

Document Number	
Document Revision	Draft 0.1
Date	2007.08.11
Author	John Hsiao
Status	First Draft



Revision History

Revision	Revision	Author	Revised	Description
Number	Date		Chapters/Sections	



Document Signoff List

Department	Name	Signature	Date	Comment

BIU Design SPEC 錯誤! 找不到參照來源。 Version 錯誤! 找不到參照來源。6



Table of Contents

BIU	Design SP	EC	i
Rev	ision Histor	·y	i
Doc	cument Sign	off List	ii
Tab	le of Conter	nts	iii
List	of Figures.		iv
List	of Tables		v
1			
2	Block diag	gram	6
3	Configurat	tion Option	7
4		escription	
	4.1 Into	erface protocol	
	4.1.1	Write	8
	4.1.2	Read	8
	4.1.3	The read minmmun latency	9
	4.1.4	The write minmmun latency	10
	4.1.5	FCU/LSU request kill flow	11
	4.1.6	Hllsc_req with Hllsc_error on success case	11
	4.1.7	Hllsc_req with Hllsc_error on fail case	12
	4.2 MMU	Interface	12
	4.3 LSU Ir	nterface	13
	4.4 FCU Ir	nterface	13
	4.5 LDMA	Interface	15
	4.6 EDM I	nterface	16
	4.7 AHB	Interface	16
	4.8 APB Ir	nterface	17
	4.9 Others		17
5	Protocol ti	ming diagram	19
6	Arbiter		30



	6.1	Read Request	30
	6.2	Write request	30
7	Req	quest queue	31
	7.1	Command queue	31
	7.2	Data queue	32
	7.3	Synchronous and asynchronous	32
8	Bus V	Wrapper	33
	8.1	AHB Bus	33
		8.1.1 Command Decoder	33
		8.1.2 Retry control	33
		8.1.3 Alignment control	34
	8.2		
	8.3	APB Bus	35
	8.4	AHB2 configuration	35
9	State	te Machine	36
10	Data	a Path	37
11	Vori	ification Plan	20



 BIU Design SPEC
 5

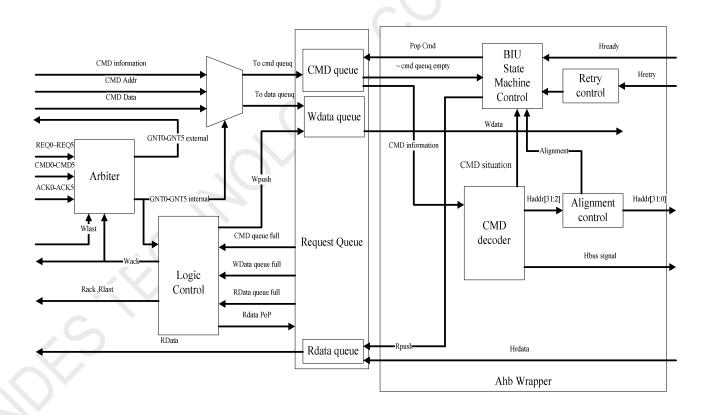
 錯誤! 找不到參照來源。 Version 錯誤! 找不到參照來源。6
 07/09/2007



1 Overview

The BIU unit is responsible for data transfer between processor core and system bus. The BIU in Condor can take six request inputs from five resources (Dcache, Icache, LDMA, EDM, and HPTWK) and support up to three bus protocols. (AHB,AHB-lite,and APB)

