BA22

32-bit RISC Processor

Hardware Specification

April 2012

IP Product Version
4.0p06

Document Signature
BA22–DES–4.0p06–104

CAST, Inc.

Document Version

This document with its associated Release Notes applies to the version(s) of the core specified on the cover. See the Release Notes for any updates and additional information not included here.

Document Version History

Version	Date	Person	Changes from Previous Version
1.00	28 July, 2011	P.D.	First draft and approval.
1.01	14 Sept, 2011	N.Z, M.B	Restructured and reviewed
1.02	15 Sept, 2011	N.S.	Improved TEE/IEE and Endian info
1.03	26 Jan 2012	N.S.	Configuration name updates
1.04	03 Apr, 2012	J.S.	Updated Figure 2

CAST, Inc.
11 Stonewall Court, Woodcliff Lake, NJ 07677
Phone: +1 201-391-8300 Fax: +1 201-391-8694
info@cast-inc.com www.cast-inc.com

Table of Contents

1. Abbreviations & Acronyms 5

2. References 6
2.1 Industry Standards and Specifications 6
2.2 Related Documents 6
2.3 Web Sites 6

3. Introduction and Features 7
3.1 Introduction 7
3.2 Features 7
3.3 Configurations 8
3.3.1 BA22-DE 8
3.3.2 BA22-EM 8
3.3.3 BA22-AP 9

4. Symbol and Interfaces 10
4.1 Clocks and Reset 11
4.2 Endianness 11

5. Architectural Specification 13
5.1 Arithmetic Units 13
5.2 Register Sets 13
5.3 System Busses 14
5.3.1 AMBA/AHB Interface 14
5.3.1.1. AMBA Data Bus Interface 14
5.3.1.2. AMBA Instruction Bus Interface 15
5.3.2 Wishbone Interface 16
5.3.2.1. Wishbone Data Bus Interface 16
5.3.2.2. Wishbone Instruction Bus Interface 17
5.4 Memory Interfaces 17
5.4.1 Tightly Coupled Memories (QMEM) 17
5.4.1.1. IQMEM 18
5.4.1.2. DQMEM 18
5.4.2 Cache Memories 20
5.4.2.1. Registers 20
5.4.2.2. Cache Memories Interfaces 21
5.4.3 Memory Management Unit 22
5.4.3.1. TLB Memories Interfaces 22
5.5 Power Management Unit 22
5.5.1 Interface 22
5.5.1.1. PMU Signal Relationship 23
5.5.2 Registers 24
5.5.2.1. Power Management Event Control Register – PMEVNT 24
5.5.2.2. Power Management Unit Power Down Register – PMUNITPD 25
5.5.3 Sample Configuration 25
5.6 Debug Unit 25
5.6.1 Interface 26
5.6.2 Registers 26
5.6.2.1. Debug Value Registers (DVR0-DVR7) 26
5.6.2.2. Debug Control Registers (DCR0-DCR7) 27
5.6.2.3. Debug Mode Register 1 (DMR1) 27
5.6.2.4. Debug Mode Register 2 (DMR2) 27
5.6.2.5. Debug Stop Register (DSR) 27
5.6.2.6. Debug Reason Register (DRR) 27
5.7 Integrated Peripherals 27
5.7.1 Programmable Interrupt Controller 27
5.7.1.1. Interface 28
5.7.1.2. Registers 28

5.7.2 Tick Timer 28
5.7.2.1. Timer Interrupt 29
5.7.2.2. Tick Timer Modes 29
5.7.2.3. Tick Timer Registers 29

List of Figures

Figure 1: BA22-DE 8
Figure 2: BA22-EM 9
Figure 3: BA22-AP 9
Figure 4: BA22 Signals 10
Figure 5: Instruction QMEM read cycle 18
Figure 6: Data QMEM read cycle 19
Figure 7: Data QMEM write cycle 20
Figure 8: Power Management Clock Mode Change 23
Figure 9:Power management Clock Gating 24
Figure 10: Clock Gating 24
Figure 11: Sample PMU Configuration 25

List of Tables

Table 1:Abbreviations & Acronyms 5
Table 2: BA22 Signal Description 10
Table 3: BA22 Special Purpose Registers 13
Table 4: AHB Data Bus Interface – Signal Description 14
Table 5 AHB Instruction Bus Interface – Signal Description 15
Table 6: Wishbone Data Interface – Signal Description 16
Table 7: Wishbone Instruction Interface – Signal Description 17
Table 8: IQMEM Signal Description 18
Table 9: DQMEM Signal Description 18
Table 10: TTCR Data Cache Block Invalidate Register 20
Table 11: Data Cache Block Flush Register 20
Table 12: Instruction Cache Block Invalidate Register 20
Table 13: Cache Memory Modules 21
Table 14: Interface Signals for Single-Port Memories 21
Table 15: Interface Signals for Dual-Port Memories 21
Table 16: MMU Memory Modules 22
Table 17: PMU Signal Description 22
Table 18: PMEVNT Register (0x4000) 24
Table 19: PMUNITPD Register (0x4002) 25
Table 20: Debug Interface – Signal Description 26
Table 21: PIC Signal Description 28
Table 22: PICMR Register (0x4800) 28
Table 23: PICSR Register (0x4802) 28
Table 24: TTMR Register (0x5000) 29
Table 25: TTCR Register (0x5001) 30
Table 26: TTRS Register (0x5002) 30

1. Abbreviations & Acronyms

Table 1lists abbreviations and acronyms used throughout this documentation.

Table 1:Abbreviations & Acronyms

Term	Description
CPU	Central Processing Unit
MMU	Memory Management Unit
SPR	Special Purpose Register
GPRS	General Purpose RegisterS
PIC	Programmable Interrupt Controller
PMU	Power Management Unit
MIPS	Million Instructions Per Second benchmark
DMIPS	Dhrystone MIPS benchmark
RISC	Reduced Instruction Set Computing
LRU	Least Recently Used (type of cache)

2. References

2.1 Industry Standards and Specifications

[1] AMBA™ Specification(Rev 2.0)

[2] WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores Revision: B.3

2.2 Related Documents

[1] BA22 Programmer's Manual

[2] BA22 Integration Manual

[3] BA22 External Peripherals Manual

2.3 Web Sites

CAST inc, – http://www.cast-inc.com

Beyond Semiconductor – http://www.beyondsemi.com

3. Introduction and Features

3.1 Introduction

The BA22 processor is 4 to 5 stage (configurable) modern RISC design. The CPU core be optionally equipped with a Floating Point Unit (FPU), a 32bit divider, and a 64bit multiply-accumulate unit. A programmable interrupt controller and a tick timer are pre-integrated with the CPU core.

The advanced architecture can include up to 32 general purpose instruction registers (GPR). Furthermore, the BA22 supports up to 31 sets of Special Purpose Registers (SPRs) with up to 1k registers per group, allowing for easy control of the processor units. Peripherals can be connected to the processor via either AMBA AHB or Wishbone SoC busses that can be either little or big endian.

Tightly coupled memories (QMEMs) allow for fast access to on-die instruction and data memories, typically required in deeply embedded applications. Additionally, up-to 4-way caches and 4-way TLBs can be optionally instantiated to enable the BA22 to run modern OS and multimedia applications.

An integrated power management unit allows for individual clock gating of all major components of the processor, and on-the-fly frequency scaling, in order to adjust power consumption to workload requirements in real time. Finally, the BA22 can optionally be equipped with a debug unit enabling hardware breakpoints and real time execution trace buffer.

3.2 Features

- High Performance 32-bit CPU
  - Single-cycle instruction execution on most instructions
  - Fast and precise internal interrupt response
  - Custom user instructions
  - 1.41 DMIPS/MHz
- Fast & Flexible Memory Access
  - Harvard-style Caches and MMU separate for Instructions and Data
  - Tightly coupled Quick Memory (QMEM) for fast and deterministic access to code and/or data
- Efficient Power Management
  - Power reduction from 2x to 100x by dynamic clock gating for individual processor units
  - Software controlled clock frequency in slow and idle modes
  - Interrupt wake-up in doze and sleep modes
- Advanced Debug Unit
  - Non-intrusive debug/trace for both RISC and system
  - Access and control of debug unit from RISC or via development interface
  - Complex chained watchpoint and breakpoint conditions
  - Uses industry standard AmontecJTAGKey USB to JTAG interface
- Integrated Peripherals
  - 32 bits-wide tick timer and Programmable interrupt controller with 32 maskable interrupt sources
- Optional Peripherals
  - AMBA bus infrastructure cores
  - GPIOs. UARTS, Timers cores
  - Serial communication cores such as I2C and SPI
  - Memory controllers, interconnect IP and more
- Easy Software Development

– Complete IDE for Windows, and Linux under Eclipse
– Ported C-libraries and OS.

3.3 Configurations

The BA22 Processor is provided at three different configurations intended for different applications:

- BA22-
- BA22-
- BA22-AP

When needed, each of the above configurations can be easily tailored to match the customer's specific requirements.

