the appropriate bridge control register controls read pre-fetching. See "Embedded Stripe Registers" on page 43 for more information.

Read/Write Buffers

Read pre-fetch and write-posting buffers in all bridges are eight entries deep, where the size of an entry is either byte, half-word or word, depending on the transaction. There is also one location for address and control information in the write buffer and one for slave response information in the read buffer.

Error Handling

Behavior following a bus error depends on the type of transaction that caused it, as detailed below.

Read Transactions

During read transactions, error responses on the destination bus are always passed back to the originating bus, while the bridge status remains unchanged. The master on the originating bus only detects the error if it reads the associated data beat. For example, if the master starts an undefined-length burst to read two beats of data, an error response on the third or subsequent pre-fetched beats is discarded.

Write Transactions

Error responses during a non-posted write transaction are passed back to the originating bus, while the bridge status remains unchanged.

Posted-Write Error Responses

During a posted-write transaction, an error response on the destination bus is never passed back to the originating bus. All beats of the transaction are written, so as to keep the bridge master and slaves in step. The address and transaction type of the error beat are logged in the bridge status registers, with the write-failure bit, WF, set, and the bridge's error interrupt is asserted.

If an error occurs before a previous error status has been cleared, it is not logged, because the bridge status registers only capture the first error to occur.

Retry Responses

If a transaction terminates with a retry response on the destination bus during a read or write, the bridge attempts to retry the transaction from the address of the beat that was terminated, and the burst-type encoding is modified as necessary.

Bridge Address Decoding

The EBI and PLD address regions have read/write no-prefetch bits in their range definition registers, which control read transaction pre-fetching. All other regions are hard wired to be pre-fetchable, except for the registers region, which is never pre-fetchable.

Read pre-fetching and write-posting can be disabled globally in each bridge.

The PLD-to-Stripe Bridge Slave Interface

The AHB slave interface is the standard AHB interface specified in the *AMBA Specification*, with exception of SLAVE_HSIZE[1..0]. SLAVE_HSIZE, on the embedded stripe, is only a 2-bit signal instead of 3-bit signal, because the embedded stripe data bus is a 32-bit data bus, which means that the most-significant bit of SLAVE_HSIZE is not needed. In addition, SLAVE_HSELREG is a chip-select pin which, when asserted, allows PLD masters to access the PLD-to-stripe bridge status register and the PLD-to-stripe bridge address status register. See "Embedded Stripe Registers" on page 43 for details.

Figure 11 on page 40 shows the PLD-to-stripe bridge.

Figure 11. PLD-to-stripe Bridge

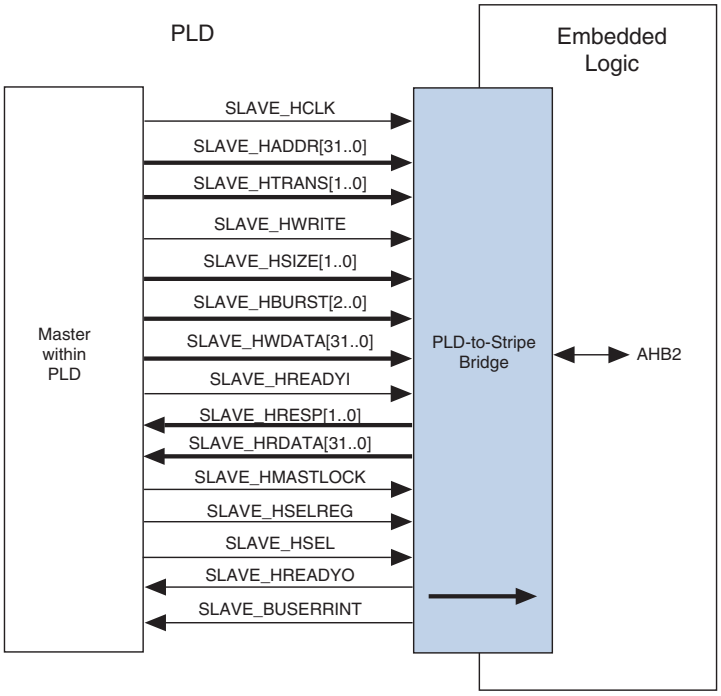


Table 8 summarizes the PLD-to-stripe bridge signals.

Table 8. PLD-to-Stripe Bridge Signals (Part 1 of 2) Note (1)

Signal	Source	Description
SLAVE_HCLK	PLD	PLD-to-stripe bridge slave port clock that times all bus transfers. Signal timings are related to its rising edge clock
SLAVE_HADDR[31..0]	PLD	PLD-to-stripe bridge 32-bit system address bus
SLAVE_HTRANS[1..0]	PLD	PLD-to-stripe bridge transfer type
SLAVE_HWRITE	PLD	When high, indicates a write transfer on the PLD-to-stripe bridge; when low, a read transfer
SLAVE_HSIZE[1..0]	PLD	Indicates the size of transfer on the PLD-to-stripe bridge
SLAVE_HBURST[2..0]	PLD	Indicates whether the transfer forms part of a burst
SLAVE_HWDATA[31..0]	PLD	Used to transfer data from the master to the bus slaves during writes across the PLD-to-stripe bridge
SLAVE_HREADYI	PLD	When high, indicates that a transfer has finished on the bus. Slaves on the bus need SLAVE_HREADY as both an input and output signal

Table 8. PLD-to-Stripe Bridge Signals (Part 2 of 2) Note (1)		
Signal	**Source**	**Description**
SLAVE_HREADYO	Stripe	When high, indicates that a transfer has finished on the bus. Slaves on the bus need SLAVE_HREADY as both an input and output signal
SLAVE_HRESP[1..0]	Stripe	Master response that provides additional information on the status of a transfer across the PLD-to-stripe bridge
SLAVE_HRDATA[31..0]	Stripe	Used to transfer data from bus slaves to the master during reads across the PLD-to-stripe bridge
SLAVE_HMASTLOCK	PLD	When high, indicates that the master requires locked access to the bus
SLAVE_BUSERRINT	Stripe	Interrupt signal signifying a bus error
SLAVE_HSELREG	PLD	PLD-to-stripe bridge register selection signal
SLAVE_HSEL	PLD	PLD-to-stripe bridge interface chip-selection signal

Note:
(1) For further details, refer to the *AMBA Specification, Revision 2.0*.

Stripe-to-PLD Bridge Master Interface

The stripe-to-PLD bridge master interface is the standard AHB interface specified in the *AMBA Specification*, with the exception of MASTER_HSIZE. As with the slave interface, MASTER_HSIZE is a 2-bit signal instead of a 3-bit signal.

Figure 12 on page 42 shows the stripe-to-PLD bridge.

Figure 12. Stripe-to-PLD Bridge

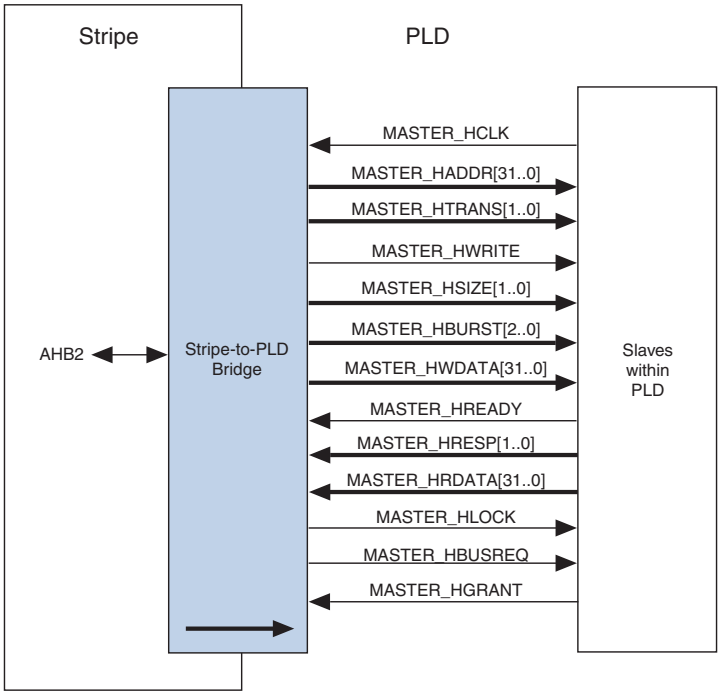


Table 9 summarizes the stripe-to-PLD bridge signals.

Table 9. Stripe-to-PLD Bridge Signals (Part 1 of 2) *Note (1)*

Signal	Source	Description
MASTER_HCLK	PLD	Stripe-to-PLD bridge slave-port clock that times all bus transfers.Signal timings are related to its rising edge clock. This signal is invertible
MASTER_HADDR[31..0]	Stripe	Stripe-to-PLD 32-bit system address bus
MASTER_HTRANS[1..0]	Stripe	Stripe-to-PLD bridge transfer type
MASTER_HWRITE	Stripe	When high, indicates a write transfer on the stripe-to-PLD bridge; when low, a read transfer
MASTER_HSIZE[1..0]	Stripe	Indicates the size of transfer on the stripe-to-PLD bridge
MASTER_HBURST[2..0]	Stripe	Indicates whether the stripe-to-PLD bridge transfer forms part of a burst
MASTER_HWDATA[31..0]	Stripe	Used to transfer data from the master to the bus slaves during writes across the stripe-to-PLD bridge
MASTER_HREADY	PLD	When high, indicates that a stripe-to-PLD bridge transfer has finished
MASTER_HRESP[1..0]	PLD	Slave response that provides additional information on the status of a transfer across the stripe-to-PLD bridge

Table 9. Stripe-to-PLD Bridge Signals (Part 2 of 2) Note (1)

Signal	Source	Description
MASTER_HRDATA[31..0]	PLD	Used to transfer data from bus slaves to the master during reads across the stripe-to-PLD bridge
MASTER_HLOCK	Stripe	When high, indicates that the master requires locked access to the bus
MASTER_HBUSREQ	Stripe	Bus request signal, from the master to the arbiter
MASTER_HGRANT	PLD	Bus grant signal that, in conjunction with MASTER_HREADY, indicates that the bus master has been granted the bus

Note:
(1) For further details, refer to the *AMBA Specification, Revision 2.0*.

General Purpose I/O Port

A parallel I/O port allows the embedded stripe to interface directly to the PLD without having to use the stripe-to-PLD bridge). The port is not available in EPXA10 devices. In EPXA4 devices, it comprises eight inputs from the PLD and eight outputs to the PLD. In EPXA1 devices, it comprises four inputs from the PLD and four outputs to the PLD.

Outputs are driven to the PLD directly from the GP_IO register. Inputs from the PLD are synchronized to the AHB2 clock before becoming available in the GP_IO register.

Embedded Stripe Registers

The bridge-control registers are located in the bridge-control module. Control signals are routed from there to each bridge, which allows all bridge-control registers to be accessed from the embedded processor. Any PLD master can access the PLD-to-stripe, stripe-to-PLD and AHB1-2 bridge-control registers via the slave port of the PLD-to-stripe bridge.

Bridge status registers capture status information for posted writes that result in an error response. They are located in the bridge-control module, apart from the AHB1-2 status registers (AHB12B_SR and AHB12B_ADDRSR) which are in the AHB1-2 bridge-control module.

The PLD-stripe bridge status registers are accessible only from the PLD. If the embedded processor needs to be able to access them, it must do so via the PLD slave port and appropriate circuitry within the PLD. The AHB1-2 and stripe-PLD bridge registers are directly accessible by the processor.

Table 10 shows the effect of a read or a write to the embedded stripe registers.

Table 10. Effects of Accessing Registers

R	Read access (no side effects)
R*	Read access with possible side effects (such as clearing some of the bits or clearing an interrupt)
W	Writes of 1 or 0 set writable bits to the values specified
S	Writes of 1 set bits. Writes of 0 do nothing.
C	Writes of 1 clear the appropriate bits. Writes of 0 do nothing.

All registers are reset to 0, unless otherwise indicated.

Register: AHB12B_CR (AHB1-2 bridge-control register)
Address: Register base + 100H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																														NW	NP

NP	R/W	When set, prevents the pre-fetching of read transactions via this bridge
NW	R/W	When set, prevents the posting of write transactions via this bridge
0	R	Reserved for future use. Write as 0 for future compatibility

Register: AHB12B_SR (AHB1-2 bridge-status register)
Address: Register base + 800H
Access: Read/clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																							HBURST			HSIZE			HTRN	WF	

WF	R/C	A bus error has been received by this bridge while executing a posted write
HTRN	R	The value of the HTRANS[1..0] bits for the write transaction which is causing the error
HSIZE*(1)*	R	The value of the HSIZE[1..0] lines for the write transaction which is causing the error
HBURST	R	The value of the HBURST[2..0] lines for the write transaction which is causing the error
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Note:
(1) Although this is a 3-bit field, the maximum value it contains is 2, signifying a 32-bit bus width.

An interrupt is generated whenever the write-failure bit, WF, is set. Writing 1 to WF clears WF and the interrupt. If a further error occurs while WF is set, the contents of the register do not change.

Register: AHB12B_ADDRSR (AHB1-2 bridge address status register)
Address: Register base + 804H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HADDR																															

HADDR	R	The value of the HADDR[31..0] lines for the write causing the error

This register is associated with the AHB12B_SR register.

Register: PLDSB_CR (stripe-to-PLD bridge-control register)
Address: Register base + 110H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																														NW	NP

NP	R/W	When set, prevents the pre-fetching of read transactions via this bridge
NW	R/W	When set, prevents the posting of write transactions via this bridge
0	R	Reserved for future use. Write as 0 for future compatibility

Register: PLDSB_SR (stripe-to-PLD bridge status register)
Address: Register base + 118H
Access: Read/clear

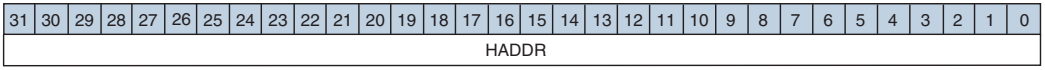
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																							HBURST			HSIZE			HTRN	WF	

WF	R/C	A bus error has been received by this bridge while executing a posted write
HTRN	R	The value of the HTRANS[1..0] bits for the write causing the error
HSIZE*(1)*	R	The value of the HSIZE[1..0] lines for the write causing the error
HBURST	R	The value of the HBURST[2..0] lines for the write causing the error
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Note:
(1) Although this is a 3-bit field, the maximum value it will contain is 2, signifying a 32-bit bus width.

An interrupt is generated whenever the write-failure bit, WF, is set. Writing 1 to WF clears WF and the interrupt. If a further error occurs while WF is set, the contents of the register do not change.

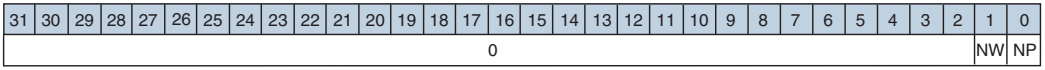
Register: PLDSB_ADDRSR (stripe-to-PLD bridge address-status register)
Address: Register base + 114H
Access: Read



HADDR	R	The value of the HADDR[31..0] lines for the write causing the error

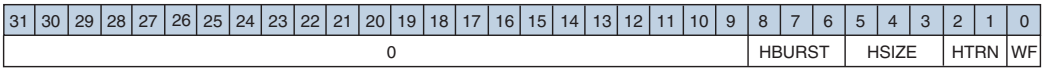
This register is associated with PLDSB_SR.

Register: PLDMB_CR (PLD-to-stripe bridge-control register)
Address: Register base + 120H
Access: Read/write



NP	R/W	When set, prevents the pre-fetching of read transactions via this bridge
NW	R/W	When set, prevents the posting of write transactions via this bridge
0	R	Reserved for future use. Write as 0 for future compatibility

Register: PLDMB_SR (PLD-to-stripe bridge status register)
Address: PLDMB_SR can only be accessed from the slave interface on the PLD side of the bridge when SLAVE_HSELREG is 1 and SLAVE_HADDR is 0. It has no entry in the AHB1 or AHB2 address maps
Access: Read/clear



WF	R/C	A bus error has been received by this bridge (while executing a posted write)
HTRN	R	The value of the HTRANS[1..0] bits for the write causing the error
HSIZE*(1)*	R	The value of the HSIZE[1..0] lines for the write causing the error
HBURST	R	The value of the HBURST[2..0] lines for the write causing the error
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Note:
(1) Although this is a 3-bit field, the maximum value it will contain is 32, signifying a 32-bit bus width.

An interrupt is generated whenever the write-failure bit, WF, is set. Writing 1 to WF clears WF and the interrupt. If a further error occurs while WF is set, the contents of the register do not change.

Register: PLDMB_ADDRSR (PLD-to-stripe bridge address status register)
Address: PLDMB_ADDRSR can only be accessed from the slave interface on the PLD side of the bridge when SLAVE_HSELREG is 1 and SLAVE_HADDR is 4. It has no entry in the AHB1 or AHB2 address maps
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HADDR																															

HADDR	R	The value of the HADDR[31..0] lines for the write causing the error

This register is associated with PLDMB_SR.

Register: GP_IO (EPXA4 and EPXA1 only)
Address: Register base + 150H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								GPI								0								GPO							

GPO	R/W	Outputs to the PLD. On reset, these are set to 0. For EPXA4—GPI[7..0] For EPXA1—GPI[3..0]
GPI	R	Inputs from the PLD. If the PLD is unconfigured, the value of these bits is indeterminate. For EPXA4—GPO[7..0] For EPXA1—GPO[3..0]
0	R	Reserved for future use. Write as 0 to ensure future compatibility

On-Chip SRAM

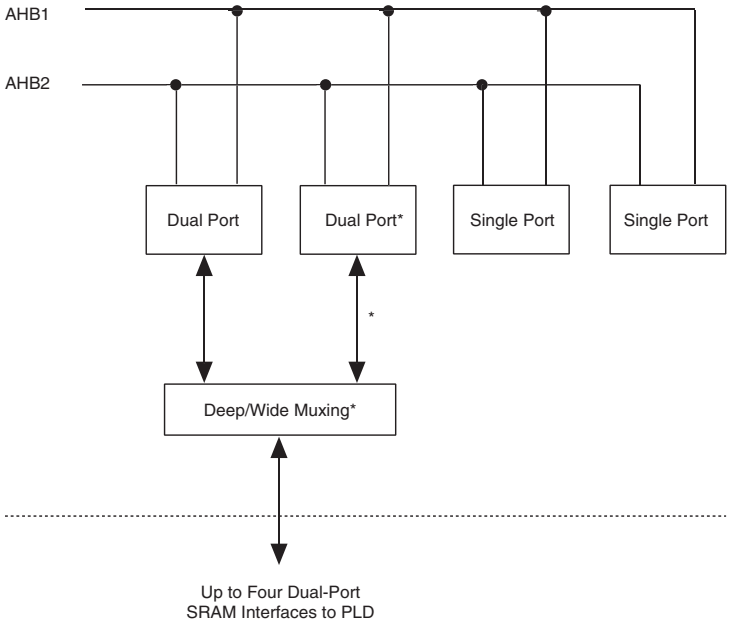
There are two blocks of single-port SRAM. Both are accessible to the AHB masters (AHB1 and AHB2) via an arbitrated interface within memory. Each block is independently arbitrated, allowing one block to be accessed by one bus master while the other block is accessed by the other bus master.

In addition, EPXA4 and EPXA10 devices also have two blocks of dual-port SRAM (DPSRAM), and EPXA1 devices have one. The dual-port SRAM is accessible to the PLD and can be configured for access by the AHB masters. The outputs of the dual-port memories can be registered if appropriately configured.

It is possible to build deeper and wider memories by using both dual-ports and multiplexing the data outputs within the stripe, although this capability is not supported on EPXA1 devices.

Figure 13 on page 48 shows the arrangement of on-chip SRAM.

Figure 13. On-Chip SRAM Structure



* Not available in EPXA1 devices

The contents of on-chip SRAM, whether single-port or dual-port, are not guaranteed after a reset.

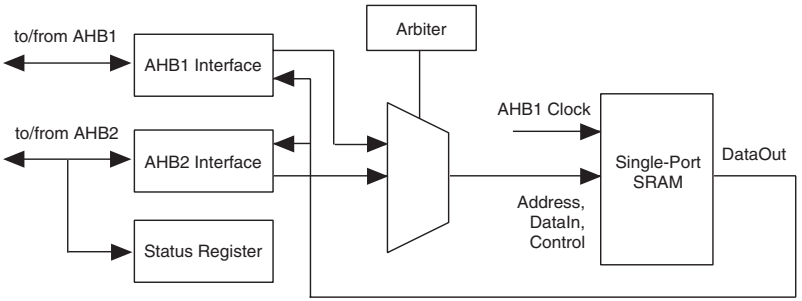
Single-Port SRAM

Each single-port SRAM block incorporates the interfaces to the AHB1 and AHB2 buses, plus the appropriate synchronization logic. The arbiter determines which bus has current access and the appropriate signal multiplexing.

The SRAM is clocked by the AHB1 clock. The input address, input data, write enable, and chip enable signals are latched by the rising edge of the clock. In read mode, data on the output bus is read from the memory location specified by the captured address.

Figure 14 on page 49 shows the layout of the single-port SRAM.

Figure 14. Single-Port SRAM Block Structure



Up to 256 Kbytes of single-port SRAM are available, as two blocks of 2 × 128 Kbytes. Each single-port SRAM block is byte-addressable. The size of the SRAM blocks depends on the device, as shown in Table 11.

Table 11. SRAM Block Sizes

Device	Block 0 Size (KBytes)	Block 1 Size (KBytes)
EPXA1	16	16
EPXA4	64	64
EPXA10	128	128

Byte, half-word and word accesses are allowed and are enabled by the slave interface. The behavior of byte and half-word reads is controlled by the system endianness.

Arbiter

The arbiter is synchronous and clocked by the AHB1 clock. It resolves competition between the two requesting interfaces, AHB1 and AHB2, for access to the SRAM. It also provides support for AHB locked transfers.

The arbitration algorithm is a fixed round-robin scheme with fairness. This means that, if several masters are requesting access to the bus, the master least-recently granted is given access. Having granted a bus, the arbiter cannot rescind it but maintains the current grant until the appropriate request line has been de-asserted, indicating the end of a transfer. At this point the arbiter selects the next master to be granted access to the block.

Locked Transfers

Asserting a locking signal to the arbiter indicates that the requesting master requires back-to-back accesses for the duration of the assertion of the lock signal. Consequently, the arbiter maintains the grant until the locking signal is removed and the transfer has ended.

Bus Interfaces

When selected, the AHB1 interface provides a single-cycle response to transactions. An initial wait state can be inserted while the arbiter acknowledges the request for access to the SRAM. All AHB transaction types are supported.

The AHB2 interface synchronizes transfers between the AHB2 clock domain and the SRAM clock domain (AHB1). It provides a slave interface to the AHB2 buses, and outputs the required SRAM signals together with an interface to the arbiter.

Read Accesses

Read operations are held in a wait state while waiting for the arbiter and for the first cycle of the transfer. One word (or less) of data is returned in each subsequent cycle.

Write Accesses

Write operations are held in a wait state while waiting for the arbiter. One word (or less) of data is written in each subsequent cycle.

Registers

Register: SRAM0_SR
Address: Register base + 20H
Access: Read

This register has the same layout as SRAM1_SR.

Register: SRAM1_SR
Address: Register base + 24H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIZE																				0											

SIZE	R	Memory block size in bytes (e.g. 32,768 bytes). The smallest-possible size is 8 Kbytes
0	R	Reserved for future use. Write as 0 to ensure future compatibility

SRAMx_SR contains the block size, in bytes, of the appropriate block of RAM.

Endianness

The single-port SRAM block supports little- or big-endian transfers as appropriate.

Dual-Port SRAM

Up to 128 Kbytes of dual-port SRAM are available, conventionally arranged as either two 64-Kbyte blocks or four 32-Kbyte sub-blocks. Alternatively, the DPSRAM blocks can be combined for wide- or deep-mode processing, as explained in "Wide/Deep Multiplexing" on page 64. In addition, each 64-Kbyte block of DPSRAM can be configured independently of the other, giving many permutations of block configurations.

Each of the two 64-Kbyte blocks has the following characteristics:

- It is byte-addressable from AHB interfaces
- It has a PLD interface capable of configuration to support various widths and depths

When a block is split into two 32-Kbyte sub-blocks, they both share the same clock.

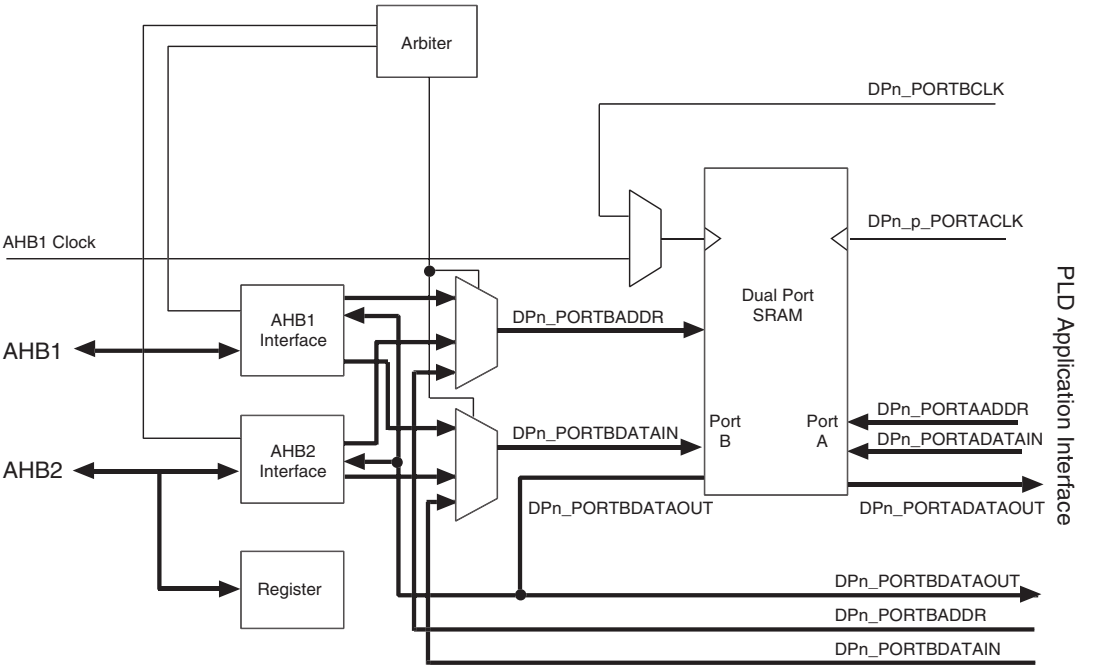
The dual-port SRAM incorporates interfaces to the AHB1 and AHB2 buses, plus the appropriate synchronization logic. The arbiter determines which bus has current access and supplies the appropriate signal multiplexing.

Memory access from the PLD is synchronous and is triggered by the rising edge of the clock, as are the input address, input data, write enable, and chip enable signals. In read mode, data on the output bus is read from the memory location specified by the captured address. In EPXA4 and EPXA1 devices, the first read or write access to the dual-port SRAM after a reset should be performed one PLD clock cycle after the initial clock cycle to ensure valid data.

For more information on configuring the stripe dual-port SRAM, refer to *Application Note 173: Excalibur Solutions—DPRAM Reference Design*.

Figure 15 on page 52 shows the layout of the dual-port SRAM.

Figure 15. Dual-Port SRAM Block



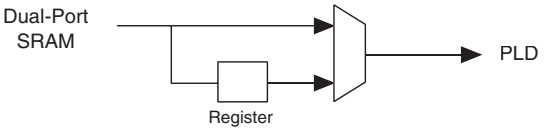
The A port of the dual port is always dedicated to the PLD application interface. It is clocked by the DP*n*_*p*_PORTACLK signal from the PLD, where *n* (0 or 1) and *p* (2 or 3) identify the block.

The B port can either be accessed from the AHB1 and AHB2 buses or from the PLD. This is selectable by configuration.

A read-only register, DPSRAM*x*_SR, enables the embedded processor to determine the configuration of the DPSRAM with respect to the PLD application interface.

The data output from the dual port can be connected directly to the PLD or it can be registered (this is selectable by configuration), as shown in Figure 16 on page 53. If the output is registered, the register is enabled by the appropriate DP*n*_PORT*y*ENA signal, where *n* identifies the block and *y* (A or B) identifies the port.

Figure 16. Dual-Port SRAM Registering



PLD Signals

Table 12 lists the signals that can traverse the PLD interface to the DPSRAM; signals are multiplexed in the different modes.

Table 12. PLD Signals (Part 1 of 2)

Signal	Source	Description
DP0_2_PORTACLK	PLD	DPRAM0/2 port A clock in the PLD
DP0_PORTAADDR[*n*..0] (1)	PLD	DPRAM0 port A address bus; registered
DP0_PORTADATAIN[*n*..0] (1)	PLD	DPRAM0 port A data input bus
DP0_PORTADATAOUT[*n*..0] (1)	Stripe	DPRAM0 port A data output bus
DP0_PORTAWE	PLD	DPRAM0 port A write enable: 1 = write 0 = read
DP0_PORTAENA	PLD	DPRAM0 port A register enable
DP0_PORTBCLK	PLD	DPRAM0 port B clock in the PLD
DP0_PORTBADDR[*n*..0] (1)	PLD	DPRAM0 port B address bus; registered
DP0_PORTBDATAIN[*n*..0] (1)	PLD	DPRAM0 port B data input bus
DP0_PORTBDATAOUT[*n*..0] (1)	Stripe	DPRAM0 port B data output bus
DP0_PORTBWE	PLD	DPRAM0 port B write enable: 1 = write 0 = read
DP0_PORTBENA	PLD	DPRAM0 port B register enable
DP1_3_PORTACLK	PLD	DPRAM1/3 port A clock in the PLD
DP1_PORTAADDR[*n*..0] (1)	PLD	DPRAM1 port A address bus; registered
DP1_PORTADATAIN[*n*..0] (1)	PLD	DPRAM1 port A data input bus
DP1_PORTADATAOUT[*n*..0] (1)	Stripe	DPRAM1 port A data output bus
DP1_PORTAWE	PLD	DPRAM1 port A write enable: 1 = write 0 = read
DP1_PORTAENA	PLD	DPRAM1 port A register enable
DP1_PORTBCLK	PLD	DPRAM1 port B clock in the PLD
DP1_PORTBADDR[*n*..0] (1)	PLD	DPRAM1 port B address bus; registered
DP1_PORTBDATAIN[*n*..0] (1)	PLD	DPRAM1 port B data input bus

Table 12. PLD Signals *(Part 2 of 2)*

Signal	Source	Description
DP1_PORTBDATAOUT[*n*..0] *(1)*	Stripe	DPRAM1 port B data output bus
DP1_PORTBWE	PLD	DPRAM1 port B write enable: 1 = write 0 = read
DP1_PORTBENA	PLD	DPRAM1 port B register enable
DP2_PORTAADDR[n..0] *(1)*	PLD	DPRAM2 port A address bus; registered
DP2_PORTADATAIN[n..0] *(1)*	PLD	DPRAM2 port A data input bus
DP2_PORTADATAOUT[n..0] *(1)*	Stripe	DPRAM2 port A data output bus
DP2_PORTAWE	PLD	DPRAM2 port A write enable: 1 = write 0 = read
DP2_PORTAENA	PLD	DPRAM2 port A register enable
DP3_PORTAADDR[n..0] *(1)*	PLD	DPRAM3 port A address bus; registered
DP3_PORTADATAIN[n..0] *(1)*	PLD	DPRAM3 port A data input bus
DP3_PORTADATAOUT[n..0] *(1)*	Stripe	DPRAM3 port A data output bus
DP3_PORTAWE	PLD	DPRAM3 port A write enable: 1 = write 0 = read
DP3_PORTAENA	PLD	DPRAM3 port A register enable

Note:
(1) The size of these ports depends on the configuration selected.

Arbiter

The dual-port SRAM arbiter is the same as the single-port SRAM arbiter.

Bus Interfaces

These interfaces are the same as for the single-port SRAM; however, if either interface is disabled, the interface provides a two-cycle ERROR response to any accesses.

Endianness

The dual-port SRAM block supports little- or big-endian transfers as appropriate.

PLD Interface Modes

The flexibility of the dual-port SRAM allows memory configuration of each block, independently of the other, to one of several conventional PLD interface modes, as required, or to use wide/deep mode. The following configuration modes are possible:

- Blocks combined
- Block provides one AHB-PLD dual-port SRAM
- Block provides two AHB-PLD dual-port SRAM
- Block provides one PLD-PLD dual-port SRAM

The MegaWizard Plug-In uses a different nomenclature for the different dual-port SRAM configurations, based on the PLD's perspective. For example, the MegaWizard Plug-In refers to dual-port SRAM configured with one port interfaced to the AHB and the other port interfaced to the PLD as a single-port RAM, which is equivalent to the AHB-PLD dual-port SRAM referred to in this document. The MegaWizard Plug-In refers to dual-port SRAM configured with both ports connected to the PLD as dual-port RAM, which is equivalent to the PLD-PLD dual-port SRAM referred to in this document.

Tables 13 through 15 list the data widths that can be configured for the EPXA1, EPXA4, and EPXA10 devices, according to the mode of access adopted.

Table 13. EPXA1 Dual-Port SRAM Width Options

Interface Width	Conventional Mode			Combined Mode
	One AHB-PLD Dual-Port SRAM	Two AHB-PLD Dual-Port SRAM	One PLD-PLD Dual-Port SRAM	
64-bit	-	-	-	-
32-bit	1 × 4 K × 32	-	-	-
16-bit	1 × 8 K × 16	2 × 4 K × 16	8 K × 16	-
8-bit	1 × 16 K × 8	2 × 8 K × 8	-	-

Table 14. EPXA4 Dual-Port SRAM Width Options

Interface Width	Conventional Mode			Combined Mode
	One AHB-PLD Dual-Port SRAM	Two AHB-PLD Dual-Port SRAM	One PLD-PLD Dual-Port SRAM	
64-bit	-	-	-	1 × 8 K × 64 (wide)
32-bit	1 × 8 K × 32	-	-	1 × 16 K × 32 (deep)
16-bit	1 × 16 K × 16	2 × 8 K × 16	16 K × 16	1 × 32 K × 16 (deep)
8-bit	1 × 32 K × 8	2 × 16 K × 8	-	1 × 64 K × 8 (deep)

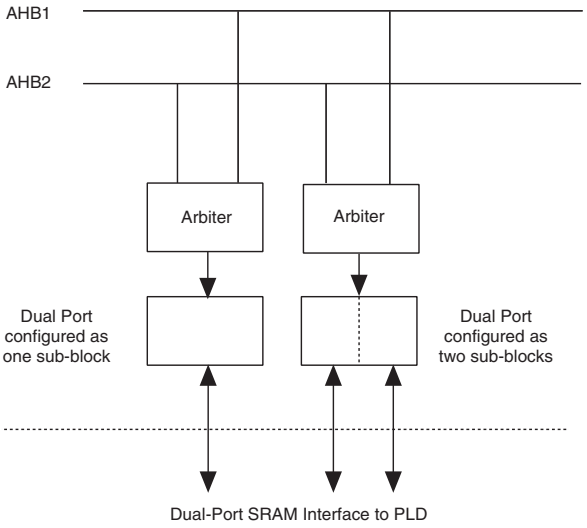
Table 15. EPXA10 Dual-Port SRAM Width Options

Interface Width	Conventional Mode			Combined Mode
	One AHB-PLD Dual-Port SRAM	Two AHB-PLD Dual-Port SRAM	One PLD-PLD Dual-Port SRAM	
64-bit	-	-	-	1 × 16 K × 64 (wide)
32-bit	1 × 16 K × 32	-	-	1 × 32 K × 32 (deep)
16-bit	1 × 32 K × 16	2 × 16 K × 16	32 K × 16	1 × 64 K ×16 (deep)
8-bit	1 × 64 K × 8	2 × 32 K × 8	-	1 × 128 K × 8 (deep)

AHB-PLD Dual-Port SRAM

AHB-PLD dual-port SRAM mode is a conventional interface mode that allows port B to access dual-port SRAM from either AHB1 or AHB2, and port A to access dual-port SRAM from the PLD. The dual-port SRAM block can be subdivided into two sub-blocks or configured as one block, as shown in Figure 17 on page 57.

Figure 17. AHB-PLD Dual-Port SRAM Configurations



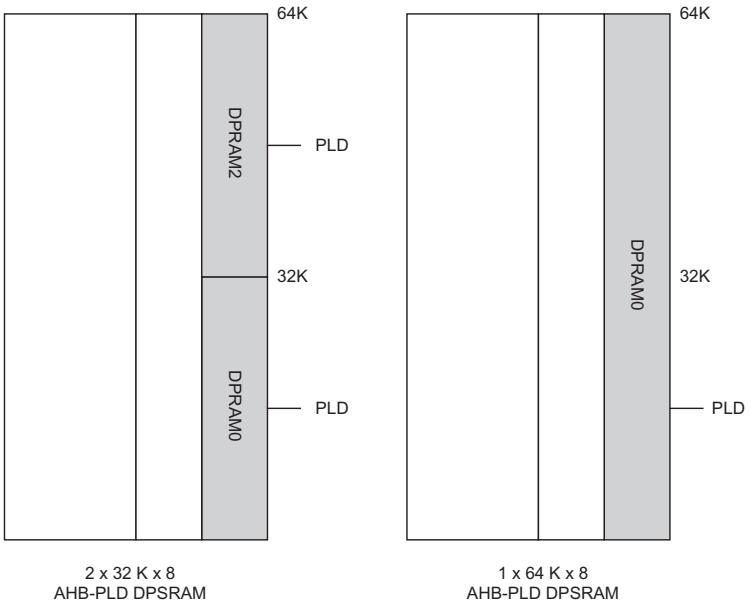
Either block of dual-port SRAM can be configured in any of the configurations shown in Figure 17 or Figure 21.

The data width of the PLD interface to the dual-port SRAM block can be 8-bit, 16-bit, or 32-bit, while the data width to the AHB buses is always 32-bit. Thus, the AHB buses always access the dual-port SRAM at a word-aligned address (i.e., 0, 4, 8, etc.) even though the data width of the PLD interface can be 8-bit, 16-bit, or 32-bit. Refer to *AN173: Excalibur Solutions—DPRAM Reference Design* for more information on how to access the dual-port SRAMs in different configurations.

The following examples illustrate the various AHB-PLD dual-port SRAM configurations for the EPXA10 device. The EPXA4 and EPXA1 devices are similar, except that the block sizes are different.

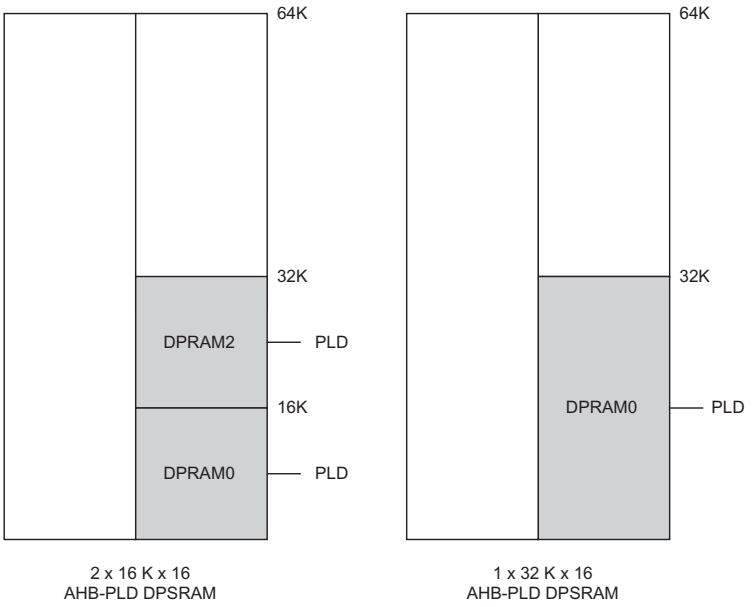
For a PLD interface with an 8-bit data width, either block of the dual-port SRAM can be configured as one 64 K × 8 block, or two sub-blocks of 32 K × 8. Figure 18 on page 58 shows these configurations for block 0, although block 1 can be configured similarly. Each 32 K × 8 sub-block has a separate port for interfacing to the PLD.

Figure 18. Dual-Port SRAM Arranged As × 8



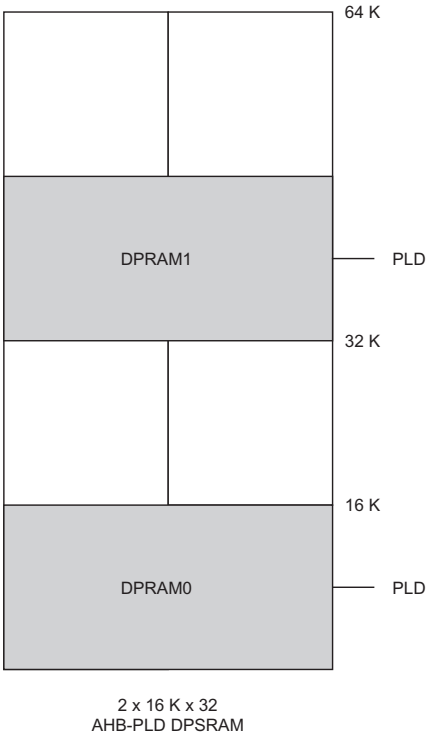
Similarly, for a 16-bit data-width PLD interface, each dual-port SRAM block can be configured as one block of 32 K × 16 or divided into two sub-blocks of 16 K × 16 as shown in Figure 19 on page 59.

Figure 19. Dual-Port SRAM Arranged As × 16



For a 32-bit data-width PLD interface, block 0 and/or block 1 of the dual-port SRAM can only be configured as independent 16 K × 32 blocks. Figure 20 shows an example of both blocks 0 and 1 configured as 32-bit wide and mapped to a particular address. They can also be configured such that one of the dual-port SRAM blocks is 32-bit wide, and the other block is 8-bit wide or 16-bit wide. In addition, the location of the blocks can be mapped anywhere within the addressable range.

