ARM RCE Generation 3 Core Module
Design Document

Ryan Herbst

April 2, 2014

Revision	Effective Date	Description Of Changes
1.2	April 2, 2014	Updates for local bus to AXI-Lite conversion. PPI interface changes. Some document sections removed.
1.1	November 14, 2013	Added clock select registers.
1.0	November 8, 2013	Cleanup.
0.3	November 5, 2013	Changed location of outbound free list FIFOs.
0.2	October 24, 2013	Cleaup, additional functional descriptions. IB/OB descriptor updates.
0.1	October 22, 2013	Initial revision

SLAC National Accelerator Center Research Engineering Division, Electronics

Contents

List of Tables

List of Figures

1 External Interfaces

This section of the document describes the external interfaces of the ARM RCE generation 3 core module. See table 14 in section 2.1 for a detailed list of the top level interface signals.

1.1 External Clock & Reset

The following table defines the clock and reset signals output from the ARM RCE generation 3 core module.

Signal	Description
axiClk	AXI Bus clock. Nominal 125Mhz. Used to clock AXI-Lite bus signals.
axiClkRst	Reset strobe synchronous to axiClk.
sysClk125	125Mhz system clock.
sysClk125Rst	Reset strobe synchronous to sysClk125.
sysClk200	200Mhz system clock.
sysClk200Rst	Reset strobe synchronous to sysClk200.

Table 1: External Clock & Reset Signals

1.2 External AXI-Lite Bus

Contained within the RCE core module is a AXI to to AXI-Lite bridge that facilitates register read and write access within the core. This controller described in section 2.2 of this document allocates a portion (0xA0000000 - 0xBFFFFFFF) of the AXI-Lite address space to an external AXI-Lite bus. The read and write portions of this bus operate independently, allowing simultaneous read and write access. Support logic for interfacing to this bus can be found in the axi sub-section of the SLAC RED Electronics standard VHDL library. Four record types, 2 for write and 2 for read, make up the AXI-Lite bus. For more information about Axi-Lite bus transactions, please see AMBA AXI Protocol Specification.

The first record type, AxiLiteReadMaster is an output containing the following signals:

Signal	Width	Description
araddr	32	Read address vector.
arprot	2	Read protection value. (usually not used).
arvalid	1	Asserted when araddr and arprot are valid.
rready	1	Asserted when master is ready to accept read data.

Table 2: AXI Lite Read Master Record

The second record type, AxiLiteReadSlave is an input containing the following signals:

Signal	Width	Description
arready	1	Asserted when slave is ready to accept a read transaction.
rdata	32	Read data vector.
rresp	2	Read response vector. Indicates status of read transaction.
rvalid	1	Asserted when read data and response are valid.

Table 3: AXI Lite Read Master Record

The third record type, AxiLiteWriteMaster is an output containing the following signals:

Signal	Width	Description
awaddr	32	Write address vector.
awprot	2	Write protection value. (usually not used).
awvalid	1	Asserted when awaddr and awprot are valid.
wdata	32	Write data vector.
wstrb	4	Write data byte enable vector.
wvalid	1	Asserted when wdata and wstrb are valid.
bready	1	Asserted when master is ready to accept write status.

Table 4: AXI Lite Write Master Record

The second record type, AxiLiteWriteSlave is an input containing the following signals:

Signal	Width	Description
awready	1	Asserted when slave is ready to accept a write transaction.
wready	1	Asserted when slave is ready to accept write data.
bresp	2	Write response vector. Indicates status of write transaction.
bvalid	1	Asserted when response is valid.

Table 5: AXI Lite Write Master Record

1.3 BSI I2C

The BSI I2C interface connects the external I2C pins to the I2C slave contained within the core module. This interface is defined as two standard logic signals (i2cSda and i2cScl) which must be connected directly to external FPGA IO pins and must be defined as inout types.

1.4 Protocol Plug In (PPI)

The protocol plug in interface (PPI) supports 4 bi-directional FIFO like interfaces for transmitting and receiving data. Each PPI frame consists of a header and an optional payload. In the receive direction the header is separated from the payload and passed to the inbound header FIFO module (see section 2.7). The payload, if present, is then processed in the inbound PPI module (see section 2.8).

In the transmit direction the outbound header FIFO module (see section 2.11) generates the header and sends it out the PPI interface. If the frame contains a payload the outbound PPI module (see section 2.12) will add the payload to the end of the transmitted frame.

More information about the protocol plug in (PPI) operation can be found in *The Reconfigurable Cluster Element User Guide*.

1.4.1 Common Signals

Each of the 4 interfaces have a seperate clock and online control signal as shown in the following table:

Signal	Width	Direction	Description
ppiClk	1	Input	PPI Clock.
online	1	Output	Online control. Asserted when the PPI interface is in the online state.

Table 6: Common PPI Signals

1.4.2 Inbound Protocol Plug In (PPI)

The inbound PPI interface connects the external logic to the Inbound PPI module defined in section 2.8.

Each of the 4 interfaces is implemented using two record types. The first record type, PpiWriteFromFifoType, is an output providing inbound flow control:

Signal	Width	Description
pause	1	Pause indication. Asserted when the inbound PPI can not longer accept a complete frame.

Table 7: Inbound PPI Output Record

The inbound PPI engine will assert the pause signal when either the input header or payload FIFOs reaches a higher water mark. The current impelentation asserts flow control when the input header FIFO or input PPI payload FIFO have less than 255 (out of 512) 64-bit entries available.

The second record type, PpiWriteToFifoType, is an input which accepts inbound PPI data:

Signal	Width	Description
data	64	Data
size	3	Indicates how many bytes are valid when eof = 1. 0x0 = 1 byte, 0x7 = 8 bytes.
eof	1	End of frame indication. Asserted coincident with the last word of frame.
eoh	1	End of header. Asserted coincident with the last word of the header portion of frame.
err	1	Frame is in error. Asserted with eof.
ftype	4	Frame type
valid	1	Frame data is valid

Table 8: Inbound PPI Input Record

1.4.3 Outbound Protocol Plug In (PPI)

The outbound PPI interface connects the external logic to the Outbound PPI module defined in section 2.12.

Each of the 4 interfaces is implemented using two record types. The first record type, PpiReadFromFifoType, is an output which provides outbound PPI data:

Signal	Width	Description
data	64	Data
size	3	Indicates how many bytes are valid when eof = 1. 0x0 = 1 byte, 0x7 = 8 bytes.
eof	1	End of frame indication. Asserted coincident with the last word of frame.
ftype	4	Frame type
valid	1	Frame data is valid
ready	1	Asserted when the FIFO contains 1 frame or at least PPI_READY_THOLD_G quad words.

Table 9: Outbound PPI Input Record

The second record type, PpiReadToFifoType, is an input providing a read strobe:

Signal	Width	Description
read	1	Read data at PPI interface. Asserting this signal advances FIFO.

Table 10: Outbound PPI Input Record

1.5 Ethernet Interface

The Ethernet interface provide direct access to the two ethernet interfaces defined in the processor_system7_v4_02a core provided from Xilinx.

Each of the 2 interfaces is implemented using two record types. The first record type, EthFromArmType, is an output containing the following signals:

Signal	Width	Description
enetGmiiTxEn	1	
enetGmiiTxEr	1	
enetMdioMdc	1	
enetMdioO	1	
enetMdioT	1	
enetPtpDelayReqRx	1	
enetPtpDelayReqTx	1	
enetPtpPDelayReqRx	1	
enetPtpPDelayReqTx	1	
enetPtpPDelayRespRx	1	
enetPtpPDelayRespTx	1	
enetPtpSyncFrameRx	1	
enetPtpSyncFrameTx	1	
enetSofRx	1	
enetSofTx	1	
enetGmiiTxD	8	

Table 11: Ethernet Output Record

The second record type, EthToArmType, is an input and contains the following signals:

Signal	Width	Description
enetGmiiCol	1	
enetGmiiCrs	1	
enetGmiiRxClk	1	
enetGmiiRxDv	1	
enetGmiiRxEr	1	
enetGmiiTxClk	1	
enetMdioI	1	
enetExtInitN	1	
enetGmiiRxd	8	

Table 12: Ethernet Input Record

Refer to the Xilinx processor_system7_v4_02a documentation and Zynq-7000 technical reference manual (UG585) for further information.

