

牛芯半导体（深圳）有限公司

**KNIULINK Co., Ltd.**

NS69993 Design Notes for Control Digital Subblock

Version 1.5

April 21, 2022

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Ver. # | Rev. Date | Rev. By | Comment |
| 1.0 | 2022/2/25 | CaoHuiyang | Initial version |
| 1.1 | 2022/3/05 | CaoHuiyang | Add inter-structure and function |
| 1.2 | 2022/3/10 | CaoHuiyang | Supplement to reg\_arbt |
| 1.3 | 2022/4/12 | CaoHuiyang | Update and content modify |
| 1.4 | 2022/4/18 | CaoHuiyang | Add some function and change document structure |
| 1.5 | 2022/4/21 | CaoHuiyang | Add the chapter of riscv/bootloader |

**Contents**

[NS69993 Design Notes for Control Digital Subblock 0](#_Toc25733)

[1 Spec/Function Requirement/Function Description 3](#_Toc8870)

[2 Signal Definition 4](#_Toc8344)

[3 Implementation Details 12](#_Toc32627)

[3.1 CMN block 12](#_Toc5051)

[3.1.1 Clock & Reset 12](#_Toc21382)

[3.1.2 Hardware Implementation 12](#_Toc12464)

[3.2 RISCV controller 28](#_Toc19523)

[3.3 Exteral RAM/ROM 29](#_Toc1606)

[3.3.1 Exteral RAM 29](#_Toc52)

[3.3.2 Exteral ROM 30](#_Toc28663)

[3.4 Hardware summarize 31](#_Toc18362)

[4 User guide 33](#_Toc19519)

[4.1 Configuration Requirements 33](#_Toc22646)

[4.3 Registers 33](#_Toc28397)

[5 Special Note 33](#_Toc8426)

[6 FPGA Verification 33](#_Toc30396)

[7 Test Mode Strategy 33](#_Toc3609)

[8 Simulation List 33](#_Toc10487)

# Spec/Function Requirement/Function Description

* Registers Management and Mapping
* Lanes.fsm’s Memory and Memory Management
* Jtag/cr\_apb\_interface mux to cr\_interface
* Extend rom and ram function
* Fsm and riscv mux and initialization