2 Block diagram





3 Configuration Option

NDS_BIU_AHB2

NDS_BIU_AHB

NDS_BIU_AHB_LITE

NDS_BIU_APB

NDS_BIU_MODE

NDS_BIU_SYNC

NDS_BIU_ASYNC

NDS_BIU_SYNC_ASYNC

NDS_CMD_QUEUE_DEPTH_1

NDS_CMD_QUEUE_DEPTH_2

NDS_CMD_QUEUE_DEPTH_4

NDS_WRITE_FIFO_DEPTH_1

NDS _HPTWK_SUPPORT

NDS_LMDMA_SUPPORT

NDS_EDM_SUPPORT



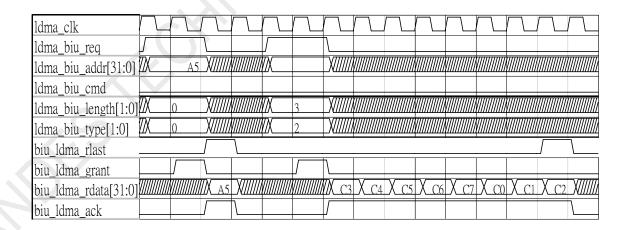
4 Interface description

4.1 Interface protocol

4.1.1 Write

ldma_biu_req												
ldma_biu_addr[31:0]	7 /X	A5	X/////////////////////////////////////	В3	<i>X</i> //////				///////////////////////////////////////			
ldma_biu_cmd												
ldma_biu_length[1:0]	<u> </u>	0	X///////////////////////X	3	X/////				///////////////////////////////////////			
ldma_biu_type[1:0]	7 /X	0	X/////////////////////////////////////	2	X//////				///////////////////////////////////////			
ldma_biu_wdata[31:0]	/ /X	A5	X//////X	В3	4	X B4 X B5	X B6	X B7	χ_{B0}	X B1	χ_{B2}	X/////
ldma_biu_bwe[3:0]	7 /X		X//////X			XX	X	X	X	X	X	<i>X//////</i>
ldma_biu_wlast												
biu_ldma_grant			1									
biu_ldma_ack												

4.1.2 Read



Andes Technology Confidential



4.1.3 The read minimum latency

ldma_clk							
ldma_biu_req							
ldma_biu_addr[31:2]	///////////////////////////////////////	//X A5	X//////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////
ldma_biu_cmd)
ldma_biu_size[1:0]	///////////////////////////////////////	7/X <u>0</u>	X//////	<u> </u>	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////
ldma_biu_length[1:0]	<u> </u>	//X 0	X//////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////
ldma_biu_type[1:0]	///////////////////////////////////////	//X 0	X//////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////
biu_ldma_rlast							
biu_ldma_grant							
biu_ldma_rdata[31:0]	 ///////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X A5	X//////
biu_ldma_ack						/	
Command push							
Command empty							
Haddr	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///X <u>A5</u>	X//////	///////////////////////////////////////	///////////////////////////////////////
Hdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X A5	X//////	///////////////////////////////////////
Hdata push						\	
Rdata_empty							

 BIU Design SPEC
 9

 錯誤! 找不到參照來源。 Version 錯誤! 找不到參照來源。6
 07/09/2007

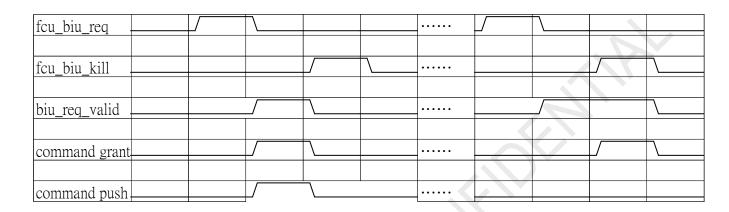


4.1.4 The write minimum latency

ldma_biu_cmd							
ldma_biu_size[1:0]	///////////////////////////////////////	//X0	X//////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////
ldma_biu_length[1:0]	///////////////////////////////////////	//X 0	X//////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////
ldma_biu_type[1:0]	///////////////////////////////////////	//X0	X//////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////
biu_ldma_wlast							
biu_ldma_grant							
biu_ldma_wdata[31:0	\ {////////////////////////////////////	//X A5	X//////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////
biu_ldma_ack							
Command push							
Command empty							
Haddr	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//XA5	X//////	 /////////////////////////////////	///////////////////////////////////////
Hdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X A5	X//////	///////////////////////////////////////
Hdata pop							



4.1.5 FCU/LSU request kill flow



4.1.6 Hllsc_req with Hllsc_error on success case

Success case without wait state

HCLK				
HADDR[31:0]		XA	\times	X
HWRITE		//	V///////	7//////////////////////////////////////
HTRANS[1:0]		Xnonseq	\times	X
HLLSC_REQ	7//////////////////////////////////////	//	V////////	7//////////////////////////////////////
HREADY	7//////////////////////////////////////	//		V////////
HLLSC_ERROR	7//////////////////////////////////////	///////////////////////////////////////	7λ success	

Success case with wait state

HCLK					
HADDR[31:0]	\boxtimes	X A	X	X	X
HWRITE	7//////////////////////////////////////		V///////	///////////////////////////////////////	7////////
HTRANS[1:0]		X <u>nonseq</u>	X	X	X
HLLSC_REQ	7//////////////////////////////////////		V///////	///////////////////////////////////////	///////////////////////////////////////
HREADY	7//////////////////////////////////////		\		V///////
HLLSC_ERROR	7//////////////////////////////////////	/////////	///////////////////////////////////////	7λ success	/////////