2 VHDL Module Descriptions

This section of the document describes the VHDL modules which make up the ARM RCE Generation 3 core. Descriptions of additional VHDL modules from the RED Electronics *Standard VHDL Library* used within this core can be found at:

`https://confluence.slac.stanford.edu/display/ppareg/Standard+VHDL+Library`.

2.1 Top Level Module (ArmRceG3Top.vhd)

The top level module serves as the interface to the RCE generation 3 core module.

2.1.1 Top Level Interfaces

The generic ports for the top level module are shown in table 13.

Value	Type	Default	Description
TPD_G	time	1 ns	Synchronous signal delay value for simulation.
AXI_CLKDIV_G	real	5.0	AXI bus clock divider. Clock rate = 1000Mhz / value. Target clock rate is 200Mhz.
PPI_READY_THOLD_G	IntegerArray	0	Indicates the number of quad words that should be in the output FIFO before asserting ready. Set to zero to only assert ready when a frame is present.

Table 13: ArmRceG3Top Generics

The signal ports for the top level module are shown in table 14.

Signal	Type	Width	Direction	Description
i2cSda	Logic	1	inout	BSI I2C slave data
i2cScl	Logic	1	inout	BSI I2C slave clock
sysClk125	Logic	1	Out	125Mhz sytem clock
sysClk125Rst	Logic	1	Out	Reset for 125Mhz sytem clock
sysClk200	Logic	1	Out	200Mhz sytem clock
sysClk200Rst	Logic	1	Out	Reset for 200Mhz sytem clock
axiClk	Logic	1	Out	Clock for AXI buss
axiClkRst	Logic	1	Out	Reset for AXI buss
localAxiReadMaster	AxiLiteReadMasterType	1	Out	AXI-Lite read master signals.
localAxiReadSlave	AxiLiteReadSlaveType	1	In	AXI-Lite read slave signals.
localAxiWriteMaster	AxiLiteWriteMasterType	1	Out	AXI-Lite write master signals.
localAxiWriteSlave	AxiLiteWriteSlaveType	1	In	AXI-Lite write slave signals.
ppiClk	Logic	4	In	PPI clock inputs
ppiOnline	Logic	4	Out	PPI online outputs
ppiReadToFifo	PpiReadToFifoType	4	In	Outbound PPI input signals
ppiReadFromFifo	PpiReadFromFifoType	4	Out	Outbound PPI output signals
ppiWriteToFifo	PpiWriteToFifoType	4	In	Inbound PPI input signals
ppiWriteFromFifo	PpiWriteFromFifoType	4	Out	Inbound PPI output signals
ethFromArm	EthFromArmType	2	Out	Ethernet port outputs
ethToArm	EthToArmType	2	In	Ethernet port inputs
clkSelA	Logic	2	Out	Clock select A bits
clkSelB	Logic	2	Out	Clock select B bits

Table 14: ArmRceG3Top Signals

2.1.2 Top Level Block Diagram

The block diagram of the RCE generation 3 core module is shown in figure 1. The following sub modules exist within the core module and are described in greater detail later in this document:

- ArmRceG3LocalAxi: AXI to AXI-Lite bus bridge, cross connect and top level registers (section 2.2)
- ArmRceG3Clock: Clock generation module (section 2.3)
- ArmRceG3DmaCntrl: DMA controller for PPI interfaces and BSI messages (section 2.4)
- ArmRceG3I2c: BSI I2C module (section 2.15)
- ArmRceG3Cpu: ARM CPU wrapper (section 2.16)

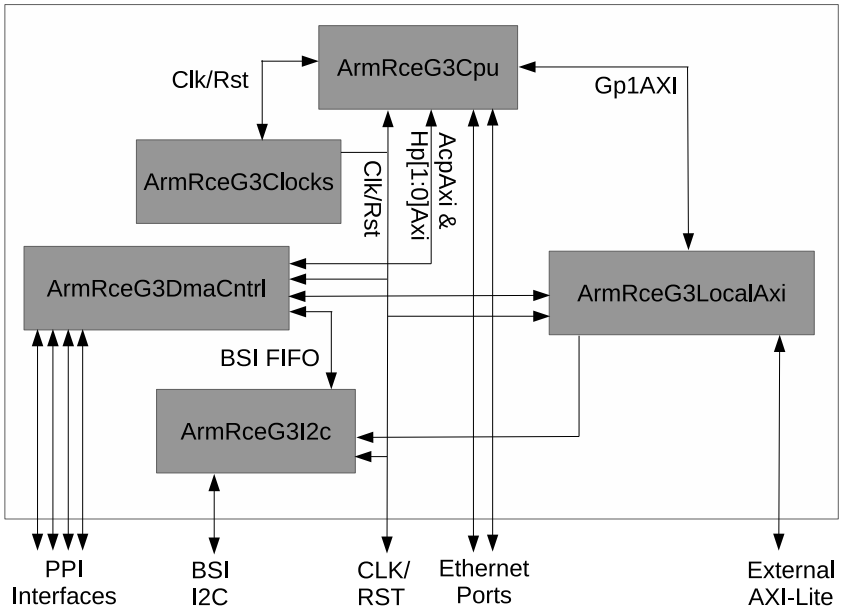

Figure 1: Top Level Block Diagram

2.2 Local AXI Controller (ArmRceG3LocalAxi.vhd)

This module implements a bridge between the general purpose AXI master port (GP1) and a number of internal AXI-Lite busses. The local AXI controller supports single dual word (32-bit) accesses.

2.2.1 RCE Core Address Space

The overall address map for the RCE core is shown in table 15.

Address Range	Assignment
0x8000_0000 - 0x8000_FFFF	Top Level Registers
0x8400_0000 - 0x8400_0FFF	BSI I2C Slave Registers
0x8800_0000 - 0x8800_0FFF	DMA Controller Registers
0x8800_1000 - 0x8800_10FF	DMA Controller Completion FIFOs
0x8800_1100 - 0x8800_11FF	DMA Controller Free List FIFOs
0xA000_0000 - 0xBFFF_FFFF	External Address Space

Table 15: RCE Core Address Space

2.2.2 Top Level Address Map

The local AXI controller module contains a handful of registers as shown in the following table.

The Version value is unique to each target FPGA design. While the format of this version value is not defined it is typical for the upper 12 bits to be unique for each target FPGA type while the lower 20 bits are incremented with each compile.

The ArmRceG3Version value is defined in the core module and is incremented any time any of the major functions within the core module are modified.

The BuildString value is a 256 character NULL terminated string which contains information about the user who built the image and the timestamp of when the image was built. This field is automatically updated by the build script at each compile.

Address	Bits	Mode	Name	Description
0x80000000	31:0	Read	Version	FPGA version value. Set in Version.vhd.
0x80000004	31:0	R/W	Scratchpad	Scratchpad register.
0x80000008	31:0	Read	ArmRceG3Version	ARM RCE Gen3 Module Version.
0x80000010	1:0	R/W	ClkSel0	Reference Clock 0 Frequency Select.
0x80000014	1:0	R/W	ClkSel1	Reference Clock 1 Frequency Select.
0x80000020	31 23:0	Read Read	DNA Valid DNA Value 56:32	Device DMA value is valid. Bits 56:32 of Xilinx device DNA value.
0x80000024	31:0	Read	DNA Vlaue 31:00	bits 31:00 of Xilinx device DNA value.
0x80001000 - 0x800010FF	31:0	Read	BuildString	NULL termination build user and timestamp string.

Table 16: Top Level Address Map

The AXI-Lite busses inside the RCE core module operate at 200Mhz while the external AXI-Lite bus operates at 125Mhz.

2.3 Clock Generation Module (ArmRceG3Clocks.vhd)

The clock generation module generates the set of clocks required both internal and external to the ARM RCE Gen 3 Core module. The clocks in this module can be derived from any of the four function clock outputs from the processor core. Currently all clocks are derived from function clock 0 which is required to be configured as 100Mhz.