# Signal Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Signal name** | | **I/O** | **From** | **To** | **Description/Function** |
| cr\_ext\_clk | | input | aon |  | cr external clock |
| cr\_ext\_clk\_rst | | input |  | cr external clock reset· |
| cr\_clk | | input |  | cr clock |
| cr\_rst | | input |  | cr reset |
| phy\_func\_reset | | output |  | aon |  |
|  | |  |  |  |  |
| ref\_range | | input | pcs\_raw |  |  |
| rx\_term\_offset[4:0] | | input | pcs\_raw |  | rx terminating resistor offset |
| txup\_term\_offset[8:0] | | input |  | rx pull-up resistor offset |
| txdn term\_offset[8:0] | | input |  | rx pull-down resistor offset |
| rtune\_req | | input |  | rtune require |
| fw\_pwrup done\_r | | output |  | aon | Send fw\_pwrup\_done\_r to AON cmn for MPLL force/skip logic |
|  | |  |  |  |  |
| fw\_rdy | | output |  | pcs\_raw\_lane | Generate fw\_rdy signal to lanes  always high |
| pma\_rx\_term\_offset  [4:0] | | output |  | pcs\_raw->PMA | rx terminating resistor to pma |
| pma\_txup\_term\_offset[8:0] | | output |  | rx pull-up resistor offset to pma |
| pma\_txdn\_term\_offset[8:0] | | output |  | rx pull-down resistor offset to pma |
|  | |  |  |  |  |
| pma\_rtune\_req | | output |  | pcs\_raw->PMA | Rtune require to pma |
|  | |  |  |  |  |
| cr\_para seli | | input | aon |  | from aon block--jtag and apb\_cr mux control signal(bus switch signal) |
| cr\_para\_addr[16:0] | | input | pcs\_raw |  | CREG parallel interface（cr bus）  (from apb or jtag) |
| cr\_para\_wr\_en | | input |  |
| cr\_para\_wr\_data[16:0] | | input |  |
| cr\_para\_rd\_en | | input |  |
| cr\_para\_rd\_data[16:0] | | output |  | pcs\_raw |
| cr\_para\_ack | | output |  |
| **Signal name** | | **I/O** | **From** | **To** | **Description/Function** |
| cr\_cmn\_aon sel[31:0] | output | |  | pcsraw\_aon\_cmn | aon register select control（decoding generated select to the registers in aon module） |
| cr\_cmn\_aon\_sel2[31:0] | output | |  |
| cr\_cmn\_aon\_rd\_data[15:0] | input | | pcsraw\_aon\_cmn |  | aon register read back data（read back data from the registers in aon module） |
| Jtag\_trst\_n | input | | pcs\_raw |  | Jtag reset |
| jtag\_clk | input | | pcsraw\_aon\_cmn |  | Jtag clock |
| jtag\_clk\_n | input | |  | Jtag clock |
| jtag\_tms | input | | pcs\_raw |  | Jtag bus |
| jtag tdi | input | |  |
| jtag tdo | output | |  | pcs\_raw |
| jtag tdo\_en | output | |  |
| jtag\_apb\_sel | output | |  |  | If jtag\_apb\_sel=1,then apb be active |
| ovrd\_clk\_sel | output | |  | aon |  |
| ovrd\_rst\_sel | output | |  |  |
| phy\_reset\_ov | output | |  |  |
| txrx0\_reset\_ov | output | |  |  |
| txrx1\_reset\_ov | output | |  |  |
| txrx2\_reset\_ov | output | |  |  |
| txrx3\_reset\_ov | output | |  |  |
| scan\_mode | input | | From pcs\_raw and input aon at the same time |  |  |
| scan\_shift | input | |  |  |
| scan shift\_cg | input | |  |  |
| scan\_set\_rst | input | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cr\_addr[15:0] | output |  | Raw PCS lane and PMA registers | CREG parallel interface (for Raw PCS lane and PMA registers)  (to Raw PCS lane and PMA) |
| cr\_wr\_en | output |  |
| cr\_wr\_data[15:0] | output |  |
| cr\_rd\_en | output |  |
|  |  |  |  |  |
| cr\_pma\_rd\_data[15:0] | input | PMA->pcs\_raw |  | CREG read data from PMA |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| cr0\_rd\_data[15:0] | input | Raw PCS lanes |  | CREG read data from Raw PCS lanes(lane0-3,lane4-7 no use) |
| cr1\_rd\_data[15:0] | input |  |
| cr2\_rd\_data[15:0] | input |  |
| cr3\_rd\_data[15:0] | input |  |
| cr4\_rd\_data[15:0] | input |  |
| cr5\_rd\_data[15:0] | input |  |
| cr6\_rd\_data[15:0] | input |  |
| cr7\_rd\_data[15:0] | input |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cr0\_mem\_ack | output |  | Raw PCS  lanes | CREG interface with Memory-Arbiter((memory bus be used by lane.fsm,  lane0-3,lane4-7 no use)) |
| cr1\_mem\_ack | output |  |
| cr2\_mem\_ack | output |  |
| cr3\_mem\_ack | output |  |