Additional microcontroller peripherals may be ordered for pre-integration and delivery with the each of the configurations, individually or in a complete platform. IP Integration Services are also available to help integrate any BA22 processor configuration with memory controllers, image compression, or other CAST IP cores.

3.3.1 BA22-DE

The BA22-DE implements a 32-bit RISC processor for deeply embedded applications that use on-chip instruction and data memories. The core has an AMBA/AHB or optional Wishbone system bus interface and dedicated buses for the on-chip instructions and data memories.

It includes 16 GPRs, the Tick Timer, the Programmable Interrupt Controller, the Power Management Unit and optionally the Debug Unit.

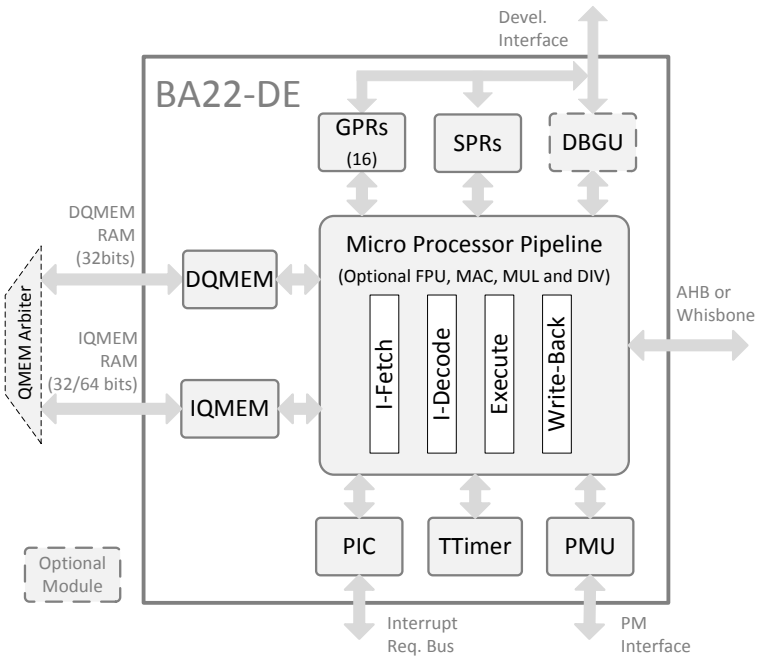

Figure 1: BA22-DE

3.3.2 BA22-EM

The BA22-EM implements a 32-bit RISC processor for deeply embedded applications that use off-chip instruction and data memories and that may need to run a real-time operating system (RTOS).

The configuration has Cache Memories, dedicated buses for the on-chip instructions and data memories, and an AMBA/AHB or optional Wishbone system bus interface. It includes 32 GPRs, the Tick Timer, the Programmable Interrupt Controller, the Power Management Unit and optionally the Debug Unit.

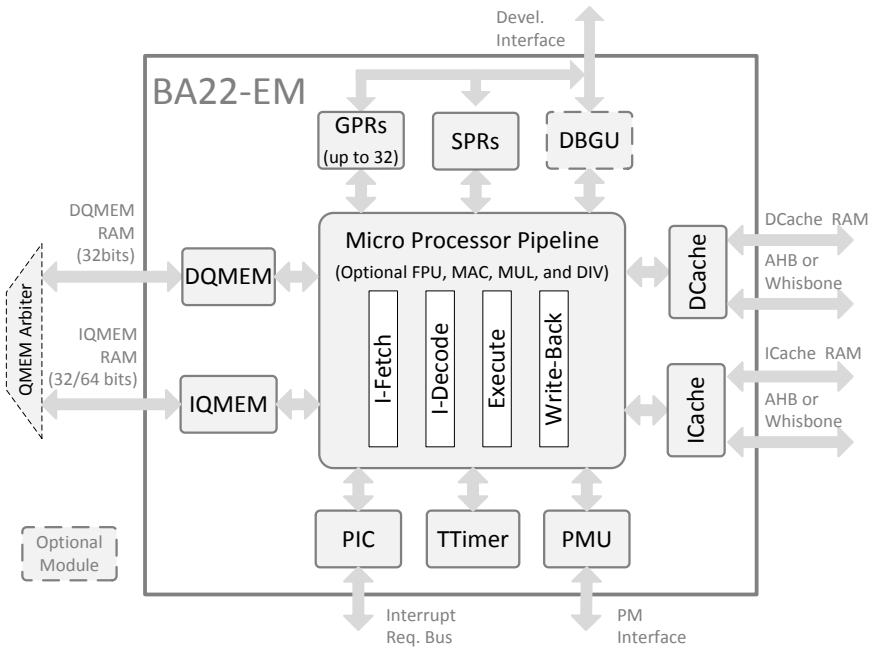

Figure 2: BA22-EM

3.3.3 BA22-AP

The BA22-AP implements a 32-bit RISC processor for demanding embedded applications that use off-chip instruction and data memories and that may need to run a real-time operating system (RTOS) or a full operating system such as Linux or Android.

The core has Instruction and Data Cache Memories,Memory Management Units, dedicated buses for on-chip instructions and data memories, and an AMBA/AHB or optional Wishbone system bus interface. Optional floating point, divider and multiply–accumulate units (Arithmetic Units) benefit DSP applications. The platform includes up to 32 GPRs, the Tick Timer, the Programmable Interrupt Controller, the Power Management Unit and optionally the Debug Unit.

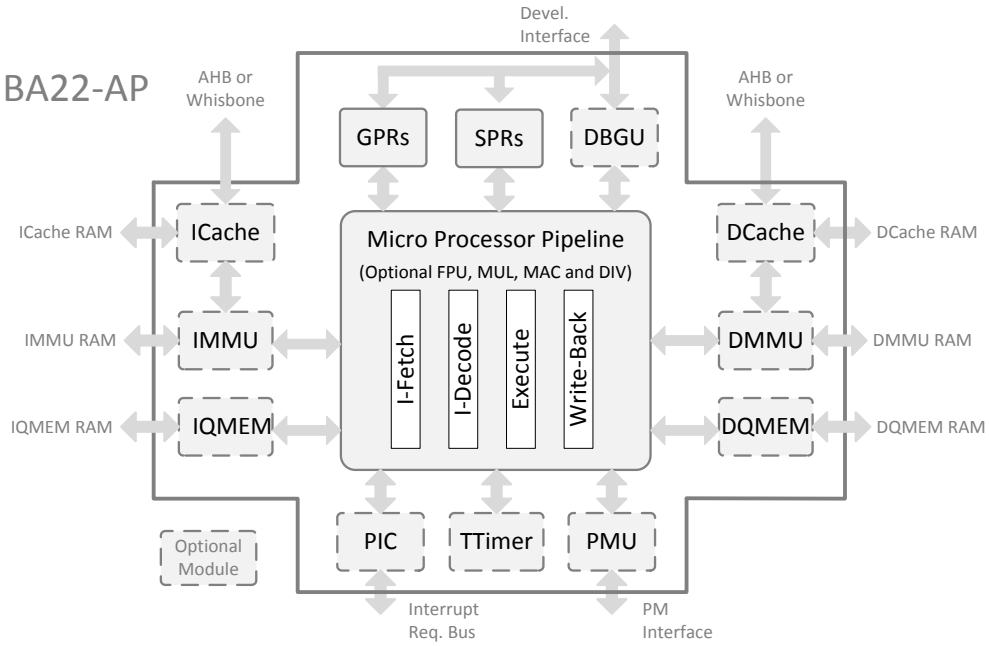

Figure 3: BA22-AP

4. Symbol and Interfaces

Figure 4 depicts a brief BA22 diagram with its I/Os. The outline presents all the possible groups of signals and interfaces however the final customer's version can include only some of them.

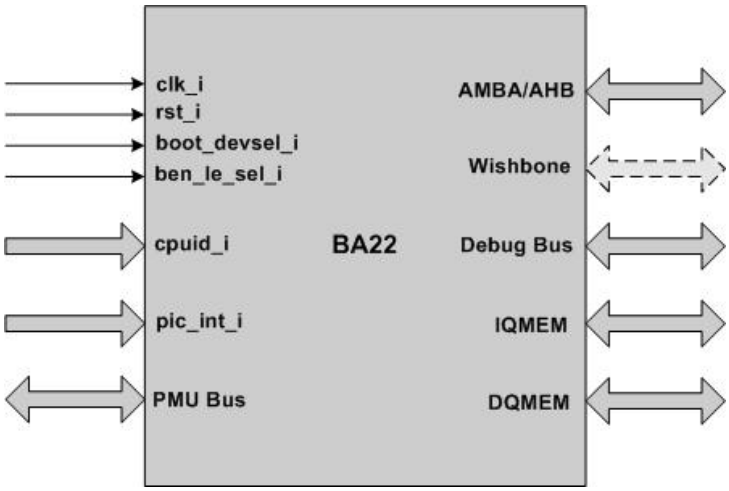

Figure 4: BA22 Signals

Table 2 briefly describes the depicted signals' sets. For more detailed description please follow the given links to the appropriate further sections of this documentation.