The following clock and reset siganls are generated within the clock generation module.

Signal	Description
dmaClk	DMA AXI bus clock output, 200Mhz
dmaClkRst	DMA AXI bus clock reset
sysClk125	System 125Mhz clock
sysClk125Rst	System 125Mhz clock reset
sysClk200	System 200Mhz clock
sysClk200Rst	System 200Mhz clock reset

Table 17: ArmRceG3Clocks Clocks

2.4 DMA Controller (ArmRceG3DmaCntrl.vhd)

The DMA controller module is a container for the logic modules which implement the protocol plug in (PPI) interfaces as well as the BSI management interface.

2.4.1 DMA Controller Block Diagram

The block diagram of the DMA controller module is shown in figure 2. The following sub modules exist within the module and are described in greater detail later in this document:

- ArmRceG3IbCntrl: Container for inbound FIFO logic modules (section 2.5)
- ArmRceG3IbPpi: Inbound PPI controller module (section 2.8)
- ArmRceG3ObCntrl: Container for outbound FIFO logic modules (section 2.10)
- ArmRceG3ObPpi: Outbound PPI controller module (section 2.12)
- ArmRceG3DmaComp: DMA completion FIFO container module (section 2.14)

The DMA controller module also contains a number of local configuration and status registers.

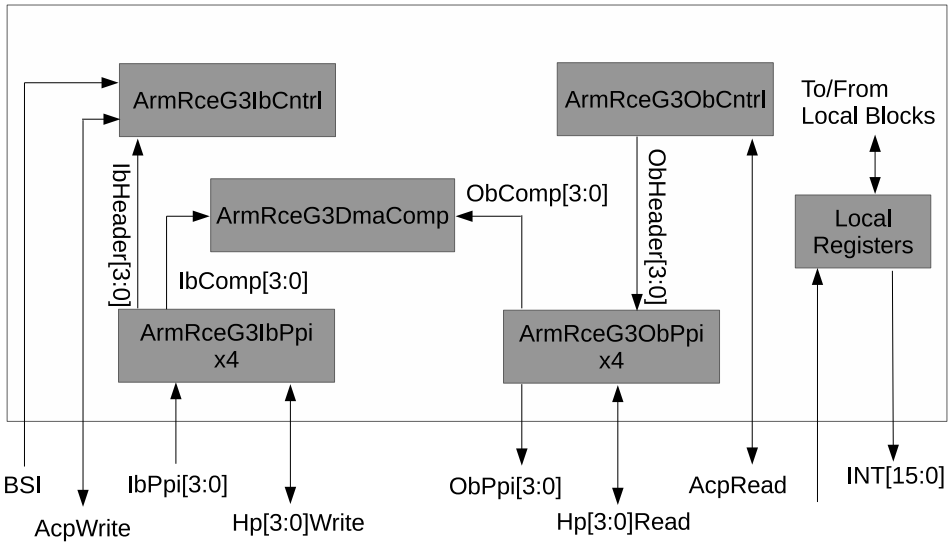

Figure 2: DMA Controller Block Diagram

2.4.2 DMA Controller Address Map

The DMA controller module contains the register read/write logic for all of it's sub modules. The resulting address map is shown below.

[H]

Address	Bits	Mode	Name	Description
0x88000000 - 0x880000FC	NA	Read	Unused	Read as zero
0x88000100 - 0x8800013C	31:0	Write	InboundHeader0Free	Inbound header 0, free list FIFO FIFO bits 35:32 = Address bits 5:2
0x88000140 - 0x8800017C	31:0	Write	InboundHeader1Free	Inbound header 1, free list FIFO FIFO bits 35:32 = Address bits 5:2
0x88000180 - 0x880001BC	31:0	Write	InboundHeader2Free	Inbound header 2, free list FIFO FIFO bits 35:32 = Address bits 5:2
0x880001C0 - 0x880001FC	31:0	Write	InboundHeader3Free	Inbound header 3, free list FIFO FIFO bits 35:32 = Address bits 5:2
0x88000200 - 0x8800023C	31:0	Write	OutboundHeader0Tx	Outbound header 0, transmit FIFO FIFO bits 35:32 = Address bits 5:2

Continued on next page

Continued from previous page

Address	Bits	Mode	Name	Description
0x88000240 - 0x8800027C	31:0	Write	OutboundHeader1Tx	Outbound header 1, transmit FIFO FIFO bits 35:32 = Address bits 5:2
0x88000280 - 0x880002BC	31:0	Write	OutboundHeader2Tx	Outbound header 2, transmit FIFO FIFO bits 35:32 = Address bits 5:2
0x880002C0 - 0x880002FC	31:0	Write	OutboundHeader3Tx	Outbound header 3, transmit FIFO FIFO bits 35:32 = Address bits 5:2
0x88000300	NA	Write	MemChan0DirtyClear	Write to clear memory channel 0
0x88000304 ...	NA	Write	MemChan1DirtyClear	Write to clear memory channel 1
0x88000310	NA	Write	MemChan4DirtyClear	Write to clear memory channel 4
0x88000314 - 0x880003FC	NA	Read	Unused	Read as zero
0x88000400	3:0 4 15:5	Read Read Read	IbDirtyStatus BsiDirtyStatus CompFifoStatus	Inbound memory dirty, one bit per channel BSI memory dirty, one bit per channel Completion FIFO ready, one bit per channel
0x88000404	15:0	R/W	InterruptEnable	Interrupt enable, one bit per interrupt
0x88000408	3:0	R/W	HeaderWriteDmaCache	Header AXI write cache configuration
0x8800040C	3:0	R/W	HeaderReadDmaCache	Header AXI read cache configuration
0x88000410	3:0 4 8:5	R/W R/W R/W	IbFifoEnable BsiFifoEnable ObFifoEnable	Inbound header enables BSI FIFO enable Outbound header enables
0x88000418	31:18	R/W	MemBaseAddress	Memory base address
0x8800041C	3:0	R/W	PpiReadDmaCache	PPI AXI read cache configuration
0x88000420	3:0	R/W	PpiWriteDmaCache	PPI AXI write cache configuration
0x88000424	3:0 7:4	R/W R/W	PpiIbOnline[3:0] PpiObOnline[3:0]	Inbound PPI online configuration Outbound PPI online configuration
0x88000428 - 0x880004FC	NA	Read	Unused	Read as zero
0x88000500 - 0x8800053C	31:0	Write	InboundPpi0Control	Inbound PPI 0 Control FIFO FIFO bits 35:32 = Address bits 5:2
0x88000540 - 0x8800057C	31:0	Write	InboundPpi1Control	Inbound PPI 1 Control FIFO FIFO bits 35:32 = Address bits 5:2
0x88000580 - 0x880005BC	31:0	Write	InboundPpi2Control	Inbound PPI 2 Control FIFO FIFO bits 35:32 = Address bits 5:2
0x880005C0 - 0x880005FC	31:0	Write	InboundPpi3Control	Inbound PPI 3 Control FIFO FIFO bits 35:32 = Address bits 5:2
0x88000600	31:0	Read	IbPpi0Id	Inbound PPI 0 ID value
0x88000604	31:0	Read	IbPpi0Id	Inbound PPI 0 version value
0x88000608	31:0	Read	IbPpi0ConfigA	Inbound PPI 0 config word A
0x8800060C ...	31:0	Read	IbPpi0ConfigB	Inbound PPI 0 config word B
0x88000630	31:0	Read	IbPpi3Id	Inbound PPI 3 ID value
0x88000634	31:0	Read	IbPpi3Id	Inbound PPI 3 version value
0x88000638	31:0	Read	IbPpi3ConfigA	Inbound PPI 3 config word A
0x8800063C	31:0	Read	IbPpi3ConfigB	Inbound PPI 3 config word B
0x88000640	31:0	Read	ObPpi0Id	Outbound PPI 0 ID value
0x88000644	31:0	Read	ObPpi0Id	Outbound PPI 0 version value
0x88000648	31:0	Read	ObPpi0ConfigA	Outbound PPI 0 config word A
0x8800064C ...	31:0	Read	ObPpi0ConfigB	Outbound PPI 0 config word B
0x88000670	31:0	Read	ObPpi3Id	Outbound PPI 3 ID value
0x88000674	31:0	Read	ObPpi3Id	Outbound PPI 3 version value
0x88000678	31:0	Read	ObPpi3ConfigA	Outbound PPI 3 config word A
0x8800067C	31:0	Read	ObPpi3ConfigB	Outbound PPI 3 config word B
0x88000680 - 0x88000FFC	NA	Read	Unused	Read as zero