 BIU Design SPEC
 11

 錯誤! 找不到參照來源。 Version 錯誤! 找不到參照來源。6
 07/09/2007



4.1.7 Hllsc_req with Hllsc_error on fail case

fail case without wait state

HCLK				
HADDR[31:0]	\boxtimes	X A	X	\times
HWRITE	7////////	7/	V////////	7/////////
HTRANS[1:0]		Nonseq	\times	\times
HLLSC_REQ	///////////////////////////////////////	7/	V///////	7/////////
HREADY	///////////////////////////////////////			V////////
HLLSC_ERROR	7//////////////////////////////////////	///////////////////////////////////////	///// fail	V////////

fail case with wait state

HCLK					
HADDR[31:0]	\boxtimes	A		X	X
HWRITE	7////////	<i>I</i> /	V////////	///////////////////////////////////////	7//////////////////////////////////////
HTRANS[1:0]	\boxtimes	nonseq	X	X	X
HLLSC_REQ	7////////	<u> </u>	V///////	///////////////////////////////////////	7//////////////////////////////////////
HREADY	7///////	<u> </u>			V////////
HLLSC_ERROR	7////////	///////////////////////////////////////	///////////////////////////////////////	///// fail	V////////

4.2 MMU Interface

NAME	I/O	WIDTH	DESCRIPTION	
NDS_MMU_SUPPORT				
hptwk_biu_req	Input	1	Hptwk read request	
hptwk_biu_addr[31:2]	Input	30	Request start addr	
hptwk_biu_length	Input	1	Transaction length – 0: word; 1: 2 words.	
hptwk_biu_rlast	Output	1	Indicate the last transfer	
biu_hptwk_rdata	Output	32	Read data	



biu_hptwk_ack	Output	3	Read acknowledge
biu_hptwk_req_valid	Output	1	Indicate where the request buffer is valid
hptwk_biu_req_kill	Input	1	To kill the previous request

4.3 LSU Interface

NAME	I/O	WIDTH	DESCRIPTION
Load Interface			
lsu_biu_req	Input	1	Load miss request
lsu_biu_cmd	Input	1	1:Write command; 0: Read command
lsu_biu_addr[31:2]	Input	30	Load address
lsu_biu_length	Input	1	Transaction length – 0: word; 1: cache line
lsu_biu_nonbuf	Input	1	Indicates the transaction is non-bufferable.
lsu_biu_noncache	Input	1	Indicates the transaction is non-cacheable access.
lsu_biu_wt	Input	1	Indication the transaction is write-through or not
lsu_biu_wdata[31:0]	Input	32	SCW data
biu_lsu_rdata[31:0]	Output	32	Load data
biu_lsu_abort	Output	1	Load abort
biu_lsu_last	Output	1	Indicates the last word of the transaction.
biu_lsu_ack	Output	1	Load acknowledge
lsu_biu_lock	Input	1	Indicates a LLW/SCW operation
biu_lsu_lock_fail	Output	1	1: fail; 0: complete
biu_lsu_req_valid	Output	1	Indicate where the request buffer is valid
lsu_biu_req_kill	Input	1	To kill the previous request
lsu_biu_isync	Input	1	Indicate there has a Isync command,



			BIU can't receive fcu request
WB Interface			
lsu_biu_wb_req	Input	1	Write request
lsu_biu_wb_addr[31:2]	Input	30	Write address
lsu_biu_wb_last	Input	1	Indicates the last word of the burst transaction.
Lsu_biu_wb_length	Input	1	Transaction length – 0: single word; 1: cache line
lsu_biu_wb_bwe[3:0]	Input	4	Write byte enable
lsu_biu_wb_wdata[31:0]	Input	32	Write data
lsu_biu_wb_nonbuf	Input	1	Indicates non-bufferable access of store operation.
lsu_biu_wb_noncache	Input	1	Indicates non-cacheable access of store operation.(to indicate the single transaction)
lsu_biu_wb_wt	Input		Indicate the transaction is write- through or write-back, 1: write- through
biu_lsu_wb_grant	Output	1	Grant for the request
biu_lsu_wb_ack	Output	1	Store acknowledge

Note: LSU write just has single case, so it doesn't need last.

