NetSpeed AHB converter design spec (requirements, spec and architecture)

There are 2 converters encompassed by this design spec.

1. AHB Master to AXI Master converter - called ahb2axi
2. AXI Master to AHB Slave converter – called axi2ahb

Table of Contents

[Top level converter diagrams 3](#_Toc378195664)

[Open issues: 4](#_Toc378195665)

[TODO 4](#_Toc378195666)

[Schedule 5](#_Toc378195667)

[Project scope/requirements. NS = NetSpeed, BS = Bay2Sierra 6](#_Toc378195668)

[verification requirements 6](#_Toc378195669)

[Block level verification env 6](#_Toc378195670)

[Design coding guidelines 7](#_Toc378195671)

[Cadence tool flow 7](#_Toc378195672)

[Design basic assumptions 8](#_Toc378195673)

[RING BUS 8](#_Toc378195674)

[AHB2AXI – AHB Master to AXI converter 10](#_Toc378195675)

[Feature summary: 10](#_Toc378195676)

[AHB2AXI block diagram 11](#_Toc378195677)

[Parameters supported (and controlled by NOCStudio) 12](#_Toc378195678)

[NOC Studio dependancies 13](#_Toc378195679)

[IO listing and mapping details 13](#_Toc378195680)

[CSR – Control Status Registers 15](#_Toc378195681)

[special design considerations (ie. IO rate calculations, tricky features/arch support). 16](#_Toc378195682)

[ahb2axi FSM 16](#_Toc378195683)

[sequence implementation of writes and reads 17](#_Toc378195684)

[Datapath control mapping 18](#_Toc378195685)

[HPROT to AxCACHE/AxPROT mapping 19](#_Toc378195686)

[Synthesis/timing/backend notes 19](#_Toc378195687)

[Latency 19](#_Toc378195688)

[Area(Gate count)/Timing 19](#_Toc378195689)

[AXI to AHB slave converter (axi2ahb) 21](#_Toc378195690)

[Feature summary: 21](#_Toc378195691)

[AXI2AHB block diagram 21](#_Toc378195692)

[Parameters supported (and controlled by NOCStudio) 23](#_Toc378195693)

[NOC Studio dependencies 23](#_Toc378195694)

[IO listing and mapping details 23](#_Toc378195695)

[CSR – Control Status Registers 26](#_Toc378195696)

[special design considerations (ie. IO rate calculations, tricky features/arch support). 26](#_Toc378195697)

[axi\_req FSM 26](#_Toc378195698)

[ahb\_seq FSM 27](#_Toc378195699)

[Ring bus IF to AHB 27](#_Toc378195700)

[Datapath control mapping 27](#_Toc378195701)

[AxCACHE/AxPROT to HPROT mapping 29](#_Toc378195702)

[Synthesis/timing/backend notes 29](#_Toc378195703)

[Latency 29](#_Toc378195704)

[Area(Gate count)/Timing 29](#_Toc378195705)

# Top level converter diagrams

AHB Master

AHB Master to AXI

converter

AHB Master

AHB arbiter

AHB Slave

AHB decoder

AHB Master to AXI

converter

AHB Master

AHB Master

Figure 1: Top level diagram. B2S design yellow blocks

# Open issues:

* Can we just use single transfers on AXI2AHB to limit Write buffering and buffering latency or do we need to be able to create larger bursts?
* Can we assume all AXI requests are constrained to be within a 64B cache boundary? No way of straddeling 1Kb AHB cachel line boundary?
* What will NOC decoder do if a request straddles address regions between 2 Slave devices? Will it break the request up?
* For AXI2AHB, can I assume no deadlock can exist between request being serialized? ie. write data will not get stuck behind some other upcoming requests write data? Seems unlikely but want to make sure this assumption is expected from NOC behavior.
* AHB2AXI – can we directly (combinatorially) use HWRITE to feed into the HREADY FF? If not then we will always need to take 1 extra cycle to determine if it is a write that we can assert HREADY and if read that it needs to be terminated with a SPLIT or stall
* review AxCACHE/AxQOS/AxPROT mapping from AXI to AHB PROT[3:0] – review
* review Datapath control mapping on AHB slave

## TODO

1. ahb2axi\_tran FSM, ahb2axi\_rdreq FSM, ahb2axi\_rsp FSM
2. axi\_req FSM
3. ahb\_seq FSM – partial word/odd burst size, wstrobe resolution
4. ring bus to AHB