Figure 20. Dual-Port SRAM Arranged As Independent × 32 Blocks



PLD-PLD Dual-Port SRAM

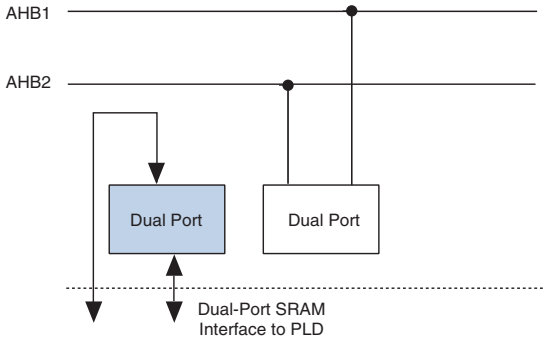
A dual port is configured such that it is dedicated to the PLD application interface and has no access to AHB1 or AHB2. The device interfaces with dual-port SRAM solely through ports A and B. In this mode, it is possible to create a 32 K × 16 dual port on EPXA10 devices, a 16 K × 16 dual port on EPXA4 devices, or an 8 K × 16 dual port on EPXA1 devices.

Take care that the same location is not accessed from both ports simultaneously.

In PLD-PLD dual-port SRAM mode, both output ports have registers clocked by the same clock as the input ports, but with separate register enables.

Figure 21 on page 61 shows the configuration of one dual-port SRAM in PLD-PLD dual-port SRAM mode.

Figure 21. PLD-PLD Dual-Port SRAM



Memory Sizes

Tables 16 to 18 and Figure 22 on page 63 summarize the arrangements of memory visible to the embedded processor for the different configuration modes of the EPXA1, EPXA4, and EPXA10 devices.

Table 16. EPXA1 Dual-Port SRAM Block Organization

PLD Application Interface		Dual-Port SRAM Size *(1)*
No. of Blocks	**Configuration**	**AHB Address Space Used**
None	None	16 Kbytes, word access
1	4 K × 32 AHB-PLD	16 Kbytes, word access
1	8 K × 16 AHB-PLD	32 Kbytes, lower half-word access
2	4 K × 16 AHB-PLD	32 Kbytes (2 × 16 Kbytes consecutive addresses), lower-half-word access
1	16 K × 8 AHB-PLD	64 Kbytes, lower byte access
2	8 K × 8 AHB-PLD	64 Kbytes (2 × 32 Kbytes consecutive addresses), lower-byte access
1	8 K × 16 PLD-PLD	Disabled

Note:
(1) From AHB interface

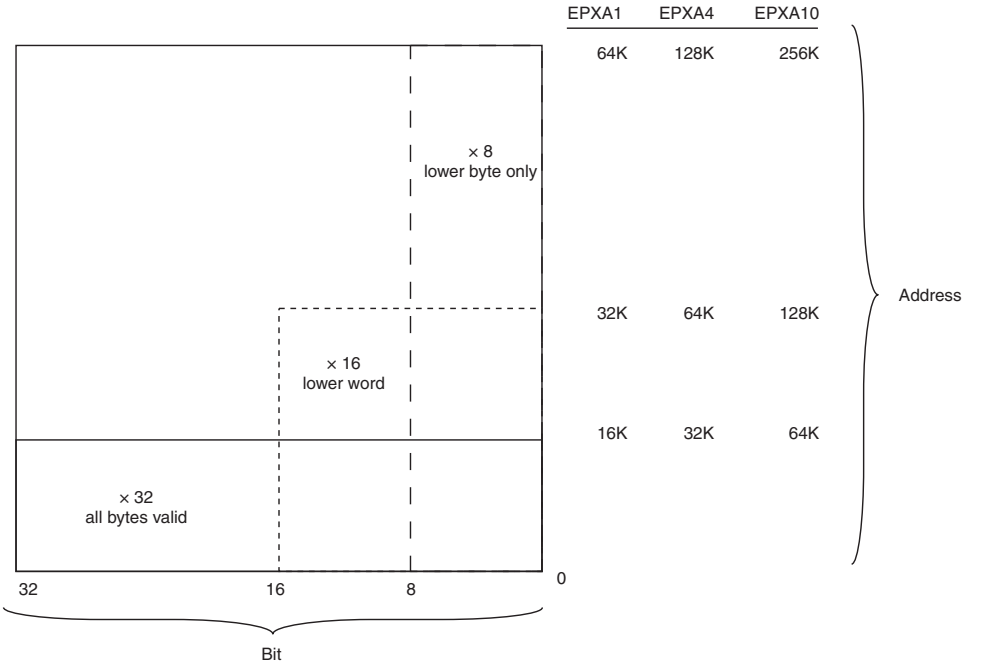
Table 17. EPXA4 Dual-Port SRAM Block Organization		
PLD Application Interface		**Dual-Port SRAM Size** *(1)*
No. of Blocks	**Configuration**	**AHB Address Space Used**
None	None	32 Kbytes, word access
1	8 K × 32 AHB-PLD	32 Kbytes, word access
1	16 K × 16 AHB-PLD	64 Kbytes, lower half-word access
2	8 K × 16 AHB-PLD	64 Kbytes (2 × 32 Kbytes consecutive addresses), lower-half-word access
1	32 K × 8 AHB-PLD	128 Kbytes, lower byte access
2	16 K × 8 AHB-PLD	128 Kbytes (2 × 64 Kbytes consecutive addresses), lower-byte access
1	16 K × 16 PLD-PLD	Disabled

Note:
(1) From AHB interface

Table 18. EPXA10 Dual-Port SRAM Block Organization		
PLD Application Interface		**Dual-Port SRAM Size** *(1)*
No. of Blocks	**Configuration**	**AHB Address Space Used**
None	None	64 Kbytes, word access
1	16 K × 32 AHB-PLD	64 Kbytes, word access
1	32 K × 16 AHB-PLD	128 Kbytes, lower half-word access
2	16 K × 16 AHB-PLD	128 Kbytes (2 × 64 Kbytes consecutive addresses), lower-half-word access
1	64 K × 8 AHB-PLD	256 Kbytes, lower byte access
2	32 K × 8 AHB-PLD	256 Kbytes (2 × 128 Kbytes consecutive addresses), lower-byte access
1	32 K × 16 PLD-PLD	Disabled

Note:
(1) From the AHB interface

Figure 22. Byte Allocation



In modes where the dual-port block is less than 32 bits wide, only some of the AHB byte lanes are used. Each location in the dual port starts on a 4-byte word address, and word accesses to that address return the value stored in the location (with zeros in the most significant 16 or 24 bits). The effects of byte and half-word accesses in these modes depend on the endianness selected.

In modes where the dual-port block is split into two sub-blocks, both sub-blocks are mapped contiguously. The lower numbered sub-block (0 or 1) starts at the memory map base address and the higher numbered sub-block is at the next higher address.

Combined Dual-Port Blocks

When the dual ports are combined together, this does not affect accesses from the AHB1 and AHB2 buses. In 8-, 16- and 32-bit width modes, both base addresses must be programmed with appropriate values to make the blocks appear contiguous within the memory map. In 64-bit width modes the AHB must make separate accesses to each half of the dual port (although the PLD accesses them simultaneously).

Wide/Deep Multiplexing

Users can configure a combination of both dual-port memories to form deeper or wider memories for EPXA4 and EPXA10 devices (this feature is not supported for EPXA1 devices), although only particular dual-port configurations are allowed in these modes.

If the dual-port SRAM mode is configured as either deep (× 8) or wide (× 32), the DP0_2_PORTACLK and DP1_3_PORTACLK clock signals are not pre-balanced. The Quartus® II fitter generates an error because it is unable to balance locally-routed clocks. To avoid generating a Quartus II fitter error, promote the DP0_2_PORTACLK or the DP1_3_PORTACLK signal to a global signal and recompile the Quartus project.

Tables 19 and 20 show the possible memory combinations for deep multiplexing.

Table 19. EPXA4 Deep Modes (1)

Individual Modes	Combined Memory
8 K × 32	16 K × 32
16 K × 16	32 K × 16
32 K × 8	64 K × 8

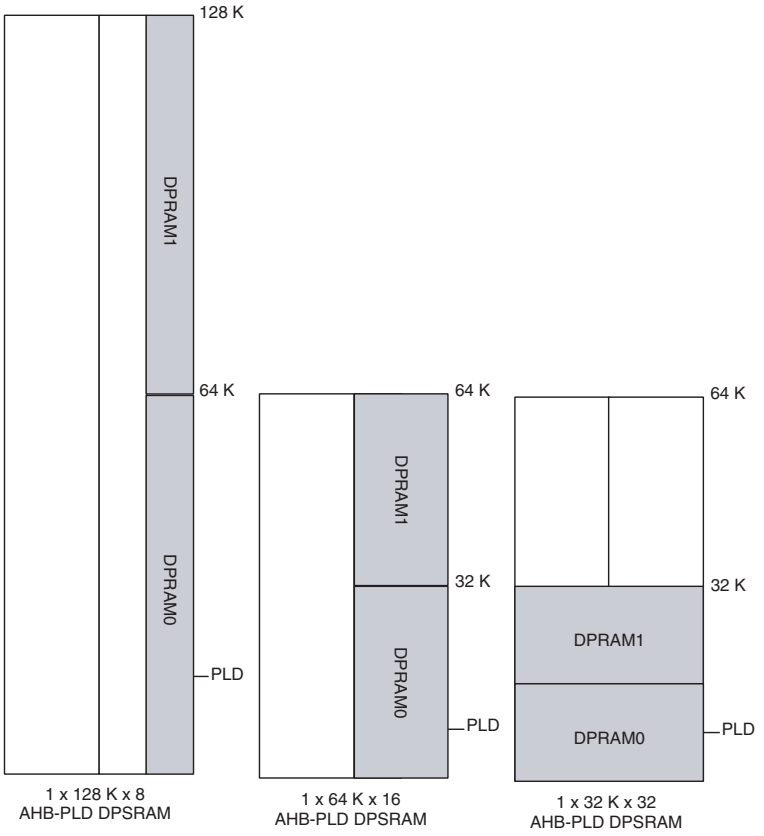
Table 20. EPXA10 Deep Modes (1)

Individual Modes	Combined Memory
16 K × 32	32 K × 32
32 K × 16	64 K × 16
64 K × 8	128 K × 8

Note:
(1) The memories appear to the AHB interfaces as individual dual-port memories (× 32, × 16, or × 8).

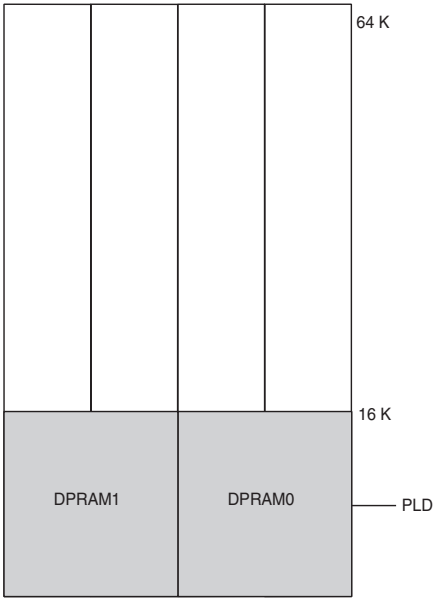
Figure 23 on page 65 shows the combined deep mode representation for the EPXA10 device. The EPXA4 combined deep mode configuration operates similarly but with different sizes.

Figure 23. Combined Deep Mode Configurations for EPXA10



For the EPXA10 device, wide mode is only applicable if both individual dual-ports are configured as 16 K × 32, to give an overall memory of 16 K × 64 as shown in Figure 24 on page 66. For EPXA4, the dual ports must be configured as 8 K × 32, to give an overall memory of 8 K × 64.

Figure 24. Combined Wide Mode Configurations for EPXA10



Registers

The registers are all reset by WARM_RESET_n.

Register: DPSRAM0_SR
Address: Register base + 30H
Access: Read

The layout of this register is the same as for DPSRAM1_SR.

Register: DPSRAM1_SR
Address: Register base + 38H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIZE																				0						GLBL		MODE			

SIZE	R	Memory block size in bytes (e.g. 32 Kbytes)
GLBL	R	Global dual-port mode: 00—Normal dual port 01—Deep single port 10—Wide single port
MODE	R	Mode[3]—Enables output registers Mode[2..0]—Dual-port mode meanings depend on GLBL; see Table 21 for details.
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Bits 31-12 of DPSRAMx_SR contain bits 31-12 of the block size, in bytes, for the appropriate block of RAM. Bits 11-0 of the block size are 0. The value held in SIZE does not change when MODE changes.

Table 21 shows how the GLBL and MODE settings in DPSRAMx_SR affect dual-port SRAM configuration.

Table 21. Dual-Port SRAM Settings in DPSRAMx_SR

GLBL	MODE [2..0]	Dual-Port Size and Mode		
		EPXA10	EPXA4	EPXA1
xx	000	Interface disabled (default)	Interface disabled (default)	Interface disabled (default)
00	001	One AHB-PLD 64 Kbyte × 8	One AHB-PLD 32 Kbyte × 8	One AHB-PLD 16 Kbyte × 8
00	010	One AHB-PLD 32 Kbyte × 16	One AHB-PLD 16 Kbyte × 16	One AHB-PLD 8 Kbyte × 16
00	011	One AHB-PLD 16 Kbyte × 32	One AHB-PLD 8 Kbyte × 32	One AHB-PLD 4 Kbyte × 32
00	100	One PLD-PLD 32 Kbyte × 16	One PLD-PLD 16 Kbyte × 16	One PLD-PLD 8 Kbyte × 16
00	101	Two AHB-PLD 32 Kbyte × 8	Two AHB-PLD 16 Kbyte × 8	Two AHB-PLD 8 Kbyte × 8
00	110	Two AHB-PLD 16 Kbyte × 16	Two AHB-PLD 8 Kbyte × 16	Two AHB-PLD 4 Kbyte × 16
01	001	One deep AHB-PLD 128 Kbyte × 8	One deep AHB-PLD 64 Kbyte × 8	Unused
01	010	One deep AHB-PLD 64 Kbyte × 16	One deep AHB-PLD 32 Kbyte × 16	Unused
01	011	One deep AHB-PLD 32 Kbyte × 32	One deep AHB-PLD 16 Kbyte × 32	Unused
10	011	One wide single port 16 Kbyte × 64	One wide single port 8 Kbyte × 64	
11	xxx	Unused	Unused	Unused
xx	111	Unused	Unused	Unused

Address decoding for the dual-port registers occurs in the centralized AHB2 address decoder, which supplies the individual dual-port instances with the appropriate register select signals.

SDRAM Controller

The SDRAM controller interfaces between the internal system buses and external synchronous DRAM. There are chip-selects for two blocks of devices, each of up to 256 Mbytes, giving support for a maximum of 512 Mbytes of memory. Eight banks are available, with up to four per block of SDRAM, for optimized performance.

EPXA10 devices always assume that the SDRAM has four banks of memory in each block.

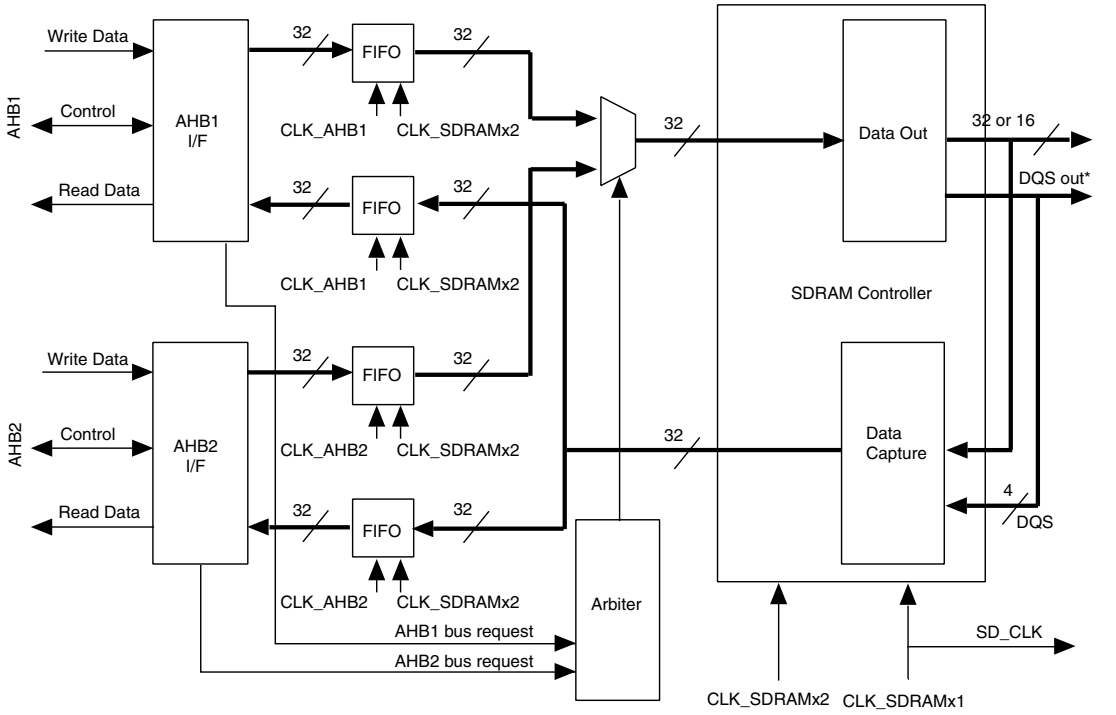
The SDRAM controller runs asynchronously to AHB1 and AHB2. It supports byte, half-word, and word transfers on the 32-bit system buses. Single beat, fixed-length incremental, fixed-length wrapping, and undefined-length incremental transfers are implemented. Eight-beat, fixed-length bursts are used for transfers and early termination is accepted.

Either 16-bit or 32-bit SDR or DDR SDRAM (but not both simultaneously) can be connected, with clock speeds up to 133 MHz (SDR) or 266 MHz (DDR). EPXA10 devices support 32-bit mode. EPXA4 devices support 32-bit and 16-bit modes. EPXA1 devices in the F672-pin package support 32-bit and 16-bit modes, while EPXA1 devices in the F484-pin package only support 16-bit mode.

The SDR interface is JEDEC-compliant; the DDR interface is JEDEC-compliant and Intel PC200/266-compliant.

Figure 25 shows the SDRAM controller logic, and further sections detail the operation of each block.

Figure 25. SDRAM Controller Block Diagram



Module I/O

Table 22 summarizes the SDRAM I/O signals.

Table 22. SDRAM I/O Signals (Part 1 of 2)

Signal	Direction	Description
SD_CLK	Output	SDRAM clock pin *(1)*
SD_CLK_n	Output	SDRAM CLK_n pin *(1)*
SD_CLKE	Output	SDRAM CLK_E clock-enable pin (dedicated pin)
SD_WE_n	Output	SDRAM active-low write enable pin *(1)*
SD_CAS_n	Output	SDRAM CAS_n pin *(1)*
SD_RAS_n	Output	SDRAM RAS_n pin *(1)*
SD_CS_n[1..0]	Output	SDRAM chip selects CS_n pins *(1)*
SD_A[14..0]	Output	SDRAM address bus *(1)*: SD_A[14..13]—bank address*(2)*
SD_DQM[3..0]	Output	SDRAM DQM data byte mask pins *(1)*
SD_DQ[31..0]	Input/Output	SDRAM bidirectional data bus pins *(1)*

Table 22. SDRAM I/O Signals (Part 2 of 2)

Signal	Direction	Description
SD_DQS[3..0]	Input/Output	SDRAM bidirectional data strobe pins *(1)*
SD_DDR_VS[2..0]	Input	DDR SDRAM voltage reference pins

Notes:
(1) Shared pin. All SDRAM pins except for SD_CLKE are shared I/O on the EPXA10. On EPXA4, all SDRAM pins are shared except for SD_DQ[31..24], SD_DQS[3] and SD_DDR_VS[2]. None of the SDRAM pins on EPXA1 are shared.
(2) Altera recommends that you aways use SD_A[14..13] to drive bank addresses BA1 and BA2.

All external signals are capable of interfacing with signals of 3.3-V LVTTL or 2.5-V SSTL-2 class II (15.2 mA). The mode is determined using the IOCR_SDRAM register (see page 168), with LVTTL as the default. The operating voltage is determined by the supply provided.

All SDRAM pins except for SD_CLKE are shared I/O on EPXA10 devices. They can be used as SDRAM pins or as PLD user I/Os. On EPXA4 devices, some SDRAM pins are dedicated, specifically SD_DDR_VS[2], SD_DQS[3], and SD_DQ[31..24]. No SDRAM pins are shared on EPXA1 devices; they are all dedicated SDRAM pins.

In SDR mode, SD_DQS[0] And SD_CLK_n must be connected together externally. In this mode, the pins SD_DQS[3..1] and SD_DDR_VS[2..0] are not used; when using the SDRAM controller in SDR mode, these pins should be connected to logic '1'.

Bus Interface

The bus interface logic accepts write and read operations for the external memory and local configuration registers.

Both AHB1 and AHB2 can access SDRAM through the SDRAM controller. The SDRAM controller is clocked by PLL2 and operates asynchronously to AHB1 and AHB2. The bus interface module contains synchronization circuitry to re-time data and control signals to the SDRAM controller. The SDRAM bus arbiter controls access.

The function of the arbiter is to multiplex the requests from the system bus interfaces to the SDRAM controller state machine. The arbitration algorithm employed is first come, first served. If both buses request together, the bus is granted to 'opposite of previous user'. Locked accesses to the SDRAM from masters on AHB1 and AHB2 are supported. While a selected bus asserts a bus lock, selection of the other bus is prevented.

Write operations are posted to an 8-word FIFO buffer, and are usually acknowledged immediately. However, if the FIFO buffer is full, the write operation is held in a wait state until the data can be accepted.

Read operations are held in a wait state while the SDRAM controller fetches the data.

All transfers within a burst must be aligned to the address boundary equal to the size of the transfer. Little-endian and big-endian data formats can be used, with the mode selected as required.

Endianness

The endianness of the SDRAM controller is determined by the endianness that the system has been configured to use. Before changing the system endianness, ensure that the SDRAM is idle.

Byte and Word Mapping for Size and Endianness

Devices on the SDRAM can be 32- or 16-bits wide. Transactions on AHB2 can be 8-, 16-, or 32-bits wide. If the size of the AHB transaction is wider than the SDRAM device width, the SDRAM controller makes multiple SDRAM accesses. If the AHB transaction is narrower than the SDRAM, the SDRAM controller masks the data read or, if appropriate, uses byte enables.

Tables 23 to 26 document the mapping of data in the AHB word to writes and reads to and from the SDRAM. Each block of the SDRAM can be 32- or 16-bits wide, and many permutations are possible. The SDRAM outputs zeros on any byte not written.

Table 23. Write Mapping for Little-Endian Transmissions *Note (1)*

		HWDATA				SD_DQ				SD_DQ		Transaction
HSIZE	HADDR[1..0]	31			0	31			0	15	0	
Byte(0)	00	×	×	×	B0	×	×	×	B0	×	B0	SDRAM write
Byte(1)	01	×	×	B1	×	×	×	B1	×	B1	×	SDRAM write
Byte(2)	10	×	B2	×	×	×	B2	×	×	×	B2	SDRAM write
Byte(3)	11	B3	×	×	×	B3	×	×	×	B3	×	SDRAM write
Hword(0)	0×	H3	H2	H1	H0	H3	H2	H1	H0	H1- H0		First SDRAM write
Hword(1)	1×	H3	H2	H1	H0	H3	H2	H1	H0	H3-H2		First SDRAM write
Word	××	W3	W2	W1	W0	W3	W2	W1	W0	W1 - W0		First SDRAM write
										W3 - W2		Second SDRAM write

Note:
(1) Bytes marked as × are undefined.

Table 24. Write Mapping for Big-Endian Transmissions *Note (1)*

		HWDATA				SD_DQ				SD_DQ		Transaction
HSIZE	HADDR[1..0]	31			0	31			0	15	0	
Byte(0)	00	B3	×	×	×	B3	×	×	×	B3	×	SDRAM write
Byte(1)	01	×	B2	X	×	×	B2	×	×	×	B2	SDRAM write
Byte(2)	10	×	×	B1	×	×	×	B1	×	B1	×	SDRAM write
Byte(3)	11	×	×	×	B0	×	×	×	B0	×	B0	SDRAM write
Hword(0)	0×	H3	H2	×	×	H3	H2	×	×	H3- H2		First SDRAM write
Hword(1)	1×	×	×	H1	H0	×	×	H1	H0	H1-H0		First SDRAM write
Word	××	W3	W2	W1	W0	W3	W2	W1	W0	W3 - W2		First SDRAM write
										W1 - W0		Second SDRAM write

Note:
(1) Bytes marked as × are undefined

Table 25. Read Mapping for Little-Endian Transmissions

		HWDATA				SD_DQ				SD_DQ		Transaction
HSIZE	HADDR[1..0]	31			0	31			0	15	0	
Byte(0)	00				B0				B0	0	B0	SDRAM read
Byte(1)	01			B1				B1		B1	0	SDRAM read
Byte(2)	10		B2				B2			0	B2	SDRAM read
Byte(3)	11	B3				B3				B3	0	SDRAM read
Hword(0)	0×			H1	H0			H1	H0	H1- H0		First SDRAM read
Hword(1)	1×	H3	H2			H3	H2			H3-H2		First SDRAM read
Word	××	W3	W2	W1	W0	W3	W2	W1	W0	W1 - W0		First SDRAM read
										W3 - W2		Second SDRAM read

Table 26. Read Mapping for Big-Endian Transmissions

		HWDATA				SD_DQ				SD_DQ		Transaction
HSIZE	HADDR[1..0]	31			0	31			0	15	0	
Byte(0)	00	B3				B3				B3	0	SDRAM read
Byte(1)	01		B2				B2			0	B2	SDRAM read
Byte(2)	10			B1				B1		B1	0	SDRAM read
Byte(3)	11				B0				B0	0	B0	SDRAM read
Hword(0)	0×	H3	H2			H3	H2			H3- H2		First SDRAM read
Hword(1)	1×			H1	H0			H1	H0	H1-H0		First SDRAM read
Word	××	W3	W2	W1	W0	W3	W2	W1	W0	W3 - W2		First SDRAM read
										W1 - W0		Second SDRAM read

Memory Access State Machine

The memory access state machine is driven by requests from the arbiter. It controls read and write access, and also supports external device initialization and refresh.

The controller is designed for DDR (e.g., PC266, PC200) and SDR operation; both operating modes are JEDEC-compliant. Slower devices can be supported with appropriate configuration of the controller parameters.

Either SDR or DDR SDRAM can be controlled, but not both simultaneously.

Transfers to the memory are made up of eight-beat reads and writes. A request from the system bus that does not map directly to this fixed-beat access (for example, a larger burst size or a wrapping transfer) is handled by performing multiple accesses. Burst termination is utilized to maximize throughput.

Two blocks of memory are supported, with up to four banks each (four banks only, for EPXA10 devices). The controller keeps the bank open after an access, closing it only when required to access a different page. Manual precharge is always used (there is no automatic precharge).

It is possible to program the interval for automatic memory refresh; all banks are closed when this is performed.

It is not possible to turn off the `SD_CLK` and `SD_CLK_n` outputs. Power savings can be achieved by putting the SDRAM PLL in bypass mode.

The contents of SDRAM are not guaranteed after a reset.

Registers

Configuration and control registers required by the SDRAM controller are mapped as part of the registers memory region and are accessed from AHB2.

Many of the parameters specified in the SDRAM register section are required in units of SDRAM clock periods. SDRAM data sheets typically specify these parameters in units of time. To obtain the number of SDRAM clock cycles from the time specified, divide the parameter's time value by the SDRAM clock period and round the result up to the nearest integer to ensure the SDRAM timing parameter is not violated.

Registers reside on a 32-bit boundary. All registers are reset to 0, unless otherwise indicated.

Register: SDRAM_TIMING1
Address: Register base + 400H
Access Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																RCD			RAS				RRD			RP			WR		

RCD	R/W	Active to read or write delay. Specified as number of clocks necessary to meet timing requirements. Valid values: 001 (1 clock) to 100 (4 clocks)
RAS	R/W	Active to precharge command. Specified as number of clocks necessary to meet timing requirement. Valid values: 0001 (1 clock) to 1000 (8 clocks)
RRD	R/W	Active bank A to active bank B command. Specified as number of clocks necessary to meet timing requirement. Valid values: 001 (1 clock) to 100 (4 clocks)
RP	R/W	Precharge command period. Specified as number of clocks necessary to meet timing requirement. Valid values: 001 (1 clock) to 100 (4 clocks)
WR	R/W	Write recovery time. Specified as number of clocks necessary to meet timing requirement. Valid values: 001 (1 clock) to 011 (3 clocks)
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: SDRAM_TIMING2
Address: Register base + 404H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																RC				CL			BL		RFC				0		

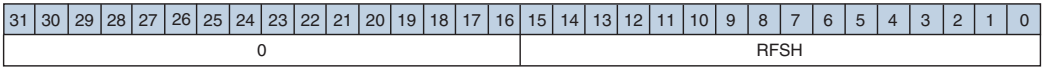
RC	R/W	Active to active command period. Specified as number of clocks necessary to meet timing requirement. Valid values: 0001 (1 clock) to 1010 (10 clocks)
CL	R/W	CAS latency. Valid values: 011—2 clocks 100—2.5 clocks 101—3 clocks
BL	R/W	SDRAM burst length. Valid value: 11—8 words
RFC	R/W	SDRAM auto refresh period. Specified as number of clocks necessary to meet timing requirement. Valid values: 0003 (3 clocks) to 1011 (11 clocks)
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: SDRAM_CONFIG
Address: Register base + 408H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																MT	0														

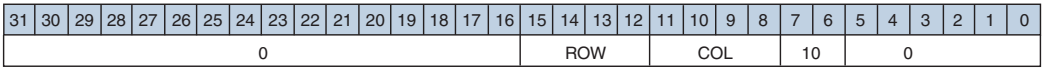
MT	R/W	SDRAM memory type: 0—SDR 1—DDR
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: SDRAM_REFRESH
Address: Register base + 40CH
Access: Read/write



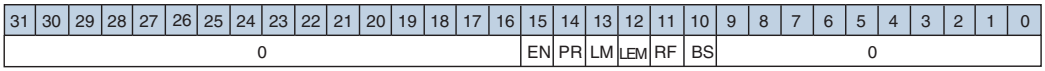
RFSH	R/W	SDRAM hidden refresh period. Specified as number of clocks between auto-refresh commands based on CLK_SDRAM
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: SDRAM_ADDR
Address: Register base + 410H
Access: Read/write



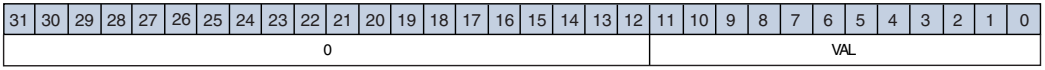
ROW	R/W	Number of row address bits (values 11 to 13 only)
COL	R/W	Number of col address bits (values 8 to 12 only)
10	R	Reserved for future use. Write as 10 to ensure future compatibility
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: SDRAM_INIT
Address: Register base + 41CH
Access: Read/write



EN	R/W	Enable SDRAM controller: 0—Disable 1—Enable
PR	W	Perform precharge all command
LM	W	Perform a load mode register command
LEM	W	Perform a load extended mode register command
RF	W	Perform a refresh command
BS	R	0—Controller is not performing LM, LEM, or precharge commands 1—Controller is performing LM, LEM, or precharge commands
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: SDRAM_MODE0
Address: Register base + 420H
Access: Read/write



VAL	R/W	Holds the value to be loaded into the SDRAM configuration register during a load mode register command. For details of how to obtain this value, refer to the mode register settings of the memory device in use
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: SDRAM_MODE1
Address: Register base + 424H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																				VAL											

VAL	R/W	Holds the value to be loaded into the SDRAM configuration register during a load extended mode register command (DDR memories only). For details of how to obtain this value, refer to the mode register settings of the memory device in use
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: SDRAM_WIDTH
Address: Register base + 7CH
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																														W	LK

LK	R/W	1—Further writes have no effect
W	R/W	Width of SDRAM port; defaults to 1 (32 bits wide). See page 68 for more details of the port width. 0—16-bit 1—32-bit
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The user specifies the SDRAM interface width by writing to this I/O control register.

If the lock bit, LK, is set, writes cause a bus error. LK remains set until either a warm reset (WARM_RESET_n) or hard reset (nPOR) is asserted.

Connecting SDRAM Devices

The following sections describe the options for connecting SDRAM devices to the SDRAM controller in Excalibur devices.

Address Lines

The address lines of the SDRAM controller can be used for two purposes when connected to SDRAM devices:

- As traditional address lines that connect to the corresponding address lines of the SDRAM device
- As bank-select (BA) pins

Typically, SDRAM devices are internally organized into two or more banks, and bank-select pins on the SDRAM device are used to select the bank required for access. The bank-select pins are connected to the address lines of the SDRAM controller, and the SDRAM controller handles the timing of the bank-address signals. The address lines of the SDRAM device are usually connected to the lowest-order address pins of the SDRAM controller. The bank-select pins of the SDRAM device are then connected to address lines A13 and A14 of the SDRAM controller.

Data Lines

The SDRAM data lines are used to transfer data to and from the SDRAM device. If the SDRAM controller is in 32-bit mode, all 32 data lines from the SDRAM controller connect to their corresponding data lines on the SDRAM device. If two 16-bit SDRAM devices will combine to create a virtual 32-bit SDRAM, `SD_DQ[15..0]` connects to `D[15..0]` of one SDRAM, and `SD_DQ[31..16]` connects to `D[15..0]` of the other SDRAM. For SDRAM devices less than 16 bits wide, a similar scheme of dividing the `SD_DQ` lines between devices achieves the same effect. Interfacing directly to 32-bit SDRAM devices is not supported in DDR mode, because 32-bit DDR SDRAM devices utilize only one `DQS` pin. The SDRAM controller requires the use of four `DQS` signals. To create a virtual 32 bit DDR SDRAM memory, multiple narrower memory devices can be combined, each with their own DQS signal.

If the SDRAM controller has been configured to operate in 16-bit mode, `SD_DQ[15..0]` connects to `D[15..0]`. Where multiple, narrower SDRAM devices combine to create a virtual 16-bit SDRAM, divide the `SD_DQ` lines between the SDRAM devices as explained above.

Control Lines

Table 27 explains how the SDRAM control lines are connected.

Table 27. SDRAM Control Lines

Signal	Connection
SD_CLK, SD_CLK_n	SD_CLK output connects to the CLK input of SDR SDRAM devices. In SDR mode, SD_CLK_n connects to the SD_DQS0 input of the SDRAM controller. In DDR SDRAM mode, SD_CLK and SD_CLK_n connect to the SDRAM's differential clock input.
SD_CLKE	SD_CLKE connects to the CKE input of SDRAM devices.
SD_CS0, SD_CS1	These chip-select pins connect to the CS# inputs of the SDRAM devices. SD_CS0 is used by the SDRAM controller to select SDRAM block0. SD_CS1 is used to select SDRAM block 1.
SD_RAS, SD_CAS, SD_WE	These command lines connect to the RAS#, CAS#, and WE# inputs of the SDRAM device, respectively.
SD_DQM0-SD_DQM3	These pins act as byte-mask enables. For 32-bit SDRAM devices, these four pins connect to the SDRAM devices' DQM0-DQM3 pins. Narrower memories have fewer than 4 DQM pins, but when stacked to create wider memory, there are 4 DQM pins in total. Each of the controller's 4 DQM pins connects to one of the DQM pins on the stacked SDRAM devices.
SD_DQS0-SD_DQS3	In SDR mode, only SD_DQS0 is used. It connects to SD_CLK_n of the SDRAM controller. In DDR mode, these pins are used as bidirectional data strobes connected to the corresponding DQS pins of the DDR SDRAM devices.

SDRAM Device Configuration

There are three steps to setting up the SDRAM controller to interface with SDRAM memory, as follows:

1. Configure PLL2 to drive the clock rate at which the SDRAM is to run.

   For SDR SDRAM, PLL2 should be set at twice the data rate at which the SDRAM is to run. For instance, if the SDRAM is to run at 100 MHz, PLL2 should be configured for 200 MHz. For DDR SDRAM, set PLL2 to the exact data rate at which the SDRAM is to run. For instance, if the DDR SDRAM is to run at PC200, PLL2 should be configured for 200 MHz.

2. Configure the SDRAM controller so that it can transfer data to and from the SDRAM correctly.

   Writing values to the SDRAM controller registers specifies the characteristics of the SDRAM devices being used.

3. Initialize the SDRAM devices to prepare them for reading and writing before any transactions occur.

   A specific sequence of commands, referred to as the initialization sequence, is sent from the SDRAM controller. The commands are controlled by the SDRAM_INIT register (see page 76).

SDR SDRAM Configuration

For SDR SDRAM configuration, the SDRAM controller must first be configured by loading all the SDRAM registers except for SDRAM_INIT and SDRAM_MODE1. SDRAM_INIT is used to send commands to the SDRAM device and must not be altered until the SDRAM controller has been configured. SDRAM_MODE1 is not used for SDR SDRAM devices, so it should not be written to for SDR configuration.

When the SDRAM controller has been configured, the SDRAM devices themselves must be initialized. The initialization sequence following power-up depends on the SDRAM device being used, but the example below shows an initialization sequence used by many SDR SDRAM devices.

Ensure that the sequence used is in compliance with the initialization sequence specified in the datasheet of the selected SDRAM device.

The sequence has the following stages:

1. Load the SDRAM controller configuration registers.
2. Enable the SDRAM controller.
3. Issue a precharge all command and two refresh commands.
4. Issue a load mode register command.

For this sequence, the following actions are required, using the SDRAM controller's configuration registers:

1. Ensure that the SDRAM PLL is locked at the operating frequency and wait 100 µs before continuing to step 2.
2. Set all of the SDRAM controller's configuration registers except SDRAM_INIT and SDRAM_MODE1.
3. Enable the controller by setting the SDRAM enable bit, EN, of SDRAM_INIT.
4. Issue a precharge all command by setting the perform precharge bit, PR, of SDRAM_INIT *(1)*.

5. Issue two refresh commands, initiating each by setting the perform refresh bit, RF, of SDRAM_INIT *(1)*.
6. Issue a load mode register command by setting the load mode register bit, LM, of SDRAM_INIT. This causes the SDRAM's mode register to be loaded with the value that has been programmed into SDRAM_MODE0 *(1)*.

Note:
(1) For EPXA10 devices, each SDRAM command must be followed by a 50 SDRAM clock delay before continuing.

At this point, the SDRAM is ready for use.

The commands from step 3 until step 6 has finished must complete within one SDRAM refresh period.

DDR SDRAM Configuration

DDR SDRAM configuration is very similar to configuration for SDR SDRAM. The example initialization sequence shown below is a typical sequence.

Ensure that the sequence used is in compliance with the initialization sequence specified in the datasheet of the selected SDRAM device.

The sequence has the following stages:

1. Load the SDRAM controller configuration registers.
2. Enable the SDRAM controller.
3. Issue a precharge all command.
4. Issue a load extended mode register command.
5. Issue a load mode register command.
6. Issue a precharge all command.
7. Issue two refresh commands.
8. Issue a load mode register command.

For this sequence, the following actions are required, using the controller's configuration registers:

1. Ensure that the SDRAM PLL is locked at the operating frequency and wait 200 µs before continuing to step 2.
2. Set all of the SDRAM controller's configuration registers except for `SDRAM_INIT`.

3. Enable the controller by setting the SDRAM enable bit, EN, of SDRAM_INIT.
4. Issue a precharge all command by setting the perform precharge bit, PR, of SDRAM_INIT *(1)*.
5. Issue a load extended mode register command by setting the perform load extended mode register bit, LEM, of SDRAM_INIT. This causes the SDRAM's mode register to be loaded with the value that has been programmed into SDRAM_MODE1 *(1)*.
6. Issue a load mode register command to reset the SDRAM device's DLL, by setting the load mode register bit, LM, of SDRAM_INIT *(1)*.
7. Issue a precharge all command by setting the perform precharge bit, PR, of SDRAM_INIT *(1)*.
8. Issue two refresh commands, initiating each by setting the perform refresh bit, RF, of SDRAM_INIT *(1)*.
9. Issue a load mode register command by setting LM of SDRAM_INIT. This causes the SDRAM's mode register to be loaded with the value that has been programmed into SDRAM_MODE0 *(1)*.

Note:
(1) For EPXA10 devices, each SDRAM command must be followed by a 50 SDRAM clock delay before continuing.

At this point, the SDRAM is ready for use.

The commands from step 3 until step 9 has finished must complete within one SDRAM refresh period.

Clocking

The memory controller is clocked by PLL2 (see "SDRAM Controller" on page 68 for details). PLL2 is configured differently depending on whether it is being used for DDR or SDR memories. For DDR devices, PLL2 should be programmed with the true data-rate clock frequency. For example, if you are using 266-MHz DDR SDRAM, PLL2 should be configured to create a 266MHz clock. However, for SDR devices, PLL2 should be configured for twice the true data-rate clock frequency. For example, if you are using 133-MHz SDR SDRAM, PLL2 should be configured to create a 266-MHz clock.

When setting up the SDRAM PLL, odd values of K, as specified in the register CLK_PLL2_KCNT, generate a non-50:50 square wave. This adversely affects the SDRAM controller in DDR mode by impacting the position of DQS in relation to DQ during a write. Specify K as either 2 or 4; other values should not be used.

Expansion Bus Interface

The expansion bus interface (EBI) is a 16-bit bidirectional external memory interface. The EBI provides a bridge between external devices, such as flash memories or memory-mapped devices, and the AHB2 bus. In addition, for slow external devices it provides for rate adaptation.

The memory controller runs synchronously to AHB2, supporting all transaction types, as defined in the *AMBA Specification, Revision 2.0*, and including split transactions. The EBI manages data packing and unpacking automatically, based on endianness, block configuration, and the size of the transaction selected by the master initiating the transaction to the EBI.

EBI Operation

The EBI is extremely flexible, and the memory interface can be configured to operate either synchronously or asynchronously. It can support four blocks of up to 32 Mbyte of external memory or memory-mapped devices of varying configurations. All blocks can be configured for 8- or 16-bit width. The base address and size of each block can be set in the memory map registers (see the section "Memory Map Control Registers" on page 100). In addition, users can monitor current activity using the EBI status register, EBI_SR.