4.4 FCU Interface

NAME	I/O	WIDTH	DESCRIPTION
fcu_biu_addr[31:2]	Input	30	Address
biu_fcu_rdata	Output	32	Read data
fcu_biu_req	Input	1	BIU request
fcu_biu_length	Input	1	Burst size.
			0: single word
			1: one cache line
biu_fcu_last	Output	1	Indicates the last word of the transaction.



biu_fcu_ack	Output	1	Read-data acknowledge
biu_fcu_abort	Output	1	Icache Read abort
biu_fcu_req_valid	Output	1	Indicate where the request buffer is valid
fcu_biu_req_kill	Input	1	To kill the previous request

4.5 LDMA Interface

NAME	I/O	WIDTH	DESCRIPTION
Ldma_biu_req	Input	1	LDMA request
Ldma_biu_addr[31:2]	Input	30	LDMA request address
ldma_biu _cmd	Input	1	1:Write command; 0: Read command
ldma_biu_length[1:0]	Input	1	Transaction length – 0: word; 1: 2word;
			2: 4 words ; 3 : cacheline
ldma_biu _wdata	Input	32	LDMA write data
ldma_biu_bwe	Input	4	LDMA write enable
ldma_biu _wlast	Input	1	Indicate last write transfer
ldma_biu_icache	Input	1	Indicate the ldma command is to icache
			or dcache; 1 : icache ; 0 : dcache
biu_ldma_rdata	Output	32	LDMA read data
biu_ldma_ack	Output	1	Read/Write-data acknowledge
biu_ldma_rlast	Output	1	Indicate last read transfer
biu_ldma_grant	Output	1	Indicate command is received



4.6 EDM Interface

NAME	I/O	WIDTH	DESCRIPTION
edm_biu_req	Input	1	EDM request
edm_biu_addr[31:2]	Input	30	EDM request address
edm_biu _cmd	Input	1	1:Write command; 0: Read command
edm_biu_length	Input	1	Transaction length – 0: word; 1: cache
			line
edm_biu _wdata	Input	32	EDM write data
edm_biu_bwe	Input	4	EDM write enable
biu_edm_rdata	Output	32	EDM read data
biu_edm_ack	Output	1	Read-data acknowledge
biu_edm_grant	Output	1	Indicate command is received

4.7 AHB Interface

NAME	I/O	WIDTH	DESCRIPTION
Hclk	Input	1	AHB clock
hreset_n	Input	1	AHB reset
Haddr	Output	32	AHB address
Htrans	Output	2	AHB transfer type
Hready	Input	1	AHB transfer done
Hresp	Input	2	AHB response
Hwrite	Output	1	AHB write; 1:write; 0:read
Hsize	Output	3	AHB size
Hburst	Output	3	AHB burst length
Hprot	Output	4	AHB protect
Hwdata	Output	32	AHB write data



Hrdata	Input	32	AHB read data
Hbusreq	Output	1	AHB bus request
Hgrant	Input	1	AHB bus grant
Hlock	Output	1	AHB lock

Note: 1 AHB2 signal postfix 2 into signa exclude Hclk,hreset_n.

Ex : Htrans = > Htrans2.

2. AHB-lite doesn't has Hgrant, Hbusreq and Hresp just 1 bit.

4.8 APB Interface

NAME	I/O	WIDTH	DESCRIPTION
pclk	Input	1	APB clock
preset_n	Input	1	APB reset
paddr	Output	32	APB address
pbyte_wenable	Output	4	APB byte Write enable(Andes definition)
psel	Output	1	APB select
penable	Output	1	APB strobe
pwrite	Output	1	APB write; 1:write; 0:read
prdata	Input	32	APB read data
Pwdata	Output	32	APB write data

4.9 Others

NAME	I/O	WIDTH	DESCRIPTION
hllsc_req,	Output	1	Load-lock store condition request
hllsc_error,	Output	1	Indicate there has a error on llsc
hwrite_through	Output	1	0: write back; 1: write through
biu_ieu_standby_ready	Input	1	Indicate IEU thre has no request at BIU



hclk_phase,	Output	1	Use to synchronous helk domain to cpu
			clk domain
core_clk,	Output	1	Cpu clock
core_reset_n	Output	1	Cpu reset
sync_mode	Input	1	1: synchronous ; 0 : asynchronous



5 Protocol timing diagram

BIU state single read_write transfer

Hgrant						
Hreq						
CMD fifo empty		\				
BIU Next State	IDLE	REQ	ADDR	Last Data	IDLE	
BIU State	IDLE		REO	X ADDR	Last Data	X IDLE
Haddr	///////////////////////////////////////	///////////////////////////////////////		//X A	X//////	///////////////////////////////////////
Hsize	///////////////////////////////////////			//X Size A	X/////	
Htrans	IDLE			XNONSEQ	X IDLE	
	OK			THOMBE		
Hresp	- OK					
Hready						
Hwrite		X	_X	X		X
HWdata	<i>\////////////////////////////////////</i>	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	X Wdata A	_X//////