# Schedule

**Any need to sequence features/deliverables?**

1. Design and project spec requirements/goal/scope, verification effort and architecture/strategy defined , define parameters, tesplan definition
2. block verification env created – using Cadence VIP, protocol check –Cadence
3. initial rtl developed, assertion checker
4. block testplan and testcases created
5. rtl feature complete
6. transactor updated with AHB instructions and driver/slave
7. NOCStudio debug effort
8. Back end synthesis/coverage effort
9. support and maintenance – TBD should be 10 avg hours for 2-3 months (? should be reasonable)
10. 100-140 hours

4) 80 – 100 hours

2) 80-100 hours

5) 120-160 hours

3) 80 – 100 hours

7) 80 – 160 hours

6) 80 – 120 hours

8) 80 – 100 hours

end point: 10- 15 weeks

start point

Figure 2: schedule – total number of project hours estimated: 700 – 980

# Project scope/requirements. NS = NetSpeed, BS = Bay2Sierra

### verification requirements

* + define AHB instructions to be generated by Nemesis
  + Nemesis to be modified by NS. What kind of errors need to be inserted?
  + Modify transactor to take AHB instruction set and generate AHB request
  + Create AHB slave models – memory only or also use FIFO/ IO?
  + block level environment for master and slave converters – using Cadence VIP (creates an impartial 3rd party check)
  + Define block level tetscases. implement and ensure running clean
  + NOCStudio – to be updated by NS
  + NOCStudio tradeoffs and NOC properties are not understood by B2S and has a large potential for issues that are not found until late in the development cycle. Currently this is NS responsibility to review scope and requirements and make sure they are consistent with requirements/scope
  + Coherency checker additions (if any) will be done by NS.
  + AHB Protocol checker for both converters will be from Cadence/Arm and will be linked to design by B2S
  + Assertion checks added in files based on functional hierarchy by B2S
  + Create method and reasonable subset of configurations to use for block level verification

### Block level verification env



Figure 3 - bloxk level verification env

### Design coding guidelines

* + Use local parameters instead of defines
  + Use lower case except for parameters and AMBA specific signals
  + top level
    - use src prefix on IO – ie ahb2axi\_\*
    - All top level module instantiations must have comments for each port // O // I

### Cadence tool flow

* + HAL used for linting. NS provide standard filter file
  + Synthesis to be run by B2S for a few expected configurations of the design – exact CFG TBD. template script to be provided by NS
  + Coverage – done by B2S on block level.. goal >95%

### Design basic assumptions

* + Synchronous designs
    - Clock will an input and we can assume all IO except for ring bus are sync to this same clock
    - Reset needs to be synchronized onto Clock does not need any additional delay
  + NOC can assume to have infinite bandwidth (eventhough any handshaking need to be respected)
  + Only support AXI4 (No support for AXI3)
  + No watchdog timers needed – if deadlock occurs then NOC will detect and system will reset sub-modules
  + No Low power AXI support but AXI2AHB will have CSRs which understand what REGIONS (Slaves) are powered down and if requests come for these regions will issue INTR
  + AxADDR – same size HADDR based on parameter
  + AxREGION used in AXI2AHB converter to decode which of the 16 AHB slaves a request is for. Region can be assume to only target existing slaves. Region not used in AHB2AXI converter.
  + AxQOS not supported
  + Max freq 400Mhz
  + all IO registered
  + W/RUSER not used
  + Ring bus must be added to both converters.
  + datapath width of 8,16,32,64,128,256,512b
    - Datapath remains unchanged on each converter’s system – no support for narrow->wide or wide->narrow
    - “FIXED” AXI bursts NOT supported

## RING BUS

|  |  |  |
| --- | --- | --- |
| **IO name** | **Dir** | **notes** |
| **External IO** |  |  |
| CLK\_ring | Input |  |
| RST\_ring | Input |  |
| CMD\_in[2:0] | Input | 000 – NOP  001 – RD  010 – WR  011 – WRD  100 – ACK  101 – ACK CLAIM  110 – Decode ERR  111 - RSVD |
| DATA\_in  [P\_RING\_DATA\_WIDTH-1:0] | Output |  |
| CMD\_out[2:0] | Output |  |
| DATA\_out  [P\_RING\_DATA\_WIDTH-1:0] |  |  |
| **Internal CSR interface** |  |  |
| ring\_en | Input | Valid ring bus request |
| ring\_rnw | Input | Read Not write |
| ring\_addr[?:0] | Input | Address for request |
| ring\_wr\_data[63:0] | Input | Write data |
| ring\_rd\_data[63:0] | Output | Read data |
| ring\_resp | Output | valid address |
| ring\_rdy | Output | response valid |