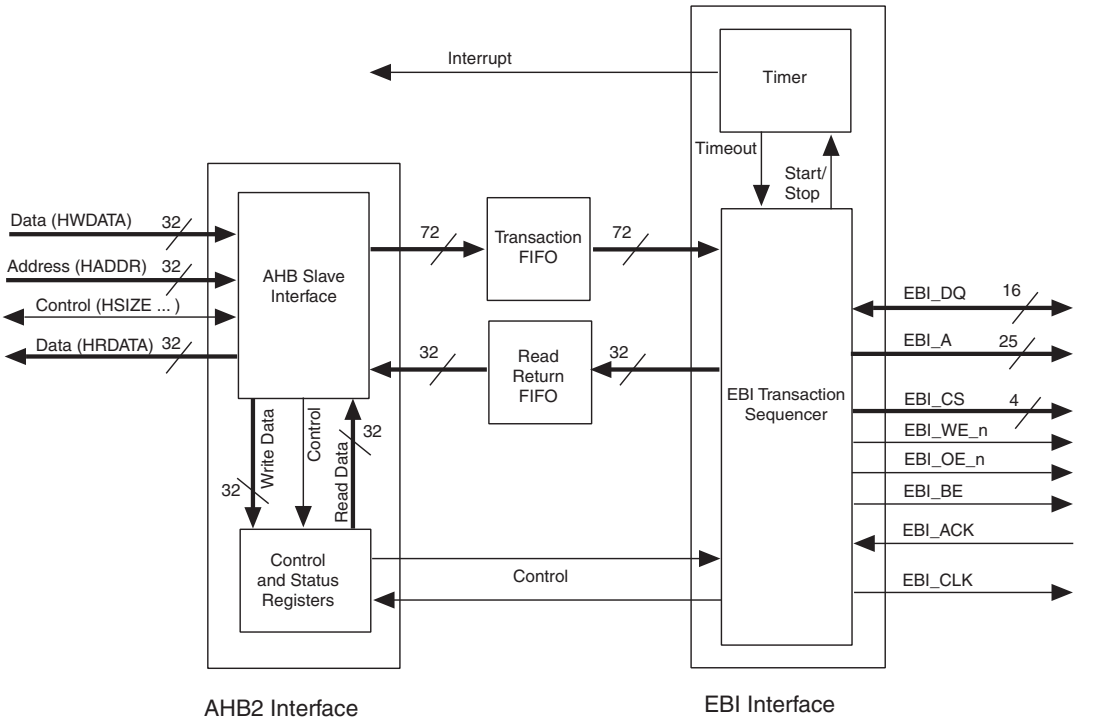
In boot-from-flash mode, the EBI0 block is mapped to address zero, which is the location from which the processor boots. Because EBI0 is used for booting the processor, EBI0 configuration is set at power-up, based on the polarity of the MSEL0 and MSEL1 pins. See "Configuration Logic" on page 149 for details on boot-from-flash mode.

The EBI is a slave on AHB2, with a fixed programmable access period or asynchronous-acknowledge input that is selectable on a block-by-block basis. A clock can be driven from the EBI for synchronous operation, if required, derived from the AHB2 clock with a programmable divide factor.

The EBI is capable of issuing a split response to AHB transactions that would otherwise tie up the bus for a long period, e.g., a long burst read from a slow 8-bit external device.

Figure 26 shows the internal layout of the EBI.

Figure 26. Expansion Bus Interface Block Diagram



As shown in Figure 26, the EBI consists of two interfaces, the AHB2 interface and the EBI interface, which communicate with each other by means of the transaction FIFO buffer and the read-return FIFO buffer. The AHB2 interface receives transactions from masters on the AHB2 bus, which it posts to the transaction FIFO buffer. The EBI interface decodes the transactions and drives the appropriate signals from the memory interface based on the settings in the control registers. For read transactions, the EBI interface posts read data to the read-return FIFO buffer. For more details about the EBI FIFO buffers, see "EBI FIFO Buffers" on page 86.

AHB Slave Interface

The AHB2 interface consists of the AHB slave interface and the control and status registers. The AHB slave interface decodes bus transactions from the AHB2 bus masters and provides a response based on the settings for the EBI block that has been targeted for the transaction. The EBI block settings are maintained in the control registers. See "Registers" on page 95 for details on the EBI registers.

For control and status register read and write operations, the slave interface can operate at the full speed of the AHB2 clock, and requires no wait-stating. For writes or reads to an EBI block, the slave interface posts a transaction to the transaction FIFO buffer, and wait-states the AHB2 bus according to the settings for the block.

Write operations to an EBI block can complete without wait states being inserted on AHB2. The EBI status registers provide sufficient information to allow the EBI transaction sequencer to be monitored and operated safely.

For a non-split read, the slave interface stalls the AHB2 bus until the data from the read is available. Stalls for a slow device on the EBI can inhibit performance. Split transactions are supported to increase the utilization of the AHB2 bus while interfacing to slow peripherals on the EBI.

EBI Split Transactions

The EBI is capable of issuing a split response to AHB2 transactions that would otherwise tie up the bus for a long period, e.g., a long burst read from a slow 8-bit external device. See the *AMBA Specification* for details about split transactions.

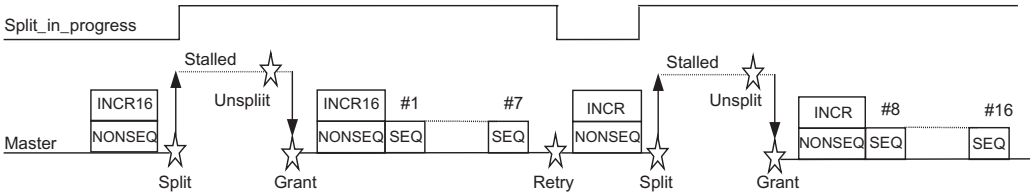
On receiving a read transaction, the AHB slave interface of the EBI issues a split response. The split response causes the arbiter to take ownership of the bus from the master that initiated the transaction. At the same time the EBI fills the read-return FIFO buffer with the data requested from the external device. Because the EBI is a narrow interface, it could take several cycles to fill the read-return FIFO buffer. When the read-return FIFO buffer is full and contains the data requested, the AHB slave interface signals that the data is ready and the arbiter no longer masks the request for the master.

A split response is issued to any master, other than the initiator, attempting to send a transaction to the EBI during a split transaction. The master can complete its transaction after the previous split transaction is finished.

If the amount of data requested exceeds the depth of the read-return FIFO buffer, the AHB slave interface issues a retry response to the transaction.

Figure 27 on page 86 shows a master attempting an INCR16 burst to the EBI. Because the read-return FIFO buffer is eight words, the 16-beat burst is broken up into two 8-beat bursts by the retry mechanism.

Figure 27. Split-Read INCR16 Transaction



Switching Between Split and Non-Split Operation

The device can be switched from non-split (default) operation to split operation without difficulty. When switching from split to non-split operation, the user must ensure that the EBI transaction FIFO buffer is empty, to provide a clean switch over. The EBI status register, `EBI_SR`, must be monitored, to check that a transaction is not in progress, so that a write to the control register, `EBI_CR`, can be performed in order to switch the split bit, `SP`. See "Registers" on page 95 for details on the EBI registers.

Do not enable split operations when peripherals with read side effects are connected to the EBI, because an undefined INCR transaction pre-fetches data from the EBI, which causes additional reads from the peripheral

EBI FIFO Buffers

The EBI contains two FIFO buffers, the transaction FIFO buffer and the read-return FIFO buffer; they provide a bridge for the AHB Interface and the EBI Interface.

Transaction FIFO Buffer

The transaction FIFO buffer provides buffering for transactions targeted for a specified EBI block. It has eight locations that are used to store transaction information. The AHB interface stores transaction address and control information in the transaction FIFO buffer. Data for the write transaction and the number of the AHB2 master that is performing the transaction is also stored in it. The AHB2 master number is needed for split transactions. The EBI transaction sequencer processes the contents of the transaction FIFO buffer.

Read-Return FIFO Buffer

The read-return FIFO buffer provides buffering for read transactions on the EBI. It contains up to eight locations to which the transaction sequencer writes. Multiple reads of an EBI peripheral might be needed to assemble a 32-bit word from two half-word transactions, or from four byte transactions, and deliver it to the read-return FIFO buffer. The AHB interface reads the contents of the FIFO buffer and returns data to masters on AHB2.

If split transactions are disabled, the read-return FIFO buffer has a depth of one word. If split transactions are enabled, it can fetch up to eight 32-bit values.

EBI FIFO Buffers and Non-Split and Split Read Transactions

For split single-beat reads, the AHB interface posts the transaction into the transaction FIFO buffer, and the EBI transaction sequencer performs the required number of reads to assemble and place the return value into the read-return FIFO buffer. The AHB interface immediately completes the split.

For a fixed-beat burst, the address phase of the burst is split, and the AHB interface posts a burst into the transaction FIFO buffer. The EBI sequencer assembles and places the read values into the read-return FIFO buffer. The transaction is unsplit when the required data has been read or the read-return FIFO buffer is full. Bursts longer than eight words are split into separate bursts.

EBI Interface

The EBI interface translates AHB transactions to memory accesses based on the settings of the control registers. The EBI interface consists of the timer and the EBI transaction sequencer, which are used in conjunction for EBI interface flexibility.

Timer

The timer is a binary counter that is used to time asynchronous memory access. In asynchronous operation, the timer counts the number of EBI_CLK cycles taken to receive an acknowledgement from an asynchronous device on the EBI and compares it with the acceptable timeout period, which is programmable via the EBI_CR register. If no acknowledgement is received before the expiration of the timeout period, the timer generates an interrupt to prevent the bus locking. The `EBI_INT_AddrSR` register maintains the address and byte access information of the memory location that caused the timeout. Clearing the interrupt from `EBI_INT_SR` restarts the timer.

EBI Transaction Sequencer

The transaction sequencer controls all of the external pins of the EBI. It does this by taking input from the transaction FIFO buffer output and using each entry as an instruction. The transaction sequencer performs a table lookup of the EBI block number to obtain key parameters for the sequence of memory accesses that it needs to make to process the instruction.

The EBI block number identifies timing information such as the polarity of chip-enables and the number of wait states for the memory access. The block number also provides other information such as whether byte enables are used for the access, whether the sequence is synchronous or asynchronous, and whether the device depends on `EBI_ACK` to control completions. See "Registers" on page 95 for details on the EBI registers.

Because the AHB2 interface and the EBI interface are different widths, the transaction sequencer is also responsible for assembling or disassembling transaction that traverse the interfaces.

Byte and Word Mapping for Size and Endianness

Devices on the EBI can be 8- or 16-bits wide. Transactions on AHB2 can be 8-, 16-, or 32-bits wide. If the size of the AHB transaction is larger than the EBI device width, the transaction sequencer makes multiple EBI accesses. If the AHB transaction is smaller than the EBI, the transaction sequencer masks the data read or, if appropriate, uses byte enables.

Tables 28 to 31 document the mapping of data in the AHB word to writes and reads to and from the EBI bus. Each block of the EBI can be 8- or 16-bits wide, and many permutations are possible. Byte-enables are provided to allow byte-writes to a 16-bit bus, although this does not work if the connected devices do not use byte-enables. The EBI outputs zeros on the byte that is not written.

Table 28. Write Mapping for Little-Endian Transmissions

		HWDATA				EBI_DQ		EBI_DQ	Transaction
HSIZE	HADDR[1..0]	31			0	15	0	7 0	
Byte(0)	00	B3	B2	B1	B0	0	B0	B0	EBI write
Byte(1)	01	B3	B2	B1	B0	B1	0	B1	EBI write
Byte(2)	10	B3	B2	B1	B0	0	B2	B2	EBI write
Byte(3)	11	B3	B2	B1	B0	B3	0	B3	EBI write
Hword(0)	0×	H3	H2	H1	H0	H1- H0		H0	First EBI write
								H1	Second EBI write
Hword(1)	1×	H3	H2	H1	H0	H3-H2		H2	First EBI write
								H3	Second EBI write
Word	××	W3	W2	W1	W0	W1 - W0		W0	First EBI write
						W3 - W2		W1	Second EBI write
								W2	Third EBI write
								W3	Fourth EBI write

Table 29. Write Mapping for Big-Endian Transmissions

		HWDATA				EBI_DQ		EBI_DQ	Transaction
HSIZE	HADDR[1..0]	31			0	15	0	7 0	
Byte(0)	00	B3	B2	B1	B0	B3	0	B3	EBI write
Byte(1)	01	B3	B2	B1	B0	0	B2	B2	EBI write
Byte(2)	10	B3	B2	B1	B0	B1	0	B1	EBI write
Byte(3)	11	B3	B2	B1	B0	0	B0	B0	EBI write
Hword(0)	0×	H3	H2	H1	H0	H3- H2		H3	First EBI write
								H2	Second EBI write
Hword(1)	1×	H3	H2	H1	H0	H1-H0		H1	First EBI write
								H0	Second EBI write
Word	××	W3	W2	W1	W0	W3 - W2		W3	First EBI write
						W1 - W0		W2	Second EBI write
								W1	Third EBI write
								W0	Fourth EBI write

Table 30. Read Mapping for Little-Endian Transmissions

		HWDATA				EBI_DQ		EBI_DQ	Transaction
HSIZE	HADDR[1..0]	31			0	15	0	7 0	
Byte(0)	00				B0	0	B0	B0	EBI read
Byte(1)	01			B1		B1	0	B1	EBI read
Byte(2)	10		B2			0	B2	B2	EBI read
Byte(3)	11	B3				B3	0	B3	EBI read
Hword(0)	0×			H1	H0	H1- H0		H0	First EBI read
								H1	Second EBI read
Hword(1)	1×	H3	H2			H3-H2		H2	First EBI read
								H3	Second EBI read
Word	×	W3	W2	W1	W0	W1 - W0		W0	First EBI read
						W3 - W2		W1	Second EBI read
								W2	Third EBI read
								W3	Fourth EBI read

Table 31. Read Mapping for Big-Endian Transmissions

		HWDATA				EBI_DQ		EBI_DQ	Transaction
HSIZE	HADDR[1..0]	31			0	15	0	7 0	
Byte(0)	00	B3				B3	0	B3	EBI read
Byte(1)	01		B2			0	B2	B2	EBI read
Byte(2)	10			B1		B1	0	B1	EBI read
Byte(3)	11				B0	0	B0	B0	EBI read
Hword(0)	0×	H3	H2			H3- H2		H3	First EBI read
								H2	Second EBI read
Hword(1)	1×			H1	H0	H1-H0		H1	First EBI read
								H0	Second EBI read
Word	××	W3	W2	W1	W0	W3 - W2		W3	First EBI read
						W1 - W0		W2	Second EBI read
								W1	Third EBI read
								W0	Fourth EBI read

Interface Signals

Table 32 lists the EBI signals. I/O standard options and the I/O mode options for the pins are programmable; they are set in the I/O control register, IOCR_EBI. For details, see page 168.

Table 32. EBI Signals

Signal	Direction	Description
EBI_CLK	Output	EBI clock pin
EBI_A[24..0]	Output	EBI address bus pins
EBI_DQ[15..0]	Input/Output	Bidirectional EBI data bus pins
EBI_BE[1..0]	Output	EBI byte-enable pins. EBI_BE0 is used for EBI_DQ[7..0]; EBI_BE1 is used for EBI_DQ[15..8]
EBI_CS[3..0]	Output	EBI chip-select pins. Chip selects correspond to memory map blocks EBI0, EBI1, EBI2, and EBI3. Programmable polarity; defaults to active-low CS_n
EBI_WE_n	Output	EBI write-enable pin
EBI_OE_n	Output	EBI output-enable pin
EBI_ACK	Input	EBI acknowledge pin

EBI Timing Diagrams

The following section shows timing diagrams for the EBI interface signals during synchronous and asynchronous operation. Figures 28 and 29 are only functional representations of the signals. For accurate representations of the EBI timing parameters. See "Electrical Characteristics & Timing Diagrams" on page 174.

Figure 28. Synchronous Read Cycle

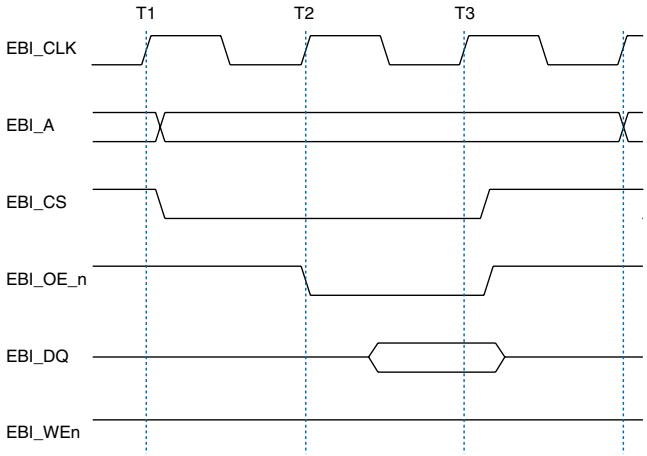
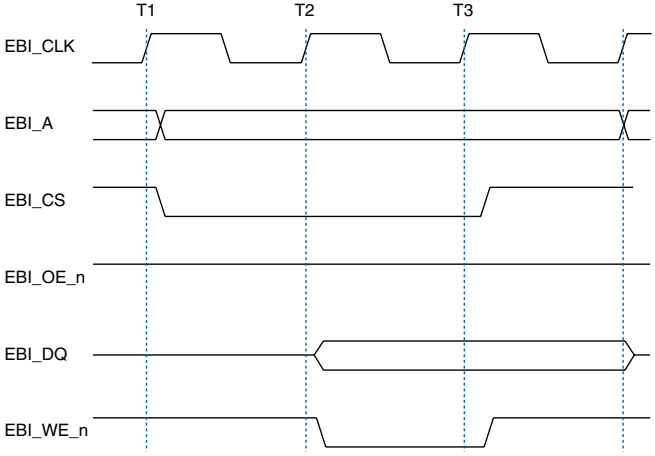


Figure 29. Synchronous Write Cycle



Figures 28 and 29 show the synchronous operation of the EBI interface. Signals are sampled on the rising edge of the EBI clock. For a read operation, when the chip has been selected, data is sampled from memory when `EBI_OE_n` is low. This occurs at T3 in Figure 28. For write transactions, when the chip has been selected, data is written to memory when `EBI_WE_n` is low.

If a block is configured as asynchronous, the EBI transaction sequencer presents the same signal sequence as a synchronous access, but samples EBI_ACK just before T3. If EBI_ACK is low, an additional T2 state is inserted. Figures 30 and 31 on page 93 show examples of extended asynchronous cycles, each with one extra T2 period. The clock is shown for reference only: EBI_CLK is still output, but is not used for asynchronous operation by the attached peripheral, although the scale factor (CLK_DIV in register EBI_CR) still has an important effect on cycle timing. The EBI interface relies on EBI_CLK to sample inputs and drive outputs. See "Registers" on page 95 for details on the EBI registers.

Figure 30. Asynchronous Read Cycle

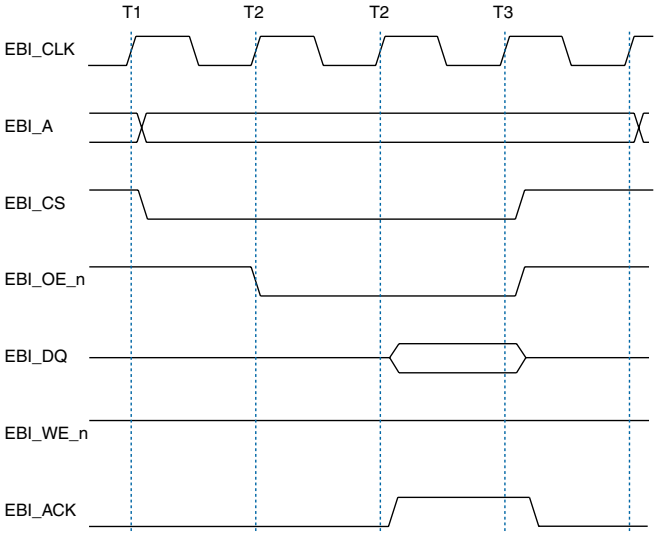
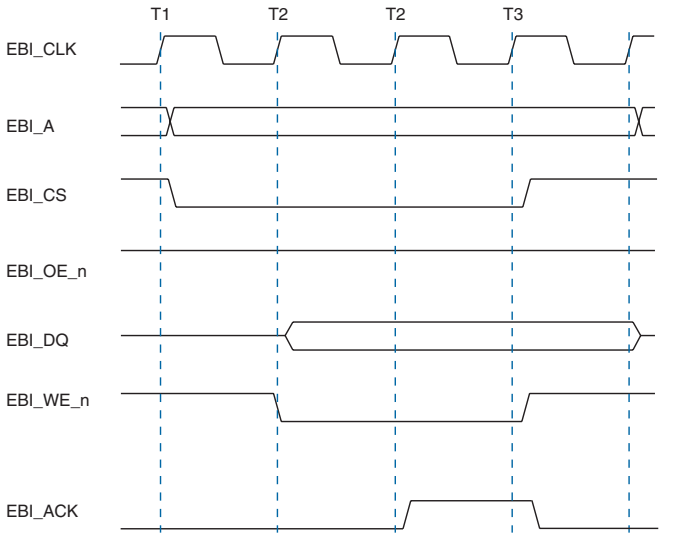


Figure 31. Asynchronous Write Cycle



EBI Clock

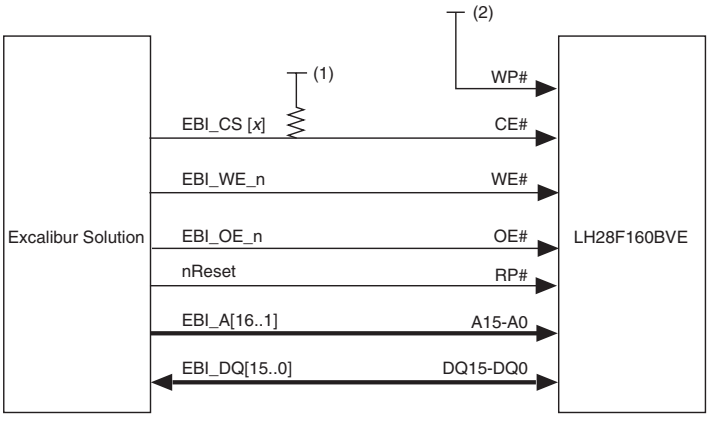
The EBI_CLK is a simple division of the AHB2 clock. Although it is always used internally by the EBI transaction sequencer, the EBI_CLK output can be enabled or disabled as required by the application. The AHB2 clock and the EBI_CLK divider must be configured before enabling the EBI_CLK output, to avoid driving too high a frequency on it. Writing to the clock register to change the programmable clock divider initiates a resynchronization cycle in the EBI. User application code must ensure that the EBI is not processing a transaction when the EBI speed is modified.

Additionally, when booting from flash it is important to set the correct number of wait states for EBI block 0 before reducing the EBI divide from its default value, to ensure that all accesses to the boot device are correctly timed.

Connecting the EBI to LH28F160BE-BTL90 Flash Memory

Figure 32 shows how to connect the EBI to an LH28F160BE-BTL90, which is a 3.3-V 16-bit flash memory with an active-low chip-select mechanism. EBI_CS[1..3] should be pulled high, because they are tri-stated during configuration.

Figure 32. Connecting the EBI to External Flash



(1) Only for EBI_CS[1], EBI_CS[2], EBI_CS[3]

(2) If high, lockable blocks are unlocked

Registers

Control and status registers are address-mapped and accessed by the system bus through the AHB slave-interface logic. Registers must be accessed with word-sized transactions, otherwise the EBI issues an ERROR response. EBI_CR, EBI_SR, EBI_INT_SR, and EBI_INT_ADDRSR relate to the whole EBI; registers EBI_BLOCK0 through EBI_BLOCK3 are for individual regions.

Unless otherwise stated, all register bits are zero after reset.

Register: EBI_CR
Address: Register base + 380H
Access: Write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											CE	TE	TIMEOUT								CLK_DIV				0	SP	EO	WP	OP	BP

BP	W	Byte enables polarity. EBI_BE0 and EBI_BE1 active level is specified here. 1—Active-high 0—Active-low
OP	W	Output enable polarity. EBI_OE active level is specified here. 1—Active-high 0—Active-low
WP	W	Write enable polarity. EBI_WE active level is specified here. 1—Active-high 0—Active-low
EO	W	Enable chip-select outputs 1-3. By default, EBI_CS0 is enabled and active-low, but all other chip-selects can have their polarity programmed and then enabled to provide a safe boot-up scenario for external SRAMs
SP	W	1—Split-read enable
CLK_DIV	W	Values 1 to 15 specify the scale factor between the AHB2 clock and EBI_CLK. Value 0 represents a scale factor of 16 (i.e., a 160-ns clock period from a 100-MHz AHB2 clock)
TIMEOUT	W	Controls the time delay before the transaction sequencer is forced out of T2 during an asynchronous transaction The timeout is a binary-coded number of EBI_CLK cycles
TE	W	Enables timeouts. If using asynchronous accesses without a timeout, reads can lock the AHB indefinitely
CE	W	Enables the external clock
R	W	Reset bit. A write with this set causes the EBI to re-initialize
0	W	Reserved for future use. Write as 0 to ensure future compatibility

The default value of this register is 0H, which provides the slowest-possible access for a synchronous device on EBI0.

Register: EBI_SR
Address: Register base + 380H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XP	XF	XE	RF	RE	0							CE	TE	TIMEOUT								CLK_DIV				0	SP	EO	WP	OP	BP

BP	R	Byte enables polarity. EBI_BE0 and EBI_BE1 active level is specified here. 1—Active-high 0—Active-low
OP	R	Output enable polarity. EBI_OE active level is specified here. 1—Active-high 0—Active-low
WP	R	Write enable polarity. EBI_WE active level is specified here. 1—Active-high 0—Active-low
EO	R	Enable chip-select outputs 1-3. By default, EBI_CS0 is enabled and active-low, but all other chip-selects can have their polarity programmed and then enabled to provide a safe boot-up scenario for external SRAMs
SP	R	1—Split-read enable
CLK_DIV	R	Values 1 to 15 specify the scale factor between the AHB2 clock and EBI_CLK. Value 0 represents a scale factor of 16 (i.e., a 160ns clock period from a 100-MHz AHB2 clock)
TIMEOUT	R	Controls the time delay before the transaction sequencer is forced out of T2 during an asynchronous transaction The timeout is a binary-coded number of EBI_CLK cycles
TE	R	Enables timeouts. If using asynchronous accesses without a timeout, reads can lock the AHB indefinitely
CE	R	Enables the external clock
RE	R	Indicates the read-return FIFO buffer is empty
RF	R	Indicates the read-return FIFO buffer is full
XE	R	Indicates the transaction FIFO buffer is empty
XF	R	Indicates the transaction FIFO buffer is full
XP	R	Indicates a transaction is in progress
0	R	Reserved for future use. Write as 0 to ensure future compatibility

EBI_SR resides at the same location as EBI_CR and therefore contains the control register bits. The status bits reside in the high-order bits of EBI_SR.

Register: EBI_INT_SR
Address: Register base + 3A0H
Access: Read/clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																															TOI

TOI	R/C	Indicates a timeout interrupt occurred
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: EBI_INT_ADDRSR
Address: Register base + 3A4H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3	B2	B1	B0	0			ADDRESS																								

ADDRESS	R	The EBI address at which a timeout occurred
B0	R	1—Timeout occurred for EBI0
B1	R	1—Timeout occurred for EBI1
B2	R	1—Timeout occurred for EBI2
B3	R	1—Timeout occurred for EBI3
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: EBI_BLOCK0
Address: Register base + 390H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																								BE	BH	CP	WAIT				SA

SA	R/W	0—Synchronous 1—Asynchronous
WAIT	R/W	Block transaction wait states. The value entered in **WAIT** is a binary-coded number of wait states to be inserted into all transactions for this block. The effect of wait states is identical to the function of ACK: the EBI transaction sequencer inserts a **WAIT** number of T2 states into each transaction. By default 0 wait states are added; up to 15 additional T2 cycles can be configured
CP	R/W	Chip-select polarity. 0—Active-low 1—Active-high
BH	R/W	Byte- or half-word width select. 0—Half-word 1—Byte
BE	R/W	1—When **BH** is 0, turns on byte enables
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The default value of EBI_BLOCK0 is 40H in 8-bit boot-from-flash modes. It is 0H in other boot modes.

The following registers are identical to EBI_BLOCK0 except that their default value is always 0.

Register: EBI_BLOCK1
Address: Register base + 394H
Access: Read/write

Register: EBI_BLOCK2
Address: Register base + 398H
Access: Read/write

Register: EBI_BLOCK3
Address: Register base + 39CH
Access: Read/write

Embedded Peripherals Memory Map

The elements listed in Table 33 are present as memory-mapped slave peripherals in the embedded stripe. The base address and range for each element can be configured. A memory range can be set to any size that is a power of two.

Table 33. Memory Map Peripherals & Elements

Memory Map Element	Range *Note (1)*		
	EPXA1	EPXA4	EPXA10
Registers, including those for embedded stripe peripherals	16 Kbytes		
Internal SRAM0, SRAM1 (total)	64 Kbytes	128 Kbytes	256 Kbytes
Internal dual-port DPSRAM0, DPSRAM1 (total)	16 Kbytes	64 Kbytes	128 Kbytes
EBI0, EBI1, EBI2, EBI3	16 Kbytes to 32 Mbytes		
SDRAM0, SDRAM1	16 Kbytes to 256 Mbytes		
PLD0, PLD1, PLD2, PLD3	16 Kbytes to 2 Gbytes		

Notes:
(1) Range is for each memory map element

☞ Ensure that memory ranges do not overlap. If an address decodes into more than one range, the results are undefined (ranges may temporarily overlap during reconfiguration, providing that no accesses are made to that range).

☞ During reconfiguration, ranges can temporarily overlap without harm; however, address decodes to overlapping regions remain undefined.

Memory Mapping in Boot-From-Flash Mode

When the device is booting in boot-from-flash mode, the initial mapping presents EBI0 at base address 0H with 32 Kbytes of addressable space. The mapping is enabled after reset and can subsequently be changed or disabled.

Memory Map Control Registers

The memory map control registers are accessed from AHB2. They can be accessed by the embedded processor, by the configuration logic master, or by a peripheral via the PLD-to-stripe bridge. Only word accesses to the memory map control registers are allowed; half-word or byte accesses generate a bus error.

To ensure that any changes have taken effect, the user must either disable write-posting in the bridge or follow each write with a safe-read instruction that will pass through the AHB1-2 bridge. The new address map does not take effect until any posted write outstanding on AHB2 has completed.

Range Definition Registers

Each memory range is controlled by a register that sets the base address, the size, and the types of access permitted. These registers all have the same format, as shown below:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE																		0		SIZE					0					NP	EN

EN	R/W	Enable. Setting this bit enables the decoding of this memory range
NP	R/W	No pre-fetch. Controls how a bridge can behave when accessing this region. If it is clear (0), accesses to this region read ahead (up to the next 1 Kbyte boundary) and writes might be delayed or aggregated for speed. If the bit is set (1), any access to the region is delayed until the previous access has completed
SIZE	R/W	Region size expressed as $\log_2$(region size) - 1. For example, 19 for a 1-Mbyte region
BASE	R/W	Base address for the regions. Bits [31..14] define the base address for this region. Minimum range is 16 K
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The minimum size of each enabled region is 16 Kbytes. If `SIZE` is set to a value less than 13 (i.e., less than 16 Kbytes), the address decoding logic defaults to 13. Setting `BASE` to a value that is not a multiple of the region size or a multiple of the underlying size produces undefined results.

For some registers, SIZE and NP are read-only. Table 34 lists their values.

Table 34. SIZE and NP Fields in the Range Definition Registers

Range Definition Register	Size	NP
MMAP_REGISTERS	13	1
MMAP_SDRAM0, MMAP_SDRAM1	-	0
MMAP_SRAM0, MMAP_SRAM1 (internal SRAM)	-	0
MMAP_DPSRAM0, MMAP_DPSRAM1 (internal dual-port RAM)	-	0
MMAP_EBI0, MMAP_EBI1, MMAP_EBI2, MMAP_EBI3	-	-
MMAP_PLD0, MMAP_PLD1, MMAP_PLD2, MMAP_PLD3 (PLD ranges)	-	-

Register: MMAP_REGISTERS
Address: Register base + 80H
Access: Read/write

At reset, the read/write bits in all range registers are set to zero, except for the registers' base address, which is enabled at address 7FFFC000H. The size of the registers region is fixed at 16 Kbytes.

Clearing the enable bit, EN, in register MMAP_REGISTERS or the register-enable bit, RE, in register BOOT_CR (see page 162), or disables the registers region. The device must be reset to re-enable it.

Other memory map regions registers are listed in Table 35. Writing to these registers perform the functions as defined.

Table 35. Memory Map Regions Registers *(Part 1 of 2)*

Register Name	Address *(1)*	Access:
MMAP_SRAM0 (on-chip SRAM block 0)	90H	R/W
MMAP_SRAM1 (on-chip SRAM block 1)	94H	R/W
MMAP_DPSRAM0 (dual-port SRAM block 0)	A0H	R/W
MMAP_DPSRAM1 (dual-port SRAM block 1)	A4H	R/W
MMAP_SDRAM0 (SDRAM block 0)	B0H	R/W
MMAP_SDRAM1 (SDRAM block 1)	B4H	R/W
MMAP_EBI0 (EBI block 0)	C0H	R/W
MMAP_EBI1 (EBI block 1)	C4H	R/W
MMAP_EBI2 (EBI block 2)	C8H	R/W
MMAP_EBI3 (EBI block 3)	CCH	R/W
MMAP_PLD0 (PLD region 0)	D0H	R/W

Table 35. Memory Map Regions Registers (Part 2 of 2)		
Register Name	**Address (1)**	**Access:**
MMAP_PLD0 (PLD region 1)	D4H	R/W
MMAP_PLD0 (PLD region 2)	D8H	R/W
MMAP_PLD0 (PLD region 3)	DCH	R/W

Note:
(1) Address is the offset from the base address of the register region

Four memory ranges are provided, which select the PLD bridge. The select signals for these four regions are combined in the bridge and are not made available separately to the PLD. The four regions provide the flexibility to map the four possible regions, with write-posting and read pre-fetching turned on or off.

UART

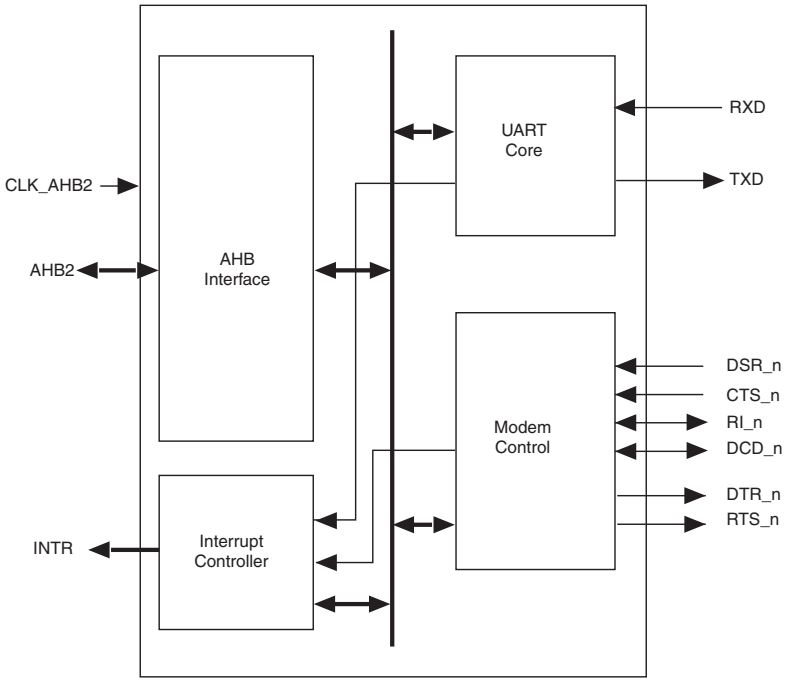
The universal asynchronous receiver transmitter module (UART) performs serial-to-parallel conversion on data characters received from a peripheral device or modem, and parallel-to-serial conversion on data characters received from the embedded processor. The UART operates in FIFO mode, with the FIFO buffers having a depth of 16 bytes. The CPU can read the status of the UART at any time during operation. The UART reports status information, including the type and condition of the transfer being performed, and any error conditions.

The UART has the following features:

- 5 to 8 data bits
- 1 or 2 stop bits
- Even, odd, stick, or no parity
- 75 to 230,400 baud rate
- 16-byte transmit FIFO buffer
- 16-byte receive FIFO buffer
- Programmable baud generator divides any input clock by 2 to 65535 and generates the 16 × baud clock
- Transmit FIFO buffer interrupt for empty indication and transmitter idle indication
- False-start bit detection
- Internal diagnostic capabilities
  - Loop-back control for communications-link fault isolation
  - Break insertion and detection in loop-back mode
- Modem communication support

Figure 33 on page 103 shows the layout of the UART.

Figure 33. UART Block Diagram



UART Pins and Signals

The UART input and output signals are 1 bit. The function of some signals depends on whether the UART is functioning at the terminal or modem end of the RS232 link. In these cases, the names of the RTS and CTS pins are exchanged, as are the names of the DSR and DTR pins.

Table 36 on page 104 lists the UART input and output signals. Unless otherwise stated, all signals are register driven and active-low. In EPXA10 and EPXA4 devices, the UART pins are shared and are available as PLD I/O if the stripe UART is not used. In EPXA1 devices, the UART pins are not shared and are dedicated to the UART function.

Table 36. UART Signals

Signal Name	Direction	Description
UART_RXD	Input	Serial data input signal to stripe UART
UART_DSR_n	Input	Data set ready, active-low signal. When active, indicates that the peer device is ready to establish the communications link with the UART. (When acting as a modem, UART_DSR_n is used as a DTR input)
UART_CTS_n	Input	Clear-to-send, active-low signal. When active, indicates that the peer device can accept characters
UART_DCD_n	Input/ Output	Data carrier detect, active-low signal. When active, indicates that the data carrier is being detected by the modem. This pin is an input when OE in UART_MC is 0, and output when OE in UART_MC is 1
UART_RI_n	Input/ Output	Ring indicator, active-low signal. When active, indicates that a telephone ringing signal is being received by the modem. This pin is an input when OE in UART_MC is 0, and output when OE in UART_MC is 1
UART_TXD	Output	Serial data output signal to the communications link. On reset, UART_TXD is set high
UART_RTS_n	Output	Request-to-send, active-low signal. When active, informs the peer device that the UART is ready to receive data
UART_DTR_n	Output	Data terminal ready, active-low signal. When active, indicates that the UART is ready to establish a communications link

Reset Behavior

The UART has four modem control signals and a receive input which are all active-low.

In EPXA10 and EPXA4 devices following a reset, the UART pins, which are shared I/O, are in PLD I/O mode and are low. This causes the UART to generate a modem interrupt, believing that a character has been received. To overcome this, all interrupts must be cleared following a reset, and the receiver must be flushed before interrupts are enabled.

This behavior does not apply to EPXA1 devices, because the EPXA1 UART pins are not shared. They are dedicated to the UART function.

Transmitter Operation

Data written to the transmit data register, UART_TD, is queued in the transmit FIFO buffer ready for transmission. The transmit FIFO buffer level, TX_LEVEL, in the transmit status register, UART_TSR, indicates the number of bytes currently stored in the transmit FIFO buffer.

Data written to UART_TD when the FIFO buffer is full is lost. If the transmit interrupt-enable bit, TE, is set in the interrupt-enable set register, UART_IES, a transmitter interrupt is generated when the number of bytes in the transmit FIFO buffer falls to the level equal to the transmit threshold level field, TX_THR, of the FIFO buffer control register, UART_FCR.

Setting the transmit idle interrupt enable bit, TIE, in the interrupt-enable set register, UART_IES, causes an interrupt when the transmitter becomes idle after sending the last byte in the transmit FIFO buffer. The transmitter becomes idle when there is no data in the transmit FIFO buffer and the transmit shift register becomes empty.

The transmit idle (TII) and transmit interrupt (TI) are both cleared by reading UART_TSR.

Receiver Operation

Received data is stored in the receive FIFO buffer and is removed by reading the received data register, UART_RD.

If the receive interrupt bit, RE, of the interrupt-enable set register, UART_IES, is set, an interrupt is generated when the number of bytes in the FIFO buffer reaches the number equal to the receive threshold level, RX_THR, in the FIFO buffer control register, UART_FCR. This interrupt is cleared by reading the receive status register, UART_RSR, which indicates the number of bytes currently in the FIFO buffer.

A receive interrupt is also generated when RE is set, the receive FIFO buffer is not empty, and no further data has been received after 32 UART bit-times.

Errors occurring during the reception of the next byte are indicated in UART_RDS, and must be read before the next byte is read. See "Registers" on page 107 for details.

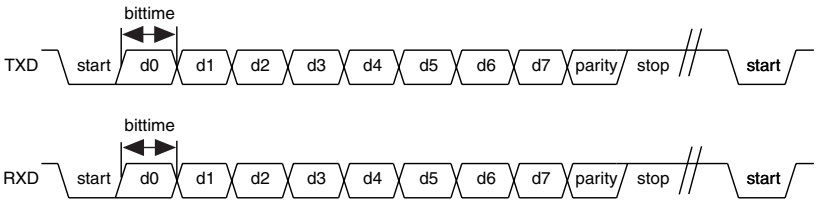
Modem Status Lines

The value of the modem status lines which are inputs can be tested by reading the UART_MSR lines. The values of the modem status outputs are controlled by bits within UART_MCR.

UART Data Formats

Figure 34 shows the data format when an 8-bit word with parity has been selected.

Figure 34. UART Data Format



BAUDRATE = CLK_AHB2/(divisor × 16) bits/sec
BITTIME = 1/BAUDRATE

Table 37 gives a brief description of the data signal components.

The UART can be set to force de-assertion on the UART_RTS_n pin when there is no space in the receive FIFO buffer, or to stop transmitting characters when the UART_CTS_n pin is de-asserted.