BIU state single read_write with wait state transfer

Hgrant							
Hreq							
ПСЦ							
CMD fifo empty	7						
BIU Next State	IDLE	X REO	ADDI	Last Data		X IDLE	
BIU State	IDLE		X REC) X ADDR	Last Data		IDLE
					\//////		
Haddr	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X A	X//////	///////////////////////////////////////	<i>/////////////////////////////////////</i>
Hsize	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	Size A	X//////	///////////////////////////////////////	///////////////////////////////////////
Htrans	IDLE			NONSEO	X//////	///////////////////////////////////////	///////////////////////////////////////
Hresp	OK						
Hready					\		
Hwrite	X	X	X	X	X	X	
HWdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X	Wdata A	X//////
HRdata	///////////////////////////////////////			///////////////////////////////////////	///////////////////////////////////////	//X RData A	X/////

 BIU Design SPEC
 20

 錯誤! 找不到參照來源。 Version 錯誤! 找不到參照來源。6
 07/09/2007



BIU state single read_write transfer with 3 byte transfer

Hgrant							
Hreq						\	
CMD fifo empty							
BIU Next State	IDLE	REQ	ADDR	Last data		X IDLE	
BIU State	IDLE		REO	X ADDR	Last data		X IDLE
Haddr	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X A1st	A2st	X//////	///////////////////////////////////////
Hsize	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	// Size A1st	Size A2st	X//////	///////////////////////////////////////
Htrans	IDLE			XNONSEO		X//////	///////////////////////////////////////
Hresp	OK		C				
Hready							
Hwrite							
HWdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	Vdata A1st	Wdata 2st	X//////
ahb_3byte_1st						\	
ahb_3byte_2nd							



BIU state multi cmd single read_write transfer

Hgrant							
Hreq							
CMD fifo empty	Į.	\					
BIU Next State	IDLE	X REO	ADDR	Last Data		IDLE	
BIU State	IDLE		X REC	X ADDR	Last Data		X IDLE
Haddr	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X A	В	X//////	///////////////////////////////////////
Hsize	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X Size A	Size B	X//////	///////////////////////////////////////
Htrans	IDLE			XNONSEO		IDLE	
Hresp	OK						
Hready							
Hwrite	X	X	X	X	X	X	X
HWdata	///////////////////////////////////////		///////////////////////////////////////	///////////////////////////////////////	Wdata A	Wdata B	X//////
HRdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	<u> </u>	RData A	Rdata B	X//////

 BIU Design SPEC
 22

 錯誤! 找不到參照來源。 Version 錯誤! 找不到參照來源。6
 07/09/2007



BIU state multi cmd single read_write with wait state transfer

Hgrant									
Hreq									
CMD fifo empty	У	\							
BIU Next State	IDLE	X REO	X ADDR	Last Data				X IDLE	
BIU State	IDLE		X REO	X ADDR	Last Data				X IDLE
Haddr	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//XA	В		X/////	///////////////////////////////////////	
Hsize	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X Size A	X Size B		X//////	///////////////////////////////////////	///////////////////////////////////////
Htrans	IDLE			NONSEO	7		X IDLE		
Hresp	OK								
Hready						/	\	/	
Hwrite	X	X	X	X	X	X	X	X	X
HWdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	Wdata A		Wdata B		X/////
HRdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X Rdata A	X//////	//X Rdata B	X/////



BIU state multi cmd singal read_write transfer with retry flow

Hreq	
CMD fifo empt	y
BIU Next State	IDLE X REQ X ADDR X ast Data X Retry X REQ X ADDR X ast Data X IDLE
BIU State	IDLE X REO X ADDR X ast Data X Retry X REO X ADDR X ast Data X IDLE
Haddr	//////////////////////////////////////
Hsize	
Htrans	IDLE XONSEO XIDLE XONSEO X IDLE
Hresp	OK Retry OK
Hready	
Hwrite	
HWdata	//////////////////////////////////////
HRdata	7/////////////////////////////////////



BIU state multi read_write transfer

Hgrant								
Hreq								
CMD fifo empty								
BIU Next State IDI	E REC	ADDR	ADDR DATA			Last Data	IDLE	
BIU State IDI	E	REO	ADDR	ADDR DATA		Θ_{λ}	Last Data	IDLE
Haddr ////////	///////////////////////////////////////	///////////////////////////////////////	//X A	A+4	A+8	A+12	X//////	(//////////////////////////////////////
Hsize ////////		///////////////////////////////////////	/// Size A				X//////	
Htrans IDI	E		NONSEQ	SEQ			IDLE	
			NONSEQ	SEQ				
Hresp O	K							
Hready								
Hwrite	X	X	X	_X	_X	_X	_X	X
HWdata /////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	Wdata A	Wdata A+4	Wdata A+8	Wdata A+12	X//////
Hrdata ///////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	RData A	Rdata A+4	Rdata A+8	Rdata A+12	X//////
Hburst /////////	///////////////////////////////////////	///////////////////////////////////////	INCR 4				X//////	///////////////////////////////////////