Table 2: BA22 Signal Description

Name	Type	Polarity / Bus Size	Description
IQMEM	Tightly coupled Instruction memory (IQMEM)interface		
DQMEM	Tightly coupled Data memory (DQMEM)		
AMBA	AMBA Interface used for connecting the external peripherals and off-chip memory		
Wishbone	Optional Wishbone Interface used for connecting the external peripherals and off-chip memory		
Debug Bus	Debug Unit Interface		
PMU Bus	Power Management Unit Interface.		
clk_i	In		CPU Clock
rst_i	In	Configurable at synthesis time (default High)	Asynchronous Reset
cpuid_i	In	32	Used to distinguish between multiple BA22 processors in a multiprocessor environment. Default value depends on the system architecture.
boot_devsel_i	In	-	The *boot_devsel_i* can be used to toggle booting the CPU from different memories. 0 – Exceptions vectors are located in memory area starting from 0x0 1 – Exceptions vectors are located in memory area starting from a synthesis time configurable address
ben_le_sel_i	In	-	Selects between little endian and big endian modes. 0 - big endian 1 - little endian
pic_int_i	In	Configurable at synthesis time Possible range: 1-32 Active high	Interrupt interface. Detailed information can be found in Programmable Interrupt Controllersection.

4.1 Clocks and Reset

The BA22 CPU operates synchronously to the rising edge of the *clk_i* signal. Other secondary clock lines are also present allowing clock gating and frequency scaling of independent modules:

- *pm_clk_i* – power management clock used to run the Tick Timer, PIC and a few PMU-related registers
- *du_clk_i* – Debug Unit clock
- *rf_clk_i* – Register File clock
- *dahb_clk_i, iahb_clk_i* – clock signals of the default AHB interfaces
- *dwb_clk_i, iwb_clk_i* – clock signals of the optional Wishbone interfaces

The following constraints should be taken into account during SoC integration of the BA22 processor:

- All of the above clocks should be synchronous to each other.
- The rf_clk_i, du_clk_i and clk_i should always run at the same clock frequency, but they can be independently gated. The *du_clk_i*should be gated when the debug unit is not used, while the *rf_clk_i*gating can be controlled by the BA22 (see Power Management Unit section).
- The *pm_clk_i* clock should always have the same or higher frequency with relationship to the other clock signals used in the BA22 chip level architecture.
- All the above clocks except the *pm_clk_i* can be gated (or scaled down) in the power down mode. More details about low power mode can be found in the Power Management Unit section.
- The BA22 is reset by using the signal *rst_i*. A pre-synthesis parameter describes the valid polarity of the signal and whether the reset is asynchronous or synchronous. The default *rst_i*value is High, and it is asynchronous .

A single reset line, the *rst_i*, resets the BA22 processor. The reset signal needs to be asserted for at least one clock cycle on each clock, and it must transition to a low state synchronously with the slowest clock.The reset exception vector is located at 0x100 vector offset. More information about other exceptions can be found in the “BA22 Programmer’s Manual” document.

The optional AHB Data and interfaces also use separate reset signals: *iahb_resetn_i*and*dabh_resetn_i.*Designer should take into account that the AHB reset signals are active-low.

4.2 Endianness

The BA22 can support big or little endian architectures through the use of a synthesizable parameter (refer to the Integration Manual for details), or if can be set up to support in-system changeable endianness using the asynchronous input signal *ben_le_sel_i*. The *ben_le_sel_i* has the following encoding:

- 0 – the BA22 operates in big-endian mode.
- 1 – the BA22 operates in little-endian mode.

The BA22 does not synchronize or do any other specific action on a change of ben_le_sel_i, so it is the task of the system to properly handle the signal. For the change to work correctly, the BA22 must have its pipeline, caches, data and instruction buses in idle state when *ben_le_sel_i* changes state. The safest and easiest way to select an endianness of the BA22 is thus during power up/reset.

If it is not possible to reset the whole system when a change of endianness is desired, then only the BA22 can be reset, but bus protocol violations may occur when doing so. If the system cannot tolerate bus protocol violations, the BA22 can safely be reset within otherwise operational system. The Power Management Interface, described in Power Management Unit section can be used for that purpose. The following procedure should be followed:

1. Assert the *pm_stall_i* signal
2. Wait for *pm_stalled_o* to get asserted.

3. Apply reset to the BA22 – the value of *pm_stalled_o* might change during reset, so it value should ignored it from this point forward.
4. Apply new *ben_le_sel_i* value.
5. De-assert *pm_stall_i*
6. Remove the BA22 reset.

The BA22 should start from the reset vector in the appropriate endianness mode.

If the system can tolerate bus protocol violations, the BA22 can be individually reset without using Power Management Interface.

Finally, the clock mode changing feature of the Power Management Interface is used, special care must be takes so that the BA22 wakes up in its default clock mode. This implies that endianness change with reset applied selectively to BA22 must be done in default clock mode for the processor to wake-up correctly after reset

5. Architectural Specification

The following sections describe the modules, and corresponding interfaces of the BA22 processor, namely the optional Arithmetic Units, the Register Sets, the System Busses, the Memory Interface, the Power Management Unit, the Debug Unit and the Integrated Peripherals.

5.1 Arithmetic Units

The BA22 can be optionally equipped with the following arithmetic units::

- DIV – provides for 32-bit division and modulo operations
- FPU – supports single precision floating point functionality
- MUL – responsible for 32-bit signed multiplication
- MAC – provides 32-bit signed multiplication and accumulation

The arithmetic units are instantiated at synthesis time, and do not affect the top-level interfaces of the core.

5.2 Register Sets

The BA22 includes two general sets of registers:

- Up to 32 User General Purpose 32-bit Registers (GPRS). The number of GPRS is defined at synthesis and can vary in the 16-32 range.
- Special Purpose Registers (SPR). The implementation of some of the SPRs depends on the chosen BA22 features. For instance, when Caches are implemented, the appropriate Cache registers are also instantiated.

The special-purpose registers of all units are grouped into thirty-two groups. Each group can have different register address decoding depending on the maximum theoretical number of registers in that particular group. A group can contain registers from several different units or processes.

Table 3: BA22 Special Purpose Registers

GROUP	UNIT DESCRIPTION
0	System Control and Status registers
1	Data Memory Management Unit (in the case of a single unified MMU, groups 1 and 2 decode into a single set of registers)
2	Instruction Memory Management Unit (in the case of a single unified MMU, groups 1 and 2 decode into a single set of registers)
3	Data Cache (in the case of a single unified cache, groups 3 and 4 decode into a single set of registers)
4	Instruction Cache (in the case of a single unified cache, groups 3 and 4 decode into a single set of registers)
5	Multiply ACcumulate unit
6	Debug unit
7	Performance counters unit
8	Power Management
9	Programmable Interrupt Controller
10	Tick Timer
12-23	Reserved for future use
24-31	Custom units

The BA22 processor is required to have implemented at least the special purpose registers from group 0. All other groups are optional, and registers from these groups are implemented only if the

implementation includes the corresponding unit. Which units are actually implemented may be determined by reading the Unit Present Register (UPR) from group 0.

A 16-bit SPR address is made of 5-bit group index (bits 15-11) and 11-bit register index (bits 10-0) assigned to every register in each group.

More information about the BA22 registers is available in the “BA22 Programmer’s Manual” document.

5.3 System Busses

5.3.1 AMBA/AHB Interface

The default AMBA/AHB 2.0 interface is integrated into the BA22 as an interface to common peripherals and memory. There is a variety of peripheral choices which are detailed in the “BA22 External Peripherals Manual” document.

For the BA22-DE configuration only the AHB Data Bus Interface is implemented. Configurations implementing the Instruction Memory Management Unit can optionally implement an additional AHB Instruction Bus Interface.

Some of the external peripherals may have the APB interface. To connect those modules, a standard AHB–APB Bridge can be used.

Below is a brief description of both the Data and Instruction AMBA interfaces. More detailed info can be found in the AMBA™ Specification (Rev 2.0).available at www.arm.com

5.3.1.1. AMBA Data Bus Interface

Table 4 lists the IO signals comprising the 32-bit data AHB interface. It also cross-references all signals to AMBA™ Specification (Rev 2.0). All signals are synchronous to *ahb_clk_i* signal. The interface is reset using the *dahb_resetn_i* signal.