Table 37. Data Signal Components

START_BIT	UART_TXD and UART_RXD are normally high. When a character is being transmitted, UART_TXD is driven low for the duration of 1-bit time. The receiver always samples the UART_RXD line. When it detects a start bit, it starts shifting a new character in
DATA	A character can be programmed for 5 to 8 bits. Both transmitting and receiving UARTs must be programmed for the same settings, otherwise communication fails
PARITY	Parity generation and checking can be enabled or disabled. If parity is disabled, no parity bit is transmitted, and the receiver does not expect to receive a parity bit. If parity is enabled, it can be even, odd, or stick parity: Even parity—Parity bit is 1, if the character has an odd number of 1s Odd parity—Parity bit is 1, if the character has an even number of 1s Stick parity—Parity bit can be forced to 1 or 0
STOP BIT	Stop bits are the last bits to be transmitted or received for each character. A stop bit is a 1. The number of stop bits can be programmed to be 1- or 2-bit times. Stop bits act like a spacer between characters if they are transmitted back-to-back. Both receiving and transmitting UARTs must be programmed for the same settings. The UART only checks the first stop bit
BREAK	A break is detected if UART_RXD is held low longer than a character-time. (A character-time is the time to transmit or receive a character including start, parity and stop bits.) This usually happens if UART_RXD is disconnected, or the transmitting UART forced a break or is turned off. A break can be forced by setting the break bit in the modem control register, UART_MCR

Registers

At reset, all registers hold the value 0 unless otherwise specified.

Register: UART_RSR (receive status register)
Address: Register base + 280H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																								RE	0			RX_LEVEL			

RX_LEVEL	R*	The number of bytes in the receive FIFO buffer
RE	R	Receive error. This bit is set when there is at least one parity error, framing error, break indication, or overrun error at any location in the receive FIFO buffer
0	R	Reserved for future use

Reading this register resets the receive-interrupt bit, RI, in UART_ISR.

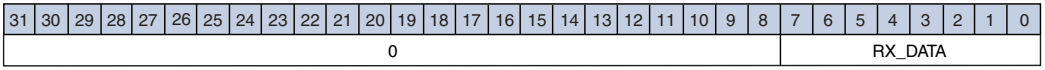
Register: UART_RDS (received data status)
Address: Register base + 284H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																												BI	FE	PE	OE

OE	R	Overrun error. Set when a receive-overrun occurs. This happens if the receive FIFO buffer is full and a character is received into the shift register, destroying the data currently in it. This status is associated with the character after the one which was lost due to overrun
PE	R	Parity error. Set if the received parity differs from the expected value. The parity error applies to the data word at the top of the FIFO buffer
FE	R	Framing error. Set if a valid stop bit is not detected. This status is always valid for the character currently at the top of the FIFO buffer
BI	R	Break indicator. Set if a break is received. This occurs when RXD is low for more than one character transmission time (from start bit to stop bit): a single 0 is received. This status is valid with the 0 character; one break-indicator flag and 0 is loaded into the receive FIFO buffer. The next character is only written into the receive FIFO buffer when the next valid start bit is detected
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The above errors are associated with the particular character in the FIFO buffer they apply to. The error is revealed when its associated character is at the top of the FIFO buffer.

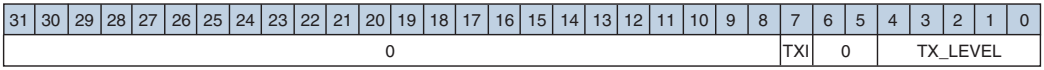
Register: UART_RD (received data)
Address: Register base + 288H
Access: Read



RX_DATA	R	Receive data
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The contents of the receive FIFO buffer are not cleared when RC in UART_FCR is set. If a read from the receive FIFO buffer happens directly after reset, and no data has been written to it, the data read is the data last stored at FIFO buffer location 0. When the FIFO buffer is full, no more data can be written into the FIFO buffer.

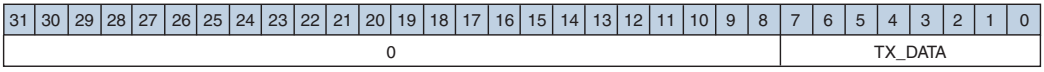
Register: UART_TSR (transmit status register)
Address: Register base + 28CH
Access: Read



TX_LEVEL	R	Transmit FIFO buffer level (the number of characters in the transmit FIFO buffer)
TXI	R	Transmitter idle. Set when the transmitter shift register becomes empty and there are no more characters in the transmit FIFO buffer. Cleared when UART_TSR is read
0	R	Reserved for future use

Reading this register clears TI and TII in UART_ISR (see page 110).

Register: UART_TD (transmit data)
Address: Register base + 290H
Access: Write



TX_DATA	W	Transmit data
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Each write to this register stores a character in the transmit FIFO buffer.

Register: UART_FCR (FIFO buffer control register)
Address: Register base + 294H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																								RX_THR			TX_THR			RC	TC

TC	R/W	Clear transmit FIFO buffer. TC is always read as 0. TC is self-clearing
RC	R/W	Clear receive FIFO buffer. RC is always read as 0 and is self-clearing
TX_THR	R/W	Transmit threshold level. The threshold level encoding is as follows: 000—0 001—2 010—4 011—8 100—10 101—12 110—14 111—15
RX_THR	R/W	Receive threshold level.The threshold level encoding is as follows: 000—1 001—2 010—4 011—6 100—8 101—10 110—12 111—14
0	R	Reserved for future use. Write as 0 to ensure future compatibility

When the receive FIFO buffer depth is equal to, or greater than, the number of characters programmed in RX_THR, the receive-interrupt bit, RI, in UART_ISR is set.

When the transmit FIFO buffer depth is equal to, or less than, the number of characters programmed in TX_THR, the transmit-interrupt bit, **TI**, in UART_ISR is set.

Writing 1 to the clear-receive bit, RC, clears the receive FIFO buffer counters. The shift register is not cleared.

Writing 1 to the clear-transmit bit, TC, clears the transmit FIFO buffer counters and sets the TII interrupt. The shift register is not cleared.

Register: UART_IES (interrupt-enable set register)
Address: Register base + 298H
Access: Read/set

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												ME	TIE	TE	RE

RE	R/S	Receive-interrupt enable
TE	R/S	Transmit-interrupt enable
TIE	R/S	Transmit-idle-interrupt enable
ME	R/S	Modem-status-interrupt enable
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Reading UART_IES indicates which bits of the interrupt mask are set.

Register: UART_IEC (interrupt-enable clear register)
Address: Register base + 29CH
Access: Read/clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												ME	TIE	TE	RE

RE	R/C	Clear receive-interrupt enable
TE	R/C	Clear transmit-interrupt enable
TIE	R/C	Clear transmit-idle-interrupt enable
ME	R/C	Clear modem-status-interrupt enable
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Reading UART_IEC indicates which bits of the interrupt mask are set.

Register: UART_ISR (interrupt status register)
Address: Register base + 2A0H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												MI	TII	TI	RI

RI	R	Receive interrupt. Set either when there has been a received-character timeout or the received-data flag goes from low to high. Cleared by reading UART_RSR
TI	R	Transmit interrupt. Set when the number of characters in the transmit FIFO buffer goes from being more than the transmit threshold to being equal to or less than the transmit threshold. (The transmit threshold is TX_THR in UART_FCR). Cleared by reading UART_TSR
TII	R	Transmitter Idle interrupt. Set when there is no data in the transmit FIFO buffer and the transmit shift register becomes empty. Cleared by reading UART_TSR
MI	R	Modem-status interrupt. Set when any of DDCD, TERI, DDSR or DCTS bits within UART_MSR are set. Cleared by reading UART_MSR
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The received data flag goes high when the level of the receive FIFO buffer is equal to or greater than the received threshold level.

The received-character timeout is an internal timeout signal, which is asserted when the receive FIFO buffer is not empty and no further data has been received over a 32-bit period.

Register: UART_IID (interrupt ID register)
Address: Register base + 2A4H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																													IID		

IID	R	Interrupt ID: 000 - no interrupts pending 001 RI *highest priority* 010 TI *next* 011 TII *next* 100 MI *next* If more than one category of interrupt is asserted, only the highest priority interrupt ID is given
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: UART_MC (mode-configuration register)
Address: Register base + 2A8H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																									OE	SP	EP	PE	ST	CLS	

OE	R/W	Controls the behavior of DCD and RI pins. When it is 1, DCD and RI are outputs controlled from the UART_MCR. When it is 0, they are inputs whose status is reflected in UART_MSR
SP	R/W	Stick parity. Forces the parity bit to either 1 or 0
EP	R/W	Selects between even parity and odd parity. When even parity is selected, the number of 1s (that is, data plus parity) is even. When odd parity is selected, the number of 1s (that is, data plus parity) is odd
PE	R/W	Parity enable. Selects whether parity is added (on transmit) and checked (on receive)
ST	R/W	Stop bits. Selects the number of stop bits transmitted: 0—1 stop bit 1—2 stop bits
CLS	R/W	Selects the number of bits used to specify character-length: 00—5 bits 01—6 bits 10—7 bits 11—8 bits
0	R	Reserved for future use. Write as 0 to ensure future compatibility

CLS selects the length of transmitted and received characters. For character lengths less than 8 bits, the least significant bits in UART_TD and UART_RD define the character, and the most significant bits are ignored on transmit and set to zero on receive.

ST selects the number of stop bits transmitted. The receiver checks only the first stop bit, regardless of the number of stop bits transmitted.

Table 38 summarizes how the interactions between PE, EP and SP affect parity mode configuration.

Table 38. Mode Configuration Bits

SP	EP	PE	Description
X	X	0	No parity
0	0	1	Odd parity
0	1	1	Even parity
1	0	1	'1' parity
1	1	1	'0' parity

Register: UART_MCR (modem control register)
Address: UART Register base + 2ACH
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																								AC	AR	BR	LB	DCD	RI	DTR	RTS

AC	R/W	Auto CTS. When set, the transmitter does not start transmitting a character unless CTS in UART_MSR is asserted. It continues to transmit the current character if CTS changes state during a character.
AR	R/W	Auto RTS. When set, RTS is forced to the de-asserted state when there are 16 bytes in the receive FIFO buffer (indicating to the transmitter that it cannot accept new data). When there are fewer than 16 bytes in the receive FIFO buffer, the state of AR is ignored. RTS also depends on the value programmed in UART_MCR
BR	R/W	Transmit break. Forces TXD to 0 immediately if no serial data is being transmitted. If data is currently being transmitted, TXD is forced to 0 after the current contents of the transmit shift register have been transmitted. The transmitter is not stopped when this bit is set
LB	R/W	When set, puts the UART into loop-back mode
DCD	R/W	Data carrier detect output. Controls the state of the DCD pin when it is an output. When DCD is 1, DCD_n is set active (i.e., low)
RI	R/W	Ring indicator output. Controls the state of the RI pin, when it is an output. When RI is 1, RI_n is set active (i.e., low)
DTR	R/W	Data terminal ready. Controls the state of the DTR pin. When it is 1, DTR_n is set active (i.e., low)
RTS	R/W	Request to send. Controls the state of the RTS pin. When it is 1, RTS_n is set active (i.e., low). Can be forced into an inactive state if AR is set
0	R	Reserved for future use. Write as 0 to ensure future compatibility

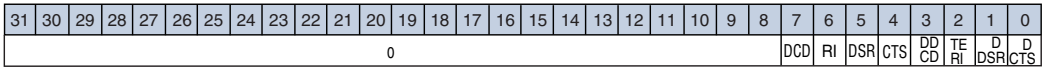
Ensure that the transmit FIFO buffer is empty before setting BR, because data in the transmit FIFO buffer might be lost or corrupted when it is set. Data in the transmit shift register is not affected.

In loop-back mode, the output pins are set high (inactive) and the input pins are ignored. Table 39 shows how the output signals from the UART are connected to the inputs.

Table 39. Input-Output Connections

Output	Connected to Input
UART_TXD	UART_RXD
UART_RTS_n	UART_CTS_n
UART_DTR_n	UART_DSR_n
UART_RI_n output	UART_RI_n input
UART_DCD_n output	UART_DCD_n input

Register: UART_MSR (modem status register)
Address: Register base + 2B0H
Access: Read



DCTS	R	Set when the CTS_n pin changes state
DDSR	R	Set when the DSR_n pin changes state
TERI	R	Set when the RI_n pin changes from low to high
DDCD	R	Set when the DCD_n pin changes state
CTS	R	Set when the CTS_n pin is at a low value
DSR	R	Set when the DSR_n pin is at a low value
RI	R	Set when the RI_n pin is at a low value
DCD	R	Set when the DCD_n pin is at a low value
0	R	Reserved for future use. Write as 0 to ensure future compatibility

When the DCD_n and RI_n pins are selected as outputs, the appropriate bits in this register are always 0.

When any of the bits DDCD, TERI, DDSR and DCTS are set, the modem-status interrupt bit, MI, is set in UART_ISR.

Reading this register resets DDCD, TERI, DDSR and DCTS to zero (and clears MI).

Register: UART_DIV_LO (divisor register)
Address: Register base + 2B4H
Access: Read/write

UART_DIV_LO is identical to UART_DIV_HI, shown on page 114.

Register: UART_DIV_HI (divisor register)
Address: Register base + 2B8H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																								DIV							

DIV	R/W	Half of the divisor value. The least significant byte is held in UART_DIV_LO and the most significant byte is held in UART_DIV_HI
0	R	Reserved for future use. Write as 0 to ensure future compatibility

To load a value, UART_DIV_LO must be loaded before UART_DIV_HI. The values in these registers combine to form the divisor latch value, UART_DIV, which is used in the clock divider to generate the UART baud clock. The baud rate generated by the UART is the AHB2 clock frequency divided by (UART_DIV × 16).

If a divisor value of 0 or 1 is programmed, the baud rate divisor divides by 2. For example, to generate a baud rate of 230,400 from an AHB2 clock of 33 Mhz the ideal divisor is:

33,000,000/(16 × 230,400) = 8.95

A programmed value of 9 gives a 0.5% error from the ideal baud rate, which is comfortably within the bounds allowed by the RS232 specification.

Timer

The timer is a dual-channel timer, with the following features:

- 32-bit clock pre-scaler
- 32-bit timer register
- Three operating modes, selectable under register control:
  - Free-running interrupt (heartbeat)
  - Software controlled start/stop (interval timer) with interrupt on limit
  - One-shot interrupt after programmable delay

Figure 35 on page 115 shows a block diagram of the timer module.

Figure 35. Timer Module Block Diagram

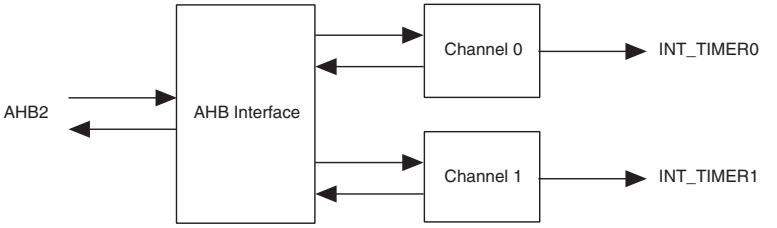
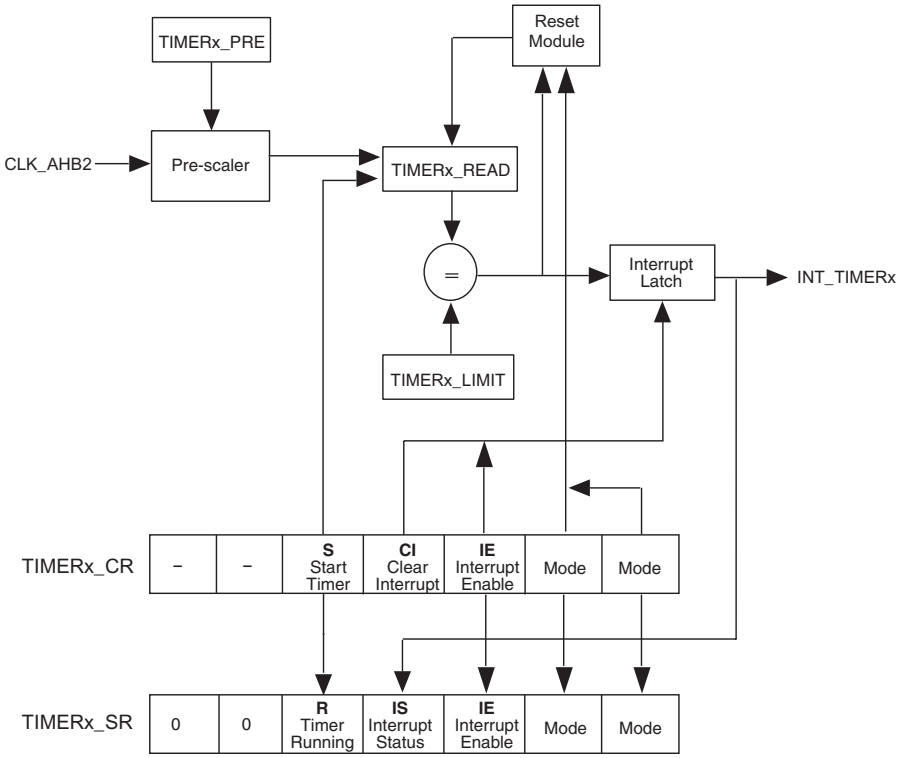


Figure 36. Individual Timer Channel Block Diagram



Initialization

The TIMER*x*_LIMIT register holds the value of the timer limit. A non-zero value must be written to the TIMER*x*_LIMIT register before enabling the interrupt, otherwise zero reset values occurring simultaneously in both TIMER*x*_READ and TIMER*x*_LIMIT cause an interrupt as soon as interrupts are enabled.

Pre-scaler

A 32-bit clock pre-scaler ratio register, TIMER*x*_PRE, is used to configure the pre-scaler which generates a periodic clock-enable input to the 32-bit timer, incremented by the AHB2 clock.

The pre-scaler resets to zero and counts until it equals the value in the pre-scaler ratio register, TIMER*x*_PRE. At this point, the timer register is then incremented, the pre-scaler resets to zero and the cycle begins again.

A value of zero in the pre-scaler ratio register permanently asserts the clock-enable to the timer register, resulting in the timer incrementing at the AHB2 clock rate. In addition, the actual timer period is one clock cycle longer than the period specified in TIMER*x*_LIMIT; logically, this can be expressed as follows:

```
if (PRESCALE = 0)
then
  period=(LIMIT + 1) / AHB2_frequency
else
  period=LIMIT * (PRESCALE + 1)/AHB2_frequency
endif
```

Free-Running Heartbeat Mode

In free-running mode, the timer is reset to 0 when the start bit, **S**, in TIMER*x*_CR changes from 0 to 1. The timer then increments until it reaches the value LIMIT in TIMER*x*_LIMIT, at which point it is reset to 0 and begins incrementing again. This cycle repeats while **S** remains set. An interrupt is requested (if enabled) at the end of each cycle.

One-Shot Delay Mode

In one-shot mode, the timer is reset to 0 when the start bit, **S**, in TIMER*x*_CR changes from 0 to 1. The timer then increments until it reaches the value LIMIT in TIMER*x*_LIMIT, at which point the timer stops, S is cleared, and an interrupt is requested (if enabled). No further activity takes place.

Software Interval Timer Mode

In interval timer mode, the timer is reset to 0 when the start bit, S, in TIMERx_CR changes from 0 to 1. The timer then increments until S is cleared, at which point the timer is frozen. An interrupt is requested (if enabled) when the timer passes through the value LIMIT in TIMERx_LIMIT. If the timer value reaches FFFFFFFFH, it wraps around to 0 and continues incrementing.

Interrupts

When asserted, interrupts remain asserted until explicitly cleared by setting the clear-interrupt bit, CI, in TIMERx_CR, or until the timer module is reset by a system rest. This corresponds to a level-sensitive interrupt scheme.

Software should use a read-modify-write sequence to set CI and preserve all other bit settings in TIMERx_CR.

Registers

Each timer channel has an identical register set. Suffixes 0 and 1 refer to channels 0 and 1 respectively. At reset, all internal registers, including interrupt outputs, are cleared.

Register: TIMER0_CR (timer 0 control register)
Address: Register base + 200H
Access: Write

Register: TIMER1_CR (timer 1 control register)
Address: Register base + 240H
Access: Write

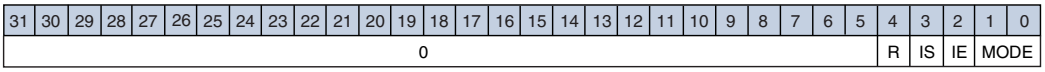
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																											S	CI	IE	MODE	

MODE	W	Timer mode: 00—Free running heartbeat mode 01—One shot delay 10—Software interval timer 11—Reserved
IE	W	Interrupt-enable
CI	W	Write 1 to clear pending interrupt
S	W	Write 1 to start timer, '0' to stop timer
0	W	Reserved for future use. Write as 0 to ensure future compatibility

S and CI can be written at any time. Other bits can only be changed (that is, written with new values) when the timer is stopped. If bits are changed while the timer is running, the behavior of the timer will be indeterminate.

Register: TIMER0_SR (timer 0 status register)
Address: Register base + 200H
Access: Read

Register: TIMER1_SR (timer 1 status register)
Address: Register base + 240H
Access: Read

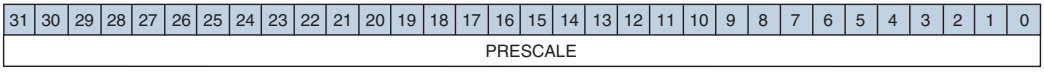


MODE	R	Timer mode
IE	R	Interrupt-enable
IS	R	Interrupt is active
R	R	1 if timer is running
0	R	Reserved for future use

The status register can be read at any time.

Register: TIMER0_PRE (timer 0 pre-scaler ratio register)
Address: Register base + 210H
Access: Read/write

Register: TIMER1_PRE (timer 1 pre-scaler ratio register)
Address: Register base + 250H
Access: Read/write



PRESCALE	R/W	The amount by which the pre-scaler should divide the clock

The pre-scaler ratio register can be read at any time, but can only be written when the timer is stopped.

Register: TIMER0_LIMIT (timer 0 limit register)
Address: Register base + 220H
Access: Read/write

Register: TIMER1_LIMIT (timer 1 limit register)
Address: Register Base + 260H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIMIT																															

LIMIT	R/W	The value of the timer limit register

The timer limit register may be read at any time but should only be written when the timer is stopped.

Register: TIMER0_READ (timer 0 read register)
Address: Register base + 230H
Access: Read

Register: TIMER1_READ (timer 1 read register)
Address: Register base + 270H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READ																															

READ	R	The latched value of the timer counter

The timer value is directly readable.

Watchdog Timer

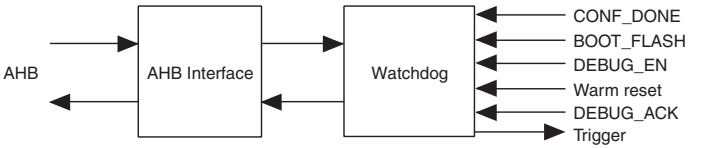
The watchdog timer is a one-shot timer, which protects the system against software failure, or against severe hardware failures (such as lock-ups), due to power-supply problems, for example. It resets the entire chip when it expires and should be regularly reset by software to maintain normal operation, as described in "Resetting the Watchdog Timer" on page 122.

Features of the watchdog timer include the following:

- 32-bit register interface
- Timeouts of up to 30 seconds with a 33-MHz clock
- Independent hardware, software, and bad reload triggers
- Protection from accidental disabling by software

Figure 37 shows the operation of the watchdog timer.

Figure 37. Watchdog Timer Block Diagram



Interface Signals

Table 40 details the signals which affect the behavior of the watchdog timer.

Table 40. Watchdog Timer Signals

Signal	Source	Description
DEBUG_EN	External	When high, allows the embedded processor to enter debug mode. When low, prevents the embedded processor from entering debug mode and enables the hardware trigger on the watchdog
DEBUG_ACK	Stripe	Indicates when the embedded processor is stopped in debug mode. This can be used to stop devices within the PLD when the embedded processor hits a breakpoint
BOOT_FLASH	Input	Boot mode select pin. When high, the embedded processor boots from flash memory on the EBI; when low, boot-from-serial mode is used
CONF_DONE	Input	Configuration: indicates that the device is fully configured
nRESET	Input/Output	Active-low bidirectional reset pin
warm reset	Stripe	Reset signal from the reset module
trigger	Stripe	Watchdog trigger signal to the reset module

Counter Operation

The value of COUNT in the watchdog count register WDOG_COUNT is incremented by the clock supplied on the CLK_REF pin. Counter operation is usually independent of the trigger modes, although there are situations where the counter is inhibited. Table 41 on page 121 summarizes the principles governing the watchdog counter operation.

Table 41. Counter Operation				
DEBUG_EN	DEBUG_ACK	CONF_DONE (1)	BOOT_FLASH	Counter
1	1	x	x	Inhibited
x	x	0	0	Inhibited
Other combinations				Enabled

Note:
(1) CONF_DONE is a pin that, when low, signifies that the PLD is undergoing configuration via the external port. When high, configuration is complete

DEBUG_ACK synchronization to CLK_REF results in a maximum delay of two CLK_REF periods between DEBUG_ACK assertion or de-assertion and the counter being inhibited or enabled.

Trigger Modes

When the watchdog triggers, it sets the TRIGGER bit in WDOG_CR and asserts the trigger output to the reset controller module. The reset controller resets all modules within the device except for the trace logic and provides a reset signal to external devices connected to the EBI. If the boot source is flash memory, the device reboots from flash memory and is reconfigured by the data stored in it. If the boot source is not flash memory, configuration data must be supplied by the external source.

The watchdog timer can be triggered by the following events:

- Counter overflow in hardware trigger mode
- Counter overflow in software trigger mode
- An unexpected value written to the reload register

The triggers operate independently of each other, and are described in detail below.

Hardware Trigger

A low value on the DEBUG_EN pin enables the hardware watchdog, which then triggers if COUNT in WDOG_COUNT overflows. The time represented by COUNT depends on the input frequency.

Software Trigger

Setting TRIGGER in WDOG_CR to a non-zero value enables the software trigger. When COUNT equals the trigger value from TRIGGER, the watchdog is triggered.

Bad Reload Value Trigger

The watchdog triggers if an unexpected value is written to WDOG_RELOAD. This process is explained more fully in "Resetting the Watchdog Timer" below.

Resetting the Watchdog Timer

The following magic values must be written to WDOG_RELOAD to maintain system operation:

- A5A5A5A5H
- 5A5A5A5AH

When the system is in software trigger mode and the lock bit, LK, of WDOG_CR is not set, writing to TRIGGER in WDOG_CR sets the trigger value. After the value is written, the watchdog timer expects the magic value A5A5A5A5H to be written to WDOG_RELOAD register. If the expected value is written to WDOG_RELOAD, it expects the next value to be the other magic value. In addition, if the watchdog expects and receives 5A5A5A5AH, WDOG_COUNT is reset to 0.

After a system reset, the watchdog expects A5A5A5A5H.

If a value other than the expected value is written to WDOG_RELOAD, it triggers the watchdog.

Software Locking

If the lock bit, LK, in WDOG_CR is set to 1, further writes to the control register result in a bus error. This feature can be used to prevent the software trigger from being changed or disabled by software after it has been enabled. As a result, LK can only be cleared by a reset.

Reset

A warm reset resets the watchdog to its initial state. All watchdog register bits are reset to zero, which means that the software trigger is disabled.

If DEBUG_EN is low, the watchdog begins counting immediately after reset. The expected value written to WDOG_RELOAD after reset is A5A5A5A5H.

Following a reset, ensure that the software process to reload the watchdog is operational before the counter overflows.

Registers

All register bits are reset to 0 unless otherwise indicated.

Register: WDOG_CR (control register)
Address: Register base + A00H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																												TRIGGER														0			LK

LK	R/S	When this bit is 1, further writes to the register cause a bus error instead of changing the register contents
TRIGGER	R/W	0—The software trigger is disabled. Other values specify bits 29..4 of the trigger value (bits 3..0 of the trigger are always zero)
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Writing to this register when the lock bit, LK, is set causes a bus error and does not modify the register contents. A watchdog timer bus error causes an exception in the embedded processor. If LK is not set, writing to this register sets the expected value in WDOG_RELOAD to A5A5A5A5H.

Reading from this register has no side-effects.

Register: WDOG_COUNT
Address: Register base + A04H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																												COUNT															

COUNT	R	Current value of watchdog count register
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Reading from this register has no side-effects.

Register: WDOG_RELOAD
Address: Register base + A08H
Access: Write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAGIC																															

MAGIC	W	Magic value to reset watchdog

Writing a sequence of magic values to this register resets WDOG_COUNT. Writing an incorrect value triggers the watchdog. See the section "Resetting the Watchdog Timer" on page 122 for more details.

Interrupt Controller

The interrupt controller provides a simple, but flexible, software interface to the interrupt system.

Figure 38 on page 124 shows the layout of the interrupt controller, which generates two interrupt signals, INT_FIQ_n and INT_IRQ_n, to the embedded processor, from the 17 interrupt sources input.

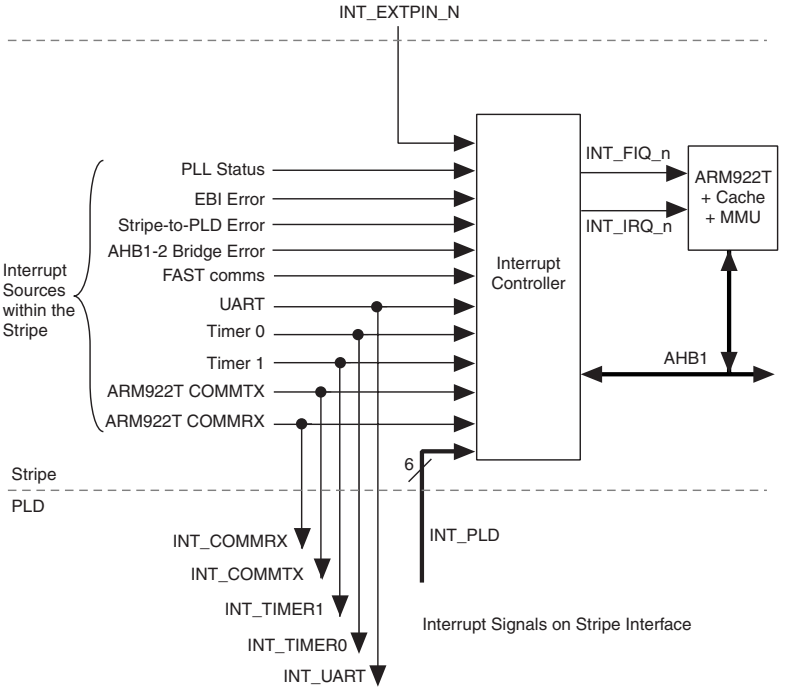
The input sources are as follows:

- 10 interrupts from modules within the stripe
- 1 external pin
- 6 from the PLD stripe interface as an interrupt bus (INT_PLD).

Additionally, five of the stripe modules interrupts are made available as inputs to the PLD - see "Indirect Access to FIQ and IRQ" on page 127. The six signals from the PLD can be treated in one of three different ways; see "Operating Modes" on page 127. By default they appear as six individual interrupt sources.

The interrupt sources are listed in Table 42 on page 130.

Figure 38. Interrupt Controllers



The external interrupt pin, INT_EXTPIN_N, is implemented on the same set of input pins (i.e., the same power bank) as the EBI interface pins. Consequently, this pin is only enabled if the EBI interface is enabled. It is a shared pin, level-triggered and active-low, whereas all other interrupt sources are active-high.

The interrupt controller contains 24 configuration and status registers that are accessible from AHB1 (i.e., only the embedded processor can access them). There are four main registers, each of which has 17 bits corresponding to each of the 17 interrupt sources, as follows:

- The interrupt mask and clear registers, INT_MASK_SET and INT_MASK_CLEAR, determine whether an individual interrupt source can interrupt the embedded processor. Interrupts effectively read and write one 17-bit mask register, but having separate registers for setting and clearing bits in this mask simplifies the software task of turning on and off individual interrupt sources, without affecting the settings for other interrupts
- The interrupt request and source registers, INT_REQUEST_STATUS and INT_SOURCE_STATUS, provide the status of each interrupt source before and after masking respectively

Each of the 17 interrupt sources also has its own priority register, INT_PRIORITY_*y* (where *y* denotes the interrupt source). Each contains a 6-bit priority value, PRI, and a 'generate FIQ if active' bit, FQ, for each interrupt source. The registers determine subsequent interrupt activity:

- Whether that particular interrupt source, when requesting, triggers an FIQ or IRQ interrupt signal to the processor
- What value is present in the interrupt identification register, INT_ID, when one or more interrupts are requesting; see "Interrupt Priority" on page 126

A requesting interrupt source is one for which the interrupt source is active and the appropriate mask bit has been set by the INT_MASK_SET register.

Interrupt Requests to the Processor

The IRQ interrupt to the embedded processor becomes active when any requesting interrupt source has a priority value, PRI, that is non-zero but less than the maximum value, 3FH.

In default operating mode (see "Operating Modes" on page 127), the FIQ interrupt to the embedded processor becomes active when either PRI in INT_PRIORITY_*y* is 3FH, or FQ is set. This signifies that the requesting interrupt source has a maximum priority value.

The FQ bits in the interrupt priority registers have no effect on IRQ interrupt generation.

Interrupt signals to the processor (FIQ and/or IRQ) remain active for as long as the interrupting source is active, provided that the register settings are not changed to disable interrupts.

Interrupt Priority

On receiving an interrupt, the embedded processor first establishes its source. If several interrupts are active, the embedded processor also has to decide which is the highest priority.The priority scheme implemented in the interrupt controller facilitates this.

To implement the priority scheme, the user writes relative priority values to each of the 17 priority registers at system initialization. Each source is usually assigned a unique priority value between 1H and 3FH. Prioritization logic in the interrupt controller constantly compares the priority values within all the priority registers (PRI in INT_PRIORITY_*y*) to discover which interrupts are pending, and provides the value of the highest in the interrupt identification register INT_ID. An interrupt service routine then reads this register to quickly identify the interrupt source.

The value of the interrupt identity register, INT_ID, changes as soon as interrupting sources change.

If FQ in INT_PRIORITY_*y* for a requesting interrupt source is set, but its priority value, PRI, is 0H, the interrupt controller generates an FIQ interrupt but does not update the interrupt identity register, INT_ID.

Indirect Access to FIQ and IRQ

In default operating mode, the interrupt mode register, INT_MODE, is set to 3H. The following settings apply (see also Table 42 on page 130):

- The INT_PLD bus from the PLD-stripe interface is configured as six individual interrupt sources
- INT_PLD[0] is set (and enabled) to trigger FIQ
- INT_PLD[5..1] are each set (and enabled) to trigger IRQ

This effectively gives the user indirect access to the IRQ and FIQ interrupt inputs to the processor, with INT_PLD[0] behaving as FIQ and any one of INT_PLD[5] through INT_PLD[1] behaving as IRQ. This allows users to implement their own interrupt control scheme in the PLD, if required. In addition, because this is the default mode, users are not obliged to disable the interrupt controller within the stripe. This is particularly suited to ASIC prototyping, where none of the stripe modules are required, other than the embedded processor.

FIQ and IRQ interrupt signals to the processor are never directly accessible, for the following reasons:

- INT_PLD signals are always active high
- INT_PLD signals are always passed through synchronizing D-types within the stripe
- In default mode, INT_PLD[5] to INT_PLD[1] are logically OR-ed together to produce IRQ

To implement an interrupt controller in the PLD instead of using the stripe interrupt controller, but still use some of the other modules within the stripe (such as the UART or timer), the interrupt outputs from these modules are made available as inputs to the PLD from the stripe interface. These five interrupt lines are shown in Figure 38 on page 124.

Operating Modes

Three operating modes are provided, to control how the six INT_PLD signals from the PLD are treated:

- As six individual interrupts (the default mode)
- As a single interrupt request, using a six-bit priority value
- As a single interrupt request, using a five-bit priority value together with one individual interrupt

The interrupt modes are controlled by the two-bit MODE value in the interrupt mode register, INT_MODE. The operating mode is not usually changed by the user during system operation, other than at system initialization. The interrupt controller operating modes are described in the sections below.

For more detail information on programming the stripe interrupt controller, refer to *AN 191: Excalibur Solutions—Using the Embedded Stripe Interrupt Controller*.

Six Individual Interrupts

This is the default mode at system reset. Each of the six interrupts has its own priority register (INT_PRIORITY_PLD0 to INT_PRIORITY_PLD5) and its own mask and status bits in the four main mask and status registers. Each interrupt is configured to generate FIQ or IRQ as appropriate. The contents of the PLD interrupt priority register, INT_PLD_PRIORITY, are undefined in default mode.

This mode is probably the most likely to be used for applications, particularly where only a maximum of six interrupts is required from the PLD. For applications requiring more than six interrupt sources from the PLD, one of the following two modes might be more appropriate.

Six-Bit Priority Value

In this mode, INT_PLD[5..0] is taken to be a six-bit encoded interrupt priority, whose value is one of the following:

- 0H—Signifying no interrupt from the PLD
- A non-zero value—Signifying a requesting interrupt with priority ranging from 1H to 3FH

The requesting interrupt priority is compared with the other requesting interrupt priorities from modules in the stripe, to produce IRQ or FIQ requests and to determine the value of the interrupt identity register, INT_ID.

Bits P5 through P0 of the interrupt mask and clear registers, INT_MASK_SET and INT_MASK_CLEAR, together with the interrupt priority registers, INT_PRIORITY_PLD5 through INT_PRIORITY_PLD0, have no effect in this mode.

Bits P5 through P0 of the interrupt source and request registers, INT_SOURCE_STATUS and INT_REQUEST_STATUS, have no effect in this mode. Instead, the PLD interrupt priority register, INT_PLD_PRIORITY, contains the 6-bit priority value that the INT_PLD bus is requesting.

This mode allows up to 63 individual interrupts in the PLD to be encoded by user logic in the PLD.

Five-Bit Priority Value Plus Individual Interrupt

In this mode, INT_PLD[5] through INT_PLD[1] are treated as the most significant bits of a single six-bit encoded interrupt priority; and INT_PLD[0] is treated as an individual interrupt.

The least-significant bit of the six-bit encoded priority value is always 0, affecting the effective values the PLD can signal to the interrupt controller priority logic using INT_PLD[5..1]. These are listed below:

- 0H—Signifying no interrupt from the PLD
- A non-zero even value—Signifying a requesting interrupt with priority ranging from 2H to 3EH

In a similar manner to six-bit priority mode, bits P5 through P1 of the the interrupt mask and clear registers, INT_MASK_SET and INT_MASK_CLEAR, together with the interrupt priority registers, INT_PRIORITY_PLD5 through INT_PRIORITY_PLD1, have no effect in this mode.

In addition, bits P5 through P1 of the interrupt source and request registers, INT_SOURCE_STATUS and INT_REQUEST_STATUS, have no effect in this mode.

INT_PLD[0] behaves as in six-individual interrupts mode

In this mode, FIQ can only be triggered if INT_PLD[0] is active, enabled, and has FQ set in the interrupt priority register, INT_PRIORITY_PLD0. The maximum value of all-ones on INT_PLD[5..1] equates to an equivalent encoded priority value of 3EH and does not cause FIQ.

Interrupt Signals

Table 42 lists all the interrupt sources supported by the hard logic interrupt controller. All interrupt sources (except for INT_EXTPIN_N) are level-triggered, active-high, and their interrupts must be cleared at source.

Table 42. Interrupt Signals

Signal Name	Source	Description	Default State
INT_PLD[0] (individual)	PLD	PLD interrupt 0	FIQ
INT_PLD[1] (individual)	PLD	PLD interrupt 1	IRQ
INT_PLD[2] (individual)	PLD	PLD interrupt 2	IRQ
INT_PLD[3] (individual)	PLD	PLD interrupt 3	IRQ
INT_PLD[4] (individual)	PLD	PLD interrupt 4	IRQ
INT_PLD[5] (individual)	PLD	PLD interrupt 5	IRQ
INT_EXTPIN_N	Input	Active-low external interrupt input pin to the embedded interrupt controller	Disabled
INT_UART	Stripe	UART interrupt signal to PLD	Disabled
INT_TIMER0	Stripe	Timer 0 interrupt signal to PLD	Disabled
INT_TIMER1	Stripe	Timer 1 interrupt signal to PLD	Disabled
INT_COMMTX	Stripe	ARM922T COMMTX interrupt to PLD	Disabled
INT_COMMRX	Stripe	ARM922T COMMRX interrupt to PLD	Disabled

INT_EXTPIN_N is not routed to the PLD, but because it is derived from an external pin, the PLD can connect directly to the pin I/O to monitor it.

The external interrupt pin is level-triggered and active-low (inversion occurs before synchronization). A weak pull-up resistor allows it to be used in a wire-OR configuration. The external interrupt is implemented on the same I/O bank as the EBI and is enabled if the EBI is enabled (see the section "I/O Control" on page 167). Alternatively, the Quartus II software can be used to enable the external interrupt during PLD configuration, without enabling the EBI.

Registers

On reset, all registers are set to 0 unless otherwise indicated.

Register: INT_MASK_SET
Address: Register base + C00H
Access: Read/set
Reset value: 3FH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															FC	CR	CT	AE	PE	EE	PS	T1	T0	UA	IP	P5	P4	P3	P2	P1	P0

P0	R/S	The INT_PLD[0] interrupt line from the PLD (when operating mode is either six individual interrupts or five-bit priority plus one individual interrupt)
P1	R/S	The INT_PLD[1] interrupt line from the PLD (when operating mode is six individual interrupts)
P2	R/S	The INT_PLD[2] interrupt line from the PLD (when operating mode is six individual interrupts)
P3	R/S	The INT_PLD[3] interrupt line from the PLD (when operating mode is six individual interrupts)
P4	R/S	The INT_PLD[4] interrupt line from the PLD (when operating mode is six individual interrupts)
P5	R/S	The INT_PLD[5] interrupt line from the PLD (when operating mode is six individual interrupts)
IP	R/S	The external interrupt pin
UA	R/S	The interrupt line from the UART
T0	R/S	Timer 0 interrupt line
T1	R/S	Timer 1 interrupt line
PS	R/S	PLL status interrupt line
EE	R/S	EBI error
PE	R/S	Stripe-to-PLD bridge error
AE	R/S	AHB1-2 bridge error
CT	R/S	COMMTX interrupt line from ARM922T debug communications channel
CR	R/S	COMMRX interrupt line from ARM922T debug communications channel
FC	R/S	FAST comms JTAG port within configuration logic (space available)
0	R	Reserved for future use. Write as 0 to ensure future compatibility

At reset, the PLD interrupts are enabled; all others are disabled.