| cr4\_mem\_ack | output |  |
| cr5\_mem\_ack | output |  |
| cr6\_mem\_ack | output |  |
| cr7\_mem\_ack | output |  |
|  |  |  |  |
| cr0\_mem\_rd\_data  [15:0] | output |  | Raw PCS lanes |
| cr1\_mem\_rd\_data  [15:0] | output |  |
| cr2\_mem\_rd\_data  [15:0] | output |  |
| cr3\_mem\_rd\_data  [15:0] | output |  |
| Cr4\_mem\_rd\_data  [15:0] | output |  |
| cr5\_mem\_rd\_data  [15:0] | output |  |
| cr6\_mem\_rd\_data  [15:0] | output |  |
| cr7\_mem\_rd\_data  [15:0] | output |  |
|  |  |  |  |
| cr0\_mem\_req | input | Raw PCS lanes |  |
| cr1\_mem\_req | input |  |
| cr2\_mem\_req | input |  |
| cr3\_mem\_req | input |  |
| cr4\_mem\_req | input |  |
| cr5\_mem\_req | input |  |
| cr6\_mem\_req | input |  |
| cr7\_mem\_req | input |  |
|  |  |  |  |
| cr0\_mem\_addr[15:0] | input | Raw PCS lanes |  |
| cr1\_mem\_addr[15:0] | input |  |
| cr2\_mem\_addr[15:0] | input |  |
| cr3\_mem\_addr[15:0] | input |  |
| cr4\_mem\_addr[15:0] | input |  |
| cr5\_mem\_addr[15:0] | input |  |
| cr6\_mem\_addr[15:0] | input |  |
| cr7\_mem\_addr[15:0] | input |  |
|  |  |  |  |
| cr0\_mem\_wr\_en | input | Raw PCS lanes |  |
| cr1\_mem\_wr\_en | input |  |
| cr2\_mem\_wr\_en | input |  |
| cr3\_mem\_wr\_en | input |  |
| cr4\_mem\_wr\_en | input |  |
| cr5\_mem\_wr\_en | input |  |
| cr6\_mem\_wr\_en | input |  |
| cr7\_mem\_wr\_en | input |  |
|  |  |  |  |
| cr0\_mem\_wr\_data  [15:0] | input | Raw PCS lanes |  |
| cr1\_mem\_wr\_data  [15:0] | input |  |
| cr2\_mem\_wr\_data  [15:0] | input |  |
| cr3\_mem\_wr\_data  [15:0] | input |  |
| cr4\_mem\_wr\_data  [15:0] | input |  |
| cr5\_mem\_wr\_data  [15:0] | input |  |
| cr6\_mem\_wr\_data  [15:0] | input |  |
| cr7\_mem\_wr\_data  [15:0] | input |  |
|  |  |  |  |
| cr0\_mem\_rd\_en | input | Raw PCS lanes |  |
| cr1\_mem\_rd\_en | input |  |
| cr2\_mem\_rd\_en | input |  |
| cr3\_mem\_rd\_en | input |  |
| cr4\_mem\_rd\_en | input |  |
| cr5\_mem\_rd\_en | input |  |
| cr6\_mem\_rd\_en | input |  |
| cr7\_mem\_rd\_en | input |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cr0\_reg\_ack | output |  | Raw PCS lanes | CREG interface with Register-Arbiter  (The address generated by decoding is register map address exactly.) |
| cr1\_reg\_ack | output |  |
| cr2\_reg ack | output |  |
| cr3\_reg\_ack | output |  |
| cr4\_reg\_ack | output |  |
| cr5\_reg\_ack | output |  |
| cr6\_reg\_ack | output |  |
| cr7\_reg\_ack | output |  |
|  |  |  |  |
| cr0\_reg\_rd\_data[15:0] | output |  | Raw PCS lanes |
| cr1\_reg\_rd\_data[15:0] | output |  |
| cr2\_reg\_rd\_data[15:0] | output |  |
| cr3\_reg\_rd\_data[15:0] | output |  |
| cr4\_reg\_rd\_data[15:0] | output |  |
| cr5\_reg\_rd\_data[15:0] | output |  |
| cr6\_reg\_rd\_data[15:0] | output |  |
| cr7\_reg\_rd\_data[15:0] | output |  |
|  |  |  |  |
| cr0\_reg\_req | input | Raw PCS lanes |  |
| cr1\_reg\_req | input |  |
| cr2\_reg\_req | input |  |
| cr3\_reg\_req | input |  |
| cr4\_reg\_req | input |  |
| cr5\_reg\_req | input |  |
| cr6\_reg\_req | input |  |
| cr7\_reg\_req | input |  |
|  |  |  |  |
| cr0\_reg\_addr[15:0] | input | Raw PCS lanes |  |
| cr1\_reg\_addr[15:0] | input |  |
| cr2\_reg\_addr[15:0] | input |  |
| cr3\_reg addr[15:0] | input |  |
| cr4\_reg\_addr[15:0] | input |  |
| cr5\_reg\_addr[15:0] | input |  |
| cr6\_reg\_addr[15:0] | input |  |
| cr7\_reg\_addr[15:0] | input |  |
|  |  |  |  |
| cr0\_reg\_wr\_en | input | Raw PCS lanes |  |
| cr1\_reg\_wr\_en | input |  |
| cr2\_reg\_wr\_en | input |  |
| cr3\_reg\_wr\_en | input |  |
| cr4\_reg\_wr\_en | input |  |
| cr5\_reg\_wr\_en | input |  |
| cr6\_reg\_wr\_en | input |  |
| cr7\_reg\_wr\_en | input |  |
|  |  |  |  |
| cr0\_reg\_wr\_data[15:0] | input | Raw PCS lanes |  |
| cr1\_reg\_wr\_data[15:0] | input |  |
| cr2 reg\_wr\_data[15:0] | input |  |
| cr3\_reg\_wr\_data[15:0] | input |  |
| cr4\_reg\_wr\_data[15:0] | input |  |
| cr5\_reg\_wr\_data[15:0] | input |  |
| cr6\_reg\_wr\_data[15:0] | input |  |
| cr7\_reg\_wr\_data[15:0] | input |  |