Table 4: AHB Data Bus Interface – Signal Description

Name	Type	Polarity/ Bus size	Description
dahb_clk_i	In	-	HCLK -this clock times all bus transfers. All signal timings are related to the rising edge of **HCLK**
dahb_resetn_i	In	Low	HRESETN - the signal is used to reset the system and the bus. The polarity of the *dahb_resetn_i* is programmable
dahb_addr_o	Out	32	HADDR - System address bus
dahb_trans_o	Out	2	HTRANS - Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE orBUSY
dahb_write_o	Out	High	HWRITE - When HIGH this signal indicates a write transfer and when LOW a read transfer
dahb_size_o	Out	3	HSIZE - indicates the size of the transfer, which is typically byte (8-bit), half word (16-bit) or word (32-bit). The protocol allows for larger transfer sizes up to a maximum of 1024 bits
dahb_burst_o	Out	3	HBURST - indicates if the transfer forms part of a burst. Four, eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping
dahb_prot_o	Out	4	HPROT - the protection control signals provide additional information about a bus access and are primarily intended for use by any module that wishes to implement some level of protection. The signals indicate if the transfer is an opcode fetch or data access, as well as if the transfer is a privileged mode access or user mode access. For bus masters with a memory management unit these signals

Name	Type	Polarity/ Bus size	Description
			also indicate whether the current access is cacheable or bufferable
dahb_wdata_o	In	32	HWDATA - the write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation
dahb_rdata_i	In	32	HRDATA - the read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation
dahb_ready_i	In	High	HREADY - when HIGH the **HREADY**signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer.
dahb_resp_i	In	2	HRESP - the transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY and SPLIT
dahb_busreq_o	Out	High	HBUSREQ - a signal from bus master to the bus arbiter which indicates that the bus master requires the bus.
dahb_lock_o	Out	High	HLOCK - when HIGH this signal indicates that the master requires locked access to the bus and no other master should be granted the bus until this signal is LOW
dahb_grant_i	In	High	HGRANT - this signal indicates that bus master x is currently the highest priority master. Ownership of the address/control signals changes at the end of a transfer when **HREADY**is HIGH, so a master gets access to the bus when both **HREADY**and **HGRANT**are HIGH

5.3.1.2. AMBA Instruction Bus Interface

Table 5 lists the IO signals comprising the 32-bit Instruction AHB interface. It also cross-references all signals to AMBA™ Specification (Rev 2.0). All signals are synchronous to *ahb_clk_i* signal. The interface is reset using *iahb_resetn_i* signal.

Here, only the port descriptions are given, for more detailed information on the AHB bus interface please refer to AMBA AHB specification.

Table 5 AHB Instruction Bus Interface – Signal Description

Name	Type	Polarity/ Bus Size	Description
iahb_clk_i	In	-	HCLK
iahb_resetn_i	In	Low	HRESETN
iahb_addr_o	Out	32	HADDR
iahb_trans_o	Out	2	HTRANS
iahb_write_o	Out	High	HWRITE
iahb_size_o	Out	3	HSIZE
iahb_burst_o	Out	3	HBURST

Name	Type	Polarity/ Bus Size	Description
iahb_prot_o	Out	4	HPROT
iahb_wdata_o	Out	32	HWDATA
iahb_rdata_i	In	32	HRDATA
iahb_ready_i	In	High	HREADY
iahb_resp_i	In	2	HRESP
iahb_busreq_o	Out	High	HBUSREQ
iahb_lock_o	Out	High	HLOCK
iahb_grant_i	In	High	HGRANT

5.3.2 Wishbone Interface

The optional Wishbone interface is used for connecting peripheral units and memories to the BA22

For the BA22-DE configuration only the Wishbone Data Bus Interface is implemented. Configurations implementing the Instruction Memory Management Unit can optionally implement an additional Wishbone Instruction Bus Interface.

5.3.2.1. Wishbone Data Bus Interface

Table 6 lists the IO signals comprising 32-bit data WISHBONE memory interface. It also cross-references all signals to WISHBONE SoC Interconnection Wishbone SystemOnChip (SoC) Interconnection Architecture for Portable Cores Architecture for Portable IP Cores specification, Revision B.

Table 6: Wishbone Data Interface – Signal Description

Name	Type	Polarity/ Bus Size	Description
dwb_clk_i	In	-	CLK_I – the Clock signal coordinates all activities for the internal logic within the WISHBONE interconnect
rst_i	In	High	RST_I - the reset input forces the WISHBONE interface to restart. Furthermore, all internal self-starting state machines will be forced into an initial state
dwb_cyc_o	Out	High	CYC_O - the cycle output, when asserted, indicates that a valid bus cycle is in progress. The signal is asserted for the duration of all bus cycles
dwb_stb_o	Out	High	STB_O - the strobe output indicates a valid data transfer cycle
dwb_we_o	Out	High	WE_O - the write enable output indicates whether the current local bus cycle is a READ or WRITE cycle. The signal is negated during READ cycles, and is asserted during WRITE cycles
dwb_cti_o	Out	3	CTI_O - the Cycle Type Identifier Address Tag provides additional information about the current cycle
dwb_bte_o	Out	2	BTE_O - Burst Type Extension
dwb_adr_o	Out	32	ADR_O - the address output array is used to pass a binary address
dwb_sel_o	Out	4	SEL_O - the select output array indicates where valid data is expected on the DAT_I()signal array during READ cycles, and where it is placed on the DAT_O() signal array

Name	Type	Polarity/ Bus Size	Description
			during WRITE cycles
dwb_dat_o	Out	32	DAT_O - the data output array is used to pass binary data
dwb_ack_i	In	High	ACK_I - the acknowledge input, when asserted, indicates the normal termination of a bus cycle
dwb_rty_i	In	High	RTY_I - the retry input indicates that the interface is not ready to accept or send data, and that the cycle should be retried
dwb_err_i	In	High	ERR_I - the error input indicates an abnormal cycle termination
dwb_dat_i	In	32	DAT_I - the data input array is used to pass binary data

5.3.2.2. Wishbone Instruction Bus Interface

The Instruction Wishbone interface is used whenever embedded memory resources are not sufficient. The interface comprises of all wishbone signals, however some signals are not used and therefore wired to inactive state.

Table 7 lists the IO signals comprising 32-bit instruction WISHBONE memory interface. It also cross-references all signals to WISHBONE SoC Interconnection Architecture for Portable IP Cores specification, Revision B.3. All signals are synchronous to *iwb_clk_i*signal. The interface is reset using *rst_i*signal.

Here, only port descriptions are given, for detailed information on Wishbone bus interface please refer to Wishbone specification.

Table 7: Wishbone Instruction Interface – Signal Description

Name	Type	Polarity / Bus Size	Description
iwb_clk_i	In	-	CLK_I
rst_i	In	-	RST_I
iwb_cyc_o	Out	High	CYC_O
iwb_stb_o	Out	High	STB_O
iwb_we_o	Out	High	WE_O Wired to logical zero
iwb_cti_o	Out	3	CTI_O
iwb_bte_o	Out	2	BTE_O
iwb_adr_o	Out	32	ADR_O
iwb_sel_o	Out	4	SEL_O
iwb_dat_o	Out	2	DAT_O Wired to logical zero
iwb_ack_i	In	High	ACK_I
iwb_rty_i	In	High	RTY_I
iwb_err_i	In	High	ERR_I
iwb_dat_i	In	32	DAT_I

5.4 Memory Interfaces

5.4.1 Tightly Coupled Memories (QMEM)

The QMEM busses are intended to give the designer a low latency option for connecting the program and data memories in typical deeply embedded applications. The tightly coupled memory interface is typically used for on-die memories although it is not restricted solely to that purpose. Both busses are strictly synchronous.

Less demanding applications may benefit from a unified QMEM structure in which case an arbiter may be used to manage the common memory module. In such a case, a special care must be taken so that the data memory access, always have a priority over the instruction memory access.

5.4.1.1. IQMEM

Table 8 lists and describes the IO signals comprising the 64-bit instruction QMEM interface.

All signals are synchronous to *clk_i*signal, the interface is reset using *rst_i* signal.

Table 8: IQMEM Signal Description

Name	Type	Polarity/Bus Size	Description
iqmem_stb_o	Out	High	Handshake request signal. It qualifies dqmem_adr_o, *dqmem_dat_o,dqmem_sel_o, dqmem_we_o,dqmem_ack_i* and *dqmem_err_i* signals.
iqmem_ack_i	In	High	Handshake response signal. When *dqmem_stb_o* and *dqmem_ack_i* are sampled asserted on the rising clock edge, *dqmem_dat_i* must be valid on the next rising clock edge.
iqmem_err_i	In	High	Handshake response signal. Memory subsystem indicates inability to provide data from memory location addressed by *dqmem_adr_o.*
iqmem_adr_o	Out	32	Memory address. Asserted simultaneous with *dqmem_stb_o* and *dqmem_ack_i.*
iqmem_dat_i	In	64	Memory data input bus. QMEM must provide data on this bus one rising clock edge after *dqmem_stb_o*and*dqmem_ack_i* were sampled asserted and *dqmem_we_o* was sampled de-asserted.
iqmem_keep_o	Out	High	In power down mode, the High *iqmem_keep_o* means that another read from the IQMEM RAM is not needed. The IQMEM data input should not be changed during this time. Such *iqmem_keep_o* usage enables power savings by avoiding reading the same location of the RAM. The iqmem_keep_o can be ignored in any other mode.