BIU state multi read_write with wait state transfer

Hgrant											
Hreq									\		
CMD fifo empty											
BIU Next State	IDLE	X REQ	X ADDR	ADXR_DATA				XLast Data		IDLE	
BIU State	IDLE		REQ	X ADDR	ADAR DATA				Last Data		X IDLE
Haddr	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X A	X A+4		X A+8	X A+12	X/////	///////////////////////////////////////	
Hsize	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X Size A					X//////	///////////////////////////////////////	
Htrans	IDLE			NONSEQ	SEQ				IDLE		
Hresp	OK										
Hready									\		
Hwrite	X	X	X	X	X	X	X	X	<u> </u>	<u> </u>	X
HWdata	7//////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	(//////////////////////////////////////	// Wdata A		Vdata A+4	Widete A 18	Wiata A+12		X/////
	7//////////////////////////////////////	(//////////////////////////////////////	///////////////////////////////////////	(//////////////////////////////////////	/// wuata A	// Pdets A				Mate A : 12	
Hrdata	<u> </u>	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X Rdata A	Adata A+4	Adata A+8	_X/////	Nata A+12	_X//////



BIU state multi read_write transfer with retry

Hgrant												^	
Hreq													
CMD fifo empty		\											
												_	
BIU Next State	IDLE	X REQ	X ADDR	DER_DATA		Retry	X REQ	ADDR	D X _DATA		Xast_Data	X IDLE	
DILL G.			V	V				V	V	V		V -	V
BIU State	IDLE		X REQ	ADDR ADDR	DIX DATA			X REQ	ADDR	DDR_DATA		ast_Data	X IDLE
Haddr	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//XA	X A+4	X A+8	X//////	///////////////////////////////////////	//X A+4	X A+8	X A+12	X//////	///////////////////////////////////////
Hsize	\/////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	VX Size A			X///////	///////////////////////////////////////	VX Size A			X//////	V////////
118126		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	/// SIZE A					// Size A				
Htrans	IDLE			NONSEQ	SEQ		X IDLE		NONSEQ			X IDLE	
Hresp	OK					X Retry)	X OK					
Hready													
Hwrite	X	X	X	X	X	X	X	X	X	X	X	X	X
HWdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	Xwdata A	Wdata A+4		_X//////	///////////////////////////////////////	Mata A+4	Wdata A+8	Wata A+12	X//////
Hrdata	<u> </u>	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	RData A	X//////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	data A+4	data A+8	Rata A+12	X/////
Hburst	<i>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\</i>	///////////////////////////////////////	///////////////////////////////////////	//X INCR 4			_X//////	///////////////////////////////////////	//\X <u>sing</u> al			_X//////	///////////////////////////////////////



BIU state multi read_write with early burst termination transfer

Hgrant													
Hreq											\		
CMD fifo emp	ty	\											
BIU Next State	IDLE	X REQ	X ADDR	ADXR_DATA		ast Data2	X REQ	ADDR	ADXR_DATA	Last_Data		X IDLE	
BIU State	IDLE		X REQ	X ADDR	ADXR_DATA		ast Data2	X REQ	X ADDR	ADXR_DATA	XLast_data		IDLE
Haddr	<i>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\</i>	///////////////////////////////////////	///////////////////////////////////////	7/X A	X A+4	X A+8	X//////		//X A+8	X A+12	X//////	///////////////////////////////////////	
Hsize	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	//X Size A									
Htrans	IDLE			NONSEQ	X SEO	X SEQ	X IDLE		XNONSEQ		X//////	///////////////////////////////////////	
				VIONSEQ	SEQ		/ IDLE		VIOUSEQ				
Hresp	OK										¬		
Hready]				\	J	
Hwrite	X	X	X	X	X	_X	_X	_X	_X	X	_X	_X	X
HWdata	7/////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	// Wdata A	Vdata A+4		_X//////	///////////////////////////////////////	Vdata A+8	Wata A+12		X//////
Hrdata	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	///////////////////////////////////////	// Rdata A	X//////	Adata A+4	X//////	///////////////////////////////////////	Adata A+8	X//////	Kiata A+12	X/////
Hburst	7/////////	///////////////////////////////////////	///////////////////////////////////////	//X INCR4			X//////	///////////////////////////////////////	SINGLE		X//////	///////////////////////////////////////	///////////////////////////////////////