# Implementation Details

## CMN block

### Clock & Reset

**cr\_ext\_clk** be generated from **aon\_cmn**,switch from **cr\_jtag\_clk** or **cr\_para\_clk** by **cr\_para\_sel\_i** selected.

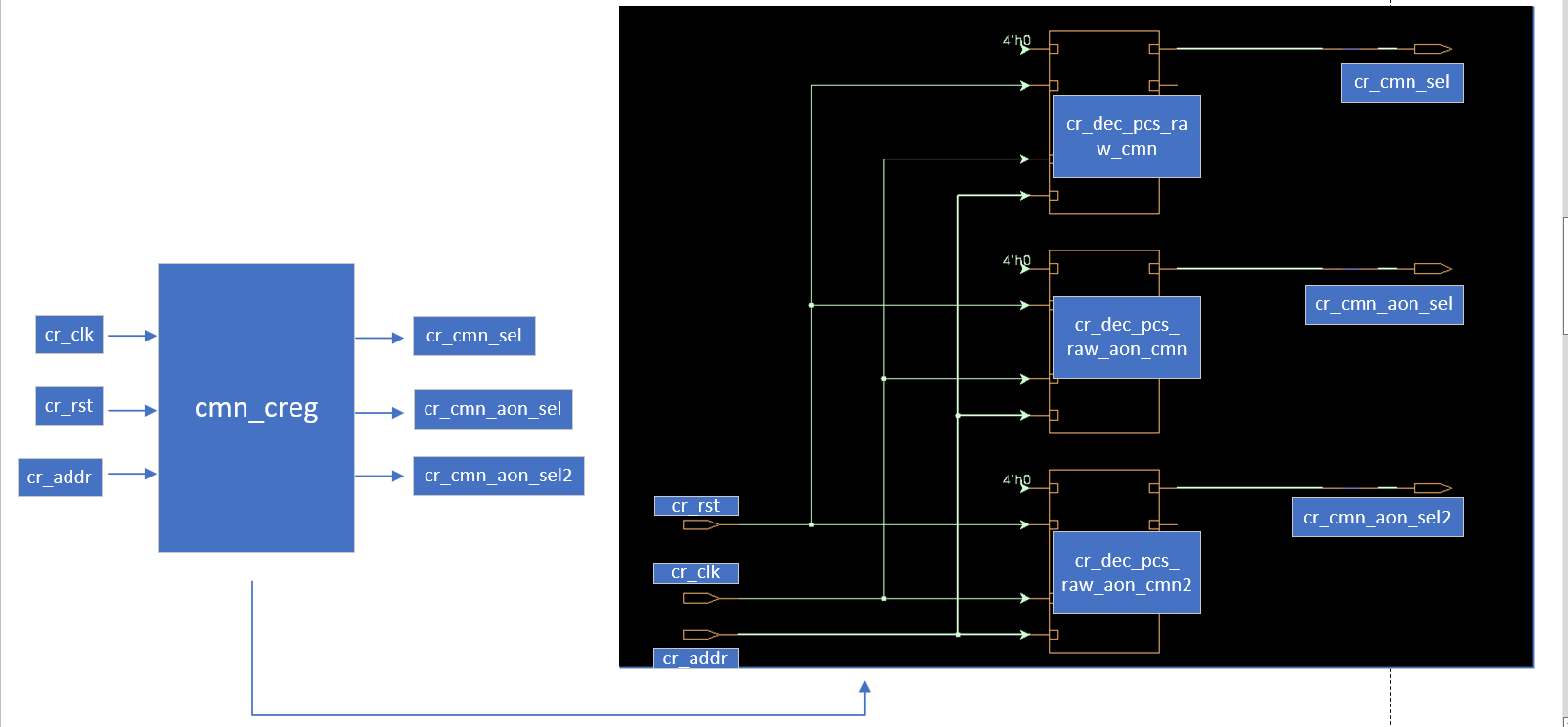
**cr\_ext\_clk\_rst** be come from **phy\_reset\_i** is synchronized by **cr\_ext\_clk**.

**cr\_rst** is synchronized by **cr\_clk**,come from **aon\_cmn**,be generated by **phy\_reset\_i** and **scan\_set\_rst**.