Writing 1 to any bit sets it, enabling that particular interrupt source. Writing 0 leaves the bit unchanged.

Reading the register returns the currently set bits, i.e., the interrupt mask.

Register: INT_MASK_CLEAR
Address: Register base + C04H
Access: Read/clear
Reset value: 3FH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															FC	CR	CT	AE	PE	EE	PS	T1	T0	UA	IP	P5	P4	P3	P2	P1	P0

This register provides access to the same internal mask register as INT_MASK_SET. However, writing 1 to any bit clears it, disabling that particular interrupt source. Writing 0 leaves the bit unchanged.

A read from INT_MASK_CLEAR is the same as a read from INT_MASK_SET: it returns the currently-set bits.

Register: INT_SOURCE_STATUS
Address: Register base + C08H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															FC	CR	CT	AE	PE	EE	PS	T1	T0	UA	IP	P5	P4	P3	P2	P1	P0

This register gives the interrupt status of individual interrupt sources before masking. A bit set to 1 signifies an active interrupt source. Undefined bits are read as 0. This register allows software to poll interrupt sources while they are disabled.

If the interrupt is enabled, reading INT_SOURCE_STATUS equates to reading INT_REQUEST_STATUS.

Register: INT_REQUEST_STATUS
Address: Register base + C0CH
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															FC	CR	CT	AE	PE	EE	PS	T1	T0	UA	IP	P5	P4	P3	P2	P1	P0

This register gives the interrupt status of individual pending interrupts. A bit set to 1 signifies that the interrupt source is active and enabled, and is therefore interrupting the embedded processor according to its priority register settings. Undefined bits are read as 0.

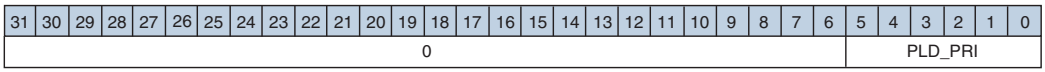
Register: INT_ID
Address: Register base + C10H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																										ID					

ID	R	The priority of the highest priority interrupt which is enabled and active
0	R	Reserved for future use

This register gives the priority of the highest-priority interrupt that is enabled and active.

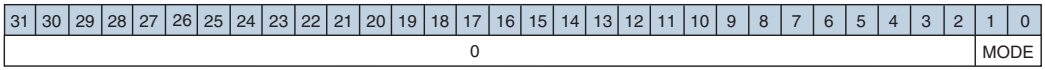
Register: INT_PLD_PRIORITY (PLD interrupt priority)
Address: Register base + C14H
Access: Read



PLD_PRI	R	The interrupt priority being interrupted by the PLD (when routed from the PLD as a priority value)
0	R	Reserved for future use

This register allows the host to read the priority the PLD is signalling, using the interrupt lines if they are configured as a single 6-bit priority value, i.e., if MODE in the interrupt operating-mode register, INT_MODE, is 0H.

Register: INT_MODE
Address: Register Base + C18H
Access: Read/write
Reset value: 3H



MODE	R/W	Interrupt controller operating mode
0	R	Reserved for future use. Write as 0 to ensure future compatibility

This register controls the operating mode of the interrupt controller through the value in MODE, as shown in Table 43.

Table 43. Operating Modes

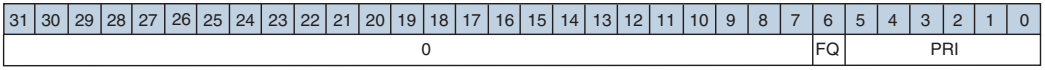
Value of MODE	Operating Mode
3H	Six individual interrupts (default mode)
0H	Six-bit priority value
1H	Five-bit priority value plus individual interrupt

See the section "Operating Modes" on page 127 for details of the effect of operating modes on the interrupt controller.

Priority Registers

There are 17 priority registers, one for each interrupt source. The register format for each is identical. The reset value of each register except INT_PRIORITY_PLD0 is 1H, which means that they only trigger IRQ. The reset value of INT_PRIORITY_PLD0 is 40H, this has its FQ bit set to 1, but PRI value set to 0, which means that it only triggers FIQ.

Register: INT_PRIORITY_PLD0 (INT_PLD[0])
Address: Register base + C80H
Access: Read/write
Reset value: 40H



PRI	R/W	Priority of this interrupt relative to others. A value of 0 equates to disabling the interrupt
FQ	R/W	Generate an FIQ interrupt regardless of the priority
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: INT_PRIORITY_PLD1 (INT_PLD[1])
Address: Register base + C84H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_PLD2 (INT_PLD[2])
Address: Register base + C88H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_PLD3 (INT_PLD[3])
Address: Register base + C8CH
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_PLD4 (INT_PLD[4])
Address: Register base + C90H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_PLD5 (INT_PLD[5])
Address: Register base + C94H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_EXTPIN (INT_EXTPIN_N)
Address: Register base + C98H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_UART (UART interrupt)
Address: Register base + C9CH
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_TIMER0 (timer 0 interrupt)
Address: Register base + CA0H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_TIMER1 (timer 1 interrupt)
Address: Register base + CA4H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_PLL (PLL status interrupt)
Address: Register base + CA8H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_EBI (EBI-error interrupt)
Address: Register base + CACH
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_STRIPE-PLD (stripe-to-PLD bridge-error interrupt)
Address: Register base + CB0H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_AHB1-2 (AHB1-2 bridge-error interrupt)
Address: Register base + CB4H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_TX (ARM922T COMMTX)
Address: Register base + CB8H
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_RX (ARM922T COMMRX)
Address: Register base + CBCH
Access: Read/write
Reset value: 1H

Register: INT_PRIORITY_FASTCOMMS (JTAG Fast communications port)
Address: Register base + CC0H
Access: Read/write
Reset value: 1H

Clocks

The Excalibur family has a number of internal clock domains. This section describes the clocking structures and control mechanisms implemented.

Table 44 summarizes the clock domains for EPXA10, EPXA4, and EPXA1 devices.

Table 44. EPXA10, EPXA4 & EPXA1 Clock Domains

Name	Frequency (MHz)	Derivation	Use
CLK_REF (EPXA10)	10-66	External Pin	Feeds PLLs and fixed-frequency logic (e.g., watchdog timer)
CLK_REF (EPXA4, EPXA1)	1-66		
CLK_AHB1	*(1)*	PLL1	Embedded processor AHB1 bus clock signal
CLK_AHB2	*(1)*	PLL1	Embedded stripe peripheral AHB2 bus clock signal
CLK_SDRAMx2	*(1)*	PLL2	Provides internal clock for SDRAM controller
CLK_SDRAMx1	*(1)*	PLL2	Provides internal clock for SDRAM controller and external clock for SDRAM device
SLAVE_HCLK	*(1)*	PLD	PLD-to-stripe bridge slave port clock that times all bus transfers. Signal timings are related to its rising edge clock
MASTER_HCLK	*(1)*	PLD	Clocks the master port of the stripe-to-PLD bridge
DP0_2_PORTACLK	*(1)(2)*	PLD	DPRAM0/2 port A clock in the PLD
DP0_PORTBCLK	*(1)(2)*	PLD	DPRAM0 port B clock in the PLD
DP1_3_PORTACLK	*(1)(2)*	PLD	DPRAM1/3 port A clock in the PLD
DP1_PORTBCLK	*(1)(2)*	PLD	DPRAM1 port B clock in the PLD

Note:
(1) See Tables 45 to 47.
(2) Only DP0_2_PORTACLK and DP0_PORTBCLK are available for EPXA1.

Table 45. EPXA10 Clock Frequency Versus Speed Grade (Part 1 of 2)

Clock	–1	–2	–3	Units
CLK_AHB1	200	166	133	MHz
CLK_AHB2	100	83	66	MHz
CLK_SDRAMx2	266	200	166	MHz
CLK_SDRAMx1	133	100	83	MHz
MASTER_HCLK	153	127	102	MHz
SLAVE_HCLK	93	78	63	MHz
DP0_2_PORTACLK	152	132	106	MHz

Table 45. EPXA10 Clock Frequency Versus Speed Grade (Part 2 of 2)

Clock	–1	–2	–3	Units
DP0_PORTBCLK	152	132	106	MHz
DP1_3_PORTACLK	152	132	106	MHz
DP1_PORTBCLK	152	132	106	MHz

Table 46. EPXA4 Clock Frequency Versus Speed Grade

Clock	–1	–2	–3	Units
CLK_AHB1	200	166	133	MHz
CLK_AHB2	100	83	66	MHz
CLK_SDRAMx2	266	200	166	MHz
CLK_SDRAMx1	133	100	83	MHz
MASTER_HCLK	172	146	117	MHz
SLAVE_HCLK	97	82	66	MHz
DP0_2_PORTACLK	193	161	129	MHz
DP0_PORTBCLK	193	161	129	MHz
DP1_3_PORTACLK	193	161	129	MHz
DP1_PORTBCLK	193	161	129	MHz

Table 47. EPXA1 Clock Frequency Versus Speed Grade

Clock	–1	–2	–3	Units
CLK_AHB1	200	166	133	MHz
CLK_AHB2	100	83	66	MHz
CLK_SDRAMx2	266	200	166	MHz
CLK_SDRAMx1	133	100	83	MHz
MASTER_HCLK	213	184	147	MHz
SLAVE_HCLK	121	101	81	MHz
DP0_2_PORTACLK	187	157	126	MHz
DP0_PORTBCLK	187	157	126	MHz

External Reference

The clock input pin is 2.5/3.3-V LVTTL. It feeds two PLLs, PLL1 and PLL2, that synthesize the internal clocks required. The reference clock can be provided by a crystal; it need not be synchronous to any of the system operation.

PLLs

The PLLs provide ClockBoost frequency multiplication only. PLL1 provides the embedded processor clock, CLK_AHB1, and the peripheral bus clock, CLK_AHB2. PLL2 provides the clocks for the SDRAM controller.

The CLK_AHB1 and CLK_AHB2 frequencies are one half and one fourth, respectively, of the PLL1 frequency. The CLK_SDRAMx2 frequency is equal to, and the CLK_SDRAMx1 frequency is one-half of, the PLL2 frequency. CLK_AHB1 operates at a maximum of 200 MHz, which limits the maximum-allowed frequency of PLL1 to 400 MHz. CLK_SDRAMx2 operates at a maximum of 266 MHz, which limits the maximum-allowed frequency of PLL2 to 266 MHz.

The embedded processor and configuration logic can program and reprogram the multiplication factor for each PLL, through a register interface.

Each PLL's default operation at power-up is to operate in bypass, which also occurs when a PLL loses lock. If a PLL loses lock, its output is bypassed until lock is again acquired.

Do not turn off a PLL without first putting it into bypass, because this can cause the system to lock up.

A PLL can be forced into bypass using a bit, BP*y*, in the CLK_DERIVE register, where the value of *y* identifies either PLL1 or PLL2.

The PLL lock state is visible to the stripe via the CLK_STATUS register. In addition, an interrupt can be generated on change of lock state.

PLL Configuration

Parameters *M*, *N*, and *K* are used to program the PLL operating frequencies, using control registers (CLK_PLL*y*_NCNT, CLK_PLL*y*_MCNT and CLK_PLL*y*_KCNT respectively). The following equations show the calculations for programming a PLL:

$$f_{VCO} = CLK\_REF \times M/N$$

where the device parameter settings given in Table 48 on page 139 apply.

Table 48. Parameter Settings for N, M, and K			
Device	**N**	**M**	**K**
EPXA10	1 to 15	1 to 15	1 to 7
EPXA4	1 to 255	1 to 255	1 to 255
EPXA1	1 to 255	1 to 255	1 to 255

Finally:

$PLL_{OUT} = f_{VCO}/K$

where f_{VCO} ranges from 160 MHz to 600 MHz.

The algorithm for mapping *M*, *N*, and *K* to CLK_PLL*y*_NCNT, CLK_PLL*y*_MCNT and CLK_PLL*y*_KCNT is given in "PLL Parameter Settings" on page 146.

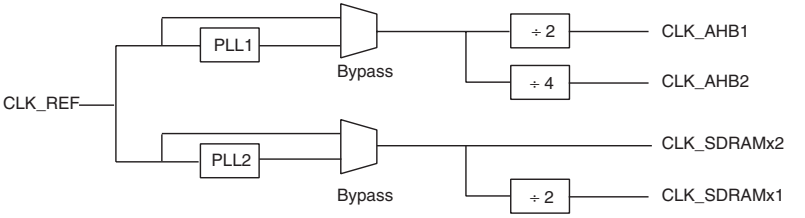
For more detailed information on programming the embedded stripe PLLs, refer to *AN 177: Excalibur Solutions—Using the Excalibur Stripe PLLs*.

When setting up the SDRAM PLL, odd values of *K*, as specified in the register CLK_PLL2_KCNT, generate a non-50:50 square wave. This adversely affects the SDRAM controller in DDR mode by impacting the position of DQS in relation to DQ during a write. Specify *K* as either 2 or 4; other values should not be used.

Clock Control

Figure 39 on page 140 shows the functional modes for the main system clocks (excluding PLD domains and configuration).

Figure 39. System Clocks' Functional Modes



The following additional points apply to the system clocks:

- CLK_AHB1 is derived from PLL1, with a divide of 2
- CLK_AHB2 is derived from PLL1, with a divide of 4
- CLK_SDRAMx2 is derived from PLL2
- CLK_SDRAMx1 is derived from PLL2, with a divide of 2

Low-Power Mode

The EPXA1 supports a low-power mode feature. In low-power mode, the EPXA1 completely turns off the PLL and the PLL consumes only 5 mA of current.

Neither the EPXA10 or EPXA4 completely turn off the PLL. The EPXA10 and EPXA4 PLLs always consume 30 mA of current, even when bypassed and disabled. The source of the 30 mA current is due to a bias circuit in the PLL, which is enabled to have a predictable locking characteristic.

Low-power mode could be used to place the embedded processor into a suspended state, and subsequently resume operation under embedded software control.

In the following procedure, *y* denotes the PLL, either 1 or 2.

Follow the steps below to place an EPXA1 into suspended mode:

1. Put the appropriate PLL into bypass by setting the BP*y* bit in the CLK_DERIVE register.
2. Wait 10 ms.
3. Disable the PLL by clearing the P bit in the CLK_PLL*y*_CTRL register.
4. Put the PLL into low power mode by setting the LP bit in the CLK_PLL*y*_CTRL register.
5. Use a PLD pin to signal to external logic to slow down the external clock generator or stop the clock.

Follow the steps below to bring the EPXA1 out of suspended mode:

1. Either start up or speed up the external clock generator.
2. Use external logic to assert a PLD pin to signal the resume event.
3. Use embedded software to poll this pin or use it as an interrupt to detect the resume event.
4. Bring the appropriate PLL out of low power mode by clearing the LP bit in the CLK_PLL*y*_CTRL register.
5. Enable the PLL by setting the P bit in the CLK_PLL*y*_CTRL register.
6. Wait 10 ms.
7. Bring the PLL out of bypass by clearing the BP*y* bit in the CLK_DERIVE register.
8. Poll the L*y* bit in the CLK_STATUS register to verify that the PLL is locked.

Registers

After reset, registers are set to 0 unless otherwise indicated.

In EPXA10 devices, only bits 18..16, 11..8, and 2..0 are implemented for the *M* and *N* parameter registers for each PLL; for the PLL *K* parameter register, only bits 18..16, 10..8, and 1..0 are implemented.

Register: CLK_PLL1_NCNT (see "PLL Configuration" on page 138 for usage details)
Address: Address base + 300H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CT2	CT1	CT0	15..8								0	6..0						

6..0	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 2..0 are used
15..8	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 11..8 are used
CT2	R/W	1—Bypass for N counter. See "PLL Parameter Settings" on page 146
CT1	R/W	1—Even multiplication factor for N counter. See "PLL Parameter Settings" on page 146
CT0	R/W	1—Odd multiplication factor for N counter. See "PLL Parameter Settings" on page 146
0	R/W	Reserved for future use. Write as 0 it ensure future compatibility

Register: CLK_PLL1_MCNT (see "PLL Configuration" on page 138 for usage details)
Address: Address base + 304H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CT2	CT1	CT0	15..8								0	6..0						

6..0	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 2..0 are used
15..8	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 11..8 are used
CT2	R/W	1—Bypass for M counter. See "PLL Parameter Settings" on page 146
CT1	R/W	1—Even multiplication factor for M counter. See "PLL Parameter Settings" on page 146
CT0	R/W	1—Odd multiplication factor for M counter. See "PLL Parameter Settings" on page 146
0	R/W	Reserved for future use. Write as 0 it ensure future compatibility

Register: CLK_PLL1_KCNT (see "PLL Configuration" on page 138 for usage details)
Address: Address base + 308H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CT2	CT1	CT0	15..8								0	6..0						

6..0	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 1..0 are used
15..8	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 10..8 are used
CT2	R/W	1—Bypass for K counter. See "PLL Parameter Settings" on page 146
CT1	R/W	1—Even multiplication factor for K counter. See "PLL Parameter Settings" on page 146
CT0	R/W	1—Odd multiplication factor for K counter. See "PLL Parameter Settings" on page 146
0	R/W	Reserved for future use. Write as 0 it ensure future compatibility

Register: CLK_PLL1_CTRL
Address: Address base + 30CH
Access: Read/write
Reset value: 1A04H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																LP	0	0	1	0	0	0	0	0	RLOCK			0	1	0	P

P	R/W	0—PLL disabled (default) 1—PLL enabled
RLOCK	R/W	Set as follows to specify the lock window: 000—0.1 ns (default) 001—0.25 ns 010—0.5 ns 011—1.0 ns 100—1.5 ns 101—2.5 ns 110—10 ns
LP	R/W	0—Low-power mode disabled (default). Low-power mode should be disabled for EPXA10 and EPXA4 devices 1—Low-power mode enabled (EPXA1 only).This mode is not supported in EPXA10 and EPXA4 devices
0	R/W	Reserved for future use. Write as 0 it ensure future compatibility
1	R/W	Reserved for future use. Write as 1 to ensure future compatibility

The LP bit is only supported in the EPXA1 device.

The MegaWizard Plug-In automatically sets the lock window setting depending on the CLK_REF value. To find out how the Quartus II software determines the lock window, or to set the lock window manually, refer to *AN 177: Excalibur Solutions—Using the Excalibur Stripe PLLs*.

Register: CLK_PLL2_NCNT (see "PLL Configuration" on page 138 for usage details)
Address: Address base + 310H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CT2	CT1	CT0	15..8								0	6..0						

6..0	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 2..0 are used
15..8	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 11..8 are used
CT2	R/W	1—Bypass for N counter. See "PLL Parameter Settings" on page 146
CT1	R/W	1—Even multiplication factor for N counter. See "PLL Parameter Settings" on page 146
CT0	R/W	1—Odd multiplication factor for N counter. See "PLL Parameter Settings" on page 146
0	R/W	Reserved for future use. Write as 0 it ensure future compatibility

Register: CLK_PLL2_MCNT (see "PLL Configuration" on page 138 for usage details)
Address: Address Base + 314H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CT2	CT1	CT0	15..8								0	6..0						

6..0	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 2..0 are used
15..8	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 11..8 are used
CT2	R/W	1—Bypass for M counter. See "PLL Parameter Settings" on page 146
CT1	R/W	1—Even multiplication factor for M counter. See "PLL Parameter Settings" on page 146
CT0	R/W	1—Odd multiplication factor for M counter. See "PLL Parameter Settings" on page 146
0	R/W	Reserved for future use. Write as 0 it ensure future compatibility

Register: CLK_PLL2_KCNT (see "PLL Configuration" on page 138 for usage details)
Address: Address base + 318H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CT2	CT1	CT0	15..8								0	6..0						

6..0	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 1..0 are used
15..8	R/W	See "PLL Parameter Settings" on page 146. For EPXA10, only 10..8 are used
CT2	R/W	1—Bypass for K counter. See "PLL Parameter Settings" on page 146
CT1	R/W	1—Even multiplication factor for K counter. See "PLL Parameter Settings" on page 146
CT0	R/W	1—Odd multiplication factor for K counter. See "PLL Parameter Settings" on page 146
0	R/W	Reserved for future use. Write as 0 it ensure future compatibility

Register: CLK_PLL2_CTRL
Address: Address base + 31CH
Access: Read/write
Reset value: 1A04H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																LP	0	0	1	0	0	0	0	0	RLOCK			0	1	0	P

P	R/W	0—PLL disabled (default) 1—PLL enabled
RLOCK	R/W	Set as follows to specify the lock window: 000—0.1 ns (default) 001—0.25 ns 010—0.5 ns 011—1.0 ns 100—1.5 ns 101—2.5 ns 110—10 ns
LP	R/W	0—Low-power mode disabled (default). Low-power mode should be disabled for EPXA10 and EPXA4 devices 1—Low-power mode enabled (EPXA1 only).This mode is not supported in EPXA10 and EPXA4 devices
0	R/W	Reserved for future use. Write as 0 it ensure future compatibility
1	R/W	Reserved for future use. Write as 1 to ensure future compatibility

The LP bit is only supported in the EPXA1 device.

The MegaWizard Plug-In automatically sets the lock window setting depending on the CLK_REF value. To find out how the Quartus II software determines the lock window, or to set the lock window manually, refer to *AN 177: Excalibur Solutions—Using the Excalibur Stripe PLLs*.

Register: CLK_DERIVE
Address: Register base + 320H
Access: Read/write
Reset value: 3010H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																		BP2	BP1	0							1	0			

0	R	Reserved for future use. Write as 0 to ensure future compatibility
1	W	Reserved. Write as 1 to ensure correct operation
BP2	R/W	1—Force bypass of PLL2
BP1	R/W	1—Force bypass of PLL1

Register: CLK_STATUS
Address: Register base + 324H
Access: Read/clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																										P2	P1	C2	C1	L2	L1

L1	R	PLL1 lock status
L2	R	PLL2 lock status
C1	R/C	1—PLL1 lock changed
C2	R/C	1—PLL2 lock changed
P1	R	1—PLL1 not bypassed
P2	R	1—PLL2 not bypassed
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The clock module generates an interrupt when either C1 or C2 is set.

PLL Parameter Settings

In EPXA10, only bits 18..16, 11..8, and 2..0 are implemented for the *M* and *N* parameter registers for each PLL; for the PLL *K* parameter register, only bits 10..8 and 1..0 are implemented.

For either of the PLLs, the *M*, *N*, and *K* parameters are mapped to the appropriate parameter register using one of the following procedures:

- For the *M* and *N* parameters:

  CLK_PLL*y*_*x*CNT[6..0] = *z*/2
  CLK_PLL*y*_*x*CNT[15..8] = *z*/2 + *z*%2

- For the *K* parameter:

  CLK_PLL*y*_*x*CNT[1..0] = *z*/2
  CLK_PLL*y*_*x*CNT[10..8] = *z*/2 + *z*%2

Subsequently, for *K*, *M*, or *N* parameters:

CLK_PLL*y*_*x*CNT[18] = 1, if *z* is 1 (PLL calculation is pointless)
CLK_PLL*y*_*x*CNT[17] = 1, if *z* is even
CLK_PLL*y*_*x*CNT[16] = 1, if *z* is odd

In the procedures above, *y* corresponds to the PLL being programmed and *x* is *M*, *N*, or *K*, corresponding to the parameter; *z* is the actual value of the parameter. The value z/2 corresponds to z shifted right by one bit and z%2 is the remainder after dividing z by 2 (actually, just z[0])

For example, if the value of *M* for PLL1 is decimal 15 (1111 in binary):

```
CLK_PLL1_MCNT[6..0]     =  0000111
CLK_PLL1_MCNT[15..8]    =  111 + 1 = 00001000

CLK_PLL1_MCNT[18]       =  0 (M is 15, not 1)
CLK_PLL1_MCNT[17]       =  0 (15 is not even)
CLK_PLL1_MCNT[16]       =  1 (15 is an odd number)
```

Therefore, register CLK_PLL1_MCNT has the value 10807H.

Similarly, if the value of *K* for PLL2 is 2 (010):

```
CLK_PLL2_KCNT[1..0]     =  01
CLK_PLL2_KCNT[10..8]    =  001 + 0 = 001

CLK_PLL2_KCNT[18]       =  0 (K is 2, not 1)
CLK_PLL2_KCNT[17]       =  1 (2 is an even number)
CLK_PLL2_KCNT[16]       =  0 (2 is not odd)
```

Therefore, register CLK_PLL2_KCNT has the value 20101H.

Profiling Register

A 32-bit clock module register counts the number of processor cycles since reset. This stops counting when the embedded processor is in debug mode (i.e. when DEBUG_ACK is active).

Register: CLK_AHB1_COUNT
Address: Register base + 328H
Access: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Count of processor cycles since reset																															

PLD Clocks

Up to six clocks can be used for the interfaces between the embedded stripe and the PLD:

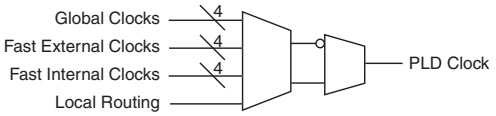
- Stripe-to-PLD bridge—`MASTER_HCLK`
- PLD-to-stripe bridge—`SLAVE_HCLK`
- Dual-port SRAM—up to four clocks

The clocks can be derived from any of the following APEX clock sources. Selectable inversion is available for each clock:

- Global clocks—up to four
- Fast external clocks—up to four
- Fast internal clocks—up to four
- Internal local routing—user definable

Figure 40 shows the multiplexing that allows the PLD clocks to be derived from the APEX clock sources. Selectable inversion is available for each clock.

Figure 40. PLD Clock Derivation



Refer to the *APEX 20K Programmable Logic Device Family Data Sheet, Version 3.2* for details of the APEX clock sources.

Configuration Logic

The configuration logic is responsible for transferring configuration data to the PLD array and setting up the system so that the embedded processor can boot. It behaves similarly to the configuration logic specified in the *APEX 20K Programmable Logic Device Family Data Sheet, Version 3.2* but there are two additional features. The Excalibur family allows users to perform the following actions:

- Set up registers and on-chip SRAM as part of the configuration bitstream
- Configure or reconfigure the PLD via the embedded processor's configuration-logic slave port

There are two configuration methods available to configure the device and make code available at the boot address. The processor always boots from address 0H. The two configuration methods are as follows:

- Boot from flash
- Boot from an external source

The external pins `BOOT_FLASH`, `MSEL1` and `MSEL2` are used to enable the various configuration schemes used for booting the device. Table 49 shows the settings for the different modes.

Table 49. Configuration Modes

BOOT_FLASH	MSEL1	MSEL0	Mode
0	0	0	Passive serial or JTAG
0	1	0	Passive parallel synchronous
0	1	1	Passive parallel asynchronous
1	0	0	Boot from 16-bit flash
1	0	1	Boot from 16-bit flash, 1.8 V
1	1	0	Boot from 8-bit flash
1	1	1	Boot from 8-bit flash, 1.8 V

Boot from Flash

Setting BOOT_FLASH to high puts the device in boot-from-flash mode upon power up. In this mode, the processor accesses the bootcode from either 8- or 16-bit flash connected to EBI0 (see "Expansion Bus Interface" on page 83 for details). The bottom 32 Kbytes of EBI0 are mapped at address 0H in boot-from-flash mode. The registers are mapped to their default addresses, based at 7FFFC000H. Remaining devices are not mapped, and interrupts are disabled.

The Altera-provided Excalibur bootloader or a customer bootloader can be used to start up the system, make the necessary mappings, and read the PLD configuration data into the device.

PLD Configuration

The Quartus II software should be used to generate PLD configuration data in slave-port binary file (.**sbi**) format, which is used to configure the PLD from the embedded processor in boot-from-flash mode. The .**sbi** is combined with the embedded register contents and the bootcode to form a HEX file that can be programmed into flash memory.

Table 50 shows the .**sbi** file format used for PLD configuration.

Table 50. SBI File Format

Offset	Size	Data
0H	4	Signature "SBI\0"
4H	4	IDCODE for target system
8H	4	Offset to configuration data (*coffset*)
CH	4	Size of configuration data in bytes (*csize*). Must be a multiple of 4
coffset	*csize*	PLD configuration data. This is a byte stream to be written to the PLD slave port

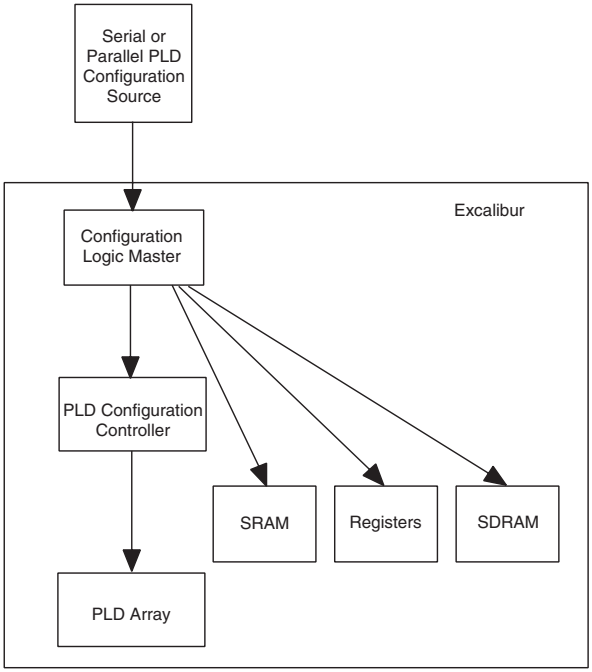
Boot from External Configuration Source

When BOOT_FLASH is low, the device can either be configured by an external configuration source such as a configuration device, or via JTAG. In boot-from-external mode, the embedded processor is held in reset while the PLD and the stripe are being configured. In passive serial or passive parallel modes, the standard APEX20KE configuration pins are used to configure the device. For JTAG configuration, the PLD TAP controller JTAG pins are used. (See "IEEE Std. 1149.1 (JTAG) Support" on page 163).

The processor is not released from reset until the configuration logic has entered user mode.

Figure 41 on page 151 shows the block diagram for boot-from-external mode.

Figure 41. Boot-from-External Configuration Source



The PLD array is configured using one of the following configuration schemes:

- Passive serial (PS)—Same as the APEX family
- Passive parallel synchronous (PPS)—Same as the APEX family
- Passive parallel asynchronous (PPA)—Same as the APEX family
- JTAG—The Quartus II software provides the appropriate SRAM object file (.**sof**)

When configuring using JTAG, ensure that no configuration via an external source is in progress simultaneously.

The configuration logic master configures the memory map, appropriate stripe memory, and the PLD. The configuration bitstream contains register writes to enable writable memory, such as on-chip SRAM, at address 0, and to write code into it. The register writes are followed by PLD configuration data. When the data transfer is complete, the bitstream writes to the boot control register, `BOOT_CR` to release the embedded processor from reset.

PLD Configuration

SRAM object files (.**sof**), raw binary files (.**rbf**), tabular text files (.**ttf**) or intel-format .**hexout** formats produced by the Quartus II software are acceptable for PLD configuration. In addition, programmer output files (.**pof**) produced by the Quartus II software can be used to program external configuration devices connected to this interface.

Registers

Register: CONFIG_CONTROL (configuration control register)
Address: Register base + 140H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																								ES			E	PC	B	CO	LK

LK	R/W	Lock. When set, writes to this register cause a bus error instead of changing the register contents
CO	R/W	Configure. When set, indicates that the PLD is being configured via the PLD configuration controller. The PLD configuration controller clears this bit when configuration is complete
B	R	Busy. When set, either the data register is not ready to accept data (and inserts wait states if written to) or the PLD controller is resetting in preparation for configuration
PC	R	When set, the PLD is configured and in user mode
E	R	Error. When set, indicates that (re)configuration was not possible for one of the reasons below. – External configuration prevents reconfiguration – The PLD is being configured via JTAG – There was an error in the PLD bitstream – CONFIG_CLOCK value is 0 This bit is latching and is cleared by setting CO while ES is 0
ES	R	Error source. These (non-latching) bits indicate that reconfiguration is not currently possible for one or more reasons. The value presented is the logical or of the following values: 001—PLD being configured via JTAG 010—CONFIG_CLOCK value is zero 100—PLD being configured via external configuration interface
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The lock bit, LK, can be used to prevent inadvertent reconfiguration of the PLD. When LK is set, writing to CONFIG_CONTROL, CONFIG_CLOCK, or CONFIG_DATA causes a bus error and does not change the register contents. LK is reset by writing the magic number to CONFIG_UNLOCK, or system reset.

Setting the configure bit, CO, to 1 (when it was previously 0) initiates PLD configuration. CO is set to 0 when configuration is complete, whether successfully or with an error condition.

If CO is set while the PLD is being configured (either via external configuration pins or via JTAG), the error bit, E, is set, and CO is reset.

Immediately after CO is set, the busy bit, B, is set, and is cleared when the PLD configuration controller is ready to receive data.

When CONFIG_CLOCK has the value 0, any attempt to set CO to 1 (when it was previously 0) is ignored, and results in E being set.

Setting CO to 0 (when it was previously 1) using a register-write aborts the configuration and sets E.

Register: CONFIG_CLOCK (configuration clock register)
Address: Register base + 144H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																RATIO															

RATIO	R/W	Half the divide ratio applied to AHB2 clock to generate the PLD download clock. No clock is generated when this has the value 0
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The clock divide ratio allows a valid clock to be generated from the AHB2 clock to drive the PLD configuration control logic in parallel-synchronous mode. The maximum clock frequency for the PLD configuration controller is 16 MHz. If the divide ratio is set to provide a higher speed, or not to provide a clock at all, the results are undefined. If CONFIG_CLOCK is written during configuration via the slave port (i.e., CO is set), the write is ignored.

For an AHB2 clock speed of 60 MHz, a divide ratio of 3 produces a PLD download-clock of 10 MHz.

Register: CONFIG_DATA (embedded controller data register)
Address: Register base + 148H
Access: Write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															

DATA	W	Data written to PLD configuration logic

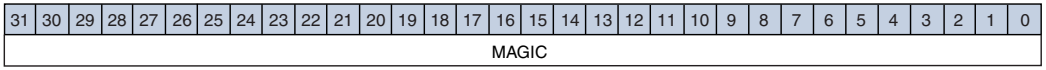
Data written to CONFIG_DATA when the configure bit, CO, in CONFIG_CONTROL is not set is ignored. Data written to CONFIG_DATA is stored in a holding register and then loaded into the PLD configuration logic in parallel-synchronous mode.

The correct order for a byte stream that has been loaded from memory using a word load is given below:

- In little-endian mode, bits 0 to 7 are loaded into the PLD, followed by bits 8 to 15, 16 to 23 and then 24 to 31
- In big-endian mode, bits 24 to 31 are loaded, then 16 to 23, 8 to 15 and finally 0 to 7

If the configure bit, CO, is set and CONFIG_DATA is unable to accept data (that is, the busy bit, B, is set), the slave interface inserts wait states until there is space for new data, or until CO is cleared. One word of buffering is provided, to ensure that configuration data can be loaded at maximum speed.

Register: CONFIG_UNLOCK (configuration unlock register)
Address: Register base + 14CH
Access: Write



MAGIC W Magic number

Writing the value 554E4C4BH to this register clears the lock bit, LK, in CONFIG_CONTROL. Writing any other value causes a bus error.

Reset & Mode Control

The devices can be reset from many sources. The reset controller logic determines the cause of the reset, synchronizes it as necessary, and resets the appropriate logic blocks.

When any type of reset is asserted, the clock generator is placed in bypass mode, which automatically bypasses the PLLs and derives all clock outputs directly from the input reference.

Only a power-on reset initializes the embedded processor trace port, watchdog status, and configuration error status.

The contents of SDRAM, single- and dual-port on-chip SRAM are not guaranteed after a reset.

Types of Reset

There are various types of reset, as follows:

- Power-on reset
- Warm reset
- JTAG reset
- Configuration/reconfiguration
- Processor reset

Power-On Reset

There is one power-on reset signal for all clock domains, which resets only the following devices:

- Embedded processor trace port
- Reset status register, RESET_SR (see page 162)
- Embedded JTAG controller

Warm Reset

There is one warm reset signal for all clock domains, which results from all reset sources. A warm reset resets all the embedded circuitry, including clearing the PLD contents and resetting the stripe registers, but does not reset the following devices:

- Embedded processor trace port
- Reset status register, RESET_SR
- Embedded JTAG controller
- Embedded configuration logic

A warm reset also asserts the external reset signal from the EBI, nRESET.

Embedded JTAG

The JTAG input TRST resets the JTAG controller in the PLD. Under certain conditions, it can also reset the JTAG controller in the embedded processor core, depending on the state of the external JSELECT pin. JSELECT determines whether PROC_TRST or TRST resets the JTAG controller in the embedded processor core. Refer to "IEEE Std. 1149.1 (JTAG) Support" on page 163 for further details.

Configuration/Reconfiguration

A warm reset clears the contents of the PLD, in addition to re-booting the stripe. If BOOT_FLASH is false, a warm reset also asserts nSTATUS to request reconfiguration. If BOOT_FLASH is true, a warm reset asserts nCONFIG to request reconfiguration.

Processor Reset

In boot-from-flash modes, the processor is held in reset for the duration of a warm reset.

In other modes, the processor is additionally held in reset while the configuration is loaded. It is released when the PLD enters user mode.

The BOOT_CR setting depends on external configuration pins; see page 162 for details.

Sources of Reset

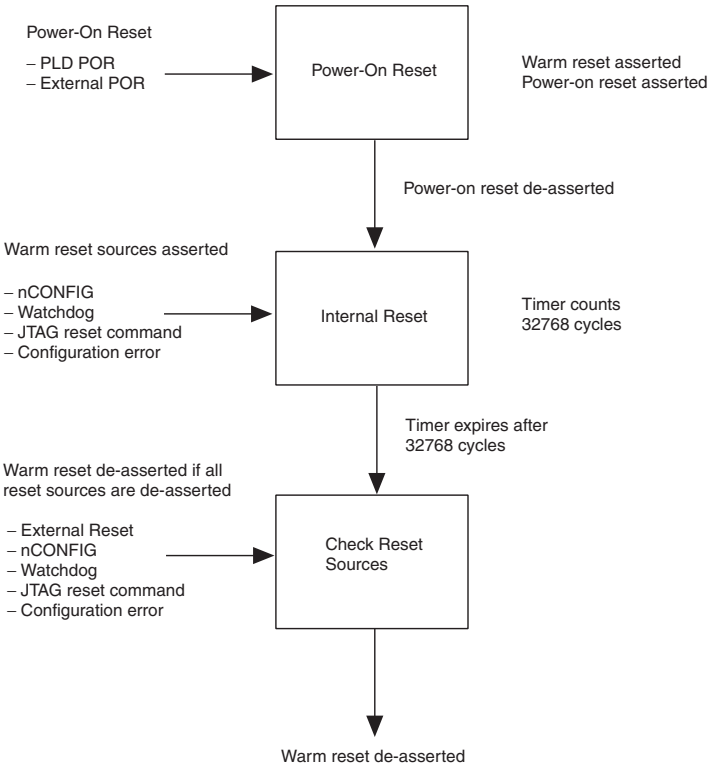
Resets can originate from several sources:

- PLD power-on reset
- Resets from external sources
  - Configuration pin nCONFIG
  - External reset pin nRESET (bidirectional open drain pin, supplying reset output to configuration or flash devices)
  - External power-on reset nPOR
- Resets from internal sources
  - Watchdog timer reset
  - JTAG module
  - Configuration error

These are explained below; "Reset Operation" on page 160 details the effects of these resets.

Figure 42 on page 158 shows the reset sequence.

Figure 42. Reset Sequence



PLD Power-On Reset

The PLD power-on reset signal asserts internal power-on and warm reset signals.

Resets from External Sources

External reset sources always assert a warm reset, although the type of reset dictates what other effects also occur.

Configuration nCONFIG Pin

`nCONFIG` forces a warm reset when pulled low; and initiates device reboot and reconfiguration when released.

External Reset Pin

The external reset pin nRESET is an active-low input that shares the pin with the reset output. It forces a warm reset when pulled low and must be asserted for sufficient time to reset any flash devices connected to the EBI. It does not start the reset counter. External de-bounce circuitry must be provided if the pin is to be connected to a mechanical reset switch.

External Power-On Reset Pin

The external power-on reset pin nPOR is active-low. When pulled low, it asserts both power-on and warm-reset signals.

Internal Sources of Reset

Internal reset sources always assert a warm reset, although the type of reset dictates what other effects also occur.

Software Watchdog Timer Reset

On reaching the programmed value, the software watchdog timer generates a trigger that invokes a warm reset.

JTAG

When the JTAG logic asserts a warm reset request, the reset module invokes a warm reset and holds the embedded processor in reset by setting the hold-microprocessor bit, HM, in the boot-control register, BOOT_CR.

Configuration Error

When the configuration logic detects a configuration error, the reset module invokes a warm reset under the following conditions:

- External pin BOOT_FLASH is false
- The auto-reconfigure option in the Quartus II software is set

Reset Operation

Reset Counter

All reset events except assertion of the external reset pin trigger the reset counter, which controls the duration of the reset event and asserts a warm reset. Under certain conditions the reset counter also asserts an external reset output for a minimum of 51.2 µs, which is long enough to reset the external flash devices. When the reset counter reaches the reset assertion limit of 32768 clock cycles at the external clock frequency, the external reset output is negated, but warm reset remains asserted until all reset sources are negated.

Power On

At power on, the reset module asserts the external reset pin, nRESET, internal power-on and warm reset signals. The reset counter does not begin counting until the power-on signal is negated.

After reset, the UART, SDRAM, and EBI pins are weakly pulled up, unless the device is to boot from flash memory, in which case the EBI pins are actively driven both during and after reset.

In the EPXA10 and EPXA4 devices, the UART, SDRAM, and EBI pins are shared I/Os (used by the stripe or the PLD image). In the EPXA1 device, these pins are dedicated to the stripe. If the stripe uses the UART, SDRAM, or EBI pins, they are actively driven by the stripe. However, if the pins are not used by the stripe, they remain weakly pulled up until the PLD is configured, and then behave according to the PLD image.

External Reset Input

Assertion of the external reset pin, nRESET, causes a warm reset.

Reset Output

The reset output, nRESET, is active-low, open-drain and shares a pin with the external reset input. It is asserted during all reset events.

Warm Reset

When a warm reset source is asserted, except when it is triggered by an external reset, the reset output, nRESET, and warm reset are asserted, and the reset counter begins counting.