Continued on next page

Continued from previous page

Address	Bits	Mode	Name	Description
0x88001000	31:0	Read	CompFifo0	Completion FIFO 0
0x88001004 ...	31:0	Read	CompFifo1	Completion FIFO 1
0x88001028	31:0	Read	CompFifo10	Completion FIFO 10
0x8800102C - 0x88001038	NA	Read	Unused	Read as zero
0x8800103C	31:0	R/W	CompFreeFifo	Completion Free List FIFO
0x88001040 - 0x880010FC	NA	Read	Unused	Read as zero
0x88001100	31:0	Read	OutboundHeader0Free	Outbound Header 0 Free List
0x88001104	31:0	Read	OutboundHeader1Free	Outbound Header 1 Free List
0x88001108	31:0	Read	OutboundHeader2Free	Outbound Header 2 Free List
0x8800110C	31:0	Read	OutboundHeader3Free	Outbound Header 3 Free List
0x88001110 - 0x880011FC	NA	Read	Unused	Read as zero

Table 18: DMA Controller Address Map

A number of FIFOs in the above table are 36-bit FIFOs which are written to over the 32-bit local bus. The upper 4 bits of the FIFO are derived by the offset address used when writing to the FIFO. The address for the FIFO write can be derived using the following equation:

Address = (FIFO base address) * 4 * (Bits 35:32)

For example to write the value 0xA_5A5A_5A5A to the Inbound header 0 free list FIFO, one would write the 32-bit value 0x5A5A_5A5A to the address 0x8800_0128.

2.4.3 DMA Controller Interrupt Mapping

The DMA controller supports 16 interrupt putputs, each with it's own enable bit. The sources of these 16 interrupts are described in the table below.

Interrupt	Name	Description
3:0	IbDesc[3:0]	Inbound header 3:0 descriptor FIFOs. Asserted when associated quad word memory location is dirty. De-asserted when the associated memory location is cleaned by writing to corresponding dirty clear address.
4	BsiData	BSI data FIFO. Asserted when associated quad word memory location is dirty. De-asserted when the associated memory location is cleaned by writing to corresponding dirty clear address.
15:5	CompFifo[10:0]	Completion FIFOs. Asserted when associated completion FIFO has at least one entry. De-asserted when the associated completion FIFO is empty.

Table 19: DMA Controller Interrupt Mapping

2.5 Inbound Controller (ArmRceG3IbCntrl.vhd)

The inbound controller module is a sub-container within the DMA controller which contains the logic to support inbound PPI traffic. This includes the 4 inbound header engines, the 5 quad word FIFOs, the memory space dirty state tracking logic and an AXI write controller. The AXI write controller is the interface between the various write engines and the write interface of the AXI ACP processor bus.

2.5.1 Inbound Controller Block Diagram

The block diagram of the inbound controller module is shown in figure 3. The following sub modules exist within the module and are described in greater detail later in this document:

- ArmRceG3IbHeaderFifo: Inbound header transfer FIFO and control logic (section 2.7)
- ArmRceG3IbQWordFifo: Inbound Quad Word FIFO and control logic (section 2.6)
- ArmRceG3AxiWriteCntrl: AXI write controller (section 2.9)

The inbound controller module also contains dirty status logic which tracks the state of the OCM memory locations associated with the 5 quad word FIFO modules.

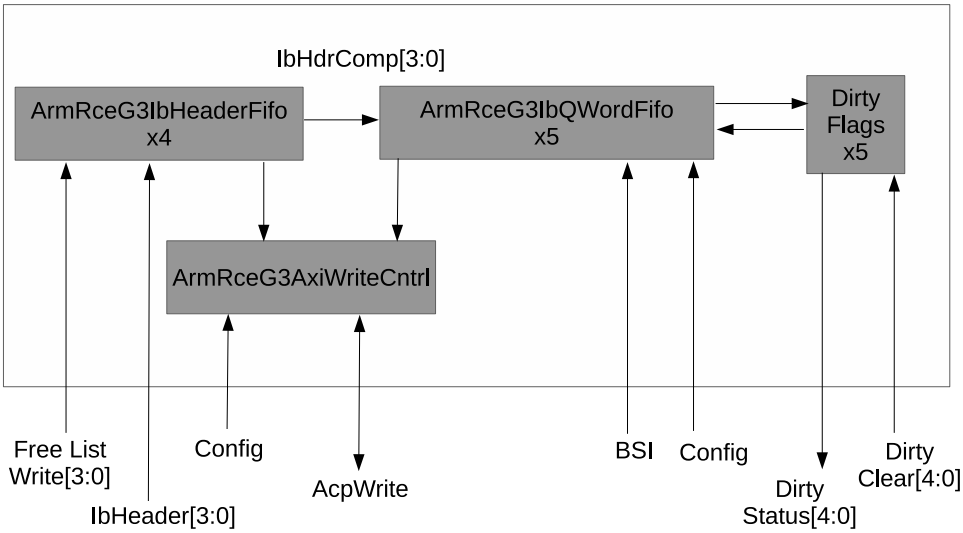

Figure 3: Inbound Controller Block Diagram

2.5.2 Quad Word FIFO Channels

The inbound controller contains 5 instances of the Quad Word FIFO module described in section 2.6. The assignment and destination memory address of each of these instances is shown in table 20.

Index	Name	Destination Address	Description
0	IbDesc0	memBaseAddress + 0x00	Inbound header 0 descriptor FIFO
1	IbDesc1	memBaseAddress + 0x08	Inbound header 1 descriptor FIFO
2	IbDesc2	memBaseAddress + 0x10	Inbound header 2 descriptor FIFO
3	IbDesc3	memBaseAddress + 0x18	Inbound header 3 descriptor FIFO
4	BsiData	memBaseAddress + 0x20	BSI data FIFO

Table 20: Quad Word FIFO Channels

The four inbound header descriptor FIFOs are populated when the inbound header engine (see section 2.7) completes a header transfer. The BSI data FIFO is populated when the IPMI controller writes a 32-bit word to the BSI shared memory over the management I2C bus (see section 2.15).

2.5.3 ACP Write ID Mapping

The ACP AXI interface only supports 8 independent transaction IDs. Since the inbound controller contains 5 potentional masters, some IDs need to be shared. Table 21 shows the allocation of AXI IDs to the masters within the inbound controller. Only one master assigned to an ID can have an outstanding write

transactions at any given time. Any other masters which share an ID with a master who has an outstanding transaction will not attempt to access the ACP bus until the outstanding transaction completes.

AXI ID	FIFO(s)	Function(s)
0	IbHeaderFifo0	Inbound header 0 engine
1	IbHeaderFifo1	Inbound header 1 engine
2	IbHeaderFifo2	Inbound header 2 engine
3	IbHeaderFifo3	Inbound header 3 engine
4	QWordFifo0 QWordFifo4	Inbound header 0 descriptor FIFO BSI FIFO
5	QWordFifo1	Inbound header 1 descriptor FIFO
6	QWordFifo2	Inbound header 2 descriptor FIFO
7	QWordFifo3	Inbound header 3 descriptor FIFO

Table 21: ACP Write ID Mapping

2.6 Quad Word FIFO Controller (ArmRceG3IbQWordFifo.vhd)

The quad word FIFO controller is a block of logic which serves as a 63-bit FIFO with direct access to the on chip memory (OCM) contained within the Zynq processor. Only 63 bits are available because the upper bit is used for handshaking between the quad word FIFO module and the software driver.

When an entry is available in the FIFO, the transfer state machine will check to see if the associated OCM memory space is clearn or dirty as indicated by the 5-bit dirty vector managed by the indbound controller module. If the associated memory location is clean the transfer state machine will pull the 63-bit value from the FIFO and write it to the associated OCM memory location. Bit 64 of the memory location will be set to zero to indicate that the location has been updated.