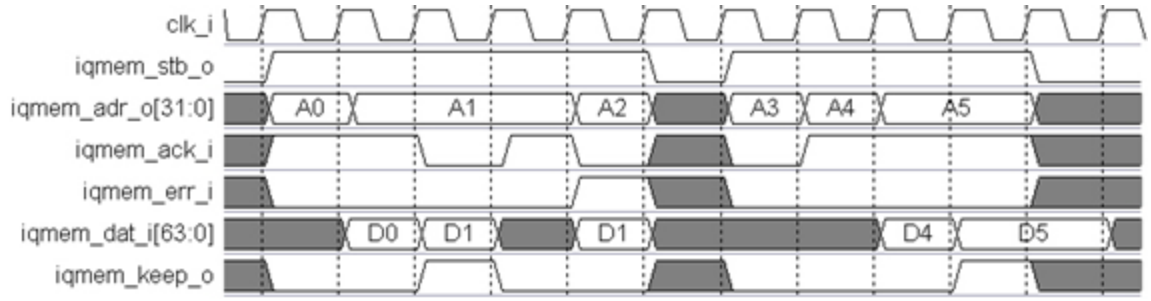

Figure 5: Instruction QMEM read cycle

5.4.1.2. DQMEM

Table 9 lists and describes the IO signals comprising the 32-bit QMEM data interface. All signals are synchronous to the *clk_i* signal; the interface is reset using the *rst_i* signal.

Table 9: DQMEM Signal Description

Name	Type	Polarity/Bus Size	Description
dqmem_stb_o	Out	High	Memory handshake request signal. It qualifies

Name	Type	Polarity/Bus Size	Description
			dqmem_adr_o, dqmem_dat_o, dqmem_sel_o, dqmem_we_o, dqmem_ack_i and *dqmem_err_i* signals.
dqmem_ack_i	In	High	Memory handshake response signal. When *dqmem_stb_o* and *dqmem_ack_i* are sampled asserted on the rising clock edge, *dqmem_dat_i* must be valid on the next rising clock edge.
dqmem_err_i	In	High	Memory handshake response signal. Memory subsystem indicates inability to provide data from memory location addressed by *dqmem_adr_o.*
dqmem_adr_o	Out	32	Memory address. Asserted simultaneous with *dqmem_stb_o* and *dqmem_ack_i.*
dqmem_dat_i	In	32	Memory data input bus. QMEM must provide data on this bus one rising clock edge after *dqmem_stb_o*and*dqmem_ack_i* were sampled asserted and *dqmem_we_o* was sampled de-asserted.
dqmem_dat_o	Out	32	Memory data output bus. Qualifed with simultaneous assertion of *dqmem_stb_o* and *dqmem_we_o*. Each data bus byte additionally qualified with *dqmem_sel_o*. Considered written when *dqmem_ack_i* is also sampled asserted.
dqmem_we_o	Out	High	Write enable signal. When *dqmem_stb_o* is asserted, indicates write (1) or read (0) cycle.
dqmem_sel_o	Out	4	Byte select bus. Signals where valid data is placed on *dqmem_dat_o* during write cycles. Signals where valid data should be returned on *dqmem_dat_i* during read cycles.

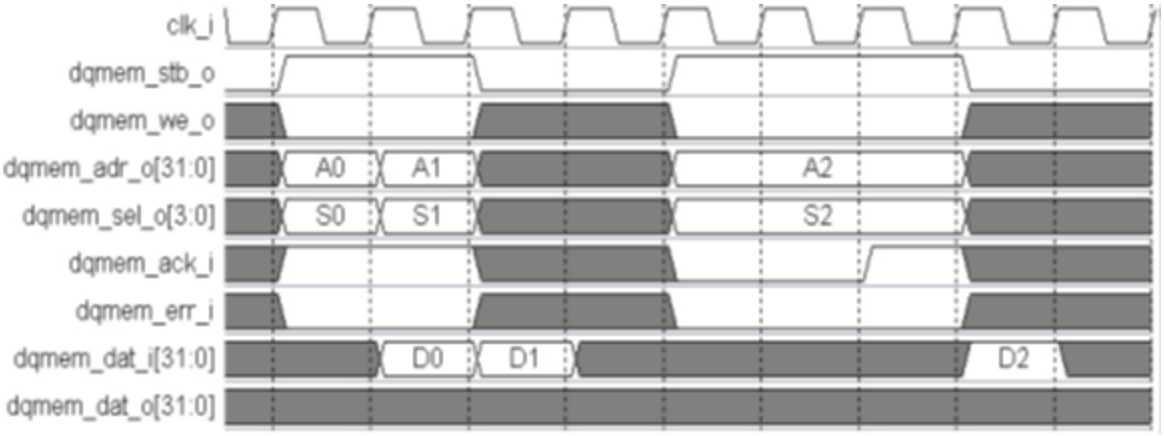

Figure 6: Data QMEM read cycle

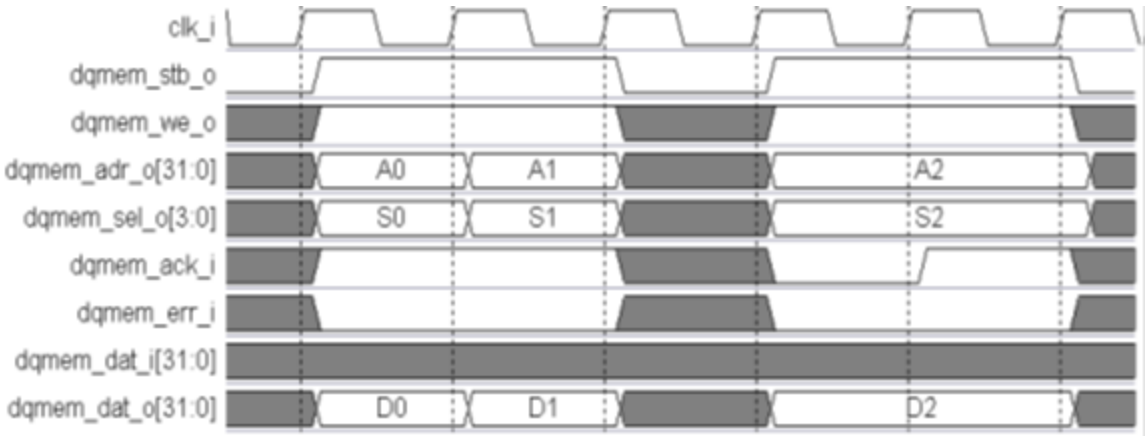

Figure 7: Data QMEM write cycle

5.4.2 Cache Memories

The BA22 can have implemented up to four ways of L0 data and instruction cache modules, each with a maximum size of 32Kbytes. The associativity together with the cache and block sizes is parameterized and should be set up before synthesis. More information about the cache parameters can be found in the "BA22 Integration Manual" document.

5.4.2.1. Registers

The BA22 Cache Module is controlled by three registers: Data Cache Block Flush, Data Cache Block Invalidate and Instruction Cache Block Invalidate. The BA22 does not support write back caches, so Data Cache Block Flush is equivalent to Data Cache Block Invalidate. For compatibility with future processors it is recommended to always assume caches may be write back.

Table 10: TTCR Data Cache Block InvalidateRegister

Bit	Symbol	Description	Type	Reset Value
0	IH	Invalidate if Hit 0 – Invalidate cache block 1 – Invalidate cache block only if EA hits in the cache	W	0
1-31	EA	Effective Address	W	0

Table 11: Data Cache Block Flush Register

Bit	Symbol	Description	Type	Reset Value
0	IH	Invalidate and write back if Hit 0 – Invalidate and write back cache block 1 – Invalidate and write back cache block only if EA hits in the cache	W	0
1-31	EA	Effective Address	W	0

Table 12: Instruction Cache Block Invalidate Register

Bit	Symbol	Description	Type	Reset Value
0	IH	Invalidate if Hit 0 – Invalidate cache block 1 – Invalidate cache block only if EA hits in the cache	W	0
1-31	EA	Effective Address	W	0

5.4.2.2. Cache Memories Interfaces

The BA22 uses both dual and single port synchronous RAM blocks for the implementation of the caches memories. Table 13lists the memory blocks and the number of ports for each of them.

Table 13: Cache Memory Modules

Name	Description	Number of Ports
dc_ram_dat	Data Cache Data Memory	1
dc_ram_tag	Data Cache Tag Memory	1
dc_ram_lru	Data Cache LRU Memory	2
ic_ram_dat	Instruction Cache Data Memory	1
ic_ram_tag	Instruction Cache Tag Memory	1
ic_ram_lru	Instruction Cache LRU Memory	2

All memory blocks are assumed to be synchronous RAMs that are read/written synchronously to the clk_i. All dual ported memory blocks need one read and one write port (ports do not have to be read/write).

The core interfaces all the single ported memories with the signals reported inTable 14.

Table 14: Interface Signals for Single-Port Memories

Name	Type	Polarity/ Bus Size	Description
<name>_cyc_o	In	High	Module select
<name>_we_o	In	High	Write enable. When <*name*>*_we_o* is asserted, indicates write (1) or read (0) cycle
<name>_sel_o	In	SP[1]	Active high byte select.
<name>_adr_o	In	SP[1]	Address bus
<name>_dat_o	In	SP[1]	Write data.
<name>_dat_i	Out	SP[1]	Read data.