## AHB2AXI – AHB Master to AXI converter

## Feature summary:

* Assume single Master with up to 16 virtualized master ports
* Modularize arbiter and attempt to use AHB like interface internally
  + round robin empty nest algorithm
* support all AHB commands
  + Does not need to use RETRY
  + LOCK only ensures Arbiter doesn’t change until LOCK de-asserted
  + AXI ID mapped to connection number
  + no specific early burst termination support needed.
* AHB – lite – no SPLIT support, assume single master to single slave – all transactions become serialized.
* SPLIT functionality
  + ID will be created based on source master 0 to 16
  + Write – minimal buffering - 64B packing buffer plus few word to support 0 delay handshaking to NOC
  + Read – Requests submitted in parallel to AXI. minimal buffer – few words to support 0 delay handshaking to NOC
    - * Defined burst length – once read data returned, burst completed.
      * Undefined burst length – create cache line (64B) pre-fetch. Assume always pre-fetchable. If more data is needed after pre-fetch data is consumed then SPLIT issued.

## AHB2AXI block diagram



Figure 4: AHB2AXI block diagram

## Parameters supported (and controlled by NOCStudio)

|  |  |  |
| --- | --- | --- |
| Parameter name | short description | implementation notes |
| P\_NUM\_MASTERS | Number of Virtual Masters supported – up to 16  default: 16 |  |
| P\_AHB\_LITE | only support AHB-LITE  default: 0 | 1: implement AHB-lite – all transactions serializes  0: full AHB functionality with SPLIT is implemented which allows multiple parallel virtual master requests to be outstanding |
| P\_ADDR\_WIDTH | width of address bus  default: 32 |  |
| P\_DATA\_WIDTH | width of data bus  default 32 | Must be in the following set:  8, 16, 32, 64, 128, 256, 512b |
| P\_NUMWRDATA\_ENTRIES | amount of buffering instantiated for write data path in terms of P\_DATA\_WIDTH  default: 66 | requires min of 64B+couple of entries for handshaking delay |
| P\_NUMRDDATA\_ENTRIES | amount of buffering instantiated for read data path in terms of P\_DATA\_WIDTH  default: 2 | requires couple of entries for handshaking delay |
| P\_AHB\_LE | AHB litte endian  default: 1 | 1 – little endian  0 – big endian |
| P\_AxPROT\_FORCE | default value 3’b000 | 1: forces value to be always asserted  0: internal logic (if any) will supersede |
| P\_AxPROT\_DEFAULTS | default values of AxPROT[2:0] signal  {instruction/data access, Nonsecure/secure acess,  privileged access} = 3’b000  [1] should in most cases be set to 1’b1 |  |
| P\_AxCACHE\_FORCE | default value 4’b0000 | 1: forces value to be always asserted  0: internal logic (if any) will supersede |
| P\_AxCACHE\_DEFAULTS | default values of AxCACHE[3:0] signal  {Other allocate,  Allocate,  Modifiable,  Bufferable} = 4’b0000 |  |

Table 1: AHB2AXI parameters

## NOC Studio dependancies

Dependency graph: all transactions are serialized for each virtual master.