External Power-On Reset

Figures 43 and 44 on page 161 show the connectivity between an embedded processor PLD and a power supervisor. They also show how an external reset must be connected to flash memory.

Figure 43. External POR Reset Connectivity

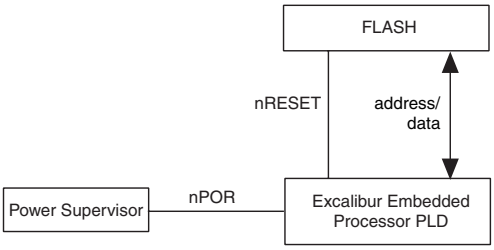
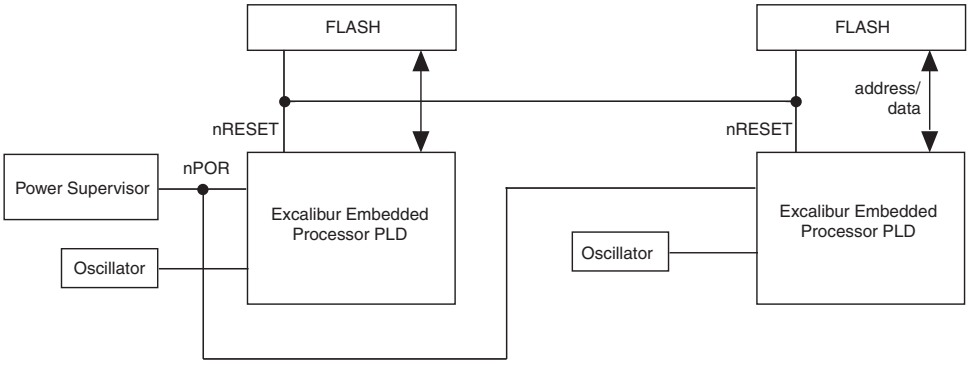


Figure 44. Multiple Excalibur System



Note:
(1) nRESET requires external weak pull-up resistor - not shown

A power supervisor is required for each system. To reset multiple devices when a warm reset is asserted in any of them, the nRESET signals must be wired together.

To guarantee correct reset behavior, nPOR must be held asserted until all power supplies are within tolerance and the CLK_REF input is operating within tolerance (with frequency stable and valid logic levels). Additionally, once these conditions are met, five CLK_REF periods must elapse before nPOR is negated.

Registers

Register: BOOT_CR (boot-control register)
Address: Register base + 0H
Access: Read/clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																													RE	HM	BM

BM	R/C	Boot memory mapping. When this bit is set, the first 32 Kbytes of EBI0 is mapped at address 0H
HM	R/C	Hold microprocessor. The embedded processor is held in reset while this bit is set
RE	R/C	Registers enabled. Clearing this bit has the same effect as clearing bit 0 of the register's base register
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The reset value for this register depends on the configuration pins attached to the chip. In boot-from-flash modes, it has the value 7 if the reset source was the JTAG configuration-DMA scan chain, but otherwise it is 5. In other boot modes, it has the value 6, although the configuration bitstream usually resets HM.

Register: RESET_SR (reset status register)
Address: Register base + 4H
Access: Read/clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																												ER	JT	CR	WR

WR	R/C	Watchdog caused warm reset
CR	R/C	Configuration error caused warm reset
JT	R/C	JTAG caused warm reset
ER	R/C	External pin caused warm reset
0	R	Reserved for future use. Write as 0 to ensure future compatibility

This register indicates the cause(s) of one or more previous reset events. nPOR clears it to 0H; other types of reset set the appropriate bit.

Register: IDCODE (identity and version register)
Address: Register base + 8H
Access: Read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDCODE																															

IDCODE	R	Chip ID code: EPXA10—0x090010DD EPXA4—0x084010DD EPXA1—0x081010DD

This register returns the value of the chip ID code, which is read from the device ID register within the embedded JTAG controller.

IEEE Std. 1149.1 (JTAG) Support

Various features are provided to facilitate debugging soft logic and running code, including the following:

- PLD JTAG TAP controller
- Embedded processor JTAG TAP controller
- SignalTap logic analyzer module

The Excalibur family provides JTAG BST circuitry that complies with the IEEE Std. 1149.1-1990 specification. The Excalibur devices contain two JTAG TAP controllers, one for the PLD JTAG boundary scan chain and one for the embedded processor. The PLD TAP controller allows boundary scan testing of the physical pins on the device and downloading of configuration data. The JTAG port can also be used to monitor the logic operation of the device with the SignalTap embedded logic analyzer. The embedded processor TAP controller gives debuggers access to the internal state of the processor and provides extensive software debugging functionality. This is described in the embedded processor debug support section.

JTAG Modes

There are two JTAG configurations:

- Serial-JTAG mode
- Parallel-JTAG mode

In serial-JTAG mode, the PLD TAP controller is followed in series by the embedded processor TAP controller. Both TAP controllers are accessible from the same physical port.

In parallel-JTAG mode, the PLD TAP controller and processor TAP controller are connected to separate ports. The PLD TAP controller is connected to the `TMS`, `TDO`, `TDI`, `TCK`, and `TRST` pins, and the processor TAP controller is connected to the `PROC_TMS`, `PROC_TDO`, `PROC_TDI`, `PROC_TCK`, and `PROC_TRST` pins.

Figures 45 and 46 show the JTAG configurations.

Figure 45. Serial-JTAG Mode (JSELECT = 0)

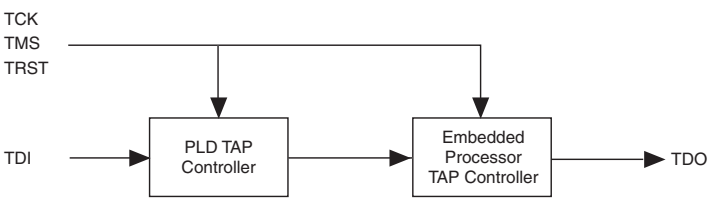
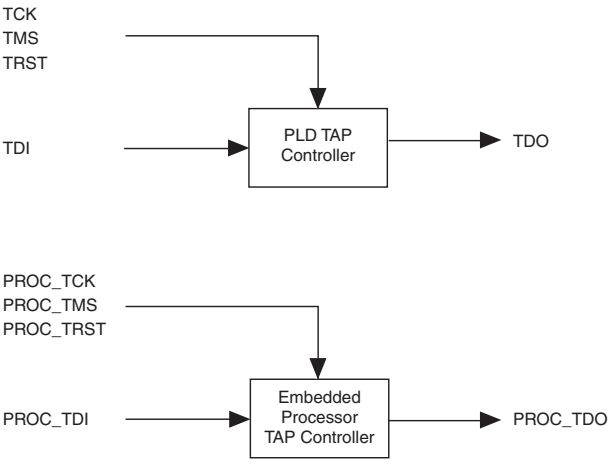


Figure 46. Parallel-JTAG Mode (JSELECT = 1)



The JTAG TAP controllers are not reset by hard or warm reset, because they have their own reset lines.

JTAG Boundary-Scan Testing

JTAG boundary-scan testing can be performed before or after configuration, but not during configuration.

The Excalibur family supports the JTAG instructions shown in Table 51. For JTAG signal timing requirements, parameters and values, refer to "JTAG" on page 195.

Table 51. JTAG Instructions (Part 1 of 2)

JTAG Instruction	Description
SAMPLE/PRELOAD	Allows a snapshot of signals at the device pins to be captured and examined during normal device operation, and permits an initial data pattern to be output at the device pins. Also used by the SignalTap embedded logic analyzer
EXTEST	Allows the external circuitry and board-level interconnections to be tested by forcing a test pattern at the output pins and capturing test results at the input pins
BYPASS	Places the 1-bit bypass register between the `TDI` and `TDO` pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation
USERCODE	Selects the 32-bit `USERCODE` register and places it between the `TDI` and `TDO` pins, allowing the usercode to be serially shifted out of `TDO`

Table 51. JTAG Instructions (Part 2 of 2)	
JTAG Instruction	**Description**
IDCODE	Selects the IDCODE register and places it between TDI and TDO, allowing the ID code to be serially shifted out of TDO
ICR Instructions	Used when configuring via the JTAG port with a MasterBlaster or ByteBlasterMV download cable, or when using a Jam File or Jam Byte-Code File via an embedded processor
Communications Instructions	Allows a host to read and write memory accessible via AHB2, and provides a fast communications channel between host and target
SignalTap Instructions	Monitors internal device operation with the SignalTap embedded logic analyzer

PLD TAP Controller

The Excalibur family can be configured through the PLD JTAG port using the Quartus II software. The device can be in serial or parallel JTAG mode. The device can also be configured with hardware using either Jam Files (**.jam**) or Jam byte-code files (**.jbc**).

The APEX 20K device instruction register length is 10 bits. The APEX 20K device USERCODE register length is 32 bits. Tables 52 and 53 show the boundary-scan register length and device ID code information for the PLD TAP controller within Excalibur family.

Table 52. Boundary-Scan Register Length Note (1)	
Device	**Boundary-Scan Register Length**
EPXA1	1,203
EPXA4	1,638
EPXA10	2,217

Note:
(1) Contact Altera Applications for up-to-date information on this device.

Table 53. 32-Bit IDCODE

Device	IDCODE (32 Bits) *(1)*			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	1 (1 Bit) *(2)*
EPXA1	0000	1000 0001 0000 0001	000 0110 1110	1
EPXA4	0000	1000 0100 0000 0001	000 0110 1110	1
EPXA10	0000	1001 0000 0000 0001	000 0110 1110	1

Notes:
(1) The most significant bit (MSB) is on the left.
(2) The IDCODE's least significant bit (LSB) is always 1.

SignalTap Embedded Logic Analyzer

The PLD section includes enhancements to support the SignalTap embedded logic analyzer. By including this circuitry, the Excalibur family provides the ability to monitor design operation over a period of time through the IEEE Std. 1149.1 (JTAG) circuitry; a designer can analyze internal logic at speed without bringing internal signals to the I/O pins. This feature is particularly important for advanced packages such as FineLine BGA packages, because it can be difficult to add a connection to a pin during the debugging process after a board is designed and manufactured.

Embedded Processor TAP Controller

The ARM processor TAP controller instruction register length is 4 bits. Refer to the *ARM922T Technical Reference Manual* for a list of JTAG instructions and scan chains supported by this TAP controller.

The embedded processor ID code for EPXA10, EPXA4, and EPXA1 devices is given in Table 54.

Table 54. Embedded Processor ID Code

IDCODE (32 Bits) *(1)*			
Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	1 (1 Bit) *(2)*
0000	0100 1001 0010 0010	000 0110 1110	1

Notes:
(1) The most significant bit (MSB) is on the left.
(2) IDCODE's least significant bit (LSB) is always 1.

I/O Features

The family of Excalibur embedded processor PLDs maintains all of the existing APEX 20KE I/O features on the PLD I/O.

As with APEX 20KE devices, high-speed LVDS is supported on only –1 and –2 speed-grade devices. Devices that are –3 speed-grade support only low-speed ×1-mode LVDS. See the *APEX 20K Programmable Logic Device Family Data Sheet* for more information on LVDS.

I/O Control

Four banks of pins can be used directly by the embedded stripe. In EPXA10 devices, the pin banks are selectable for use by either the stripe or the PLD. In EPXA4 and EPXA1 devices, the pin banks are dedicated to the embedded stripe. The banks are as follows:

- SDRAM interface
- EBI (including miscellaneous signals)
- UART
- Trace port

The stripe selects whether each bank is controlled by the stripe or the PLD. When the stripe controls the bank, it can select the following I/O standard properties for it:

- Output control (open drain, slow/fast slew rate, enable PCI diode, enable JTAG debug)
- Input mode (2.5/3.3-V LVTTL, 1.8-V LVTTL, SSTL-3/GTL+, V_{REF})

Registers

A control register for each shared I/O bank dictates its I/O mode. In addition, for each shared I/O enabled as embedded I/O, there is a dedicated V_{REF} pin that drives the appropriate power bank, although only in modes that require V_{REF} (i.e., SSTL-3/GTL+).

If the selected input mode requires use of a voltage reference, the appropriate V_{REF} pins are enabled.

A lock bit, `LK`, in each register prevents inadvertent modification of the shared I/O standard or mode. If `LK` is set, a write to this register causes a bus error. `LK` remains set until a reset occurs.

All registers reset to 0, unless otherwise indicated.

Register: IOCR_SDRAM (SDRAM I/O bank control register)
Address: Register base + 40H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																									IC		OC			IO	LK

LK	R/W	When this bit is 1, further writes to the register generate a bus error
IO	R/W	Select bank: 1—Stripe 0—PLD
OC	R/W	Output control: xx0—Slow slew rate xx1—Fast slew rate x1x—PCI diode 1xx—Open drain
IC	R/W	Input control: 00—2.5/3.3-V LVTTL 01—1.8-V LVTTL 10—SSTL-3/GTL+ 11—Reserved
0	R	Reserved for future use. Write as 0 to ensure future compatibility

Register: IOCR_EBI (EBI I/O bank control register)
Address: Register base + 44H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																									IC		OC			IO	LK

The meaning of bits in this register is the same as for IOCR_SDRAM.

At reset, the value of this register depends on the BOOT_FLASH pin. If booting from flash, this register is reset to 2H or 22H, (depending on the value of MSEL0), otherwise it is reset to 0.

Register: IOCR_UART (UART I/O bank control register)
Address: Register base + 48H
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																									IC		OC			IO	LK

The meaning of bits in this register is the same as for IOCR_SDRAM.

Register: IOCR_TRACE (trace I/O bank control register)
Address: Register base + 4CH
Access: Read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																											OC			IO	LK

LK	R/W	When this bit is 1, further writes to the register generate a bus error
IO	R/W	Select bank: 1—Stripe 0—PLD
OC	R/W	Output control: xx0—Slow slew rate xx1—Fast slew rate x1x—PCI diode 1xx—Open drain
0	R	Reserved for future use. Write as 0 to ensure future compatibility

The meaning of bits in this register is the same as for IOCR_SDRAM.

Dedicated I/O

Some I/O pins are dedicated to the embedded stripe, and others are dedicated to the PLD portion of the device. Some embedded stripe dedicated I/O pins and PLD dedicated I/O pins are listed in Table 55. For detailed information about embedded stripe dedicated I/O, refer to the UART, Expansion Bus Interface, SDRAM controller, and Trace Support sections of this document.

Table 55. Device Dedicated I/O Pins (Part 1 of 2)

Pin	Direction	Use
nRESET	Input/Output	Active-low bidirectional reset pin
nPOR	Input	Active-low power-on reset pin
BOOT_FLASH	Input	Boot mode select pin. When high, the embedded processor boots from flash memory on the EBI; when low, boot-from-serial mode is used
DEBUG_EN		Debug: enables the hardware trigger in the watchdog
JSELECT	Input	JTAG mode select pin. When high, the device is in parallel JTAG mode. When low, the device is in serial JTAG mode. See "IEEE Std. 1149.1 (JTAG) Support" on page 163
TCK	Input	JTAG TAP controller test clock pin
TDI	Input	JTAG TAP controller test data input pin
TDO	Output	JTAG TAP controller test data output pin
TMS	Input	JTAG TAP controller test mode select pin
TRST	Input	JTAG TAP controller test reset input pin
PROC_TCK	Input	Processor JTAG TAP controller test clock pin

Table 55. Device Dedicated I/O Pins (Part 2 of 2)

Pin	Direction	Use
PROC_TDI	Input	Processor JTAG TAP controller test data input pin
PROC_TDO	Output	Processor JTAG TAP controller test data output pin
PROC_TMS	Input	Processor JTAG TAP controller test mode select pin.
PROC_TRST	Input	Active-low processor JTAG TAP controller test reset input pin.
nSTATUS	Input/Output	Configuration pin that indicates that the device is being initialized or has encountered an error during initialization
nCONFIG	Input	Active-low input that initiates device reconfiguration
MSEL0	Input	Configuration mode select pin
MSEL1	Input	Configuration mode select pin
CONF_DONE	Input/Output	Active-high bidirectional pin that indicates completion of device configuration
nCE	Input	Active-low chip enable used in multi-device configuration schemes
nCEO	Output	Active-low chip enable output used in multi-device configuration schemes
DATA0	Input	Configuration data input
DATA[1..7]	Input	Configuration data input for parallel modes. In other programming modes this pin can be used as user I/O
DCLK	Input	Configuration clock input
CLK1p	Input	Dedicated global clock input pin and input to PLD PLL1
CLK2p	Input	Dedicated global clock input pin and input to PLD PLL2
CLK3p	Input	Dedicated global clock input pin and input to PLD PLL3
CLK4p	Input	Dedicated global clock input pin and input to PLD PLL4
LOCK[1..4]	Output	PLL lock status pins. When the PLL circuitry is locked to the incoming clock and generates an internal clock, LOCK is driven high. If not used for the PLL function, the pins are available as user I/Os
CLKLK_ENA	Input	Active-high enable pin for all PLL circuits in the device. When de-asserted, all PLLs are reset to their default, unlocked state and stop clocking. When re-asserted, the PLLs lock again and resume clocking. If this pin function is not needed, the pin should be connected to VCCINT
CLKLK_OUT1p	Output	Dedicated external output for PLD PLL1
CLKLK_OUT2p	Output	Dedicated external output for PLD PLL2
CLKLK_FB1p	Input	Dedicated input that allows external feedback to PLD PLL1
CLKLK_FB2p	Input	Dedicated input that allows external feedback to PLD PLL2
FAST[1..3]	Input	Input pins that drive dedicated fast input lines
CLK_REF	Input	Embedded stripe clock input pin to the embedded stripe PLLs

The PLD fast I/O pins are retained for PLD use and are not shared.

Separate power banks are supplied to the following peripherals:

- SDRAM interface—With additional V_{CCIO} pins and V_{REF} supply, enables SSTL-2 operation and allows the other embedded logic I/O to interface to a different I/O standard
- EBI
- UART
- Trace port

SDRAM Controller

The SDRAM controller supports SDR SDRAM and DDR SDRAM, which require two different IO standards to be supported:

- LVTTL for SDRAM
- SSTL-2 for DDR SDRAM

Operating Conditions

Tables 56 through 58 provide information on absolute maximum ratings, recommended operating conditions, DC operating conditions, and capacitance for 1.8-V APEX 20KE devices.

Table 56. Absolute Maximum Ratings *Note (1)*

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CCINT}	Supply voltage	With respect to ground *(2)*	–0.5	2.5	V
V_{CCIO}			–0.5	4.6	V
V_I	DC input voltage		–0.5	4.6	V
I_{OUT}	DC output current, per pin		–25	25	mA
T_{STG}	Storage temperature	No bias	–65	150	° C
T_{AMB}	Ambient temperature	Under bias	–65	135	° C
T_J	Junction temperature	PQFP, RQFP, TQFP, and BGA packages, under bias		135	° C
		Ceramic PGA packages, under bias		150	° C

Table 57. Recommended Operating Conditions *(Part 1 of 2)*

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CCINT}	Supply voltage for internal logic and input buffers	*(3)*, *(4)*	1.71 (1.71)	1.89 (1.89)	V
V_{CCIO}	Supply voltage for output buffers, 3.3-V operation	*(3)*, *(4)*	3.00 (3.00)	3.60 (3.60)	V
	Supply voltage for output buffers, 2.5-V operation	*(3)*, *(4)*	2.375 (2.375)	2.625 (2.625)	V
V_I	Input voltage	*(2)*, *(5)*	–0.5	4.1	V

Table 57. Recommended Operating Conditions *(Part 2 of 2)*

Symbol	Parameter	Conditions	Min	Max	Unit
V_O	Output voltage		0	V_{CCIO}	V
T_J	Operating temperature	For commercial use	0	85	°C
		For industrial use	–40	100	°C
t_R	Input rise time			40	ns
t_F	Input fall time			40	ns

For DC operating specifications on APEX 20KE I/O standards, please refer to *Application Note 117—Using Selectable I/O Standards in Altera Devices*.

Table 58. Capacitance *Note (12)*

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input capacitance	V_{IN} = 0 V, f = 1.0 MHz		8	pF
C_{INCLK}	Input capacitance on dedicated clock pin	V_{IN} = 0 V, f = 1.0 MHz		12	pF
C_{OUT}	Output capacitance	V_{OUT} = 0 V, f = 1.0 MHz		8	pF

Table 59. DC Operating Conditions *(Part 1 of 2) Notes (6), (7)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High-level LVTTL, CMOS, or 3.3-V PCI input voltage		1.7, 0.5 × V_{CCIO} *(8)*		4.1	V
V_{IL}	Low-level LVTTL, CMOS, or 3.3-V PCI input voltage		–0.5		0.8, 0.3 × V_{CCIO} *(8)*	V
V_{OH}	3.3-V high-level LVTTL output voltage	I_{OH} = –12 mA DC, V_{CCIO} = 3.00 V *(9)*	2.4			V
	3.3-V high-level LVCMOS output voltage	I_{OH} = –0.1 mA DC, V_{CCIO} = 3.00 V *(9)*	V_{CCIO} – 0.2			V
	3.3-V high-level PCI output voltage	I_{OH} = –0.5 mA DC, V_{CCIO} = 3.00 to 3.60 V *(9)*	0.9 × V_{CCIO}			V
	2.5-V high-level output voltage	I_{OH} = –0.1 mA DC, V_{CCIO} = 2.30 V *(9)*	2.1			V
		I_{OH} = –1 mA DC, V_{CCIO} = 2.30 V *(9)*	2.0			V
		I_{OH} = –2 mA DC, V_{CCIO} = 2.30 V *(9)*	1.7			V

Table 59. DC Operating Conditions *(Part 2 of 2)* *Notes (6), (7)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{OL}	3.3-V low-level LVTTL output voltage	I_{OL} = 12 mA DC, V_{CCIO} = 3.00 V *(10)*			0.4	V
	3.3-V low-level LVCMOS output voltage	I_{OL} = 0.1 mA DC, V_{CCIO} = 3.00 V *(10)*			0.2	V
	3.3-V low-level PCI output voltage	I_{OL} = 1.5 mA DC, V_{CCIO} = 3.00 to 3.60 V *(10)*			$0.1 \times V_{CCIO}$	V
	2.5-V low-level output voltage	I_{OL} = 0.1 mA DC, V_{CCIO} = 2.30 V *(10)*			0.2	V
		I_{OL} = 1 mA DC, V_{CCIO} = 2.30 V *(10)*			0.4	V
		I_{OL} = 2 mA DC, V_{CCIO} = 2.30 V *(10)*			0.7	V
I_I	Input pin leakage current	V_I = 4.1 to –0.5 V	–10		10	µA
I_{OZ}	Tri-stated I/O pin leakage current	V_O = 4.1 to –0.5 V	–10		10	µA
I_{CC0}	V_{CC} supply current (standby) (All ESBs in power-down mode)	V_I = ground, no load, no toggling inputs, –1 speed grade		10		mA
		V_I = ground, no load, no toggling inputs, –2, –3 speed grades		5		mA
R_{CONF}	Value of I/O pin pull-up resistor before and during configuration	V_{CCIO} = 3.0 V *(11)*	20		50	kΩ
		V_{CCIO} = 2.375 V *(11)*	30		80	kΩ
		V_{CCIO} = 1.71 V *(11)*	60		150	kΩ

Notes to tables:

(1) See the *Operating Requirements for Altera Devices Data Sheet*.
(2) Minimum DC input is –0.5 V. During transitions, the inputs may undershoot to –0.5 V or overshoot to 4.6 V for input currents less than 100 mA and periods shorter than 20 ns.
(3) Numbers in parentheses are for industrial-temperature-range devices.
(4) Maximum V_{CC} rise time is 100 ms, and V_{CC} must rise monotonically.
(5) All pins, including dedicated inputs, clock, I/O, and JTAG pins, may be driven before V_{CCINT} and V_{CCIO} are powered.
(6) Typical values are for T_A = 25° C, V_{CCINT} = 1.8 V, and V_{CCIO} = 1.8 V, 2.5 V or 3.3 V.
(7) These values are specified under the APEX 20K device recommended operating conditions, shown in Table 57 on page 171.
(8) The APEX 20K input buffers are compatible with 1.8 -V, 2.5-V and 3.3-V (LVTTL and LVCMOS) signals. Additionally, the input buffers are 3.3-V PCI compliant. Input buffers also meet specifications for GTL+, CTT, AGP, SSTL-2, SSTL-3, and HSTL.
(9) The I_{OH} parameter refers to high-level TTL, PCI, or CMOS output current.
(10) The I_{OL} parameter refers to low-level TTL, PCI, or CMOS output current. This parameter applies to open-drain pins as well as output pins.
(11) Pin pull-up resistance values will be lower if an external source drives the pin higher than V_{CCIO}.
(12) Capacitance is sample-tested only.

Electrical Characteristics & Timing Diagrams

The following sections describe the electrical characteristics and show timing diagrams for the modules in the Excalibur embedded processor PLDs.

Embedded Stripe Bridges

See the *AMBA Specification* for the definition and timing diagrams of the AHB timing parameters.

Stripe-to-PLD Bridge Timing Parameters

The following sections list the input and output timing parameters for the stripe-to-PLD bridge.

Stripe-to-PLD Bridge Input Timing Parameters

Tables 60 to 62 list the stripe-to-PLD bridge timing parameters for the Excalibur devices.

Table 60 shows the input timing parameters for the EPXA10 stripe-to-PLD bridge.

Table 60. EPXA10 Input Timing Parameters for the Stripe-to-PLD Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tclk	MASTER_HCLK minimum clock period time	6.54		7.88		9.85		ns
Tisgnt	MASTER_HGRANT setup time before MASTER_HCLK		4.16		5.10		6.25	ns
Tihgnt	MASTER_HGRANT hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrdy	MASTER_HREADY setup time before MASTER_HCLK		3.61		4.35		5.43	ns
Tihrdy	MASTER_HRESP hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrsp	MASTER_HRESP setup time before MASTER_HCLK		3.97		4.78		5.97	ns
Tihrsp	MASTER_HRESP hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrdy	MASTER_HDATA setup time before MASTER_HCLK		1.50		1.78		2.20	ns
Tihrd	MASTER_HDATA hold time after MASTER_HCLK	0.00		0.00		0.00		ns

Table 61 on page 175 shows the input timing parameters for the EPXA4 stripe-to-PLD bridge.

Table 61. EPXA4 Input Timing Parameters for the Stripe-to-PLD Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tclk	MASTER_HCLK minimum clock period time	5.82		6.85		8.57		ns
Tisgnt	MASTER_HGRANT setup time before MASTER_HCLK		2.80		3.36		4.46	ns
Tihgnt	MASTER_HGRANT hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrdy	MASTER_HREADY setup time before MASTER_HCLK		3.21		3.86		5.13	ns
Tihrdy	MASTER_HRESP hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrsp	MASTER_HRESP setup time before MASTER_HCLK		3.41		4.10		5.44	ns
Tihrsp	MASTER_HRESP hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrdy	MASTER_HDATA setup time before MASTER_HCLK		1.00		1.10		1.45	ns
Tihrd	MASTER_HDATA hold time after MASTER_HCLK	0.00		0.00		0.00		ns

Table 62 shows the input timing parameters for the EPXA1 stripe-to-PLD bridge.

Table 62. EPXA1 Input Timing Parameters for the Stripe-to-PLD Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tclk	MASTER_HCLK minimum clock period time	4.70		5.45		6.80		ns
Tisgnt	MASTER_HGRANT setup time before MASTER_HCLK		3.15		3.79		5.03	ns
Tihgnt	MASTER_HGRANT hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrdy	MASTER_HREADY setup time before MASTER_HCLK		3.54		4.28		5.68	ns
Tihrdy	MASTER_HRESP hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrsp	MASTER_HRESP setup time before MASTER_HCLK		3.20		3.85		5.11	ns
Tihrsp	MASTER_HRESP hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tisrdy	MASTER_HDATA setup time before MASTER_HCLK		1.67		2.04		2.63	ns
Tihrd	MASTER_HDATA hold time after MASTER_HCLK	0.00		0.00		0.00		ns

Stripe-to-PLD Bridge Output Timing Parameters

Tables 63 to 65 list the stripe-to-PLD bridge timing parameters for the Excalibur devices.

Table 63 shows the output timing parameters for the EPXA10 stripe-to-PLD bridge.

Table 63. EPXA10 Output Timing Parameters for the Stripe-to-PLD Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tovtr	MASTER_HTRANS valid time after MASTER_HCLK		3.23		3.89		4.86	ns
Tohtr	MASTER_HTRANS hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tova	MASTER_HADDR valid time after MASTER_HCLK		3.43		4.14		5.16	ns
Toha	MASTER_HCLK hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovctl	Control signals valid time after MASTER_HCLK							ns
Tohctl	Control signal hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovwd	MASTER_HWDATA valid time after MASTER_HCLK		3.65		4.40		5.49	ns
Tohwd	MASTER_HWDATA hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovreq	MASTER_HBUSREQ valid time after MASTER_HCLK		3.37		4.49		5.06	ns
Tohreq	MASTER_HBUSREQ hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovlck	MASTER_HLOCK valid time after MASTER_HCLK		3.26		3.93		4.90	ns
Tohlck	MASTER_HLOCK hold time after MASTER_HCLK	0.00		0.00		0.00		ns

Table 64 shows the output timing parameters for the EPXA4 stripe-to-PLD bridge.

Table 64. EPXA4 Output Timing Parameters for the Stripe-to-PLD Bridge (Part 1 of 2)

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tovtr	MASTER_HTRANS valid time after MASTER_HCLK		3.53		4.24		5.63	ns
Tohtr	MASTER_HTRANS hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tova	MASTER_HADDR valid time after MASTER_HCLK		3.45		4.15		5.51	ns
Toha	MASTER_HCLK hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovctl	Control signals valid time after MASTER_HCLK							ns
Tohctl	Control signal hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovwd	MASTER_HWDATA valid time after MASTER_HCLK		3.72		4.47		5.93	ns
Tohwd	MASTER_HWDATA hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovreq	MASTER_HBUSREQ valid time after MASTER_HCLK		3.45		4.15		5.51	ns

Table 64. EPXA4 Output Timing Parameters for the Stripe-to-PLD Bridge (Part 2 of 2)

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tohreq	MASTER_HBUSREQ hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovlck	MASTER_HLOCK valid time after MASTER_HCLK		3.22		3.87		5.14	ns
Tohlck	MASTER_HLOCK hold time after MASTER_HCLK	0.00		0.00		0.00		ns

Table 65 shows the output timing parameters for the EPXA1 stripe-to-PLD bridge.

Table 65. EPXA1 Output Timing Parameters for the Stripe-to-PLD Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tovtr	MASTER_HTRANS valid time after MASTER_HCLK		2.89		3.48		4.62	ns
Tohtr	MASTER_HTRANS hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tova	MASTER_HADDR valid time after MASTER_HCLK		3.31		3.98		5.28	ns
Toha	MASTER_HCLK hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovctl	Control signals valid time after MASTER_HCLK		3.45		4.14		5.50	ns
Tohctl	Control signal hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovwd	MASTER_HWDATA valid time after MASTER_HCLK		3.19		3.83		5.10	ns
Tohwd	MASTER_HWDATA hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovreq	MASTER_HBUSREQ valid time after MASTER_HCLK		2.99		3.60		4.78	ns
Tohreq	MASTER_HBUSREQ hold time after MASTER_HCLK	0.00		0.00		0.00		ns
Tovlck	MASTER_HLOCK valid time after MASTER_HCLK		3.10		3.70		4.80	ns
Tohlck	MASTER_HLOCK hold time after MASTER_HCLK	0.00		0.00		0.00		ns

PLD-to-Stripe Bridge Timing Parameters

The following sections list the input and output timing parameters for the PLD-to-stripe bridge.

PLD-to-Stripe Bridge Input Timing Parameters

Tables 66 to 68 list the stripe-to-PLD bridge timing parameters for the Excalibur devices.

Table 66 shows the input timing parameters for the EPXA10 PLD-to-stripe bridge.

Table 66. EPXA10 Input Timing Parameters for the PLD-to-Stripe Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tclk	SLAVE_HCLK minimum clock period	10.76		12.83		10.31		ns
Tisselreg	SLAVE_HSELREG setup time before SLAVE_HCLK		3.54		4.27		5.33	ns
Tihselreg	SLAVE_HSELREG hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tissel	SLAVE_HSEL setup time before SLAVE_HCLK		3.84		4.62		5.77	ns
Tihsel	SLAVE_HSEL hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tistr	SLAVE_HSEL setup time before SLAVE_HCLK		4.10		4.93		6.15	ns
Tihtr	SLAVE_HSEL hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tisa	SLAVE_HADDR setup time before SLAVE_HCLK		4.00		4.70		5.87	ns
Tiha	SLAVE_HADDR hold time after SLAVE_HSIZE	0.00		0.00		0.00		ns
Tisctl	SLAVE_HWRITE, SLAVE_HSIZE and SLAVE_HBURST setup time before SLAVE_HCLK		3.71		4.47		5.58	ns
Tihctl	SLAVE_HWRITE, SLAVE_HSIZE and SLAVE_HBURST hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tiswd	SLAVE_HWDATA setup time before SLAVE_HCLK		1.98		2.40		2.97	ns
Tihwd	SLAVE_HWDATA hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tisrdy	SLAVE_HREADY setup time before SLAVE_HCLK		3.76		4.53		5.66	ns
Tihrdy	SLAVE_HREADY hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tismlck	SLAVE_HMASTLOCK setup time before SLAVE_HCLK		1.55		1.87		2.33	ns
Tihmlck	SLAVE_HMASTLOCK hold time after SLAVE_HCLK	0.00		0.00		0.00		ns

Table 67 shows the input timing parameters for the EPXA4 PLD-to-stripe bridge.

Table 67. EPXA4 Input Timing Parameters for the PLD-to-Stripe Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tclk	SLAVE_HCLK minimum clock period	10.31		12.20		15.25		ns
Tisselreg	SLAVE_HSELREG setup time before SLAVE_HCLK		1.97		2.37		3.15	ns
Tihselreg	SLAVE_HSELREG hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tissel	SLAVE_HSEL setup time before SLAVE_HCLK		2.69		3.23		4.29	ns
Tihsel	SLAVE_HSEL hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tistr	SLAVE_HSEL setup time before SLAVE_HCLK		2.67		3.21		4.26	ns
Tihtr	SLAVE_HSEL hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tisa	SLAVE_HADDR setup time before SLAVE_HCLK		2.65		3.19		4.23	ns
Tiha	SLAVE_HADDR hold time after SLAVE_HSIZE	0.00		0.00		0.00		ns
Tisctl	SLAVE_HWRITE, SLAVE_HSIZE and SLAVE_HBURST setup time before SLAVE_HCLK		2.66		3.20		4.25	ns
Tihctl	SLAVE_HWRITE, SLAVE_HSIZE and SLAVE_HBURST hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tiswd	SLAVE_HWDATA setup time before SLAVE_HCLK		1.00		1.00		1.00	ns
Tihwd	SLAVE_HWDATA hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tisrdy	SLAVE_HREADY setup time before SLAVE_HCLK		2.66		3.20		4.25	ns
Tihrdy	SLAVE_HREADY hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tismlck	SLAVE_HMASTLOCK setup time before SLAVE_HCLK		1.00		1.00		1.28	ns
Tihmlck	SLAVE_HMASTLOCK hold time after SLAVE_HCLK	0.00		0.00		0.00		ns

Table 68 shows the input timing parameters for the EPXA1 PLD-to-stripe bridge.

Table 68. EPXA1 Input Timing Parameters for the PLD-to-Stripe Bridge (Part 1 of 2)

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tclk	SLAVE_HCLK minimum clock period	8.27		9.91		12.38		ns
Tisselreg	SLAVE_HSELREG setup time before SLAVE_HCLK		1.28		1.54		2.04	ns
Tihselreg	SLAVE_HSELREG hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tissel	SLAVE_HSEL setup time before SLAVE_HCLK		2.02		2.43		3.22	ns
Tihsel	SLAVE_HSEL hold time after SLAVE_HCLK	0.00		0.00		0.00		ns

Table 68. EPXA1 Input Timing Parameters for the PLD-to-Stripe Bridge (Part 2 of 2)

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tistr	SLAVE_HSEL setup time before SLAVE_HCLK		1.90		2.28		3.03	ns
Tihtr	SLAVE_HSEL hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tisa	SLAVE_HADDR setup time before SLAVE_HCLK		2.40		2.88		3.82	ns
Tiha	SLAVE_HADDR hold time after SLAVE_HSIZE	0.00		0.00		0.00		ns
Tisctl	SLAVE_HWRITE, SLAVE_HSIZE and SLAVE_HBURST setup time before SLAVE_HCLK		1.87		2.25		2.98	ns
Tihctl	SLAVE_HWRITE, SLAVE_HSIZE and SLAVE_HBURST hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tiswd	SLAVE_HWDATA setup time before SLAVE_HCLK		1.21		1.46		1.93	ns
Tihwd	SLAVE_HWDATA hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tisrdy	SLAVE_HREADY setup time before SLAVE_HCLK		1.95		2.35		3.11	ns
Tihrdy	SLAVE_HREADY hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tismlck	SLAVE_HMASTLOCK setup time before SLAVE_HCLK		1.00		1.10		1.46	ns
Tihmlck	SLAVE_HMASTLOCK hold time after SLAVE_HCLK	0.00		0.00		0.00		ns

PLD-to-Stripe Bridge Output Timing Parameters

Tables 69 to 73 list the PLD-to-stripe bridge timing parameters for the Excalibur devices.

Table 69 shows the output timing parameters for the EPXA10 PLD-to-stripe bridge.

Table 69. EPXA10 Output Timing Parameters for the PLD-to-Stripe Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tovrsp	SLAVE_HRESP valid time after SLAVE_HCLK		2.95		3.56		4.44	ns
Tohrsp	SLAVE_HRESP hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tovrdy	SLAVE_HREADY valid time after SLAVE_HCLK		2.83		3.41		4.26	ns
Tohrdy	SLAVE_HREADY hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tovrd	SLAVE_HRDATA valid time after SLAVE_HCLK		3.97		4.79		5.97	ns
Tohrd	SLAVE_HRDATA hold time after SLAVE_HCLK	0.00		0.00		0.00		ns

Table 70 shows the output timing parameters for the EPXA4 PLD-to-stripe bridge.

Table 70. EPXA4 Output Timing Parameters for the PLD-to-Stripe Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tovrsp	SLAVE_HRESP valid time after SLAVE_HCLK		3.33		4.00		5.32	ns
Tohrsp	SLAVE_HRESP hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tovrdy	SLAVE_HREADY valid time after SLAVE_HCLK		3.21		3.86		5.13	ns
Tohrdy	SLAVE_HREADY hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tovrd	SLAVE_HRDATA valid time after SLAVE_HCLK		3.60		4.33	5.75		ns
Tohrd	SLAVE_HRDATA hold time after SLAVE_HCLK	0.00		0.00		0.00		ns

Table 71 shows the output timing parameters for the EPXA1 PLD-to-stripe bridge.

Table 71. EPXA1 Output Timing Parameters for the PLD-to-Stripe Bridge

Parameter	Description	Speed –1		Speed –2		Speed –3		Units
		Min	Max	Min	Max	Min	Max	
Tovrsp	SLAVE_HRESP valid time after SLAVE_HCLK		2.80		3.38		4.49	ns
Tohrsp	SLAVE_HRESP hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tovrdy	SLAVE_HREADY valid time after SLAVE_HCLK		2.60		3.13		4.15	ns
Tohrdy	SLAVE_HREADY hold time after SLAVE_HCLK	0.00		0.00		0.00		ns
Tovrd	SLAVE_HRDATA valid time after SLAVE_HCLK		3.39		4.10		5.41	ns
Tohrd	SLAVE_HRDATA hold time after SLAVE_HCLK	0.00		0.00		0.00		ns

EBI

Figures 47 to 50 show the timing requirements for the EBI. Figure 47 shows the timing requirements for EBI_CLK.

Figure 47. EBI Timing Diagrams: EBI_CLK

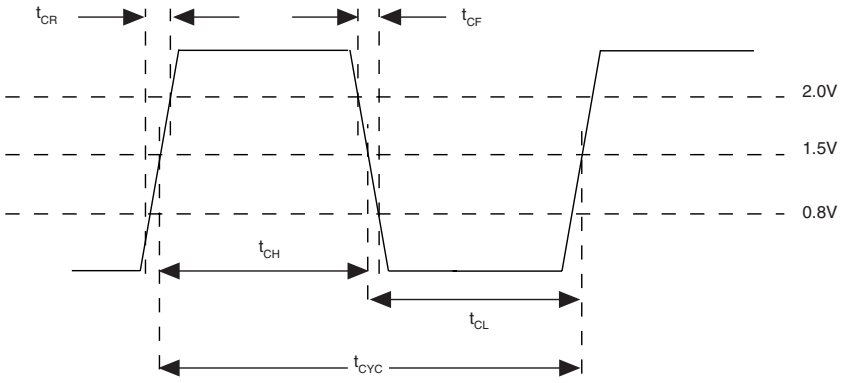


Table 72 lists the EBI timing parameters for the EBI_CLK waveforms shown in Figure 47.

Table 72. EBI Waveform Parameters & Values: EBI_CLK

Symbol	Parameter	–1		–2		–3		Units
		Min	Max	Min	Max	Min	Max	
t_{CH}	EBI_CLK high-level width	4.00		4.66		5.32		ns
t_{CL}	EBI_CLK low-level width	4.00		4.66		5.32		ns
t_{CR}	EBI_CLK rise time	.50	3.00	.58	3.50	.67	4.00	ns
t_{CF}	EBI_CLK fall time	.50	3.00	.58	3.50	.67	4.00	ns
t_{CYC}	EBI_CLK period *(1)*	1.00	16.00	1.00	16.00	1.00	16.00	t_{ahb2p}

Table 73 shows the EBI parameter values used in timing algorithms.

Table 73. EBI Parameter Values used in Timing Algorithms

Symbol	Parameter	–1		–2		–3		Units
		Min	Max	Min	Max	Min	Max	
t_{ahb2p}	AHB2 period	10.00		12.00		15.00		ns
ECD	EBI clock divide (CLK_DIV in EBI_CR) *(1)*	1.00	16.00	1.00	16.00	1.00	16.00	ns
EWS	EBI wait state(s) (WAIT field in register EBI_BLOCKn)	0.00	15.00	0.00	15.00	0.00	15.00	ns

Notes:
(1) The EBI clock period is a function of AHB2 clock period and the EBI clock-divide ratio

Figure 48 shows the timing requirements for a synchronous EBI read with no wait states. Up to 15 additional T2 cycles can be added.