The core interfaces all the dual ported memories with the signals reported in Table 15.

Table 15: Interface Signals for Dual-Port Memories

Name	Type	Polarity/ Bus Size	Description
<name>_wcyc_o	In	High	Write enable
<name>_rcyc_o	In	High	Read enable
<name>_wadr_o	In	SP[1]	Write address
<name>_radr_o	In	SP[1]	Read address
<name>_dat_o	In	SP[1]	Write data
<name>_dat_i	Out	SP[1]	Read data

5.4.3 Memory Management Unit

The BA22 processor can have implemented up to four ways of instruction and data memory management unit translation lookaside buffers (TLBs). Those buffers are used to keep recently used page address translation on chip.

The details of the MMU implementation can be specified by used the pre-synthesis parameters. More information about the MMU pre-synthesis settings can be found in the BA22 Implementation Manual documentation.

To complete any memory access, the effective address must be translated to a physical address. An MMU exception occurs if this translation fails (because TLBs do not hold the relevant translation). More details about the Memory Management exceptions can be found in the “BA22 Programmer’s Manual” document.

There are six 32-bit registers that the operating system uses to program the MMU. Access to the MMU registers is allowed only when the corresponding MMU is disabled, either because of an exception or by having its Data TLB Miss Exception (DME) or Instruction TLB Miss Exception (IME) bits explicitly cleared in the Supervision Register (Group 0). Access to MMU registers is undefined while the corresponding MMU is enabled. The MMU registers are described in detail in the ”BA22 Programmer’s Manual” document.

5.4.3.1. TLB Memories Interfaces

The BA22 uses both dual and single port synchronous RAM blocks for the implementation of the MMU memories. Table 13 lists the memory blocks and the number of ports for each of them.

Table 16: MMU Memory Modules

Name	Description	Number of Ports
dtlb_ram_dat	Data MMU Data Memory	1
dtlb_ram_lru	Data MMU LRU Memory	2
itlb_ram_dat	Instruction MMU Data Memory	1
itlb_ram_lru	Instruction MMU LRU Memory	2

All memory blocks are assumed to be synchronous RAMs that are read/written in a synchronous manner. All dual ported memory blocks need one read and one write port (ports do not have to be read/write).

The core interfaces all the single ported memories with the signals reported in Table 13. The core interfaces all the dual ported memories with the signals reported in Table 14.

5.5 Power Management Unit

The Power Management Unit (PMU) provides a flexible handshaking interface to the clock management circuitry, to enable clock gating of individual units and frequency scaling depending on the workload.

5.5.1 Interface

The clock management asserts (high) *pm_stall_i*signal anytime it wants to change the CPU's power state. It has to wait for the PMU to assert its *pm_stalled_o*signal before changing the clock mode and/or putting the CPU to power down mode. While *pm_stall_i*and*pm_stalled_o*are both asserted, the external clock management can change clock speeds and ratios or gate the clocks off. If enabled, the PMU will assert its *pm_event_o*signal in response to internal tick timer and/or programmable interrupt controller interrupt assertion – more details can be found in the Power Management Event Control Register – PMEVNT section. This can be used to wake the CPU from gated clocks mode.

The *pm_clmode_i* signal can be only changed when *pm_stalled_o* is asserted; in other words, the CPU acknowledged the power down mode. The *pm_clmode_i* sets the ratio between *clk_i* (main CPU clock) and the clocks for the AHB (Wishbone) peripheral bus. All handshaking signals are synchronous to the *pm_clk_i* signal.

Table 17: PMU Signal Description

Name	Type	Polarity / Bus Size	Description
pm_clk_i	In	-	Clock signal used to run BA22 and internal peripherals in the power down mode
pm_stall_i	In	High	Request signal to change the CPU power state
pm_stalled_o	Out	High	Handshaking signal that confirms CPU readiness for the power down mode
pm_clmode_i[7..0]	In	8	Informs the processor about the AHB (Whisbone) to CPU clock ratio. *pm_clmode_i*[7]: 0 – AHB clock is faster, 1 CPU clock is faster *1/pm_clmode_i*[6:0]: the CPU to AHB clock ratio, when *pm_clmode_i*[7[=1, or the AHB to CPU clock ratio, when *pm_clmode_i*[7[=0 If *pm_clmode_i*[6:0]=0 AHB and CPU uses the same clock frequency
pm_event_o	Out	High	Request to change/finish the CPU power down mode. The *pm_event_o*signal can be generated by internal timer or PIC.
du_clmode_o	Out	32	Reserved; should be ignored
du_clk_i	In	-	Debug Unit clock
du_clk_en_o	Out	High	Debug Unit clock gating enable
rf_clk_i	In	-	Register File clock
rf_clk_en_o	Out	High	Register File clock gating enable

It is required that all clocks are turned-on and stable and that the *pm_clmode_i* value is appropriate, before de-asserting the *pm_stall_i* signal. The *pm_clmode_i* sets the ratio between the *clk_i*, the main CPU clock and the clocks for the AHB peripheral bus. All handshaking signals are synchronous to *pm_clk_i.* The *pm_clk_i* signal should always active and not changed in any mode.

The *rf_clk_en_o* signal is asserted when the CPU does not use the register file in any mode.

The *du_clk_en_o* signal can be programmed by setting up an appropriate bit in the Power Management Unit Power Down Register – PMUNITPD. It allows the firmware to power down the Debug Unit at the startup when necessary.

5.5.1.1. PMU Signal Relationship

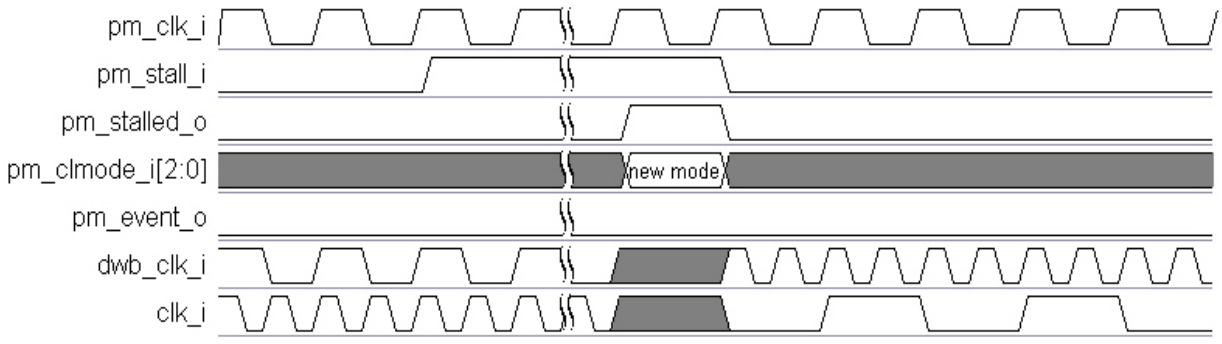

Figure 8: Power Management Clock Mode Change

The time between *pm_stall_*i and *pm_stalled_o* assertion varies depending on what instructions are present in the CPU's pipeline at the time *pm_stall_*i is asserted. Note that combinatorial path exists between *pm_stall_i* and *pm_stalled_o* in order to achieve timing in the diagram. It is recommended to take a special care of the *pm_stalled_o* signal by registering it using the *pm_clk_i* signal or a different custom logic should be implemented instead.

At the time *pm_stall_i* and *pm_stalled_o* are both asserted, *clk_i* and *dwb_clk_*i are marked as “don’t care” in the figure to represent that an arbitrary clock ratio is possible at this time. This does not imply clocks can become asynchronous or violate achieved minimum clock period – clock switching must be clean and glitch free.

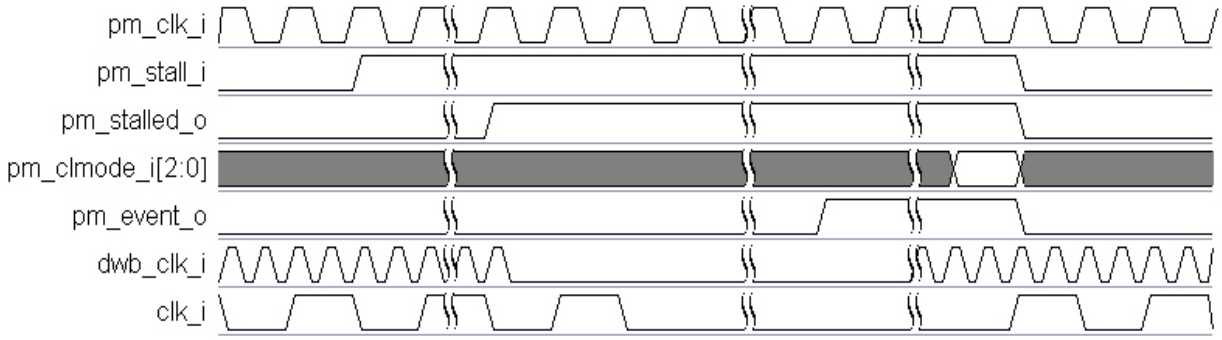

Figure 9:Power management Clock Gating

Figure 9 depicts a typical clock gating application. The external clock management circuit can turn clocks off while both *pm_stall_i*and *pm_stalled_o*are asserted. It may use *pm_event_o*signal to trigger the wakeup procedure or the wakeup procedure can be triggered automatically on other, external conditions.