## IO listing and mapping details

|  |  |  |  |
| --- | --- | --- | --- |
| **IO name** | **Dir** | **Mapping** | **notes** |
| **Global** |  |  |  |
| CLK | Input | HCLK, ACLK | Same clock used for both AHB and AXI sides |
| RST | Input | HRESETN, SresetN | Same reset used for both AHB and AXI sides – need to sync to clock? |
| **AHB Master** |  |  |  |
| HWRITE | Input | if 1 enable AWVALID else ARVALID. WVALID needs to be enabled on AXI to transmit data and strobes | transfer direction.1 – write, 0- read |
| HTRANS[1:0] | Input |  | Transfer type – IDLE, BUSY, NONSEQ, SEQ |
| HSIZE[2:0] | Input | if hwrite ==1 🡪 AWSIZE else🡪ARSize. See special design considerations below | transfer size – 8, 16, 32, 64, 128, 256, 512b |
| HBURST[2:0] | Input | if hwrite==1 🡪AWBURST else 🡪ARBURST See special design considerations below | Burst operation, - SINGLE, INCR – unspecified length, WRAP4, INCR4, WRAP8, INCR8, WRAP16, INCR16 |
| HPROT[3:0] | Input | See special design considerations below | protection control signals – see details below  [3] – cacheable  [2] – bufferable  [1] – privileged  [0] – data/opcode |
| HADDR[P\_ADDR\_WIDTH-1:0] | Input | if hwrite==1🡪 AWADDR  else 🡪ARADDR |  |
| HWDATA[P\_DATA\_WIDTH-1:0] | Input | AWDATA | write data bus |
| HREADY | Output | RVALID | transfer Done |
| HRESP[1:0] | Output | AXIRESP/AXIWRESP | transfer response – OKAY, ERROR, RETRY,SPLIT |
| HRDATA[P\_DATA\_WIDTH-1:0] | Output | RVALID=1 🡪AXIRDATA | read data bus |
| **Arbiter** |  |  |  |
| HBUSTREQ[P\_NUM\_MASTERS-1:0] | Input |  | master asking for request access for one of its virtual ports |
| HLOCK[P\_NUM\_MASTERS-1:0] | Input |  | keeps arbiter until request line de-asserted |
| HGRANT[P\_NUM\_MASTERS-1:0] | Output |  | Virtual master granted access to use bus |
| **Ring bus – CSR access** |  |  |  |
| CLK\_ring | Input | N/A |  |
| RST\_ring | Input | N/A |  |
| CMD\_in[2:0] | Input | N/A | 000 – NOP  001 – RD  010 – WR  011 – WRD  100 – ACK  101 – ACK CLAIM  110 – Decode ERR  111 - RSVD |
| DATA\_in  [P\_RING\_DATA\_WIDTH-1:0] | Output | N/A |  |
| CMD\_out[2:0] | Output | N/A |  |
| DATA\_out  [P\_RING\_DATA\_WIDTH-1:0] |  | N/A |  |
| **AXI Master** |  |  |  |
| **Write address channel** |  |  |  |
| AWID[3:0] | Output |  | Write channel ID - use Master number |
| AWADDR[31:0] | Output |  | Write address |
| AWLEN[3:0] | Output | See special design considerations below | Burst length. |
| AWSIZE[2:0] | Output | See special design considerations below | Burst size. |
| AWBURST[1:0] | Output | See special design considerations below | Burst type |
| AWLOCK[1:0] | Output |  | Lock type. |
| AWCACHE[3:0] | Output | See special design considerations below | Cache type. |
| AWPROT[2:0] | Output | See special design considerations below | Protection type. |
| AWVALID | Output |  | Write address valid. |
| AWREADY | Input |  | Write address ready. |
| **Write data Channel** |  |  |  |
| WID[3:0] | Output |  | Write ID tag – use Master number |
| WDATA[31:0] | Output |  | Write data. |
| WSTRB[3:0] | Output |  | Write strobes. |
| WLAST | Output |  | Write last. |
| WVALID | Output |  | Write valid |
| WREADY | Input |  | Write ready |
| **Write response channel signals** |  |  |  |
| BID[3:0] | Input |  | Response ID. |
| BRESP[1:0] | Input |  | Write response |
| BVALID | Input |  | Write response valid. |
| BREADY | Output |  | Response ready. |
| **Read address channel signals** |  |  |  |
| ARID[3:0] | Output |  | Read address ID – use Master number |
| ARADDR[31:0] | Output |  | Read address |
| ARLEN[3:0] | Output | See special design considerations below | Burst length. |
| ARSIZE[2:0] | Output | See special design considerations below | Burst size. |
| ARBURST[1:0] | Output | See special design considerations below | Burst type |
| ARLOCK[1:0] | Output |  | Lock type – only used to keep arbiter from changing |
| ARCACHE[3:0] | Output | See special design considerations below | Cache type. |
| ARPROT[2:0] | Output |  | Protection type. |
| ARVALID | Output |  | Read request valid |
| ARREADY | Input |  | Read request accepted |
| **Read data channel signals** |  |  |  |
| RID[3:0] | Input |  | Read ID tag – use Master number |
| RDATA[31:0] | Input |  | Read data |
| RRESP[1:0] | Input |  | Read response. |
| RLAST | Input |  | Read last. |
| RVALID | Input |  | Read valid. |
| RREADY | Output |  | Read ready. |

Table 2 AHB2AXI IO and mapping

## CSR – Control Status Registers

|  |  |  |  |
| --- | --- | --- | --- |
| CSR Name | field name | Address | Description |
| TBD[63:0] | TBD[15:0] | 0x0 | currently undefined |
| rsvd[63:16] |  |
|  |  |  |  |
|  |  |  |  |

Table 3: CSRs

## special design considerations (ie. IO rate calculations, tricky features/arch support).

### ahb2axi FSM

FSM description and Operational details for FSM.