All transactions generated by the quad word FIFO controller are a single 64-bit write transaction on the ACP bus. Once the write data portion of the transaction is completed the transfer state machine will release the AXI bus. The associated AXI ID will be marked busy while the state machine waits for the write transaction to be acknowledged by the AXI bus. When the write has completed the transfer state machine will clear the ID busy signal, mark the memory location as dirty and return to the idle state.

If enabled the dirty flag of the memory location will trigger a processor interrupt. Some time later, after accessing the associated memory location in OCM, software will clean the memory location by performing a write access to the associated address space. The transfer state machine will then transfer the next FIFO entry when available.

The FIFO logic will not operated until its associated fifoEnable signal is set via register access.

2.6.1 Quad Word FIFO Block Diagram

The quad word FIFO module consists of a 72-bit wide by 512 entry deep input FIFO and a transfer state machine.

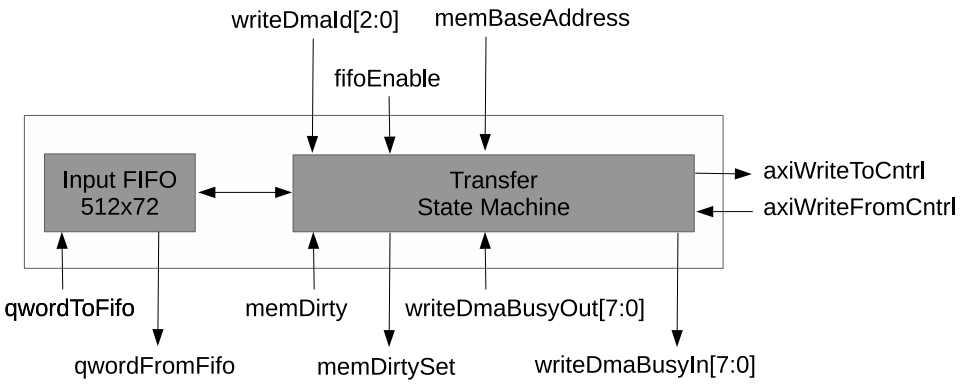

Figure 4: Quad Word FIFO Block Diagram

2.7 Inbound Header FIFO (ArmRceG3IbHeaderFifo.vhd)

The function of the inbound header FIFO module is to receive a PPI header and transfer it into on chip memory (OCM).

When data is available in the FIFO the transfer engine will exit the idle state and pull a free descriptor entry from the free list FIFO. The write address vector within the transfer engine is then preset with the sum of the memBaseAddress and offset address contained in the descriptor. The offset value in the descriptor must be aligned to a cache line boundary.

Each AXI bus transaction originated by the transfer engine is a fixed size containing 32 bytes of data. The transfer engine will wait until a 32-byte block of data is ready in the FIFO or the end of header (EOH) signal is detected. The transfer engine will request and release access to the AXI bus for each 32 byte write transaction. If the EOH is in a position short of the 32 byte boundary, the transfer engine will continue to write undefined data past the EOH location to the OCM.

When the entire frame has been transfered (as indicated by the EOH flag) the transfer engine will wait for all of the outstanding writes to complete. The transfer engine will then form a receive descriptor and write it to the associated inbound descriptor quad word FIFO. Along with the header length the type and mgmt flags from the received frame are included in the receive descriptor along with the error flag.

The inbound header FIFO will not operate if the associated fifoEnable configuration bit is not set.

2.7.1 Inbound Header FIFO Block Diagram

The inbound header FIFO module consists of a 72-bit wide by 512 entry deep input FIFO and a transfer state machine. A 36-bit x 512 entry FIFO is used for the inbound free list.

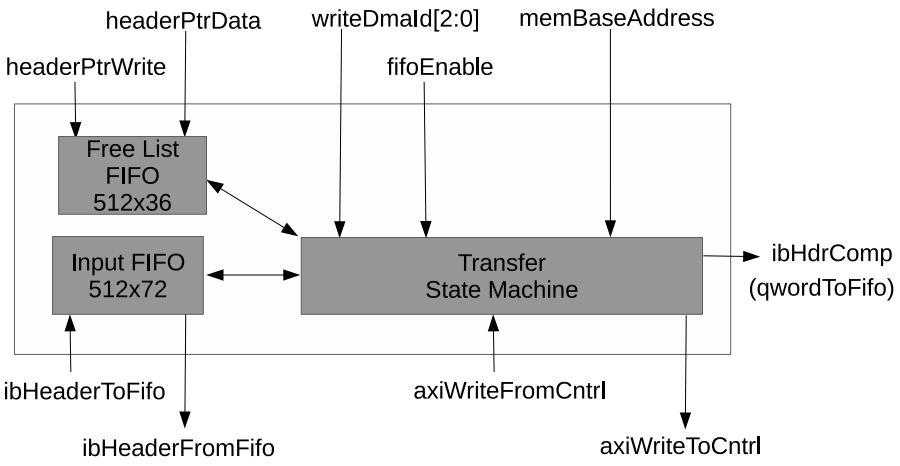

Figure 5: Inbound Header FIFO Block Diagram

2.7.2 Inbound Header Free List

The inbound header free list contains a pool of memory address to which incoming headers are to be transfered. This free list is populated by writing the allocated buffer address to the appropriate register address. Since the upper data bits (35:32) of the free list FIFOs are not used only the base address for each FIFO is utilized. Each free list FIFO is capable of holding 511 free list entries.

The format of the receive free list entry is shown in table 22.

Bits	Name	Description
35:19	unused	ignored
17:3	address	This field contains the offset address to which the inbound frame will be transfered. This address is relative to the configured base address and must be cache line aligned.
2:0	address	Unused. The lower three address bits are assumed to be zero.

Table 22: Inbound Header Free List Entry

2.7.3 Inbound Header Receive Descriptor

When an inbound header is received the inbound header logic will pull an address from the free list and DMA the header data to the associated address. When the operation has completed a receive descriptor will be placed in the associated receive queue. The receive queue is implemented in a Quad Word FIFO described in section 2.6. The mapping of the quad word FIFOs are described in section 2.5.

The format of the receive queue descriptor is shown in table 23.

Bits	Name	Description
63	handshake	Set to zero by firmware when valid
62:61	unused	Always zero
60	error	This bit is set when the error bit was set on the inbound header. This state is only possible when the inbound PPI frame has no payload.
59:52	unused	Always zero
51:48	htype	frame type field from inbound frame
47:40	unused	Always zero
39:32	length	Length of received header. One based length (1=1, 2=2) Specified in number of 64-bit quad words transfered.
31:18	unused	Always zero
17:3	address	This field contains the offset address to which the inbound frame was transfered. This address is relative to the configured base address and must be cache line aligned.
2:0	address	These bits are always zero.

Table 23: Inbound Header Receive Descriptor

2.7.4 Inbound Header Flow Control

The inbound header module asserts three flow control signals depending on the state of the input FIFO:

- Full = The FIFO is full, no free entries available.
- Almost Full = The FIFO is almost full with one free entry left.
- Partially Full = The FIFO has less than 255 entries (out of 512) available.

2.8 Inbound PPI Controller (ArmRceG3IbPpi.vhd)

The inbound PPI controller receives a complete PPI frame and separates the header from the payload. The header portion of the frame is forwarded to the associated inbound header controller module. Meanwhile payload portion of the frame is stored in a local FIFO. The internal state machine will wait until a inbound PPI descriptor is available in the PPI control FIFO. This descriptor described in table 24 contains the information needed to transfer the PPI frame into processor memory. When the inbound DMA operation has completed, a completion descriptor is written to a targeted completion FIFO. The contents of this completion descriptor are detailed in table 25.

Each inbound PPI engine is attached to one of the four HP AXI write interfaces. This dedicated connection means that the inbound PPI engine can assume complete ownership of the interface. A simplified version of the AXI write controller (described in section 2.9) is instantiated in the module in order to simplify the state machine and improve timing performance. In order to ensure AXI bus efficiency all write transfers are a fixed size of 128 bytes. The write enable strobes are used to insure that only relevant payload data is written to memory and that the transfer never exceeds the maximum frame size as indicated by the receive control descriptor.