BIU Design SPEC 28

Andes Technology Confidential



APB Transfer

PCLK								
CMD fifo empty	T	\						
APB State	SEL		NABLE	X SEL	NABLE	X SEL	NABLE	X SEL
PADDR	///////////////////////////////////////	//X A		X A+4		X A+8		X//////
PSEL								\
Penable				\			/	\
Pwrite	X	X	X	X	X	X	X	X
PWdata	///////////////////////////////////////	//XA		X_A+4		X_A+8		X//////
PRdata	///////////////////////////////////////	///////////////////////////////////////	//XA	X//////	//X_A+4	X//////	//X_A+8	X/////



6 Arbiter

There could have 6 requests to BIU at the same time. So BIU needs to have a arbiter to decide who is the transaction owner. When the ownership of the transaction is determined, the corresponding address and data are placed into command FIFO and data FIFO. The priority of the arbitration obeys the following rule.

Dcache-load > Hptwk > Icache -fetch > Dcache-write > LDMA > EDM

6.1 Read Request

General case: When a request grants the arbitration, the address and transaction information are pushed into command queue.

Also FCU will send fcu_biu_req_kill/lsu_biu_req_kill to kill the request sending to BIU. If the fcu/lsu request is into command queue, the kill will failed, others the kill will succeed.

For Isyn related commands, LSU unit will sent a ISYN signal to BIU, when this bit is 1, BIU will block the icache request to arbiter

6.2 Write request

The write requests can be from either LSU or LDMA. When both requests are issued simultaneously, an arbitration between these two request is made before sending to the main arbiter. When one write command got grant and command queue is not full, it will write to command queue and release grant. If the next grant owner is a read command and command queue is not full, then it can write to command queue. If the next grant owner is a write command, it need to wait previous write command writing data completed.



7 Request queue

Request queue comprises a command queue and a data queue. When a request gets the arbitration, its corresponding information are pushed into request queue. The BU in condor supports three type of operation modes, synchronous, asynchronous, and both. Therefore, the size of request queue may need to be adjusted accordingly.

7.1 Command queue

Command queue contain information of request. It consists of non-cachable, non-bufferable, ls-lock, write-enable, read ID, Icache request id, command addr[31:2], ahb-type (single increment,wrap).

40	39	38	37:34	33	32:3	2:1	0
Nonbuf	Noncache	Lslock	We/rid	Cmd	Addr	Length	write_through

Nonbuf: The command is non-bufferable.

Noncache: The command is non-cacheable.

Lslock : The command is Load-store lock command.

We/rid: When the command is write and not Load-store lock command, it

mean write enable, others mean read id.

Cmd : 1: write command ; 0 : read command.

Length : 2'b00 : single word; 2'b01 : 2 word ; 2b'10 : 4word ; 2'b11 : 8word

External cache line option: write-through or write back



7.2 Data queue

Write Data queue just contains write data. But read data queue will contain error status when ls-lock command needs to be checked whether the write operation succeeds or not.

The fifo depth of request queue

Clock type	Command queue	Wdata queue	Rdata queue
Sync	4	1 or 4 words	1 word
Async or Sync-Async	4	CL32:8words;	CL32:8words;
		CL16:4words	CL16:4words

CL32: Cacheline32B; CL16:Cacheline16B

7.3 Synchronous and asynchronous

There has a hardcore-configuration in Condor. The Synchronous and asynchronous will exist at the same time. So there need a pin sync_mode (1:syn; 0: asyn) to indicate which mode is operation.



8 Bus Wrapper

8.1 AHB Bus

8.1.1 Command Decoder

The information stored in request queue needs to be decoded and translated to meet the bus protocol. For AHB protocol, the write-enable will be translated to ahb_addr[1:0] and ahb_size, Ahb_addr and Ahb_size will be discussed in 10.6.1.3.

Transfer length	Cacheline16B:011(incr4); Cacheline32B:101(incr8)
\mathcal{C}	

8.1.2 Retry control

When the AHB bus responses retry operation, it needs some control logics to remember information of the previous request.(Ongoing request?) Because Htrans , Hburst, Hsize, Haddr[5:0] have already had registers to control bus transaction. So it only needs additional registers to store Haddr[31:6]. This is because the command in command fifo will be popped out at the last data cycle instead of starting of command.