FIXME

### arbiter

round robbin empty nest

IO

|  |  |  |
| --- | --- | --- |
| **IO name** | **Dir** | **Description** |
| **from external IO** |  |  |
| HBUSTREQ[P\_NUM\_MASTERS-1:0] | Input | master asking for request access for one of its virtual ports |
| HLOCK[P\_NUM\_MASTERS-1:0] | Input | keeps arbiter until request line de-asserted |
| HGRANT[P\_NUM\_MASTERS-1:0] | Output | Virtual master granted access to use bus. equivalent to HSEL[P\_NUM\_MASTERS-1:0] |
| HTRANS[1:0] | Input | If SPLIT response is seen then need to block any further grants to this virtual master until see HSPLIT clear signal. |
| **from internal ahb2axi logic** |  |  |
| HMASTER[3:0] | Output | indicates which virtual master is currently in control of the bus |
| HMASTLOCK | Output | indicates if current virtual master is asserting HLOCK – not used by logic |
| HSPLIT[P\_NUM\_MASTERS-1:0] | Input | 1 cycle high pulse to clear state that is currently blocking any new requests from the virtual master due to previous SPLIT response from AHB2AXI block |

### sequence implementation of writes and reads



Figure 5 – write and read sequencing

### Datapath control mapping

|  |  |  |  |
| --- | --- | --- | --- |
| **AHB** | **Values** | **AXI** | **Map** |
| HADDR  [P\_ADDR\_WIDTH-1:0] | Aligned based on HSIZE  Will not cross 1KB boundary | AxADDR  [P\_ADDR\_WIDTH-1:0]  WSTROBE  [LOG2(P\_DATA\_WIDTH)-1:0] | direct map  de-assert bits associated with HADDR |
| HSIZE[2:0] | 000 – 1B | AxSIZE[2:0] | direct map |
| 001 – 2B |
| 010 – 4B |
| 011 – 8B |
| 100 – 16B |
| 101 – 32B |
| 110 – 64B |
| 111 – 128B – not supported  rtl assertion |
| HBURST[2:0] | 000 – Single transfer | {AxLEN[7:0],  AxBURST[1:0]}   1. FIXED – not supported 2. INCR 3. WRAP 4. WRAP | 0,1 |
| 001 - INCR - unspecified length | posted and non-posted buffer up to 64B before issuing AXI request, 1 |
| 010 – WRAP4 – 4-beat wrapping burst | 3,  3 |
| 011 – INCR4 – 4-beat incrementing burst | 3,  1 |
| 100 – WRAP8 – 8-beat wrapping burst | 7,  3 |
| 101 – INCR8 – 8-beat incrementing burst | 7,  1 |
| 110 – WRAP16 – 16-beat wrapping burst | 15,  3 |
| 111 – INCR16 – 16-beat incrementing burst | 15,  1 |
| Endianess | Little (P\_AHB\_LE=1) | no action | byte invariance |
| Big (P\_AHB\_LE=0) | swap byte position | swap byte position |

Notes:

1. Assume AHB/AXI data bus is same width
2. RTL assertion needed to ensure HSIZE <= Datapath width
3. HSIZE \* HBURST (from aligned address) determine wrap address
4. Assertion check to ensure BUSY not asserted for anything but INCR
5. Assertion for AxLEN = 0

#### Timing diagrams (waveforms) for various corner cases

FIXME

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

### HPROT to AxCACHE/AxPROT mapping

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AHB | value | description | AXI map | description | comment |
| HPROT[0] | 0 | Opcode fetch | AxPROT[2]=1 | Instruction access |  |
|  | 1 | Data access | AxPROT[2]=0 | Data access |  |
| HPROT[1] | 0 | User access | AxPROT[0]=0 | Unprivileged access |  |
|  | 1 | Privileged access | AxPROT[0]=1 | Privileged access |  |
| HPROT[2] | 0 | Not bufferable | AxCACHE[0] = 0 | Bufferable |  |
|  | 1 | Bufferable | AxCACHE[0] = 1 |  |
| HPROT[3] | 0 | Not cacheable | AxCACHE[3:2] = 0 | 3: Other allocate  2: Allocate |  |
|  | 1 | Cacheable | AxCACHE[3:2] = 3 |  |
|  |  |  | AxPROT[1] | 0 - secure access  1 – Non secure access | default to AxPROT\_DEFAULTS parameter |
|  |  |  | AxCACHE[1] | Modifiable | default to AxCACHE\_DEFAULTS Parameter |

Note:

* Xilinx bridge asuumes HPROT[3] is always 1’b0 – It assume all requests crossing AHB-AXI interface are non-cacheable – do we want to follow suite? Or do we like the mapping in the table above?

## Synthesis/timing/backend notes

* All IO are registered
* Ring bus connections are all async to rest of core logic

### Latency

FIXME: Create table of latency values for various scenarios

### Area(Gate count)/Timing

FIXME: Create table with various typical parameter configurations

Worst case area (parameter setting ==??):

Worst case timing(parameter setting ==??):

Best case area (parameter setting ==??):

Best case timing(parameter setting ==??):