A byte realignment block is used to allow byte aligned transfer sizes and start addresses. The transfer size of the initial write block at the start of a new payload frame is adjusted in order to align the remaining transfers to 128 byte memory boundaries. This is done to ensure that none of the write transfers cross a 4-KByte boundary.

The inbound PPI engine does not have a mechanism to indicate errors on the incoming payload frame. If the PPI client indicates an error by asserting the ERR flag coincident with EOF or if the inbound payload frame overruns the allocated space, the error state is not communicated to the software layer.

2.8.1 Inbound PPI Controller Block Diagram

The inbound PPI controller module consists of a header receive engine which separates the incoming PPI frame into header and payload portions. The payload portion of the frame is stored in a 72-bit wide by 512 deep payload FIFO. Receive descriptors are buffered in 36-bit by 512 entry FIFO. A transfer state machine controls the transfer of the input FIFO data to the Arm processor memory space. An ArmRceG3AxiWriteCntrl (see section 2.9) block serves as the bridge between the transfer state machine and the HP AXI bus. A 36-bit by 16 entry FIFO serves as a staging FIFO for completion records.

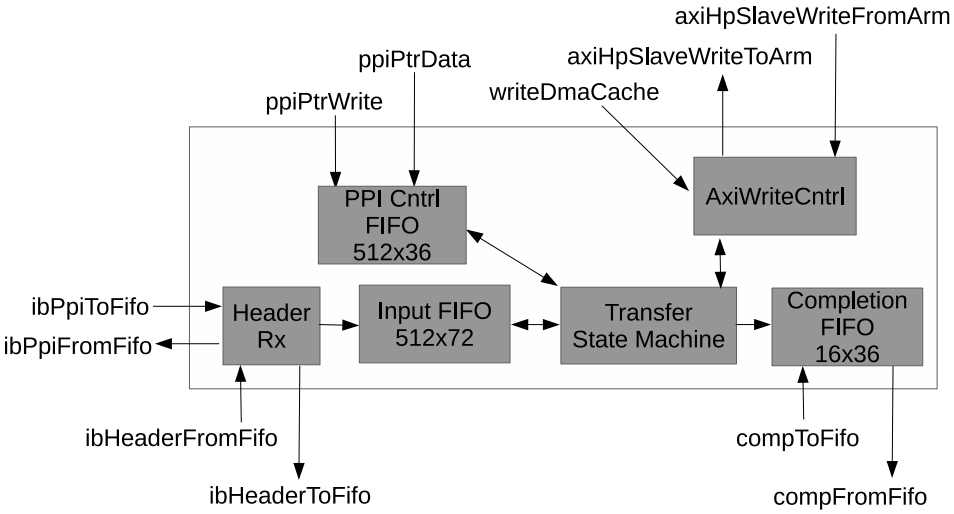

Figure 6: Inbound PPI Controller Block Diagram

2.8.2 Inbound PPI Receive Control

When software wishes to initiate the reception of an inbound PPI payload it will write a receive control descriptor to the inbound PPI control FIFO. The receive control descriptor requires three separate writes to the associated FIFO. Bits 35:32 of the FIFO entry are generated by adjusting offset address within the FIFO address space.

The format of the inbound PPI receive control descriptor is shown in table 24.

DWord	Bits	Name	Description
0	35:33	unused	ignored
0	32	IbDrop	Set this bit to discard frame
0	31:0	IbAddr	Inbound frame destination address
1	35:33	unused	ignored
1	32	CompEn	Set this bit to enable completion record generation
1	31:0	Length	Length of inbound frame in bytes.
2	35:32	CompIndex	Completion FIFO selection index. Valid values are 0 - 10.
2	31:4	CompId	Id value for completion record

Table 24: Inbound PPI Receive Descriptor

2.8.3 Inbound PPI Receive Completion Record

When the inbound frame DMA operation is completed, the inbound PPI engine has the ability to add a completion record to one of the 11 completion FIFOs. The CompEn bit in the inbound receive control descriptor determines if a completion record is generated and the CompIndex field determines which of the 11 FIFOs to route the completion record to. The value written to the completion record is defined in the

CompId field of the inbound receive control record. When the IbDrop bit is set a completion record is not generated.

The format of the receive completion record is shown in table 25.

Bits	Name	Description
31:4	CompId	Completion ID field from receive descriptor.
3	User Error	The PPI block asserted the error signals on the received frame.
2	Length Error	The incoming frame length did not match the receive descriptor.
1	AXI Error	An AXI write error occured.
0	Invalid	Asserted when the completion value is emtpy upon read.

Table 25: Inbound PPI Receive Completion Descriptor

2.8.4 Inbound PPI Flow Control

The inbound PPI module will assert the pause signal to the PPI client firmware when the inbound frame FIFO has less than 255 entries (out of 512) available. The pause signal will also be asserted when the associated inbound header FIFO also has less than 255 entires (out of 512) available.

2.9 AXI Write Controller (ArmRceG3AxiWriteCntrl.vhd)

The AXI write control module serves two purposes. The first is to provide arbitration between AXI masters in cases where more than one write source is attached to a shared AXI bus. The second is to provide address and data FIFOs between the attached write state machine and the processor's AXI interface. This simplifies the implementation of the attached state machine and decouples the AXI interface flow control handshaking from the bursting nature of the attached write state machines.

The AXI write controller supports two modes of operation. The first supports 9 separate write masters when used in the inbound controller block (see section 2.5). The second supports a single master when used in the in the inbound PPI controller (see section 2.8). When only master is attached the arbitration stage is optimized away to reduce latency.

2.9.1 Axi Write Controller Block Diagram

The AXI write controller contains an arbitration block which selects between one of 9 possible AXI masters. Address data and write data are buffered in separate 36-bit x 512 entry FIFOs. Address and data engines convert the address and data FIFO entries into transactions on the AXI bus.

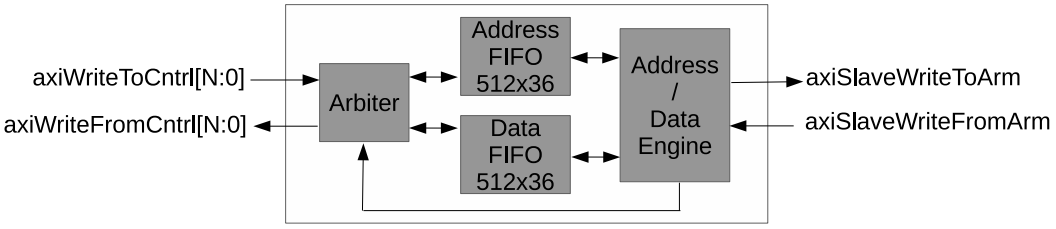

Figure 7: Axi Write Controller Block Diagram

2.10 Outbound Controller (ArmRceG3ObCntrl.vhd)

The outbound controller module is a wrapper that contains four instances of the outbound header FIFO block and a single instances of the AXI read control block.

2.10.1 Outbound Controller Block Diagram

The block diagram of the outbound controller module is shown in figure 8. The following sub modules exist within the module and are described in greater detail later in this document:

- ArmRceG3ObHeaderFifo: Inbound header transfer FIFO and control logic (section 2.7)
- ArmRceG3AxiReadCntrl: AXI write controller (section 2.9)

The outbound controller module also contains 4 outbound free list FIFOs.