8.1.3 Alignment control

In condor, the address will have un-alignment case. BIU needs to solve the case. There are 3 cases of un-alignment transaction. The first one is 1110, the second is 0111, and the third is 0110. These 3 cases need 2 AHB cycles to complete due to AHB hasn't byte enable information. For 1110 case, the first cycle is Haddr[1:0] = 2'b01, Hsize = byte; the second cycle is Haddr[1:0] = 2'b10, Hsize = hword. For 0111 case, the first cycle is Haddr[1:0]=2'b00, Hszie = hword; the second cycle is Haddr [1:0] = 2'b10, Hsize = byte; the second cycle is Haddr[1:0]=2'b10, Hsize = byte; the second cycle is Haddr[1:0]=2'b10, Hsize = byte.

Write Enable	First cycle	Second cycle
0110	Haddr:2'b01;Hsize:byte	Haddr:2'b10;Hsize:byte
0111	Haddr:2'b00;Hsize:hword	Haddr:2'b10;Hsize:byte
1110	Haddr:2'b01;Hsize:byte	Haddr:2'b10;Hsize:hword

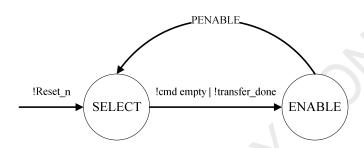
8.2 AHB-lite Bus

The different of AHB-lite bus with AHB bus is that AHB-lite bus doesn't has split/retry transfer. So AHB-lite bus implement just need to remove retry control of AHB bus. Because AHB-lite bus doesn't support split/retry transfer, so it doesn't need to remember address[31:6] and also remove retry control. AHB-lite bus can reduce 500 gate counts compared with AHB bus.



8.3 **APB Bus**

The APB bus implementation uses a one bit state machine to control. The select state will assert the APB PSEL signal. The enable state will assert APB PENABLE single. Once PSEL signal is asserted, it will keep until data counter is equal to zero. PENABLE will only be asserted on PENABLE status.



8.4 AHB2 configuration

	AHB_1	AHB_2
With LDMA feature	FCU request,	Others request
	LDMA – FCU request	Command queue: 4
	Command queue: 2	
Without LDMA feature	FCU request	Others request
	Command queue: 1	Command queue: 4

35 **BIU Design SPEC** 錯誤! 找不到參照來源。 Version 錯誤! 找不到參照來源。6 07/09/2007

Andes Technology Confidential

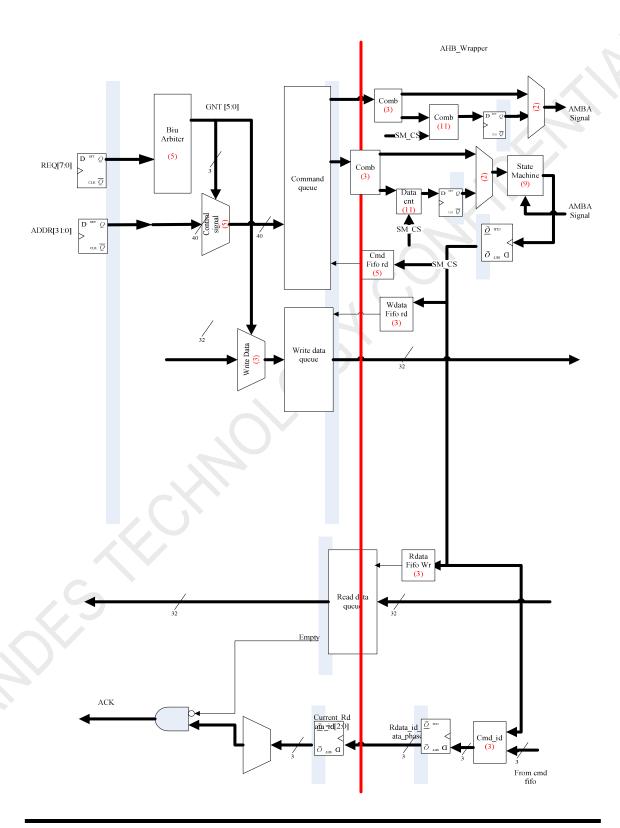


9 State Machine





10 Data Path





11 Verification Plan

- 1. Each module interface protocol test. (FCU;LSU....).
- 2. Each wrapper interface protocol test. (AHB;AHB-lite;APB).
- 3. State machine fully test.(Test every condition).
- 4. Synchronous and asynchronous clock domain test.
- 5. FIFO test. (Test when FIFO full and empty has problem?)
- 6. Test retry and retry with error/early terminate coming.
- 7. Test alignment control and alignment with error/early terminate coming.
- 7.1 Test alignment with byte enable: 1110;0111;0110.
- 8. Test grant back to back condition. And before master has retry/error case.
- 9. Test single with multi-world command sequence with different variance (single-multi-single-multi-multi.....)
- 10. Test different command length: 1.2.4.8 words.