# AXI to AHB slave converter (axi2ahb)

## Feature summary:

* Decoder - Use direct mapping from REGION value
* All transactions serialized
* AHB commands limited by NOC/AXI
  + undefined length INCR transactions will never occur
  + LOCK will never occur
  + SPLIT are legal
    - Can occur anywhere in a burst and must be replayed. (– affectively create early burst terminate)
    - Entire request will serialized
* Addressing/request sequencing
  + Support 1KB address boundaries for AHB
  + SPLIT early terminate and replay of request
  + SIZE is always the same value as the datapath width
  + LEN of any arbitrary value
  + WSTRB of any arbitrary value
  + Address alignment of any arbitrary value
* Ring bus support for CSRs as well as direct access to AHB slaves
  + 16b CSR which details which of the 16 possible slaves are powered-up (1). If request comes to a powered-down device then interrupt needs to be asserted

## AXI2AHB block diagram



Figure 6 : AXI2AHB block diagram

## Parameters supported (and controlled by NOCStudio)

|  |  |  |
| --- | --- | --- |
| Parameter name | short description | implementation notes |
| P\_NUM\_SLAVES | Number of Slaves supported – up to 16  default: 16 |  |
| P\_ADDR\_WIDTH | width of address bus  default: 32 |  |
| P\_DATA\_WIDTH | width of data bus  default: 32 | Must be in the following set:  8, 16, 32, 64, 128, 256, 512b |
| P\_NUMWRDATA\_ENTRIES | amount of buffering instantiated for write data path in terms of P\_DATA\_WIDTH  default: 66 | requires couple of entries for handshaking delay |
| P\_NUMRDDATA\_ENTRIES | amount of buffering instantiated for read data path in terms of P\_DATA\_WIDTH  default: 2 | requires couple of entries for handshaking delay |
| P\_AHB\_LE | AHB litte endian  default: 1 | 1 – little endian  0 – big endian |
| P\_PROT\_FORCE | defaults to 4’b0000 | 1: forces value to be always asserted  0: internal logic (if any) will supersede |
| P\_PROT\_DEFAULTS | default values of PROT[3:0] signal = 4’b0000 (non-cacheable, nun-buffered, privileged, data access) |  |