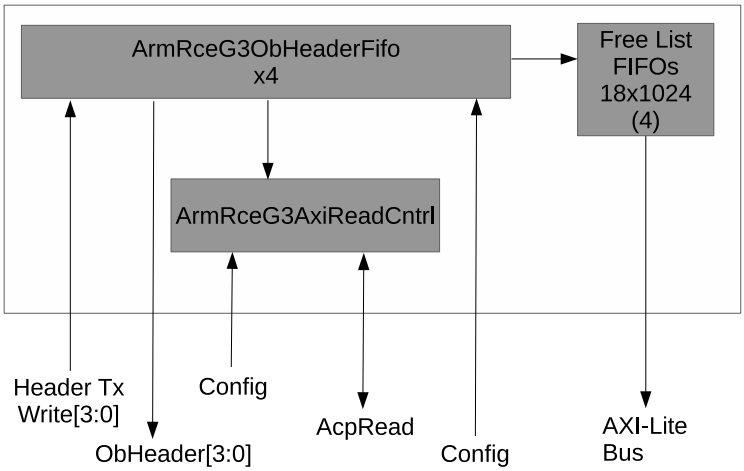

Figure 8: Outbound Controller Block Diagram

2.11 Outbound Header FIFO (ArmRceG3ObHeaderFifo.vhd)

The function of the outbound header FIFO module is to transfer PPI header data from OCM to the outbound PPI module (see section 2.12).

The transfer is started when software writes a transmit descriptor to the transmit list FIFO. The transfer state machine will then exit the idle state, pull the offset address and length from the transmit descriptor and begin reading from the OCM. Each read request is a fixed size of 32-bytes of data regardless of the length. The outbound transfer state machine will queue as many read requests as it has space left in it's outbound FIFO. Anytime the transfer engine pauses for flow control it will release control of the AXI bus allowing it to be re-arbitrated.

When the transmission operation is complete the offset address from the transmit descriptor will be placed back on the free list.

The outbound header FIFO will not operate if the associated fifoEnable configuration bit is not set.

2.11.1 Outbound Header FIFO Block Diagram

The outbound header FIFO module consists of a 72-bit wide by 512 entry deep input FIFO and a transfer state machine. A 36-bit x 512 entry FIFO is used for outbound transmit control.

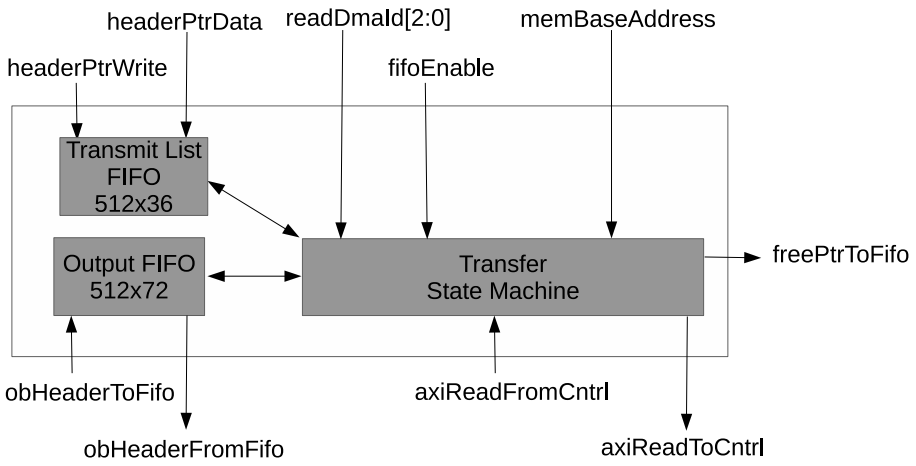

Figure 9: Outbound Header FIFO Block Diagram

2.11.2 Outbound Header Free List

Each of the outbound header FIFOs provide a descriptor free list. This free list is implemented in a local FIFO which is read over the local bus. At startup the software will populate the free list by submitting transmit descriptors with the transfer bit set (described in section 2.11.3). During normal operation the software will pull a free transmit buffer from the free list and provide it to the header engine as part of the transmit descriptor. When the transmit operation has completed the address will be returned to the free list.

The format of the outbound free list entry is shown in table 26.

Bits	Name	Description
31	valid	Bit to indicate if read entry is valid.
30:18	unused	Read as zero.
17:3	address	This field contains the offset address for the outbound header free list. This address is relative to the configured base address and must be cache line aligned.
2:0	address	These bits are always zero.

Table 26: Outbound Header Free List Entry

2.11.3 Outbound Header Transmit Descriptor

An outbound header transmission is started by writing a descriptor to the appropriate header transmit FIFO. Since the upper data bits (35:32) of the transmit FIFOs are not used only the base address per each FIFO is utilized. Each transmit FIFO is capable of holding 511 free list entries. The format of the transmit header descriptor is shown in table 27.

Bits	Name	Description
31:30	transaction	Transaction type. 0 = Move address to the free list without data transmission. 1 = Send header only frame. 2 = Send header + payload without completion. 3 = Send header + payload with completion.
29:26	htype	frame type field for outbound frame
25:18	length	Length of header portion of outbound frame. One based length (1=1, 2=2) Specified in number of 64-bit quad words to transfer.
17:3	address	This field contains the offset address for the outbound header transmission. This address is relative to the configured base address and must be cache line aligned.
2:0	address	These bits are always zero.

Table 27: Outbound Header Transmit Descriptor

When the outbound header transmission has completed the address will be returned to the outbound header free list Quad Word FIFO.

2.12 Outbound PPI Controller (ArmRceG3ObPpi.vhd)

The outbound PPI controller transmits a complete PPI frame by first forwarding the contents of the outbound header FIFO and attaching an optional payload. The last 4 32-bit words contained in the header received from the outbound header FIFO are used as a PPI transmit descriptor and are not included as part of the outbound frame. The contents of this descriptor are described in table 28. When the outbound DMA operation has completed, a completion descriptor is written to a targeted completion FIFO. The contents of this completion descriptor are detailed in table 29.

Each outbound PPI engine is attached to one of the four HP AXI read interfaces. This dedicated connection means that the outbound PPI engine can assume complete ownership of the interface. A simplified version of the AXI read controller (described in section 2.13) is instantiated in the module in order to simplify the state machine and improve timing performance. In order to ensure AXI bus efficiency all read transfers are a fixed size of 128 bytes.

A byte realignment block is used to allow byte aligned transfer sizes and start addresses. The transfer size of the initial read block at the start of a new payload frame is adjusted in order to align the remaining transfers to 128 byte memory boundaries. This is done to ensure that none of the read transfers cross a 4-KByte boundary.

2.12.1 Outbound PPI Block Diagram

The outbound PPI controller module consists of a transfer state machine which is bridged to the AXI bus using an ArmRceG3AxiReadCntrl (see section 2.9) module. Transmit control descriptors are buffered using a 36-bit x 512 entry transmit control FIFO. Outbound data is buffered in a 72-bit x 512 entry FIFO. A 36-bit by 16 entry FIFO serves as a staging FIFO for completion records.

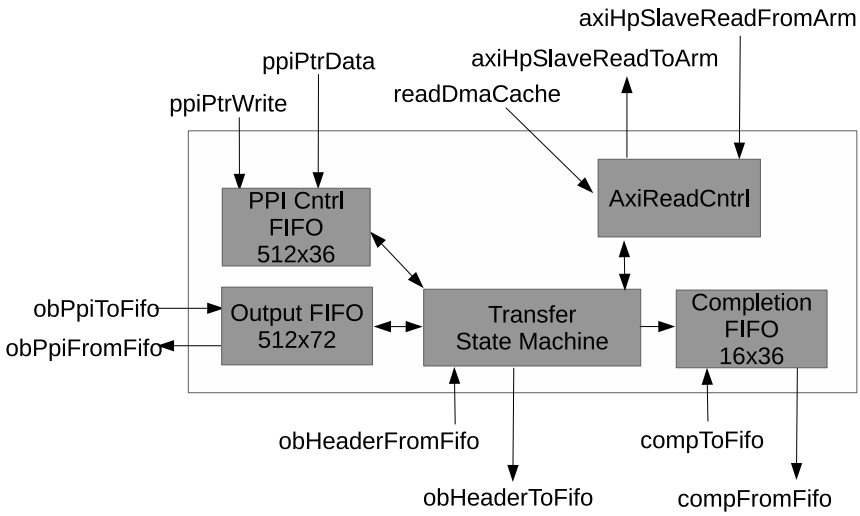

Figure 10: Outbound PPI Block Diagram

2.12.2 Outbound PPI Transmit Control

Outbound PPI frame transmission is controlled by the last four 32-bit dual words in the outbound PPI header frame. These four words form the outbound PPI transmit descriptor and are not included in the outbound frame. All four 32-bit dual words are ignored for transaction types 0 and 1. The last two dual words are ignored for transaction types 0, 1 and 2.