Figure 48. EBI Timing Diagrams: Synchronous Read, Zero Wait States (EWS=0)

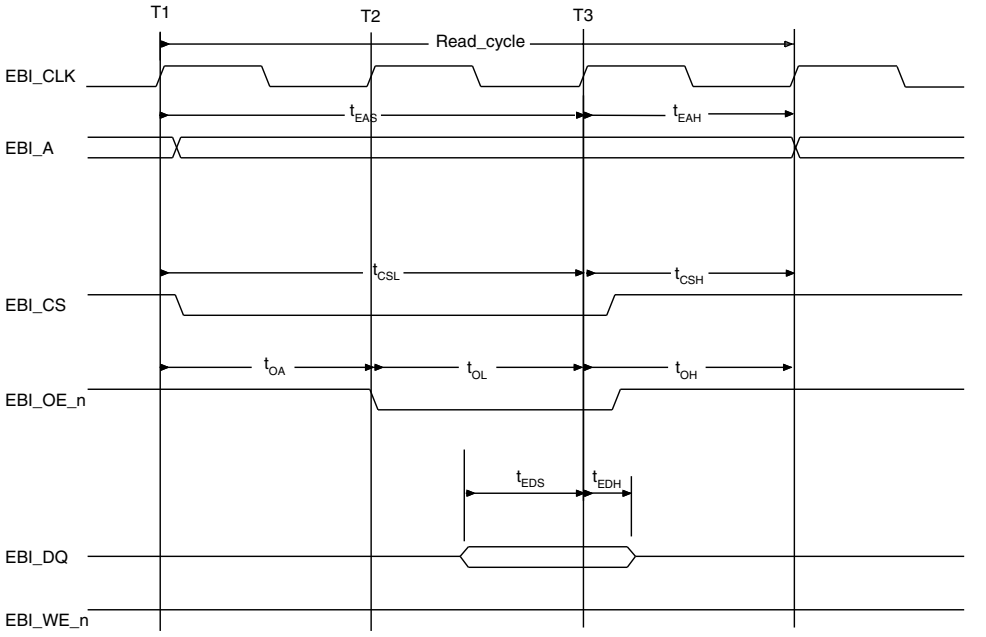


Figure 49 on page 184 shows the timing requirements for a synchronous EBI write with no wait states. Up to 15 additional T2 cycles can be added.

Figure 49. EBI Timing Diagrams: Synchronous Write, Zero Wait States (EWS=0)

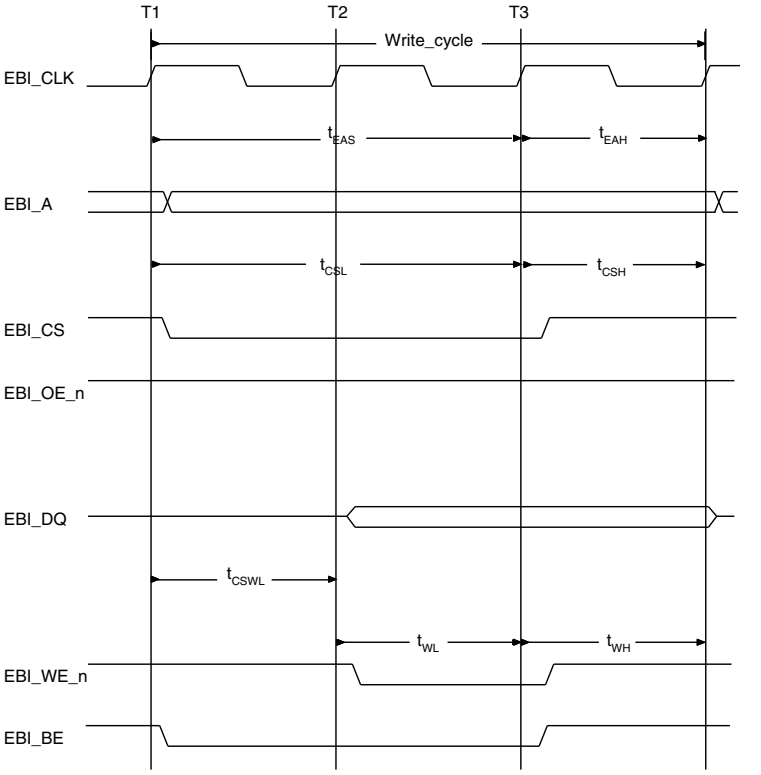


Figure 50 shows the timing requirements for synchronous EBI signals.

Figure 50. EBI Timing Diagrams: t_{OV_S} Output Delay

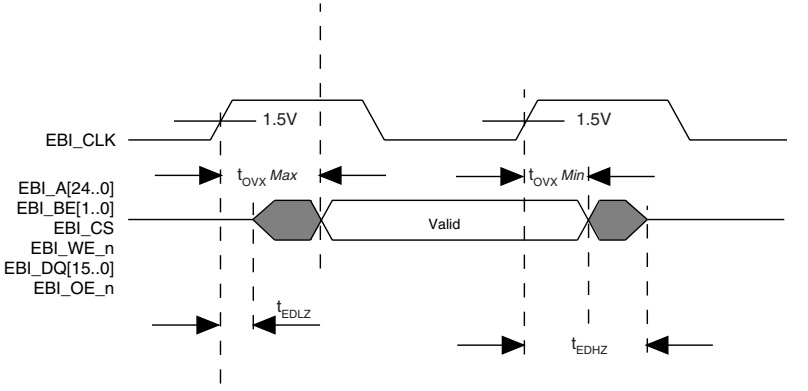


Table 74 shows the EBI timing parameters and values for synchronous transfers.

Table 74. EBI Timing Parameters & Values

Symbol	Parameter	–1		–2		–3		Units (EBI_CLK)
		Min	Max	Min	Max	Min	Max	
tread_cycle	Read-cycle time	3.00	(3 + EWS) × ECD	3.00	(3 + EWS) × ECD	3.00	(3 + EWS) × ECD	t_{ahb2p}
twrite_cycle	Write-cycle time	3.00	(3 + EWS) × ECD	3.00	(3 + EWS) × ECD	3.00	(3 + EWS) × ECD	t_{ahb2p}
t_{EAS}	Address setup to EBI_CS high	4.00	(2 + EWS) × ECD	4.00	(2 + EWS) × ECD	4.00	(2 + EWS) × ECD	t_{ahb2p}
t_{EAH}	Address hold from EBI_CS high	1.00		1.00		1.00		t_{ahb2p}
t_{CSL}	Chip-select low	2.00	(2 + EWS) × ECD	2.00	(2 + EWS) × ECD	2.00	(2 + EWS) × ECD	t_{ahb2p}
t_{CSH}	Chip-select high *(1)*	3.00		3.00		3.00		t_{ahb2p}
t_{OA}	EBI_CS low to EBI_OE_n low	1.00	ECD	1.00	ECD	1.00	ECD	t_{ahb2p}
t_{OL}	EBI_OE_n low	1.00	(1 + EWS) × ECD	1.00	(1 + EWS) × ECD	1.00	(1 + EWS) × ECD	t_{ahb2p}
t_{OH}	EBI_OE_n high *(2)*	4.00		4.00		4.00		t_{ahb2p}
t_{WL}	Write-enable low	1.00	(1 + EWS) × ECD	1.00	(1 + EWS) × ECD	1.00	(1 + EWS) × ECD	t_{ahb2p}
t_{WH}	Write-enable high *(2)*	4.00		4.00		4.00		t_{ahb2p}
t_{CSWL}	Chip select to write-enable low	1.00	ECD	1.00	ECD	1.00	ECD	t_{ahb2p}
t_{OV5}	EBI_A[24..0] output valid delay	0.10	3.00	0.12	3.50	0.13	3.99	ns
t_{OV6}	EBI_OE-n, EBI_WE_n output valid delay	0.30	8.00	0.35	9.32	0.84	10.64	ns
t_{OV7}	EBI_DQ output valid delay	0.00	1.00	0.00	1.17	0.00	1.33	ns
t_{EDS}	EBI_DQ setup time	4.00		4.66		5.32		ns
t_{EDH}	EBI_DQ hold time	0.00		0.00		0.00		ns

Notes

(1) For ECD = 1, minimum EBI_CS high time is 3 EBI clock periods (3 AHB2 clock periods). In all other cases it is 2 EBI clock periods (2 × ECD - AHB periods)

(2) For ECD = 1, minimum EBI_WE and EBI_OE high time is 4 EBI clock periods (4 AHB2 clock periods). In all other cases it is 3 EBI clock periods (3 × ECD - AHB periods)

SDRAM

Figures 51 to 56 show the SDR SDRAM timing requirements. Figures 57 to 59 show the DDR SDRAM timing requirements.

Figure 51 shows the timing requirements for an SDRAM read.

Figure 51. SDRAM Timing Diagram: Read

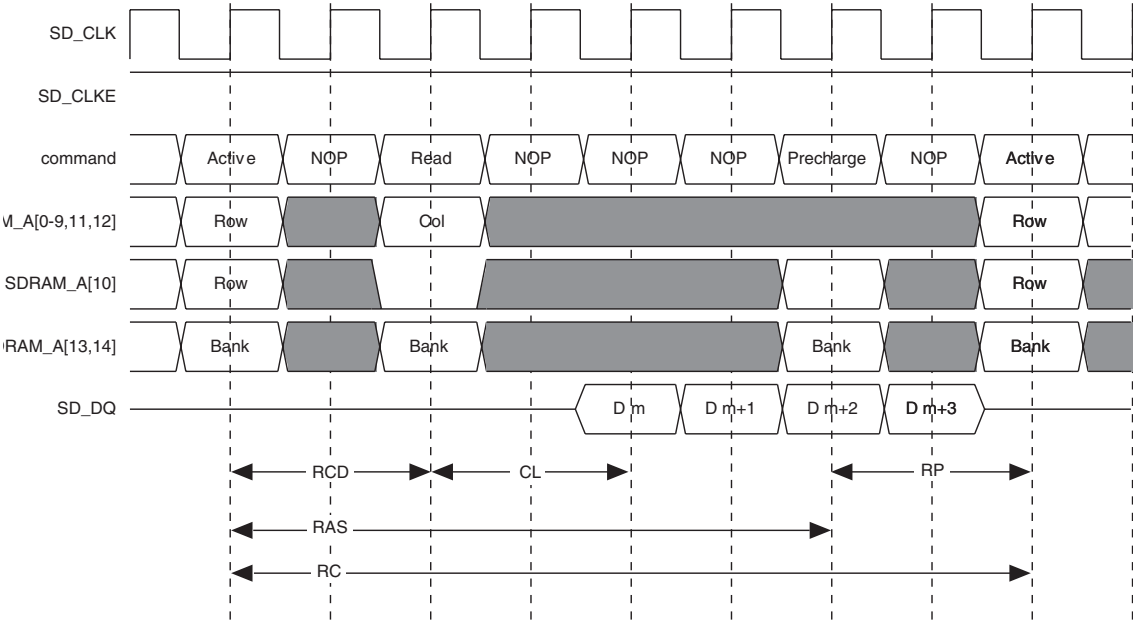


Figure 52 on page 187 shows the timing requirements for an SDRAM write.

Figure 52. SDRAM Timing Diagram: Write

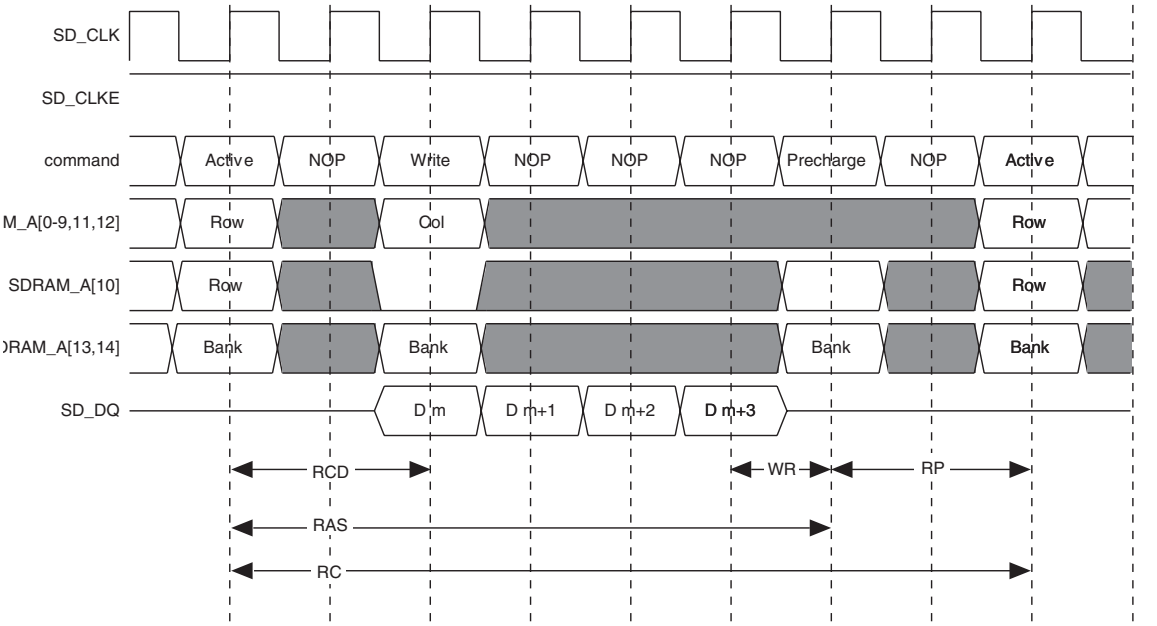


Figure 53 shows the timing requirements for an SDRAM auto refresh.

Figure 53. SDRAM Timing Diagram: Auto Refresh

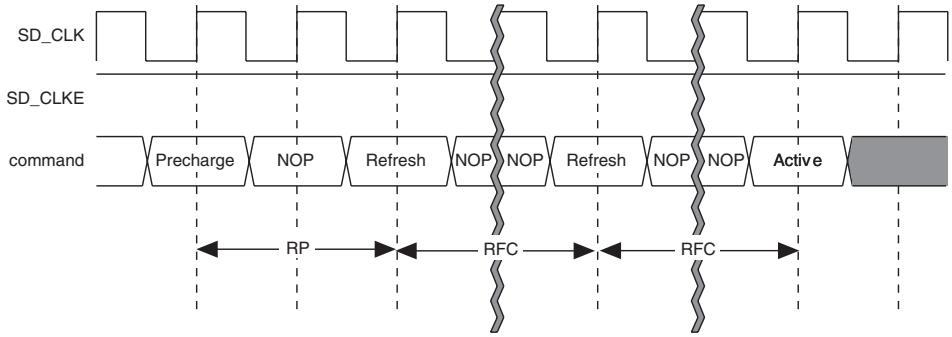


Table 75 on page 188 shows the SDR SDRAM timing parameters and values.

Table 75. SDRAM Timing Parameters & Values				
Symbol	**Parameter**	**Min**	**Max**	**Units**
CL	CAS latency	2	3	Cycles
RAS	Active to precharge command	1	8	Cycles
RC	Active to active command	1	10	Cycles
RCD	Active to read or write delay	1	4	Cycles
RFC	Auto refresh period	3	11	Cycles
RP	Precharge command period	1	4	Cycles
WR	Write recovery time	1	3	Cycles

Figure 54 shows the timing requirements for the SDRAM clock.

Figure 54. SDRAM Timing Diagram: Clock

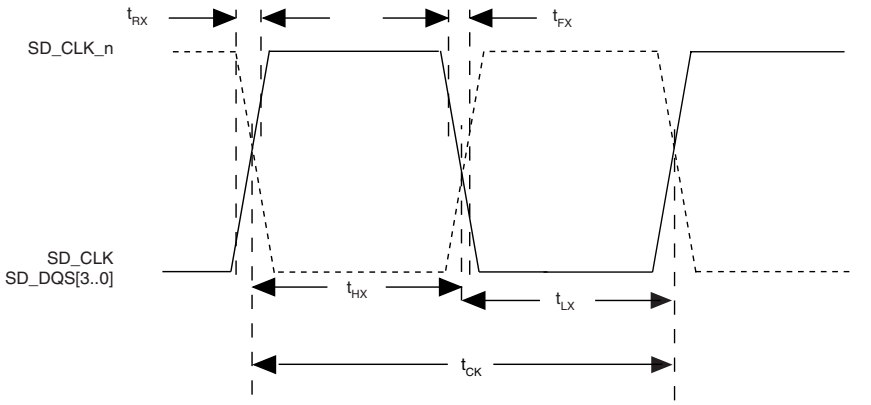


Table 76 lists the SDR and DDR clock timing parameters for the SD_CLK waveforms shown in Figure 54.

Table 76. SDRAM Parameters & Values: DDR Clock/ SD_DQS[31..0] Note (1) (Part 1 of 2)				
Symbol	**Parameter**	**Min**	**Max**	**Units**
t_{CK}	SD_CLK period	7.45	7.57	ns
t_{H1}	SD_CLK high-level width	3.61	3.66	ns
t_{L1}	SD_CLK low-level width	3.84	3.91	ns
t_{R1}	SD_CLK rise time	1.40	1.49	ns
t_{F1}	SD_CLK fall time	1.84	1.95	ns
t_{H2}	SD_DQS output high pulse width	3.54	3.66	ns
t_{L2}	SD_DQS output low pulse width	4.05	4.13	ns

Table 76. SDRAM Parameters & Values: DDR Clock/ SD_DQS[31..0] Note (1) (Part 2 of 2)				
Symbol	**Parameter**	**Min**	**Max**	**Units**
t_{R2}	SD_DQS rise time	0.99	1.25	ns
t_{F2}	SD_DQS fall time	1.02	1.26	ns

Note:
(1) All values are measured under typical conditions.

Figure 55 shows the SDR output delay timing requirements.

Figure 55. SDRAM Timing Diagram: SDR t_{OV} Output Delay

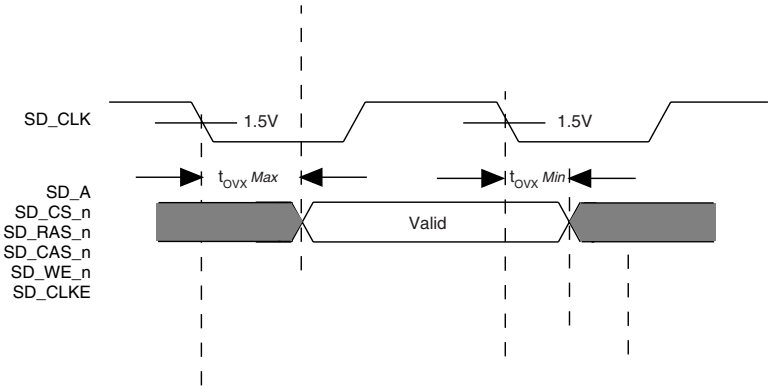


Table 77 lists the SDR output delay timing parameters in Figure 55.

Table 77. SDRAM Parameters & Values: SDR Output Delay Notes (1) (2)				
Symbol	**Parameter**	**Min**	**Max**	**Units**
t_{OV1}	SD_CLK to DQ[31..0] delay	0.23	0.46	ns
t_{LZ1}	SD_CLK to DQ[31..0] driven delay	0.31		ns
t_{HZ1}	SD_CLK to DQ[31..0] high-Z delay		0.66	ns
t_{OV2}	SD_CLK to A[14..0] delay	0.18	0.47	ns
t_{OV3}	SD_CLK to SD_CLKE delay		1.65	ns
t_{OV4}	SD_CLK to CS, RAS, CAS, WE delay		0.13	ns
t_{OV5}	SD_CLK to DQM[3..0] delay	0.26	0.44	ns
t_{OV7}	SD_CLK_n to SD_CLK delay		0.70	ns

Note:
(1) All delays are relative to the rising edge of SD_CLK.
(2) All values are measured under typical conditions.

Figure 56 shows the SDR input data setup and hold timing requirements.

Figure 56. SDRAM Timing Diagram: SDR Input Setup & Hold

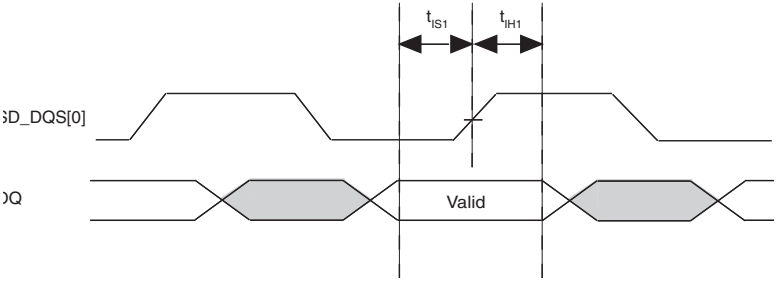


Table 78 lists the SDR input data setup and hold timing parameters shown in Figure 56.

Table 78. SDRAM Parameters & Values: SDR Input Setup & Hold Notes (1) (2)

Symbol	Parameter	Min	Max	Units
t_{IS1}	SD_DQ[31..0] input setup time	0.45		ns
t_{IH1}	SD_DQ[31..0] input hold time	0.32		ns

Note:
(1) All delays are relative to the rising edge of SD_DQS[0].
(2) All values are measured under typical conditions.

Figure 57 on page 191 shows the DDR output address and control timing requirements.

Figure 57. SDRAM Timing Diagram: DDR Output Address & Control

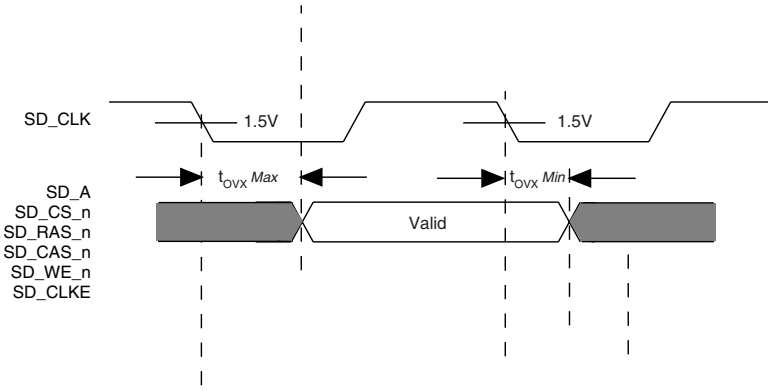


Table 79 lists the DDR output address and control timing parameters shown in Figure 57.

Table 79. SDRAM Parameters & Values: DDR Output Address & Control Note (1)

Symbol	Parameter	Min	Max	Units
t_{OV6}	DDR address & control (CS, RAS, CAS, WE) output delay	0.07	0.36	ns

Note:
(1) All values are measured under typical conditions.

Figure 58 shows the DDR input data setup and hold timing requirements.

Figure 58. SDRAM Timing Diagram: DDR t_{s1} Input Setup & Hold

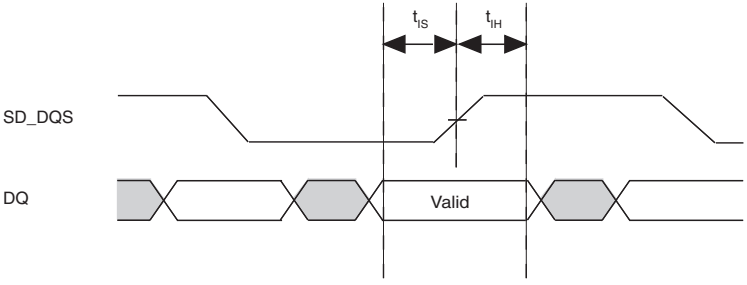


Table 80 lists the DDR input data setup and hold timing parameters shown in Figure 58.

Table 80. SDRAM Parameters & Values: DDR Input Setup and Hold Note (1)

Symbol	Parameter	Min	Max	Units
t_{IS}	SD_DQ[31:0] input setup time	–1.10		ns
t_{IH}	SD_DQ[31:0] input hold time	1.90		ns

Note:
(1) All values are measured under typical conditions.
(2) All times are relative to rising or falling edge of SD_DQS[3..0].

Figure 59 shows the DDR data output timing requirements.

Figure 59. SDRAM Timing Diagram: DDR Data Output Timings

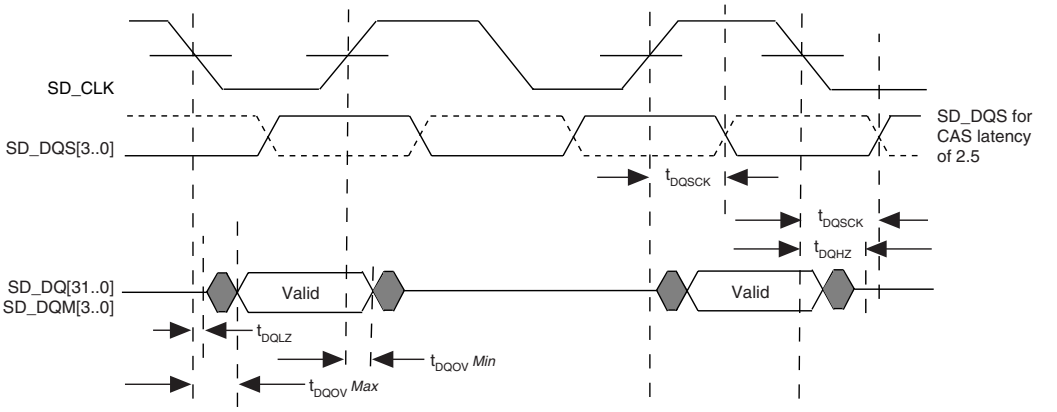


Table 81 lists the DDR data output timing parameters in Figure 59.

Table 81. SDRAM Parameters & Values: DDR Data Output Note (1)

Symbol	Parameter	Min	Max	Units
t_{DQSCK}	SD_CLK to SD_DQS[3..0] delay	1.60	1.68	ns
t_{DQOV}	SD_CLK to SD_DQM[3..0], SD_DQ[31..0] valid delay,	0.02	0.39	ns
t_{DQLZ}	SD_CLK to DQ[31..0] low-Z delay	0.02		ns
t_{DQHZ}	SD_CLK to DQ[3..0] high-Z delay		0.39	ns

Note:
(1) All values are measured under typical conditions.

Trace Port

Figure 60 on page 193 shows the trace port clock timing characteristics. See the *Embedded Trace Macrocell Specification* for more information.

Figure 60. Timing Diagram: TRACE_CLK

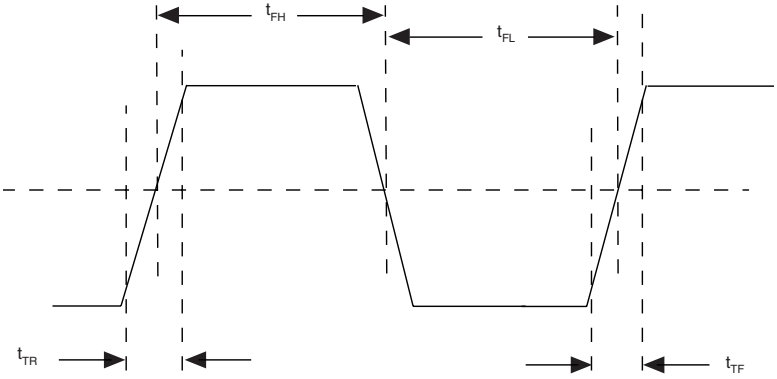


Table 82 lists the TRACE_CLK timing parameters shown in Figure 60.

Table 82. Trace Port Parameters & Values: TRACE_CLK

Symbol	Parameter	Min	Units
t_{FH}	TRACECLK high time	2	ns
t_{FL}	TRACECLK low time	2	ns
t_{TR}	TRACECLK rise time	3	ns
t_{TF}	TRACECLK fall time	3	ns

Figure 61 on page 194 shows the output timing requirements.

Figure 61. Timing Diagram: Trace Output

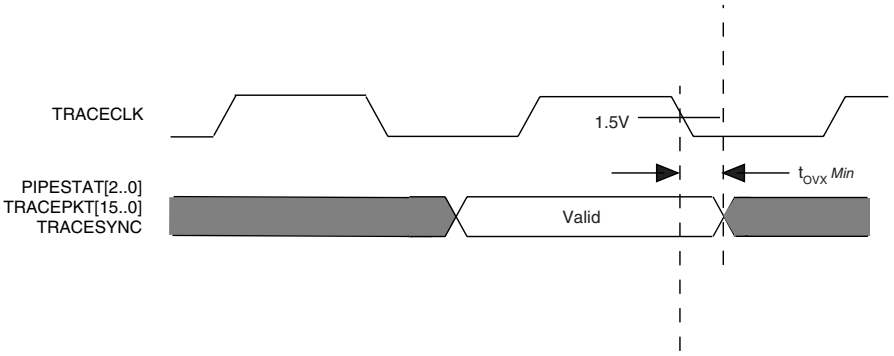


Table 83 lists the data output timing parameters shown in Figure 61.

Table 83. Trace Port Parameters & Values: Trace Output

Symbol	Parameter	Min	Units
t_{ovx}	PIPESTAT[2..0] TRACEPKT[15..0] TRACESYNC output valid delay	2	ns

UART

Figure 62 shows the timing requirements for the UART.

Figure 62. UART Timing Diagram

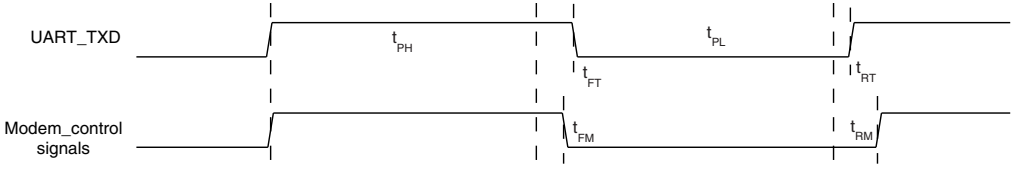


Table 84 shows the UART timing parameters and values.

Table 84. UART Parameters & Values

Symbol	Parameter	Baudrate=*4800*
		Min (ns)
t_{RT}	TXD rise time	2.5
t_{FT}	TXD fall time	2.5
t_{PH}	TXD pulse high	222
t_{PL}	TXD pulse low	221
t_{FM}	MODEM control fall time	2.5
t_{RM}	MODEM Control Rise Time	2.5

JTAG

Figure 63 shows the timing requirements for the JTAG signals.

Figure 63. JTAG Timing Diagram

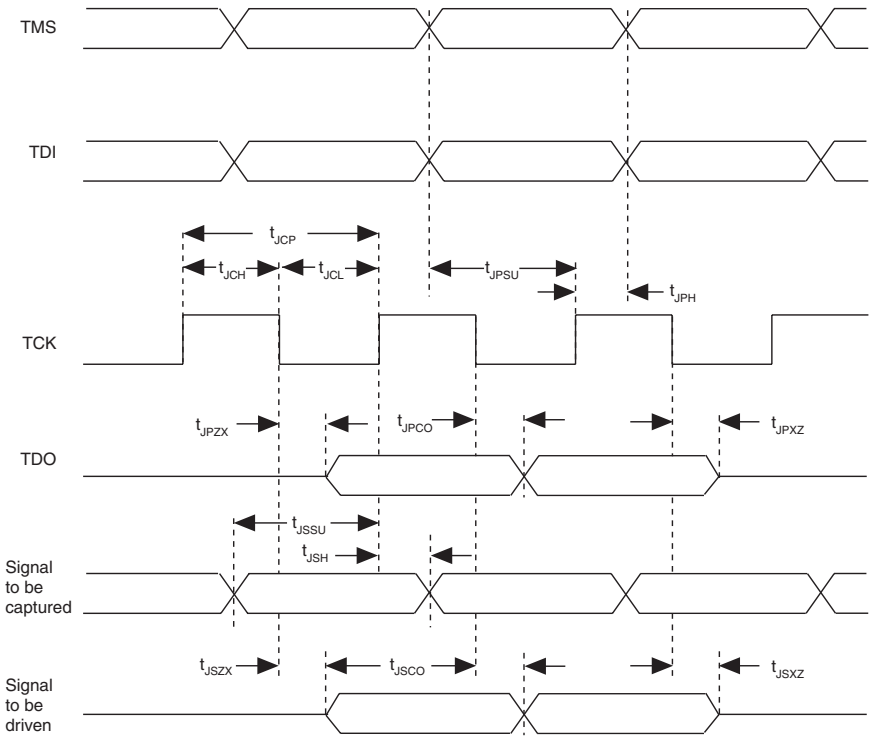


Table 85 lists the JTAG timing parameters and values.

Table 85. JTAG Timing Parameters & Values

Symbol	Parameter	Min	Max	Unit
t_{JCP}	TCK clock period	100		ns
t_{JCH}	TCK clock high time	50		ns
t_{JCL}	TCK clock low time	50		ns
t_{JPSU}	JTAG port setup time	20		ns
t_{JPH}	JTAG port hold time	45		ns
t_{JPCO}	JTAG port clock to output		25	ns
t_{JPZX}	JTAG port high impedance to valid output		25	ns
t_{JPXZ}	JTAG port valid output to high impedance		25	ns
t_{JSSU}	Capture register setup time	20		ns
t_{JSH}	Capture register hold time	45		ns
t_{JSCO}	Update register clock to output		35	ns
t_{JSZX}	Update register high impedance to valid output		35	ns
t_{JSXZ}	Update register valid output to high impedance		35	ns

For more information, see the following documents:

- *Application Note 39 (IEEE Std. 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices)*
- *Jam Programming & Test Language Specification*

Dual-Port SRAM

Figure 64 on page 197 shows the timing requirements of the dual-port SRAM at the stripe interface.

Figure 64. Dual-Port SRAM Timing Diagram

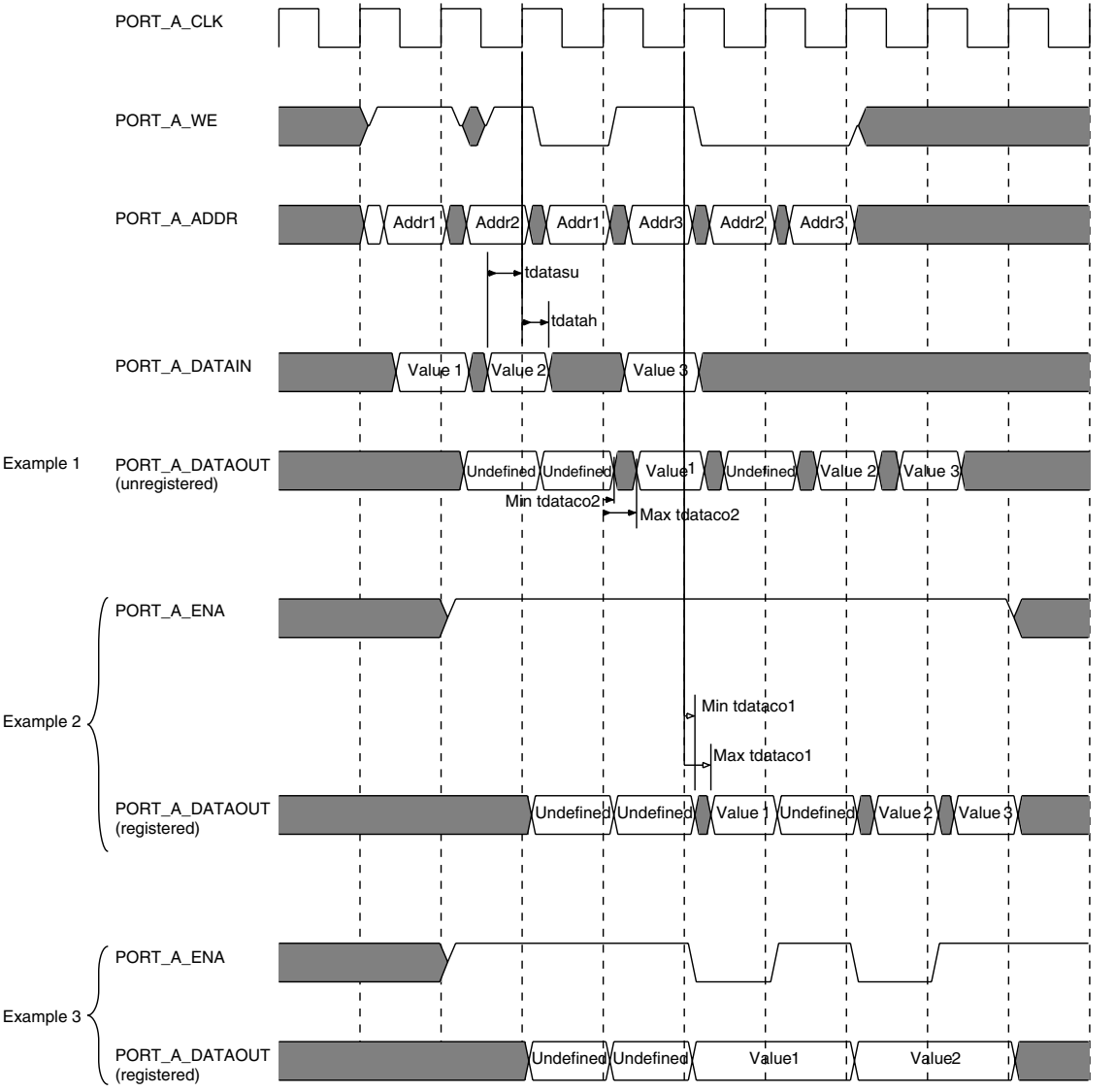


Table 86 on page 198 shows the timing parameters for the EPXA10 dual-port SRAM signals.

Table 86. EPXA10 Dual-Port SRAM Timing Parameters & Values (ns)

Parameter	–1		–2		–3	
	Min	Max	Min	Max	Min	Max
PORT_A_WE set-up time	2.59		3.12		3.89	
PORT_A_ENA set-up time (registered mode)	2.50		3.01		3.76	
PORT_A_ADDR set-up time	2.83		3.41		4.25	
PORT_A_DATAIN set-up time	2.81		3.38		4.22	
PORT_A_WE hold time	0		0		0	
PORT_A_ENA hold time (registered mode)	0		0		0	
PORT_A_ADDR hold time	0		0		0	
PORT_A_DATAIN hold time	0		0		0	
PORT_A_DATAOUT clock-to-output delay (registered mode)	0	4.85	0	5.85	0	7.30
PORT_A_DATAOUT clock-to-output delay (unregistered mode)	0	8.19	0	9.87	0	12.32

Table 87 shows the timing parameters for the EPXA4 dual-port SRAM signals.

Table 87. EPXA4 Dual-Port SRAM Timing Parameters & Values (ns)

Parameter	–1		–2		–3	
	Min	Max	Min	Max	Min	Max
PORT_A_WE set-up time	1.86		2.23		2.96	
PORT_A_ENA set-up time (registered mode)	1.08		1.30		1.72	
PORT_A_ADDR set-up time	1.95		2.35		3.12	
PORT_A_DATAIN set-up time	2.04		2.45		3.25	
PORT_A_WE hold time	0.74		0.89		1.18	
PORT_A_ENA hold time (registered mode)	0.74		0.89		1.18	
PORT_A_ADDR hold time	0.74		0.89		1.18	
PORT_A_DATAIN hold time	0.74		0.89		1.18	
PORT_A_DATAOUT clock-to-output delay (registered mode)	0.74	3.61	0.89	4.34	1.18	5.76
PORT_A_DATAOUT clock-to-output delay (unregistered mode)	0.74	7.72	0.89	9.28	1.18	12.33

Table 88 on page 199 shows the timing parameters for the EPXA1 dual-port SRAM signals.

Table 88. EPXA1 Dual-Port SRAM Timing Parameters & Values (ns)

Parameter	–1		–2		–3	
	Min	Max	Min	Max	Min	Max
PORT_A_WE set-up time	2.01		2.42		3.22	
PORT_A_ENA set-up time (registered mode)	1.83		2.20		2.92	
PORT_A_ADDR set-up time	2.27		2.73		3.63	
PORT_A_DATAIN set-up time	2.27		2.73		3.63	
PORT_A_WE hold time	0		0		0	
PORT_A_ENA hold time (registered mode)	0		0		0	
PORT_A_ADDR hold time	0		0		0	
PORT_A_DATAIN hold time	0		0		0	
PORT_A_DATAOUT clock-to-output delay (registered mode)	0	4.01	0	4.46	0	5.93
PORT_A_DATAOUT clock-to-output delay (unregistered mode)	0	6.92	0	8.32	0	11.05

Device Pinouts

See the Altera web site (http://www.altera.com) for pin-out information

Packaging

A variety of packaging options are available for the Excalibur embedded processor PLDs. Table 89 shows the device/package combinations that are available.

Table 89. Device Combinations Available

Device	Package		
	1020-Pin FBGA	672-Pin FBGA	484-Pin FBGA
EPXA10	✓		
EPXA4	✓	✓	
EPXA1		✓	✓

Embedded Processor Register Summary

The tables in this section document the embedded processor register map. Embedded processor registers can only be accessed using word (32-bit) accesses; 8-bit or 16-bit accesses generate a bus error. Only registers containing packed bytes are susceptible to endian considerations.

Table 90 on page 200 shows register offsets, size and usage.

Table 90. Register Size and Usage			
Offset	**Size (Bytes)**	**Name**	**Bus**
000H	64	Reset and mode control	2
040H	64	I/O control	2
080H	128	Memory map	2
100H	64	Bridge control	2
140H	64	PLD Configuration	2
200H	128	Timer	2
280H	128	UART	2
300H	128	Clock control	2
380H	128	External bus interface	2
400H	128	SDRAM interface	2
800H	512	AHB1-2 bridge control	1
A00H	512	Watchdog	1
C00H	512	Interrupt controller	1

Reset and Mode Control Module

Table 91 shows the registers used in the reset and mode control module.

Table 91. Reset and Mode Control Registers				
Offset	**Name**	**Access**	**Bus**	**Page**
000H	BOOT_CR	R/C	2	162
004H	RESET_SR	R/C	2	162
008H	IDCODE	R	2	162
020H	SRAM0_SR	R	2	50
024H	SRAM1_SR	R	2	50
030H	DPSRAM0_SR	R	2	66
038H	DPSRAM1_SR	R	2	67

I/O Control Module

Table 92 shows the registers used in the I/O control module.