Table 4: AXI2AHB parameters

## NOC Studio dependencies

Dependency graph: all transactions are serialized through AXI node

## IO listing and mapping details

|  |  |  |  |
| --- | --- | --- | --- |
| **IO name** | **Dir** | **Mapping** | **notes** |
| **Global** |  |  |  |
| CLK | Input | HCLK, ACLK | Same clock used for both AHB and AXI sides |
| RST | Input | HRESETN, SresetN | Same reset used for both AHB and AXI sides – need to sync to clock? |
| INTR | Output |  | Asserted when request received to an unpowered AHB slave |
| **Ring bus – CSR access** |  |  |  |
| FIXME |  |  |  |
|  |  |  |  |
| **AHB Master** |  |  |  |
| HWRITE[P\_NUM\_MASTERS-1:0] | Output | AWVALID or ARVALID. | transfer direction.1 – write, 0- read |
| HTRANS[1:0]  [P\_NUM\_MASTERS-1:0] | Output |  | Transfer type – IDLE, BUSY, NONSEQ, SEQ |
| HSIZE[2:0]  [P\_NUM\_MASTERS-1:0] | Output | AWSIZE or ARSize. See special design considerations below | transfer size – 8, 16, 32, 64, 128, 256, 512b |
| HBURST[2:0]  [P\_NUM\_MASTERS-1:0] | Output | AWBURST or ARBURST See special design considerations below | Burst operation, - SINGLE, INCR – unspecified length, WRAP4, INCR4, WRAP8, INCR8, WRAP16, INCR16 |
| HPROT[3:0]  [P\_NUM\_MASTERS-1:0] | Output | See special design considerations below | protection control signals – see details below  [3] – cacheable  [2] – bufferable  [1] – privileged  [0] – data/opcode |
| HADDR[P\_ADDR\_WIDTH-1:0]  [P\_NUM\_MASTERS-1:0] | Output | AWADDR or ARADDR |  |
| HWDATA[P\_DATA\_WIDTH-1:0]  [P\_NUM\_MASTERS-1:0] | Output | AWDATA | write data bus |
| HLOCK[P\_NUM\_MASTERS-1:0] | Output |  | what do we do with this? |
| HREADY[P\_NUM\_MASTERS-1:0] | Input | RVALID | transfer Done |
| HRESP[1:0]  [P\_NUM\_MASTERS-1:0] | Input | AXIRESP/AXIWRESP | transfer response – OKAY, ERROR, RETRY,SPLIT |
| HRDATA[P\_DATA\_WIDTH-1:0]  [P\_NUM\_MASTERS-1:0] | Input | RVALID=1 🡪AXIRDATA | read data bus |
| **Ring bus – CSR access** |  |  |  |
| CLK\_ring | Input | N/A |  |
| RST\_ring | Input | N/A |  |
| CMD\_in[2:0] | Input | N/A | 000 – NOP  001 – RD  010 – WR  011 – WRD  100 – ACK  101 – ACK CLAIM  110 – Decode ERR  111 - RSVD |
| DATA\_in  [P\_RING\_DATA\_WIDTH-1:0] | Output | N/A |  |
| CMD\_out[2:0] | Output | N/A |  |
| DATA\_out  [P\_RING\_DATA\_WIDTH-1:0] |  | N/A |  |
| **AXI Master** |  |  |  |
| **Write address channel** |  |  |  |
| AWID[3:0] | Input |  | Write channel ID |
| AWADDR[31:0] | Input | HADDR | Write address |
| AWLEN[3:0] | Input | See special design considerations below | Burst length. |
| AWSIZE[2:0] | Input | See special design considerations below | Burst size. |
| AWBURST[1:0] | Input | See special design considerations below | Burst type |
| AWLOCK[1:0] | Input |  | Lock type. |
| AWCACHE[3:0] | Input |  | Cache type. |
| AWPROT[2:0] | Input |  | Protection type. |
| AWREGION[3:0] | Input | NOC creates direct decoding | AHB slave being targeted |
| AWVALID | Input |  | Write request valid. |
| AWREADY | Output | HREADY | Write request ready. |
| **Write data Channel** |  |  |  |
| WID[3:0] | Input |  | Write ID tag. |
| WDATA[31:0] | Input | HWDATA | Write data. |
| WSTRB[3:0] | Input |  | Write strobes. |
| WLAST | Input |  | Write last. |
| WVALID | Input |  | Write valid |
| WREADY | Output | HREADY | Write ready |
| **Write response channel signals** |  |  |  |
| BID[3:0] | Output |  | Response ID. |
| BRESP[1:0] | Output | HRESP | Write response |
| BVALID | Output |  | Write response valid. |
| BREADY | Input | HREADY | Response ready. |
| **Read address channel signals** |  |  |  |
| ARID[3:0] | Input |  | Read address ID. |
| ARADDR[31:0] | Input | HADDR | Read address |
| ARLEN[3:0] | Input | See special design considerations below | Burst length. |
| ARSIZE[2:0] | Input | See special design considerations below | Burst size. |
| ARBURST[1:0] | Input | See special design considerations below | Burst type |
| ARLOCK[1:0] | Input |  | Lock type. |
| ARCACHE[3:0] | Input |  | Cache type. |
| ARPROT[2:0] | Input |  | Protection type. |
| ARREGION[3:0] | Input | NOC creates direct decoding | AHB slave being targeted |
| ARVALID | Input |  | Read request valid |
| ARREADY | Output | HREADY | Read address accepted |
| **Read data channel signals** |  |  |  |
| RID[3:0] | Output |  | Read ID tag. |
| RDATA[31:0] | Output | HRDATA | Read data |
| RRESP[1:0] | Output | HRESP | Read response. |
| RLAST | Output |  | Read last. |
| RVALID | Output |  | Read valid. |
| RREADY | Input | HREADY | Read ready. |

Table 5: AXI2AHB IO and mapping

## 

## CSR – Control Status Registers

|  |  |  |  |
| --- | --- | --- | --- |
| CSR Name | field name | Address | Description |
| Slave\_properties[63:0] | Slave\_has\_power[15:0] | 0x0 | Slave has power when 1. If received request when not powered up then issue INTR |
| rsvd[63:16] |  |
|  |  |  |  |
|  |  |  |  |

Table 6: CSRs

## special design considerations (ie. IO rate calculations, tricky features/arch support).

### axi\_req FSM

Summary: FSM deals with AXI requests and passes on proper control info onto ahb\_seq FSM. All requests are dealt with serially

FSM description and Operational details for FSM.

FIXME

### ahb\_seq FSM

Summary: FSM deals with any sequencing/break-up of AXI requests onto AHB compliant requests. A complete AXI transaction will be finished even if this requires many individual AHB requests before next AXI request is accepted

Must break up AXI request into highest common denominator - writes use write buffer to ensure most efficient requests created base on wstrobe and addressing

FSM description and Operational details for FSM. – add diagrams to show corner cases