The clock mode is also allowed to change during the wakeup procedure as clock gating and clock mode switch are functionally equivalent.

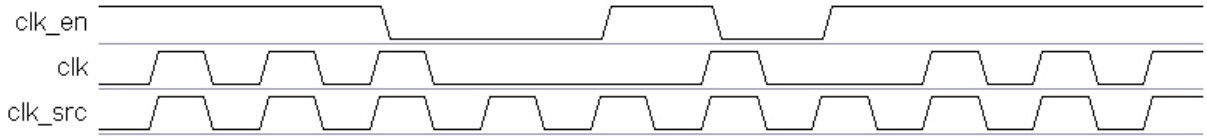

Figure 10: Clock Gating

Figure 10 shows the required relationship between source clock, gated clock and clock gating signal for applicable units (Register File, Debug Unit).

5.5.2 Registers

The Power Management SPRs start at offset 0x4000 Register implementations are optional – if a particular register is not implemented, then the functionality it provides is not available. Reads of an unimplemented register return undefined value and writes are ignored.

5.5.2.1. Power Management Event Control Register – PMEVNT

The Power Management Event Control Register is mapped to address 0x4000. The optional PMEVNT register is used to configure the behavior of the power management event output signal (*pm_stalled_o*) while the processor is stalled via power the power management IO interface.

Table 18: PMEVNT Register (0x4000)

Bit	Symbol	Description	Type	Reset Value
0	PMEPICEN	Tick Timer Power Management Event Enable 0 - Tick timer interrupt does not trigger powermanagement events 1 - Tick timer interrupt triggers power management events	R/W	0
1	PMETTEN	Programmable Interrupt Controller Power Management Event Enable	R/W	0

		0 - Programmable interrupt controller interrupt does not trigger power management events 1 - Programmable interrupt controller interrupt triggers power management events		
[31..2]	-	Reserved. Write with 0x0, reads undefined.	NA	-

5.5.2.2. Power Management Unit Power Down Register – PMUNITPD

The Power Management Unit Power Down Register is mapped to address 0x4002. The optional PMUNITPD register is used to control clock enable signals for individual units.

Table 19: PMUNITPD Register (0x4002)

Bit	Symbol	Description	Type	Reset Value
0	PMDUCLKEN	Debug Unit Clock Enable 0 - Debug unit clock enable output signal deasserted 1 - Debug unit clock enable output signal asserted	R/W	1
[31..1]	-	Reserved. Write data discarded, read data undefined	NA	-

5.5.3 Sample Configuration

Figure 11 illustrates the typical power management configuration for the BA22 processor.

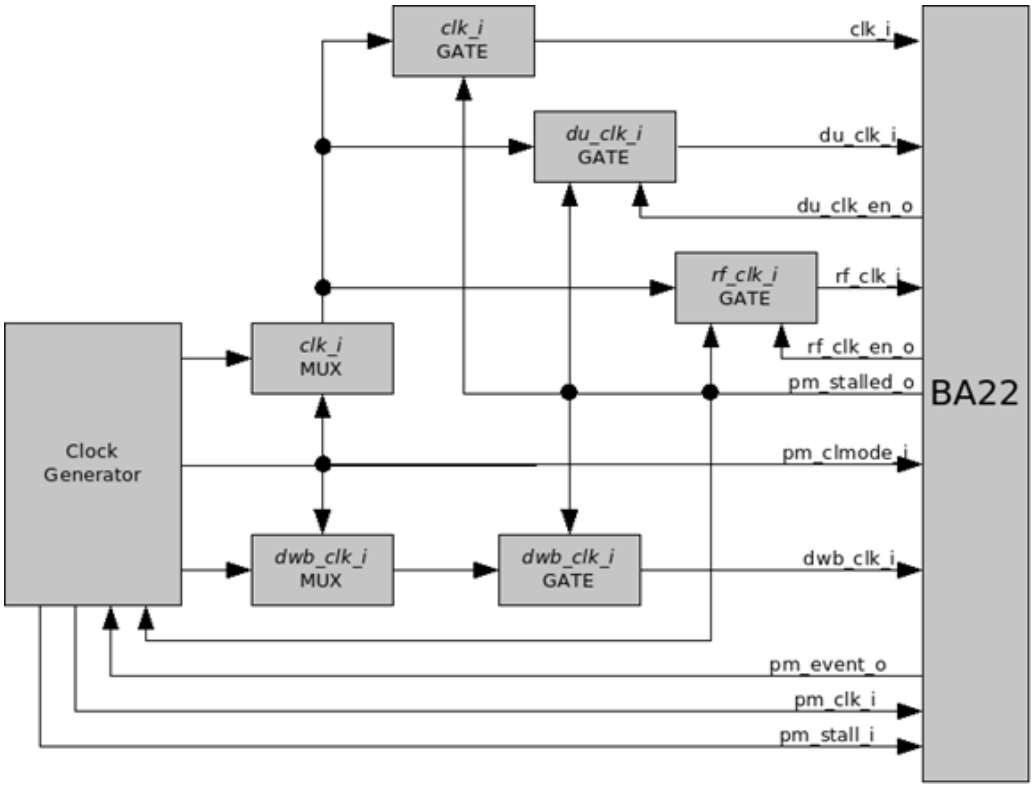

Figure 11: Sample PMU Configuration

5.6 Debug Unit

The optional Debug Unit assists software developers in debugging their systems. It provides support for watchpoints, breakpoints and program-flow control registers. The watchpoints and breakpoints may trigger events causing the debug interface to stall the processor or when a debug interface is not used, invoke the resident debugger software.

The main features of the debug units are as follows:

- Eight architecture defined sets of debug value/compare registers

- Match signed/unsigned conditions on instruction fetch Effective Address (EA), load/store EA and load/store data
- Match conditions on instruction fetch address can generate a breakpoint (trap exception). The program is interrupted before the instruction matching conditions is committed.
- Match conditions on load/store EA or load/store data or combining match conditions for complex watchpoints (trap exception). The program is interrupted/stalled after the instruction matching the condition is committed.

5.6.1 Interface

Table 20: Debug Interface – Signal Description

Signal	Type	Polarity/ Bus Size	Description
dbg_stall_i	In	High	Execution stop request to processor.
dbg_stb_i	In	High	Special purpose register data transfer handshake signal.
dbg_we_i	In	-	Indicates current SPR data transfer request direction. 1 – write (to processor), 0 – read (from processor). Qualified with *dbg_stb_i*.
dbg_adr_i	In	32	The SPR address for current data transfer request. Qualified with *dbg_stb_i*.
dbg_dat_i	In	32	SPR data transfer request write data (to processor). Qualified with *dbg_stb_i*.
dbg_dat_o	Out	32	SPR data transfer request read data (from processor). Qualified with *dbg_stb_i* and *dbg_ack_o*.
dbg_ack_o	Out	-	SPR data transfer handshake signal. When both this signal and *dbg_stb_i* are high, the SPR transfer is complete.
dbg_lss_o	Out	4	Load/Store Unit's status. Reserved, always 0x0.
dbg_is_o	Out	2	Instruction Fetch Unit's status. Reserved, always 0x0.
dbg_wp_o	Out	11	Watchpoints' status. Reserved, always 0x0.
dbg_bp_o	Out	High	Breakpoint hit status. Activated for one or more *clk_i* cycles when hardware breakpoint condition is detected. Should result in *dbg_stall_i* assertion if hardware handling of breakpoints is desired.
dbg_ewt_i	In	-	External Watchpoint trigger. Reserved, ignored. It's recommended to keep it Low.

5.6.2 Registers

Here, only brief information about the Debug Unit registers is presented. More details can be found in the "BA22 Programmer's Manual" document.

5.6.2.1. Debug Value Registers (DVR0-DVR7)

The Debug Value Registers are mapped to addresses 0x6000 to 0x6007. The 32-bit DVRs are programmed with the comparison addresses or data by the resident debug software or by the development interface.

Their value is compared to the fetch or load/store Effective Address (EA) or to the load/store data according to the corresponding Debug Control Registers. Based on the settings of the corresponding DCR a match condition is generated.

5.6.2.2. Debug Control Registers (DCR0-DCR7)

The Debug Control Registers are mapped to addresses 0x6008 to 0x6015. The 32-bit DCRs are programmed with the watchpoint settings that define how DVRs are compared to the instruction fetch or load/store Effective Address or to the load/store data.

Among others, the user can set the following compare conditions:

- Equal
- Less than
- Less than or equal
- Greater than
- Greater than or equal
- Not equal

The comparison can be executed using signed or unsigned integers.