Table 92. I/O Control Registers

Offset	Name	Access	Bus	Page
040H	IOCR_SDRAM	R/W	2	168
044H	IOCR_EBI	R/W	2	168
048H	IOCR_UART	R/W	2	168
04CH	IOCR_TRACE	R/W	2	169

Memory Map Module

Table 93 shows the registers used in the memory map module.

Table 93. Memory Map Registers

Offset	Name	Access	Bus
080H	MMAP_REGISTERS	R/W	2
090H	MMAP_SRAM0	R/W	2
094H	MMAP_SRAM1	R/W	2
0A0H	MMAP_DPSRAM0	R/W	2
0A4H	MMAP_DPSRAM1	R/W	2
0B0H	MMAP_SDRAM0	R/W	2
0B4H	MMAP_SDRAM1	R/W	2
0C0H	MMAP_EBI0	R/W	2
0C4H	MMAP_EBI1	R/W	2
0C8H	MMAP_EBI2	R/W	2
0CCH	MMAP_EBI3	R/W	2
0D0H	MMAP_PLD0	R/W	2
0D4H	MMAP_PLD1	R/W	2
0D8H	MMAP_PLD2	R/W	2
0DCH	MMAP_PLD3	R/W	2

Bridge-Control Module

Table 94 shows the bridge-control and status registers that are located in the bridge-control module.

Table 94. Bridge-Control and Status Registers

Offset	Name	Access	Bus	Page
100H	AHB12B_CR	R/W	2	44
110H	PLDSB_CR	R/W	2	45
118H	PLDSB_SR	R/C	2	45
114H	PLDSB_ADDRSR	R	2	46
120H	PLDMB_CR	R/W	2	46
—	PLDMB_SR	R/C	PLD	45
—	PLDMB_ADDRSR	R	PLD	46

AHB1-2 Bridge-Control Module

Table 95 shows the AHB1-2 bridge status registers, which are located in the AHB1-2 bridge-control module.

Table 95. AHB1-2 Bridge Status Registers

Offset	Name	Access	Bus	Page
800H	AHB12B_SR	R/C	1	44
804H	AHB12B_ADDRSR	R	1	45

PLD Configuration

Table 96 shows the registers used for PLD configuration.

Table 96. PLD Configuration Registers

Offset	Name	Access	Bus	Page
140H	CONFIG_CONTROL	R/W	2	153
144H	CONFIG_CLOCK	R/W	2	154
148H	CONFIG_DATA	W	2	154
14CH	CONFIG_UNLOCK	W	2	155

Timer

Table 97 shows the registers used in the timer module.

Table 97. Timer Registers

Offset	Name	Access	Bus	Page
200H	TIMER0_CR	W	2	117
	TIMER0_SR	R	2	118
210H	TIMER0_PRE	R/W	2	118
220H	TIMER0_LIMIT	R/W	2	118
230H	TIMER0_READ	R	2	119
240H	TIMER1_CR	W	2	117
	TIMER1_SR	R		118
250H	TIMER1_PRE	R/W	2	118
260H	TIMER1_LIMIT	R/W	2	119
270H	TIMER1_READ	R	2	119

UART

Table 98 shows the registers used to control the UART.

Table 98. UART Registers

Offset	Name	Access	Bus	Page
280H	UART_RSR	R*	2	107
284H	UART_RDS	R	2	107
288H	UART_RD	R*	2	108
28CH	UART_TSR	R*	2	108
290H	UART_TD	W	2	108
294H	UART_FCR	R/W	2	109
298H	UART_IES	R/S	2	110
29CH	UART_IEC	R/C	2	110
2A0H	UART_ISR	R	2	110
2A4H	UART_IID	R	2	111
2A8H	UART_MC	R/W	2	111
2ACH	UART_MCR	R/W	2	112
2B0H	UART_MSR	R*	2	113
2B4H	UART_DIV_LO	R/W	2	113
2B8H	UART_DIV_HI	R/W	2	114

Clock Control

Table 99 shows the registers used in the clock control module.

Table 99. Clock Control Registers

Offset	Name	Access	Bus	Page
300H	CLK_PLL1_NCNT	R/W	2	142
304H	CLK_PLL1_MCNT	R/W	2	142
308H	CLK_PLL1_KCNT	R/W	2	143
30CH	CLK_PLL1_CTRL	R/W	2	143
310H	CLK_PLL2_NCNT	R/W	2	144
314H	CLK_PLL2_MCNT	R/W	2	144
318H	CLK_PLL2_KCNT	R/W	2	144
31CH	CLK_PLL2_CTRL	R/W	2	145
320H	CLK_DERIVE	R/W	2	145
324H	CLK_STATUS	R/C	2	146
328H	CLK_AHB1_COUNT	R	2	147

Expansion Bus Interface

Table 100 shows the registers used in the expansion bus interface module.

Table 100. Expansion Bus Interface Registers

Offset	Name	Access	Bus	Page
380H	EBI_CR	W	2	96
	EBI_SR	R		96
390H	EBI_BLOCK0	R/W	2	98
394H	EBI_BLOCK1	R/W	2	98
398H	EBI_BLOCK2	R/W	2	98
39CH	EBI_BLOCK3	R/W	2	99
3A0H	EBI_INT_SR	R/C	2	97
3A4H	EBI_INT_ADDRSR	R	2	98

SDRAM Interface

Table 101 shows the registers used in the SDRAM interface module.

Table 101. SDRAM Interface Registers

Offset	Name	Access	Bus	Page
400H	SDRAM_TIMING1	R/W	1, 2	75
404H	SDRAM_TIMING2	R/W	1, 2	75
408H	SDRAM_CONFIG	R/W	1, 2	75
40CH	SDRAM_REFRESH	R/W	1, 2	76
410H	SDRAM_ADDR	R/W	1, 2	76
41CH	SDRAM_INIT	R/W	1, 2	76
420H	SDRAM_MODE0	R/W	1, 2	76
424H	SDRAM_MODE1	R/W	1, 2	77
07CH	SDRAM_WIDTH	R/W	2	77

Watchdog Timer

Table 102 shows the registers used in the watchdog timer module.

Table 102. Watchdog Timer Registers

Offset	Name	Access	Bus	Page
A00H	WDOG_CR	R/W	1	123
A04H	WDOG_COUNT	R	1	123
A08H	WDOG_RELOAD	W	1	123

Interrupt Controller

Table 103 shows the registers used in the interrupt controller module.

Table 103. Interrupt Controller Registers

Offset	Name	Access	Bus	Page
C00H	INT_MASK_SET	R/S	1	131
C04H	INT_MASK_CLEAR	R/C	1	131
C08H	INT_SOURCE_STATUS	R	1	132
C0CH	INT_REQUEST_STATUS	R	1	132
C10H	INT_ID	R	1	132
C14H	INT_PLD_PRIORITY	R	1	133
C18H	INT_MODE	R/W	1	133
C80H	INT_PRIORITY_PLD0	R/W	1	134
C84H	INT_PRIORITY_PLD1	R/W	1	134
C88H	INT_PRIORITY_PLD2	R/W	1	134
C8CH	INT_PRIORITY_PLD3	R/W	1	134
C90H	INT_PRIORITY_PLD4	R/W	1	134
C94H	INT_PRIORITY_PLD5	R/W	1	134
C98H	INT_PRIORITY_EXTPIN	R/W	1	134
C9CH	INT_PRIORITY_UART	R/W	1	134
CA0H	INT_PRIORITY_TIMER0	R/W	1	135
CA4H	INT_PRIORITY_TIMER1	R/W	1	135
CA8H	INT_PRIORITY_PLL	R/W	1	135
CACH	INT_PRIORITY_EBI	R/W	1	135
CB0H	INT_PRIORITY_STRIPE-PLD	R/W	1	135
CB4H	INT_PRIORITY_AHB1-2	R/W	1	135
CB8H	INT_PRIORITY_TX	R/W	1	135
CBCH	INT_PRIORITY_RX	R/W	1	135
CC0H	INT_PRIORITY_FASTCOMMS	R/W	1	135

Comprehensive Pin & Signal Listing

Table 104 lists the external pins in the Excalibur embedded processor PLD.

On EPXA10 devices, all of the embedded stripe UART, EBI, SDRAM controller, and trace port pins (except for the SD_CLKE pin) are available as user I/O, if they are not used for their stripe function.

On EPXA4 devices, all of the embedded stripe pins are available as user I/O except for EBI_A[24..22], SD_DQ[31..24], SD_DQS[3], SD_DDR_VS[2] and all trace port pins.

All of the embedded stripe pins on EPXA1 devices are dedicated.

Table 104. External Pins in the Excalibur Embedded Processor PLD (Part 1 of 5)

Signal	Source	Description
BOOT_FLASH	Input	Boot mode select pin. When high, the embedded processor boots from flash memory on the EBI; when low, boot-from-serial mode is used
CLK_REF	Input	Embedded stripe clock input pin to the embedded stripe PLLs
DEBUG_EN	Input	When high, allows the embedded processor to enter debug mode.
JSELECT	Input	JTAG mode select pin. When high, the device is in parallel JTAG mode. When low, the device is in serial JTAG mode.
nPOR	Input	Active-low power-on reset pin
nRESET	Input/Output	Active-low bidirectional reset pin
MSEL0	Input	Configuration mode select pin
MSEL1	Input	Configuration mode select pin
CLK1p	Input	Dedicated global clock input pin and input to PLD PLL1
CLK2p	Input	Dedicated global clock input pin and input to PLD PLL2
CLK3p	Input	Dedicated global clock input pin and input to PLD PLL3
CLK4p	Input	Dedicated global clock input pin and input to PLD PLL4
CLK1n	Input	Complementary signal for LVDS pair on dedicated input and outputs that can be configured for LVDS. If not used for the LVDS pair, it is a user I/O
CLK2n	Input	Complementary signal for LVDS pair on dedicated input and outputs that can be configured for LVDS. If not used for the LVDS pair, it is a user I/O
CLK3n	Input	Complementary signal for LVDS pair on dedicated input and outputs that can be configured for LVDS. If not used for the LVDS pair, it is a user I/O

Table 104. External Pins in the Excalibur Embedded Processor PLD (Part 2 of 5)

Signal	Source	Description
CLK4n	Input	Complementary signal for LVDS pair on dedicated input and outputs that can be configured for LVDS. If not used for the LVDS pair, it is a user I/O
CLKLK_ENA	Input	Active-high enable pin for all PLL circuits in the device. When de-asserted, all PLLs are reset to their default, unlocked state and stop clocking. When re-asserted, the PLLs lock again and resume clocking. If this pin function is not needed, the pin should be connected to VCCINT
CLKLK_FB1p	Input	Dedicated input that allows external feedback to PLD PLL1
CLKLK_FB2p	Input	Dedicated input that allows external feedback to PLD PLL2
CLKLK_FB1n	Output	Complementary signal for LVDS pair on dedicated input and outputs that can be configured for LVDS. If not used for the LVDS pair, it is a user I/O.
CLKLK_FB2n	Output	Complementary signal for LVDS pair on dedicated input and outputs that can be configured for LVDS. If not used for the LVDS pair, it is a user I/O.
CLKLK_OUT1p	Output	Dedicated external output for PLD PLL1
CLKLK_OUT2p	Output	Dedicated external output for PLD PLL2
CLKLK_OUT1n	Output	Negative terminal output for differential output from PLD PLL1. If not used for the LVDS pair, it is a user I/O.
CLKLK_OUT2n	Output	Negative terminal output for differential output from PLD PLL2. If not used for the LVDS pair, it is a user I/O.
LOCK[1..4]	Output	PLL lock status pins. When the PLL circuitry is locked to the incoming clock and generates an internal clock, LOCK is driven high. If not used for the PLL function, the pins are available as user I/Os
nCONFIG	Input	Active-low input that initiates device reconfiguration
CONF_DONE	Input/Output	Active-high bidirectional pin that indicates completion of device configuration
INIT_DONE	Output	Active-high signal that indicates the completion of initialization following configuration
DATA0	Input	Configuration data input
DCLK	Input	Configuration clock input
FAST[1..3]	Input	Input pins that drive dedicated fast input lines
CLKUSR	Input	External clock input for configuration
CS	Input	Active-high configuration chip-select for parallel programming modes. Either CS or nCS can be used for selecting the device. If an active-high enable is desired, CS is used; and if an active-low enable is desired, nCS is used. In other programming modes this pin can be used as user I/O

Table 104. External Pins in the Excalibur Embedded Processor PLD (Part 3 of 5)

Signal	Source	Description
nCS	Input	Active-low configuration chip-select for parallel programming modes. Either CS or nCS can be used for selecting the device. If an active-high enable is desired, CS is used; and if an active-low enable is desired, nCS is used. In other programming modes this pin can be used as user I/O
nCE	Input	Active-low chip enable used in multi-device configuration schemes
nCEO	Output	Active-low chip enable output used in multi-device configuration schemes
nSTATUS	Input/Output	Configuration pin that indicates that the device is being initialized or has encountered an error during initialization
nRS	Input	Active low read strobe for parallel configuration modes
nWS	Input	Active low read strobe for parallel configuration modes
RDYnBSY	Output	Status pin for parallel configuration modes
DATA[1..7]	Input	Configuration data input for parallel modes. In other programming modes this pin can be used as user I/O
DEV_CLR	Input	Active low device-wide clear input. Can be used as user I/O if not used as a device-wide clear
DEV_OE	Input	Device-wide output enable input. Can be used as user I/O if not used as a device-wide output-enable
LVDSDESKEW	Input	LVDS deskew pin. If not used for LVDS, this pin is available as a user I/O
LVDSRX[1..16]p	Input	LVDS receiver positive data pin. If not used for LVDS, this pin is available as a user I/O
LVDSRX[1..16]n	Input	LVDS receiver negative data pin. If not used for LVDS, this pin is available as a user I/O
LVDSRXINCLK1p	Input	LVDS receiver input clock positive pin. If not used for LVDS, this pin is available as a user I/O
LVDSRXINCLK1n	Input	LVDS receiver input clock negative pin. If not used for LVDS, this pin is available as a user I/O
LVDSTX[1..16]p	Output	LVDS transmitter positive data pin. If not used for LVDS, this pin is available as a user I/O
LVDSTX[1..16]n	Output	LVDS transmitter negative data pin. If not used for LVDS, this pin is available as a user I/O
LVDSTXINCLK1p	Input	LVDS transmitter input clock positive pin. If not used for LVDS, this pin is available as a user I/O
LVDSTXINCLK1n	Input	LVDS transmitter input clock negative pin. If not used for LVDS, this pin is available as a user I/O

Table 104. External Pins in the Excalibur Embedded Processor PLD (Part 4 of 5)

Signal	Source	Description
LVDSTXOUTCLK1p	Output	LVDS transmitter output clock positive pin. If not used for LVDS, this pin is available as a user I/O
LVDSTXOUTCLK1n	Output	LVDS transmitter output clock negative pin. If not used for LVDS, this pin is available as a user I/O
PIPESTAT[2..0] *(1) (3)*	Output	Trace port pipeline status pins
TRACEPKT[15..0] *(1) (3)*	Output	Trace port packet output pins
TRACECLK *(1) (3)*	Output	Trace port clock input pin
TRACESYNC *(1) (3)*	Output	Trace port signal indicating start of a branch sequence on packet port
INT_EXTPIN_N *(2)*	Input	Active-low external interrupt input pin to the embedded interrupt controller
EBI_CLK *(2)*	Output	EBI clock pin
EBI_A[24..0] *(4)*	Output	EBI address bus pins
EBI_DQ[15..0] *(2)*	Input/Output	Bidirectional EBI data bus pins
EBI_BE[1..0] *(2)*	Output	EBI byte-enable pins. EBI_BE0 is used for EBI_DQ[7..0]; EBI_BE1 is used for EBI_DQ[15..8]
EBI_CS[3..0] *(2)*	Output	EBI chip-select pins. Chip selects correspond to memory map blocks EBI0, EBI1, EBI2, and EBI3. Programmable polarity; defaults to active-low CS_n
EBI_WE_n *(2)*	Output	EBI write-enable pin
EBI_OE_n *(2)*	Output	EBI output-enable pin
EBI_ACK *(2)*	Input	EBI acknowledge pin
SD_CLK *(2)*	Output	SDRAM clock pin (1)
SD_CLK_n *(2)*	Output	SDRAM CLK_n pin (1)
SD_CLKE *(2)*	Output	SDRAM CLK_E clock-enable pin (dedicated pin)
SD_WE_n *(2)*	Output	SDRAM active-low write enable pin (1)
SD_CAS_n *(2)*	Output	SDRAM CAS_n pin (1)
SD_RAS_n *(2)*	Output	SDRAM RAS_n pin (1)
SD_CS_n[1..0] *(2)*	Output	SDRAM chip selects CS_n pins (1)
SD_A[14..0] *(2)*	Output	SDRAM address bus (1): SD_A[14..13]—bank address(2)
SD_DQM[3..0] *(2)*	Output	SDRAM DQM data byte mask pins (1)
SD_DQ[31..0] *(5)*	Input/Output	SDRAM bidirectional data bus pins (1)
SD_DQS[3..0] *(6)*	Input/Output	SDRAM bidirectional data strobe pins (1)
UART_RXD *(2)*	Input	Serial data input signal to stripe UART
UART_DSR_n *(2)*	Input	Data set ready, active-low signal. When active, indicates that the peer device is ready to establish the communications link with the UART. (When acting as a modem, UART_DSR_n is used as a DTR input)

Table 104. External Pins in the Excalibur Embedded Processor PLD *(Part 5 of 5)*

Signal	Source	Description
UART_CTS_n *(2)*	Input	Clear-to-send, active-low signal. When active, indicates that the peer device can accept characters
UART_DCD_n *(2)*	Input/ Output	Data carrier detect, active-low signal. When active, indicates that the data carrier is being detected by the modem. This pin is an input when OE in UART_MC is 0, and output when OE in UART_MC is 1
UART_RI_n *(2)*	Input/ Output	Ring indicator, active-low signal. When active, indicates that a telephone ringing signal is being received by the modem. This pin is an input when OE in UART_MC is 0, and output when OE in UART_MC is 1
UART_TXD *(2)*	Output	Serial data output signal to the communications link. On reset, UART_TXD is set high
UART_RTS_n *(2)*	Output	Request-to-send, active-low signal. When active, informs the peer device that the UART is ready to receive data
UART_DTR_n *(2)*	Output	Data terminal ready, active-low signal. When active, indicates that the UART is ready to establish a communications link
TCK	Input	JTAG TAP controller test clock pin
TDI	Input	JTAG TAP controller test data input pin
TDO	Output	JTAG TAP controller test data output pin
TMS	Input	JTAG TAP controller test mode select pin
TRST	Input	JTAG TAP controller test reset input pin
PROC_TCK	Input	Processor JTAG TAP controller test clock pin
PROC_TDO	Input	Processor JTAG TAP controller test data input pin
PROC_TDO	Output	Processor JTAG TAP controller test data output pin
PROC_TDI	Input	Processor JTAG TAP controller test mode select pin.
PROC_TRST	Input	Active-low processor JTAG TAP controller test reset input pin.
SD_DDR_VS[2..0] *(7)*	Input	SDRAM voltage reference inputs

Notes:
(1) EPXA1 devices do not contain an embedded trace module, so this pin is not available on EPXA1 devices.
(2) If not used for its stripe function, this pin is available as user I/O on EPXA4 and EPXA10 devices.
(3) If not used for its stripe function, this pin is available as user I/O on EPXA10 devices only.
(4) All EBI_A pins are available as user I/O on EPXA10 devices. On EPXA4 devices, only EBI_A[21..0] are available as user I/O.
(5) All SD_DQ pins are available as user I/O on EPXA10 devices. On EPXA4 devices, only SD_DQ[23..0] are available as user I/O.
(6) All SD_DQS pins are available as user I/O on EPXA10 devices. On EPXA4 devices, only SD_DQS[2..1] are available as user I/O.
(7) All SD_DDR_VS pins are available as user I/O on EPXA10 devices. On EPXA4 devices, only SD_DDR_VS[1..0] are available as user I/O.

Table 105 lists the remaining signals in the Excalibur devices.

Table 105. Signals (Part 1 of 3)

Signal	Source	Description
DEBUG_RQ	PLD	Forces the embedded processor into debug mode
DEBUG_EXT0, DEBUG_EXT1	PLD	These inputs are matched by the breakpoint/watchpoint unit in the embedded processor, and matches are reported on the DEBUG_RNGx signals
DEBUG_ACK	Stripe	Indicates when the embedded processor is stopped in debug mode. This can be used to stop devices within the PLD when the embedded processor hits a breakpoint
DEBUG_RNG0, DEBUG_RNG1	Stripe	Indicates when the breakpoint/watchpoint unit has found a match (whether enabled or not). The signal is valid for at least one PLD clock cycle
DEBUG_EXTIN[3..0]	PLD	Trace port input signals
DEBUG_EXTOUT[3..0]	Stripe	Trace port output signals
MASTER_HCLK	PLD	Stripe-to-PLD bridge slave-port clock that times all bus transfers.Signal timings are related to its rising edge clock. This signal is invertible
MASTER_HADDR[31..0]	Stripe	Stripe-to-PLD 32-bit system address bus
MASTER_HTRANS[1..0]	Stripe	Stripe-to-PLD bridge transfer type
MASTER_HWRITE	Stripe	When high, indicates a write transfer on the stripe-to-PLD bridge; when low, a read transfer
MASTER_HSIZE[1..0]	Stripe	Indicates the size of transfer on the stripe-to-PLD bridge
MASTER_HBURST[2..0]	Stripe	Indicates whether the stripe-to-PLD bridge transfer forms part of a burst
MASTER_HWDATA[31..0]	Stripe	Used to transfer data from the master to the bus slaves during writes across the stripe-to-PLD bridge
MASTER_HREADY	PLD	When high, indicates that a stripe-to-PLD bridge transfer has finished
MASTER_HRESP[1..0]	PLD	Slave response that provides additional information on the status of a transfer across the stripe-to-PLD bridge
MASTER_HRDATA[31..0]	PLD	Used to transfer data from bus slaves to the master during reads across the stripe-to-PLD bridge
MASTER_HLOCK	Stripe	When high, indicates that the master requires locked access to the bus
MASTER_HBUSREQ	Stripe	Bus request signal, from the master to the arbiter
MASTER_HGRANT	PLD	Bus grant signal that, in conjunction with MASTER_HREADY, indicates that the bus master has been granted the bus
SLAVE_HCLK	PLD	Stripe-to-PLD bridge slave-port clock that times all bus transfers.Signal timings are related to its rising edge clock. This signal is invertible
SLAVE_HADDR[31..0]	PLD	PLD-to-stripe bridge 32-bit system address bus
SLAVE_HTRANS[1..0]	PLD	PLD-to-stripe bridge transfer type
SLAVE_HWRITE	PLD	When high, indicates a write transfer on the PLD-to-stripe bridge; when low, a read transfer

Table 105. Signals (Part 2 of 3)

Signal	Source	Description
SLAVE_HSIZE[1..0]	PLD	Indicates the size of transfer on the PLD-to-stripe bridge
SLAVE_HBURST[2..0]	PLD	Indicates whether the transfer forms part of a burst
SLAVE_HWDATA[31..0]	PLD	Used to transfer data from the master to the bus slaves during writes across the PLD-to-stripe bridge
SLAVE_HREADYI	PLD	When high, indicates that a transfer has finished on the bus. Slaves on the bus need SLAVE_HREADY as both an input and output signal
SLAVE_HREADYO	Stripe	When high, indicates that a transfer has finished on the bus. Slaves on the bus need SLAVE_HREADY as both an input and output signal
SLAVE_HRESP[1..0]	Stripe	Master response that provides additional information on the status of a transfer across the PLD-to-stripe bridge
SLAVE_HRDATA[31..0]	Stripe	Used to transfer data from bus slaves to the master during reads across the PLD-to-stripe bridge
SLAVE_HMASTLOCK	PLD	When high, indicates that the master requires locked access to the bus
SLAVE_BUSERRINT	Stripe	Interrupt signal signifying a bus error
SLAVE_HSELREG	PLD	PLD-to-stripe bridge register selection signal
SLAVE_HSEL	PLD	PLD-to-stripe bridge interface chip-selection signal
CLK_AHB1	PLL1	Embedded processor AHB1 bus clock signal
CLK_AHB2	PLL1	Embedded stripe peripheral AHB2 bus clock signal
CLK_SDRAMx1	PLL2	Provides internal clock for SDRAM controller and external clock for SDRAM device
CLK_SDRAMx2	PLL2	Provides internal clock for SDRAM controller
DP0_2_PORTACLK	PLD	DPRAM0/2 port A clock in the PLD
DP0_PORTBCLK	PLD	DPRAM0 port B clock in the PLD
DP1_3_PORTACLK	PLD	DPRAM1/3 port A clock in the PLD
DP1_PORTBCLK	PLD	DPRAM1 port B clock in the PLD
DP0_PORTAADDR[n..0]	PLD	DPRAM0 port A address bus; registered
DP0_PORTADATAIN[n..0]	PLD	DPRAM0 port A data input bus
DP0_PORTADATAOUT[n..0]	Stripe	DPRAM0 port A data output bus
DP0_PORTAWE	PLD	DPRAM0 port A write enable: 1 = write 0 = read
DP0_PORTAENA	PLD	DPRAM0 port A register enable
DP0_PORTBADDR[n..0]	PLD	DPRAM0 port B address bus; registered
DP0_PORTBDATAIN[n..0]	PLD	DPRAM0 port B data input bus
DP0_PORTBDATAOUT[n..0]	Stripe	DPRAM0 port B data output bus
DP0_PORTBWE	PLD	DPRAM0 port B write enable: 1 = write 0 = read
DP0_PORTBENA	PLD	DPRAM0 port B register enable
DP1_PORTAADDR[n..0]	PLD	DPRAM1 port A address bus; registered
DP1_PORTADATAIN[n..0]	PLD	DPRAM1 port A data input bus

Table 105. Signals (Part 3 of 3)

Signal	Source	Description
DP1_PORTADATAOUT[n..0]	Stripe	DPRAM1 port A data output bus
DP1_PORTAWE	PLD	DPRAM1 port A write enable: 1 = write 0 = read
DP1_PORTAENA	PLD	DPRAM1 port A register enable
DP1_PORTBADDR[n..0]	PLD	DPRAM1 port B address bus; registered
DP1_PORTBDATAIN[n..0]	PLD	DPRAM1 port B data input bus
DP1_PORTBDATAOUT[n..0]	Stripe	DPRAM1 port B data output bus
DP1_PORTBWE	PLD	DPRAM1 port B write enable:
DP1_PORTBENA	PLD	DPRAM1 port B register enable
DP2_PORTAADDR[n..0]	PLD	DPRAM2 port A address bus; registered
DP2_PORTADATAIN[n..0]	PLD	DPRAM2 port A data input bus
DP2_PORTADATAOUT[n..0]	Stripe	DPRAM2 port A data output bus
DP2_PORTAWE	PLD	DPRAM2 port A write enable: 1 = write 0 = read
DP2_PORTAENA	PLD	DPRAM2 port A register enable
DP3_PORTAADDR[n..0]	PLD	DPRAM3 port A address bus; registered
DP3_PORTADATAIN[n..0]	PLD	DPRAM3 port A data input bus
DP3_PORTADATAOUT[n..0]	Stripe	DPRAM3 port A data output bus
DP3_PORTAWE	PLD	DPRAM3 port A write enable: 1 = write 0 = read
DP3_PORTAENA	PLD	DPRAM3 port A register enable

Abbreviations

The *Excalibur Devices Hardware Reference Manual* uses the following abbreviations and acronyms.

Table 106. Common Acronyms *(Part 1 of 2)*

Acronym	Name
AHB	advanced high-performance bus
AMBA	advanced micro-controller bus architecture
APEX	advanced programmable embedded matrix
ARM	advanced RISC machine
ASIC	application-specific integrated circuit
BDM	background debugging mode
BGA	ball-grid array
CMOS	complementary metal-oxide semiconductor
CPU	central processing unit
CRC	cyclic redundancy check
DDR	double-data rate
DDR RAM	double-data rate (DDR) RAM
DPSRAM	dual-port SRAM
DRAM	dynamic random access memory
EBI	expansion bus interface
EDA	electronic-design automation
EOF	end of file
ESB	embedded system block
ETM	embedded trace macrocell
FIFO	first-in first-out
GOL	general operating language
GTL+	gunning transceiver logic plus
IC	integrated circuit
I/O	input/output
IP	intellectual property
IRQ	interrupt controller
JED	JEDEC file (**.jed**)
JEDEC	Joint Electronic Device Engineering Council
JTAG	Joint Test Action Group
LAB	logic array block
LSB	least significant bit
LVTTL	low-voltage transistor-transistor logic
MMU	memory management unit
MSB	most significant bit

Table 106. Common Acronyms (Part 2 of 2)

Acronym	Name
PCI	peripheral component interconnect
PLD	programmable logic device
PLL	phased-lock loop
POF	programmer object file (**.pof**)
POR	power-on reset
RAM	random-access memory
RISC	reduced instruction set computing
ROM	read-only memory
RTL	register transfer language
RTOS	run-time operating system
SDR	single-data rate
SDRAM	synchronous dynamic random-access memory
SOPC	system-on-a-programmable chip
SRAM	static random access memory
SSTL	stub series terminated logic
TAP	terminal access point
UART	universal asynchronous receiver/transmitter UART
WWW	world-wide web

Glossary

A

APEX 20K An Altera embedded programmable logic device family based on the Advanced Programmable Embedded Matrix (APEX™) architecture, which integrates look-up table logic, product-term logic, and memory in a single device. This family offers complete system integration on a single device. The APEX 20K device family includes the EP20K100, EP20K100E, EP20K160E, EP20K200, EP20K200E, EP20K300E, EP20K400, EP20K400E, EP20K600E, and EP20K1000E devices.

B

ball-grid array (BGA) A high-performance device package offered by Altera that allows for higher pin counts in significantly less board area than quad flat pack (QFP) packages and have better thermal characteristics than most QFP packages. BGA packages are rapidly becoming the preferred packages for high-density PLDs. See the *Altera Device Package Information Data Sheet* and *Ordering Information* for more information.

beat .A movement of data in a single clock period, which could be a byte, a half-word (2 bytes) or a word (4 bytes). Thus a burst transaction with 4 beats can be 4, 8 or 16 bytes, depending on the transaction size.

ByteBlasterMV cable A parallel download cable that allows PC users to program and configure devices in-system. The ByteBlasterMV™ parallel port download cable provides configuration support for APEX 20K, FLEX 10K, FLEX 8000, and FLEX 6000 devices. APEX 20K, FLEX 6000 and FLEX 10K devices can be configured together in a chain.

C

configuration device Altera's family of serial devices, which are designed to configure APEX and FLEX devices. See the *Configuration Devices for APEX & FLEX Devices* Data Sheet for more information.

configuration scheme The method used to load data into APEX 20K and FLEX devices.

Five configuration schemes are available for APEX 20K and FLEX 10K devices: configuration device, passive serial (PS), passive parallel asynchronous (PPA), passive parallel synchronous (PPS), and IEEE Std. 1149.1 Joint Test Action Group (JTAG). For complete information on FLEX 10K configuration schemes, see *Application*

Note 116 (Configuring APEX 20K, FLEX 10K, and FLEX 6000 Devices).

Three configuration schemes are available for FLEX 6000 devices: configuration device, passive serial (PS), and passive serial asynchronous (PSA). For complete information on FLEX 6000 configuration schemes, see *Application Note 116 (Configuring APEX 20K, FLEX 10K, and FLEX 6000 Devices)*.

D

dedicated input pin A pin that can only be used as an input to the device.

device Refers to an Altera programmable logic device, including APEX 20K, FLEX 10K, FLEX 8000, FLEX 6000, MAX 9000, MAX 7000, MAX 3000A, MAX 5000, and Classic devices. Altera also offers configuration devices that are used to configure APEX 20K, FLEX 10K, FLEX 8000, and FLEX 6000 devices.

device family A group of Altera programmable logic devices with the same fundamental architecture. Altera device families include the APEX 20K, FLEX 10K, FLEX 8000, FLEX 6000, MAX 9000, MAX 7000, MAX 3000A, MAX 5000, and Classic device families. Altera also offers a configuration device family that includes devices used for configuring APEX 20K, FLEX 10K, FLEX 8000, and FLEX 6000 devices.

E

Embedded logic Logic that is implemented in the stripe.

EPXA Family signature on a part number that refers to the Excalibur family.

F

FineLine BGA FineLine BGA packages available for Excalibur, APEX 20K, FLEX 10K, and MAX 7000 families use only half the board area of traditional BGA packages and are offered with as many as 784 pins for the EP20K1000E device. This new package allows designs to be effectively implemented into higher density, higher pin count devices into designs while decreasing board space and costs. See the *Altera Device Package Information Data Sheet* and *Ordering Information* for more information.

I

I/O cell Also known as an I/O element. A register that exists on the periphery of an APEX 20K, FLEX 10K, FLEX 8000, or MAX 9000 device, or a fast input-type logic cell that is associated with an I/O pin in MAX 7000E, MAX 7000S, or MAX 7000A devices. I/O cells give short setup and clock-to-out times.

J

Joint Test Action Group (JTAG) A set of specifications that enables a designer to perform board- and device-level functional verification of a board during production.

JTAG boundary-scan testing Testing that isolates a device's internal circuitry from its I/O circuitry. This testing is made possible by the JTAG boundary-scan test (BST) architecture that is available in all APEX 20K, FLEX 10K devices, all FLEX 8000 devices except the EPF8452A and EPF81188A, and all FLEX 6000, MAX 9000, MAX 7000S, MAX 7000A and MAX 3000A devices. Serial data is shifted into boundary-scan cells in the device; observed data is shifted out and externally compared to expected results. Boundary-scan testing offers efficient PC board testing, providing an electronic substitute for the traditional "bed of nails" test fixtures.

L

locked transaction A term used to qualify transactions (usually reads and writes) that take place without losing bus access. Masters request locked access to memory by asserting a lock signal at the same time as requesting the bus. In this situation, the bus remains granted to the master until the lock is de-asserted.

logic cell The generic term for the basic building block of an Altera device. In APEX 20K, FLEX 10K, FLEX 8000, and FLEX 6000 devices, logic cells are called logic elements. In MAX 9000, MAX 7000, MAX 3000A, MAX 5000, and Classic devices, logic cells are called macrocells.

logic element (LE) A basic building block of APEX 20K, FLEX 10K, FLEX 8000, and FLEX 6000 devices. A logic element consists of a look-up table (LUT)—i.e., a function generator that quickly computes any function of four variables—and a programmable register to support sequential functions. The register can be programmed as a flow-through latch, as a D, T, JK, or SR flipflop, or bypassed entirely for pure combinatorial logic. The register can feed

other logic cells or feed back to the logic cell itself. Some logic elements feed output or bidirectional I/O pins on the device.

M

MasterBlaster Communications Cable The MasterBlaster™ communications cable uses a PC serial or USB port hardware interface. This cable provides configuration data to APEX 20K, FLEX 10K, FLEX 8000, and FLEX 6000 devices, as well as programming data to MAX 9000, MAX 7000S, MAX 7000A, and MAX 3000A devices. The MasterBlaster communications cable also supports in-circuit debugging with the SignalTap embedded logic analyzer in APEX 20K devices.

P

plastic J-lead chip carrier (PLCC) A device package option offered by Altera. Both ceramic J-lead chip carrier (JLCC) and PLCC packages are available. See the *Altera Device Package Information Data Sheet* and *Ordering Information* for more information.

posted write A write that is posted to a bridge FIFO buffer and usually acknowledged immediately. However, the write need not have occurred at the time of acknowledgement. It is queued before sending to the destination.

product term Two or more factors in a Boolean expression combined with an AND operator constitute a product term, where "product" means "logic product".

programmable logic devices (PLDs) Digital, user-configurable integrated circuits used to implement custom logic functions. PLDs can implement any Boolean expression or registered function with built-in logic structures.

Q

Quartus The Quartus™ II software is Altera's fourth generation development system for programmable logic and allows designers to process multi-million gate designs. Features of the Quartus II software include: work group computing, integrated logic analysis functionality, electronic design automation (EDA) tool integration, multi-processor support, incremental recompilation, and intellectual property (IP) integration.

S

split transaction Split transactions allow other masters to access the buses while a high-latency slave access, such as reading flash memory, is in progress. If a transfer is likely to take a large number of cycles to perform, a split response signals to the bus arbiter that further requests from the master attempting the access should be masked until the slave indicates that it is ready to complete the transfer (by asserting the appropriate split signal for the master that has been split)

T

thin quad flat pack (TQFP) A device package offered by Altera. See the *Altera Device Package Information Data Sheet* and *Ordering Information* for more information.

U

user I/O The total number of I/O pins and dedicated inputs on a device.

Notes:

Appendix A
Excalibur Device Comparison

Table 1 shows a comparison of the devices in the Excalibur family.

Table 1. Excalibur Device Comparison (Part 1 of 2)

Feature(2)		EPXA10		EPXA4				EPXA1			
Package		F1020		F1020		F672		F672		F484	
Max processor frequency	-1	200 MHz		200 MHz		200 MHz		200 MHz		200 MHz	
	-2	166 MHz		166 MHz		166 MHz		166 MHz		166 MHz	
	-3	133 MHz		133 MHz		133 MHz		133 MHz		133 MHz	
ETM9 Version(3)		1		2a		2a		None		None	
SRAM (Bytes)		256K		128K		128K		32K		32K	
DPRAM (Bytes)		128K		64K		64K		16K		16K	
DPRAM Instances		2		2		2		1		1	
SDRAM Controller Data Bus Width(4)		32-Bit		16 / 32-Bit		16 / 32-Bit		16 / 32-Bit		16-Bit	
Max SDR SDRAM frequency	-1	133 MHz		133 MHz		133 MHz		133 MHz		133 MHz	
	-2	100 MHz		100 MHz		100 MHz		100 MHz		100 MHz	
	-3	83 MHz		83 MHz		83 MHz		83 MHz		83 MHz	
Max DDR SDRAM Frequency	-1	266 MHz		266 MHz		266 MHz		266 MHz		266 MHz	
	-2	200 MHz		200 MHz		200 MHz		200 MHz		200 MHz	
	-3	166 MHz		166 MHz		166 MHz		166 MHz		166 MHz	
Interrupt controller		Y		Y		Y		Y		Y	
Watchdog timer		Y		Y		Y		Y		Y	
General purpose timer		32-bit with pre-scale & interrupts		32-bit with pre-scale & interrupts		32-bit with pre-scale & interrupts		32-bit with pre-scale & interrupts		32-bit with pre-scale & interrupts	
UART		Y		Y		Y		Y		Y	
Embedded stripe PLL ranges		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
N		1	15	1	255	1	255	1	255	1	255
M		1	15	1	255	1	255	1	255	1	255
K		1	7	1	255	1	255	1	255	1	255
Standby PLL Bypass Current(5)		30mA		30mA		30mA		5mA		5mA	

Table 1. Excalibur Device Comparison (Part 2 of 2)

Feature(2)	EPXA10	EPXA4		EPXA1	
Package	F1020	F1020	F672	F672	F484
Internal I/O between stripe(1) & PLD(6)	None	8 Inputs 8 Outputs	8 Inputs 8 Outputs	4 Inputs 4 Outputs	4 Inputs 4 Outputs
Available User I/O(7)	711	488	426	246	186
Dedicated Stripe I/Os(8)	SD_CLKE	See Note	See Note	All	All
Stripe I/O Standards Control(9)	CRAM Bits	IOCR LVTTL0, LVTTL1, SSTL	IOCR LVTTL0, LVTTL1, SSTL	IOCR LVTTL0, LVTTL1, SSTL	IOCR LVTTL0, LVTTL1, SSTL
I/O voltages(10)	1.8 V	1.8 V	1.8 V	1.8 V	1.8 V
	2.5 V	2.5 V	2.5 V	2.5 V	2.5 V
	3.3 V	3.3 V	3.3 V	3.3 V	3.3 V

Notes:

(1) The Term 'stripe' refers to the region of the die containing the processor and the hard peripherals. The remainder of the die is the PLD (programmable logic device). Both are internally connected by bridges and internal I/Os.

(2) -1, -2 and –3 refer to PLD speed grades, where –1 is the fastest PLD speed and –3 the lowest. Faster PLD speeds also imply a faster processor and faster communication with the SDRAM. The speed grade is the last digit in the ordering code of a production device (e.g. EPXA10F1020C**2** for –2 speed grade).

(3) ETM9 version 1 is supported by the ARM Trace Debug Tools version 1.1 and later, available from ARM Limited. ETM9 version 2a is supported by the ARM Trace Debug Tools version 1.2 and later, available from ARM in Q2, 2002. Support from other vendors is available today, contact Excalibur marketing for details.

(4) SDRAM devices using address line A10 to initiate refresh are supported. 8 and 16-bit wide SDRAM devices are supported by all members of the EPXA family, with an aggregate width of 16 or 32- bits.

(5) EPXA1 uses software enabling of the PLL for low power applications.

(6) EPXA4 and EPXA1 incorporate general purpose I/Os between the stripe and the PLD, with identical numbers of input and outputs. Full support will be provided in Quartus II V2.1. In this instance, output means stripe to PLD and input means PLD to stripe connection.

(7) The total available user I/Os = shared stripe I/O + PLD I/O.

(8) SDRAM Clock Enable is the only dedicated Stripe pin on the EPXA10. All other stripe pins are shared with the PLD. All pins on the EPXA4 are shared by the stripe and PLD except the following:
`EBI_A[24..22]`, `SD_DQ[31..24]`, `SD_DQS3`, `DDR_VS2`, `SD_CLKE` and all trace port pins.
All stripe I/Os on the EPXA1 are dedicated.

(9) All stripe I/Os support the following I/O standards:
LVTTL0, LVTTL1, SSTL.
For the EPXA10, all stripe I/O modes are set using CRAM bits. CRAM Bits are Configuration RAM bits, which can only be written during a device configuration.
For the EXPA4 and EPXA1, the stripe I/O standards are set through the IO_CR register of the ports (in the stripe). There is an IOCR register for each stripe peripheral, allowing each peripheral to have its IO standard selected independent of all other peripherals. All PLD I/Os standards are available as per the APEX20KE device specification.

(10) The Vcc voltage range for 1.8V and 2.5V I/O voltages on commercial and industrial temperature parts is +/-5%. Vcc voltage range for 3.3V I/O voltages is +/-5% on commercial and +/-10% on industrial temperature parts.