The format of the outbound PPI transmit control descriptor is shown in table 28.

DWord	Bits	Name	Description
0	31:0	ObAddr	Outbound frame source address
1	31:0	ObLength	Length of outbound frame in bytes
2	31:4	CompId	Id value for completion record.
2	3:0	unused	ignored
3	31:4	unused	ignored
3	3:0	CompIndex	Completion FIFO selection index. Valid values are 0 - 10.

Table 28: Outbound PPI Transmit Descriptor

2.12.3 Outbound PPI Transmit Completion Record

When the outbound frame DMA operation is completed, the outbound PPI engine has the ability to add a completion record to one of the 11 completion FIFOs. The CompEn bit in the outbound transmit control descriptor determines if a completion record is generated and the CompIndex field determines which of the 11 FIFOs to route the completion record to. The value written to the completion record is defined in the CompId field of the outbound transmit control record.

The format of the receive completion record is shown in table 29.

Bits	Name	Description
31:4	CompId	Completion ID field from transmit descriptor.
3:1	Unused	Read as zero.
0	Invalid	Asserted when the completion value is emtpy upon read.

Table 29: Outbound PPI Transmit Completion Descriptor

2.13 AXI Read Controller (ArmRceG3AxiReadCntrl.vhd)

Similar to the AXI write control module, the AXI read controller provides arbitration between AXI masters in cases where more than one read source is attached to a shared AXI bus. It also provides an address FIFO between the attached read state machine and the processor's AXI interface. Returned read data is first passed to a FIFO to decouple flow control which may be originated by the attached read state machine(s) from the signals on the AXI bus interface.

The AXI read controller supports two modes of operation. The first supports 4 separate write masters when used in the outbound controller block (see section 2.5). The second supports a single master when used in the in the outbound PPI controller (see section 2.12). When only master is attached the arbitration stage is optimized away to reduce latency.

2.13.1 Axi Read Controller Block Diagram

The AXI read controller contains an arbitration block which selects between one of 4 possible AXI masters. Address data is buffered in a 36-bit x 512 entry FIFO. An address engines converts the address into transactions on the AXI bus.

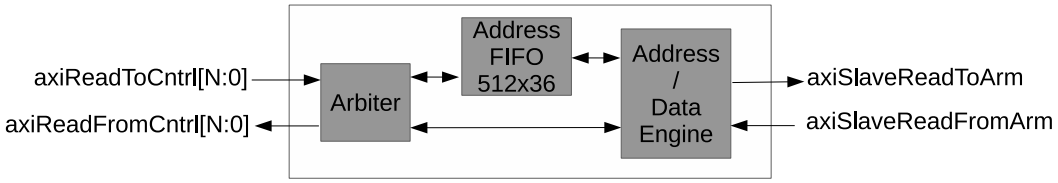

Figure 11: Axi Read Controller Block Diagram

2.14 DMA Completion FIFO Controller (ArmRceG3DmaComp.vhd)

The purpose of the DMA completion FIFO controller is to accept completion records from the four inbound PPI engines and the four outbound PPI engines. Each of the PPI engines has the ability to direct it's completion entries to one of 11 completion FIFOs. Each completion entry is treated independently and can be targeted to a different completion FIFO than other completion entries from the same source.

When a completion FIFOs is non-empty it asserts a ready status bit which can be read over the AXI bus and be configured to trigger an interrupt. The mapping of the 11 completion FIFOs to interrupts is described earlier in this document in table 19. The contents of the completion FIFOs are read over the AXI bus. When software reads from the address associated with a completion FIFO the entry at the head of the queue is returned to software and the contents of the FIFO are advanced.

The completion engine works by iterating through each of the 8 possible sources, one per clock cycle. When a particular source is selected the completion record is examined and the destination ID is determined. If the requested destination FIFO is not full the completion entry will then be moved to the selected completion FIFO.

2.14.1 DMA Completion Block Diagram

The DMA completion module consists of a completion engine which transfers data from a series of small inbound header and outbound header completion FIFOs to one of 9 independent 36-bit x 512 entry completion FIFOs. A read control block allows the 9 completion to be read from the local bus.

The DMA completion module also contains a single completion free list FIFO. This FIFO can be written to and read from through the local AXI address space.

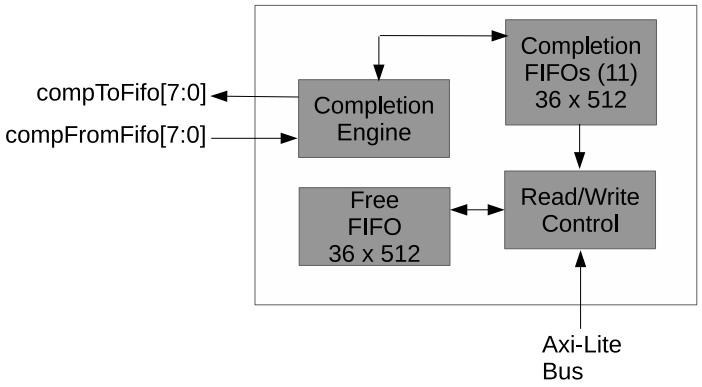

Figure 12: DMA Completion Block Diagram

2.15 I2C Controller (ArmRceG3I2c.vhd)

The I2C controller block supports the BSI operation by providing a message path between the CPU software and the management I2C bus. Anytime a byte is written over the I2C bus the appropriate entry in the I2C BRAM is updated with the data contained in the write. When a complete 32-bit word is written the FIFO writer block will place the 32-bit value along with the target address in the BSI quad word FIFO.

A 32-bit value is written over the I2C bus by writing the least significant byte first (offset address 0), followed by the second byte (offset 1) and third byte (offset 2). When the finally byte (offset 3) is written it's value is combined with the previous three received bytes and added to the quad word FIFO. Software also has the ability to directly read and write the I2C BRAM.

The I2C management controller may poll the flow control state of the BSI FIFO by reading from offset address 0x800.

2.15.1 I2C Controller Block Diagram

The I2c controller module consists of a I2C slave core which converts I2c bus accesses to local byte transfers. A 2Kbyte dual port block ram allows read and write access from either the local bus or the I2C bus. A FIFO write module converts I2C write access to quad word FIFO writes.

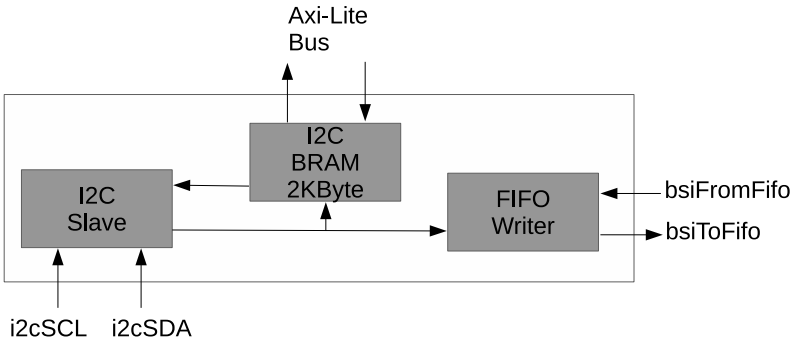

Figure 13: I2C Controller Block Diagram

2.15.2 Local Bus Address Space

RCE software may access the contents of the 2K I2C block ram in the address space 0x8400_0000 - 0x8400_07FC.

2.15.3 I2C Bus Address Space

The I2C controller may access the contents of the 2K I2C block ram in the address space 0x00 - 0x7FF. The BSI Quad Word almost full status appears in bit 0 at address 0x800. The I2C bus address of the slave device is 0x49.

2.16 CPU Interface Module (ArmRceG3Cpu.vhd)

The CPU interface module is a wrapper to the processor_system7_v4_02a core provided from Xilinx. The interfaces used in the ARM RCE generation 3 core are converted to record types. Unused interfaces are terminated with constants.

Further information and documentation for the processor_system7 core can be found on the Xilinx website at:

`http://www.xilinx.com/products/intellectual-property/processing_system7.htm`