FIXME

### Ring bus IF to AHB

### Datapath control mapping

|  |  |  |  |
| --- | --- | --- | --- |
| AXI | Values | AHB | Map |
| AxADDR  [P\_ADDR\_WIDTH-1:0] | any alignment | HADDR - aligned based on HSIZE. Will not cross 1KB boundary | set by ahb\_seq FSM. |
| AxSIZE[2:0] | always equal to datapath width | HSIZE[2:0] | set by ahb\_seq FSM. Datapath width or less depending on required request sequencing |
| AxBurst[1:0] | 0-FIXED – not supported | illegal |  |
| 1- INCR | HBURST[0]=1 |  |
| 2-WRAP | HBURST[0]=0 |  |
| 3-RSVD | illegal |  |
| AxLEN[7:0] | INCR – any arbitrary number up to total request length of 64B (NOC constrains)  WRAP – only 2,4,8,16 | HBURST[2:1] | set by ahb\_seq FSM. request sequencer must limit value to 1,4,8 or 16 |
| WSTROBE  [LOG2(P\_DATA\_WIDTH)-1:0] | any arbitrary values | no direct map but input to ahb\_seq FSM | ahb\_seq FSM must buffer write data to ensure create correct request sequencing |
| Endianess | Little (P\_AHB\_LE=1) | no action | byte invariance |
| Big (P\_AHB\_LE=0) | swap byte position | swap byte position |

Notes:

1. Assume AXI/AHB data bus is same width
2. RTL assertion needed to ensure AxSIZE <= Datapath width
3. RTL assertion on AxBURST = FIXED or RSVD
4. AxSIZE \* AxBURST (from aligned address) determine wrap address
5. Assertion for AxLEN = 0AxCACHE/AxPROT to HPROT mapping
6. Assume all AXI transactions will not cross 64B address boundary and therefore each transaction will not straddle 1Kb AHB cache line boundary – check statement??!!!

#### Corner cases

|  |  |
| --- | --- |
| AXI transaction (assume 32b data bus – SIZE=4B) | AHB transaction |
| INCR burst LEN =1 WSTROBE all set | single transaction |
| INCR burst LEN =1 WSTROBE = single byte set | Single transaction with HSIZE =1B |
| INCR burst LEN =1 WSTROBE = two byte set aligned | Single transaction with HSIZE =2B |
| INCR burst LEN =1 WSTROBE = two byte set unaligned | two transactions both with HSIZE=1B |
| INCR burst LEN =1 WSTROBE = three byte set aligned | two tranactions.  1st to addr 0 with HSIZE=2B  2nd to addr 2 with HSIZE = 1B |
| INCR burst LEN =1 WSTROBE = three byte set unaligned | two tranactions.  1st to addr 1 with HSIZE=1B  2nd to addr 2 with HSIZE = 2B |
| Read INCR burst LEN>1 WSTROBE all set | Map to 4,8,16 or undefined length transfers. |
| Write INCR burst LEN>1 WSTROBE – sparse or all set | If we devolve to writing only a single transfer at a time then this devolves into LEN=1 scenarios above??? Otherwise For writes will need to buffer data to ensure no WSTROBES de-asserted before issuing command – unless we just default to single transfers??? |
| Read WRAP burst LEN 4,8,16 WSTROBE all set | Map to WRAP 4,8,16 |
| Read WRAP burst LEN 2 WSTROBE all set | 2 single transactions |
| Write WRAP burst LEN 2,4,8,16 WSTROBE sparse or all set | Writes - Break down into single transfers |

#### Timing diagrams (waveforms) for various corner cases

### AxCACHE/AxPROT to HPROT mapping

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AXI | value | description | AHB map | description | comment |
| AxCACHE[0] | 0 | bufferable | HPROT[2] = 0 | Not bufferable |  |
| 1 | HPROT[2]= 1 | Bufferable |  |
| AxCACHE[1] | 0 | Modifiable | no mapping |  |  |
|  | 1 | no mapping |  |  |
| AxCACHE[2] | 0 | Allocate | HPROT[3] = AxCACHE[3] |  | HPROT[3] = |AxCACHE[3:2] |
|  | 1 | HPROT[3] = 1 |  |  |
| AxCACHE[3] | 0 | Other allocate | HPROT[3] = AxCACHE[2] |  |  |
|  | 1 | HPROT[3] = 1 |  |  |
| AxPROT[0] | 0 | Unprivileged access | HPROT[1]=0 | User access |  |
| 1 | Privileged access | HPROT[1]=1 | Privileged access |  |
| AxPROT[1] | 0 | Secure access | no mapping |  |  |
|  | 1 | Non-secure access | no mapping |  |  |
| AxPROT[2] | 0 | Data access | HPROT[0]=0 | Data access |  |
| 1 | Instruction access | HPROT[0]=1 | Opcode fetch |  |

## Synthesis/timing/backend notes

* All IO are registered
* Ring bus connections are all async to rest of core logic

### Latency

FIXME: Create table of latency values for various scenarios

### Area(Gate count)/Timing

FIXME: Create table with various typical parameter configurations

Worst case area (parameter setting ==??):

Worst case timing(parameter setting ==??):

Best case area (parameter setting ==??):

Best case timing(parameter setting ==??):