5.6.2.3. Debug Mode Register 1 (DMR1)

The Debug Mode Register 1 is mapped to address 0x6016. The DMR1 is a 32-bit programmed with the watchpoint/breakpoint settings that define how DVR/DCR pairs operate and is set by the resident debug software or by the development interface.

By the appropriate settings, the user can set single or chained watchpoints depending on the needs.

5.6.2.4. Debug Mode Register 2 (DMR2)

The Debug Mode Register 2 is mapped to address 0x6017. The DMR2 32-bit register is programmed with the watchpoint/breakpoint settings that define which match conditions generate a breakpoint/watchpoint.

When a breakpoint/watchpoint happens Watchpoints Breakpoint Status (WBS) provides information which of several match conditions caused breakpoint/watchpoint condition.

5.6.2.5. Debug Stop Register (DSR)

The Debug Stop Register is mapped to address 0x6020. The DSR is a 32-bit register which specifies which exceptions cause the core to stop the execution of the exception handler and turn over control to development interface. It can be programmed by the resident debug software or by the development interface.

5.6.2.6. Debug Reason Register (DRR)

The Debug Reason Register is mapped to address 0x6021. The DRR is a 32-bit register which specifies which event caused the core to stop the execution of program flow and turned control over to the development interface. It should be cleared by the resident debug software or by the development interface.

5.7 Integrated Peripherals

5.7.1 Programmable Interrupt Controller

The Programmable Interrupt Controller (PIC) unit in the BA22 processor has a configurable number of interrupt lines, from 1 to 32. If more interrupts are required, the user may use shared interrupts or stack external interrupt controllers on the PIC unit.

The interrupt controller facility is optional and the developer may choose whether or not to include it. If the PIC is not implemented, assertion of any interrupt input on the *pic_int_i* bus causes the CPU to

execute the interrupt exception handler located at 0x800. Note that the Interrupt Exception is enabled via bit IEE of the Supervision Register – SR[IEE]. Refer to the programmers manual for details.

The BA22 PIC hardware unit has only a single interrupt priority, however, the interrupt enables are fully under software control, and hence the user may opt for an arbitrary number of interrupt priorities.

The latency from interrupt assertion to instruction fetch of the first instruction in the interrupt handler is usually 10 cycles, but certain dynamic effects like cache misses or memory latency can prolong it.

The BA22 architectural state is automatically stored in EPCR and ESR registers when the IRQ exception handler is to be executed.

5.7.1.1. Interface

Table 21: PIC Signal Description

Name	Type	Polarity / Bus Size	Description
pic_int_i	In	Possible range: 1 - 32	Interrupt interface.

5.7.1.2. Registers

The Programmable Interrupt Controller has two special-purpose registers to mask and read the status of the interrupt inputs.

5.7.1.2.1. PIC Mask Register - PICMR

The PIC Mask Register is mapped at address 0x4800. The PICMR register is a 32-bit special-purpose supervisor-level. The register is used to mask or unmask 32 programmable interrupt sources.

Table 22: PICMR Register (0x4800)

Bit	Symbol	Description	Type	Reset Value
31-0	IUM	Interrupt UnMask 0x00000000 – All interrupts are masked 0x00000001 – Interrupt input 0 is enabled, all others are masked 0xFFFFFFFF – All interrupt inputs are enabled	R/W	0x0

5.7.1.2.2. PIC Status Register – PICSR

The PIC Status Register is mapped at address 0x4802. The PICSR register is a 32-bit special-purpose supervisor-level register. This register is used to determine the status of each PIC interrupt input. PIC only supports level-triggered interrupts where bits in PICSR simply represent level of interrupt inputs.

Interrupts are cleared by taking an appropriate action at the device to negate the source of the interrupt.

The appropriate bit in the PICSR should be cleared by the SW in the exception handler.

Table 23: PICSR Register (0x4802)

Bit	Symbol	Description	Type	Reset Value
31-0	IS	Interrupt Status 0x00000000 – All interrupts are inactive 0x00000001 – Interrupt input 0 is pending 0xFFFFFFFF – All interrupt inputs are pending	R/W	0x0

5.7.2 Tick Timer

The optional 32-bit Tick Timer is used to schedule operating system and user tasks on a regular time basis or as a high precision time reference.The Tick Timer is triggered by *pm_clk_i* clock signal.

The main features of the tick timer are:

- Maximum timer count of 2^32 clock cycles

- Maximum time period of 2^28 clock cycles between interrupts
- Maskable tick timer interrupt
- Disabled, single run, restart able counter, or continues counter modes

The Tick Timer unit is enabled and configured by appropriate settings in Tick Timer Mode Register (TTMR). The Tick Timer Count Register (TTCR) is incremented with each clock cycle. A new timer period can be set by using the Tick Timer ReStart Register (TTRS)

5.7.2.1. Timer Interrupt

The timer interrupt occurs every time when the Interrupt Enable bit (TTMR reg.) is set and Time Period (TTMR reg.) matches the lower 28-bits of the TTCR register. The top 4 bits are ignored for the comparison.

The timer interrupt is asserted when the Interrupt Enable for Tick Timer interrupt is set - TTMR[IE], and also the Tick Timer Exception Enable bit is set (Supervision Register - SR[TEE]).

5.7.2.2. Tick Timer Modes

It should be ensured that the TTRS register is set to the appropriate value before setting any timer's mode.

Tick Timer can operate in one of the following four modes:

- **Disabled Timer** – in this mode, the timer does not increment the TTCR. It's recommended though to make sure that a permanent tick timer exception does not occur.

  Such a situation would happen when all of the following conditions are met:

  TTCR[27:0] = TTMR[27:0]; match condition has occurred

  TTMR[31:30] = 0x0; the timer is disabled

  TTMR[29] = 1; the timer interrupt is enabled

  SR[1] = 1; Tick Timer exception is enabled in Supervision Register

- **Auto-restart Timer** - when the timer is set to the auto-restart mode, the timer will reset the TTCR register to 0 as soon as the lower 28-bits of the timer (TTCR register) match the Time Period (TTMR register). The timer interrupt will be asserted if the Interrupt Enable is set in the TTMR register.
- **Single Run Timer** – in one-shot timing mode, the timer stops counting as soon as a match condition has been reached. A special care should be taken that the timer interrupt has been masked (or disabled) after the match condition has been reached. Otherwise the CPU core will get a spurious timer interrupt.
- **Continuous Timer** - In the event that a match condition has been reached, the counter does not stop but rather keeps counting from the value of the TTCR. The timer interrupt will be asserted if the Interrupt Enable bit has been set.

5.7.2.3. Tick Timer Registers

The timer unit contains three registers which are used to control and hold the timer's value.

5.7.2.3.1. Tick Timer Mode Register - TTMR

The Tick Time Mode Register is mapped at address 0x5000. The TTMR is programmed with the Time Period of the tick timer as well as with the mode bits that control operation of the tick timer. The register is also used to configure power management event output signal behavior while the processor is stalled via the power management IO interface.

Table 24: TTMR Register (0x5000)

Bit	Symbol	Description	Type	Reset Value
0 -27	Time Period	Time Period	R/W	0

	(TP)	0x0000000 – shortest comparison time period 0xFFFFFFF – longest comparison time period		
28	Interrupt Pending (IP)	0 – Tick timer irq is not pending 1 – Tick timer irq pending The IP bit needs to be cleared by SW when the Timer exception is entered or exited.	R/W	0
29	Interrupt Enable (IE)	0 – Tick timer does not generate irq 1 – Tick timer generates irq when Time Period matches TTCR[27:0]	R/W	0
30-31	Mode(M)	00 – Tick Timer is disabled 01 – Auto-restart mode. The timer is restarted when Time Period matches TTCR[27:0] 10 - Single Run mode. The timer stops when Time Period matches TTCR[27:0]. The TTCR value should be changed to resume counting. 11 – Continuous Run. The timer does not stop when Time Period matches TTCR[27:0]	R/W	0x0

5.7.2.3.2. Tick Timer Count Register – TTCR

The tick timer count register is mapped to address 0x5001. The TTCR is a 32-bit special-purpose register which holds the current value of the timer.

Table 25: TTCR Register (0x5001)

Bit	Symbol	Description	Type	Reset Value
0 -31	Count (CNT)	32-bit incrementing counter	R/W	0

5.7.2.3.3. Tick Timer ReStart Register – TTRS

The tick timer count register is mapped to address 0x5002. Accesses to the TTRS register influence other timer's registers. When the TTRS is read, it clears the Interrupt Pending bit in the TTMR register. The TTRS writing access should be used only when the timer is set in the Singe Run mode. When TTRS[0:27] is written, it sets a new period in the TTMR register (a new value is being assigned as Time Period) and resets the TTCR to 0.

When the mode is not set to Single Run, the result of a write to TTRS is undefined and should be avoided

Table 26: TTRS Register (0x5002)

Bit	Symbol	Description	Type	Reset Value
0 -27	Period	Period	R/W	0
28-31	-	Ignored	-	-