### Hardware Implementation

#### cmn\_creg

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| cr\_cmn\_sel[31:0] | output |  | pcs\_raw\_cmn | pcs\_raw\_cmn interal regsiter select |
| cr\_cmn\_aon\_sel[31:0] | output |  | pcs\_raw\_aon\_cmn | pcs\_raw\_aon\_cmn regsiter select |
| cr\_cmn\_aon\_sel2[31:0] | output |  | pcs\_raw\_aon\_cmn | pcs\_raw\_aon\_cmn regsiter select |
|  |  |  |  |  |
| cr\_clk | input | pcs\_raw\_cmn |  | cr clock |
| cr\_rst | input | pcs\_raw\_cmn |  | cr reset |
| cr\_addr[15:0] | input | pcs\_raw\_cmn |  | cr address bus(is register map address exactly) |



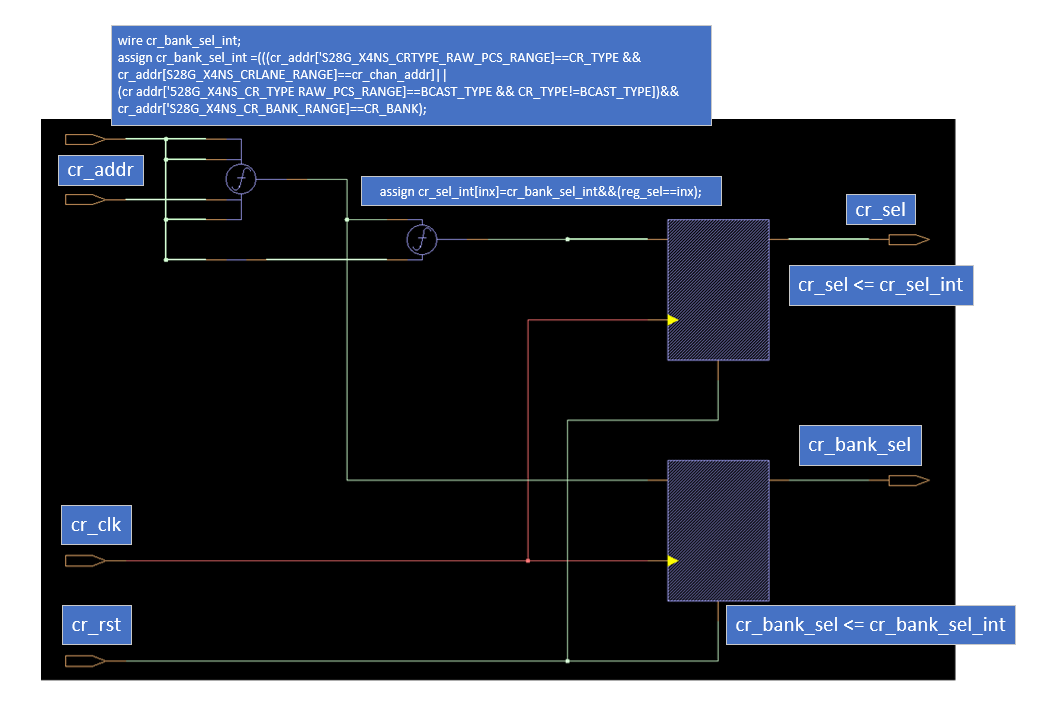
As shown in FIG, **cmn\_creg** module include **cr\_dec\_pcs\_raw\_cmn** module/**cr\_dec\_pcs\_raw\_aon\_cmn** module and **cr\_dec\_pcs\_raw\_aon\_cmn2** module.

The function of the **cmn\_creg** module is to use **cr\_addr** to generate the **pcs\_raw\_cmn** module’s internal registers chip select and the **pcs\_raw\_aon\_cmn** module’s registers chip select.

The function of the **cr\_dec\_pcs\_raw\_cmn** module is to use **cr\_addr** to decoding and generate the **cr\_cmn\_sel[31:0]** that is signal of chip select.

The function of the **cr\_dec\_pcs\_raw\_aon\_cmn** module is to use **cr\_addr** to decoding and generate the **cr\_cmn\_aon\_sel[31:0]** that is signal of chip select.

The function of the **cr\_dec\_pcs\_raw\_aon\_cmn2** module is to use **cr\_addr** to decoding and generate the **cr\_cmn\_aon\_sel2[31:0]** that is signal of chip select.

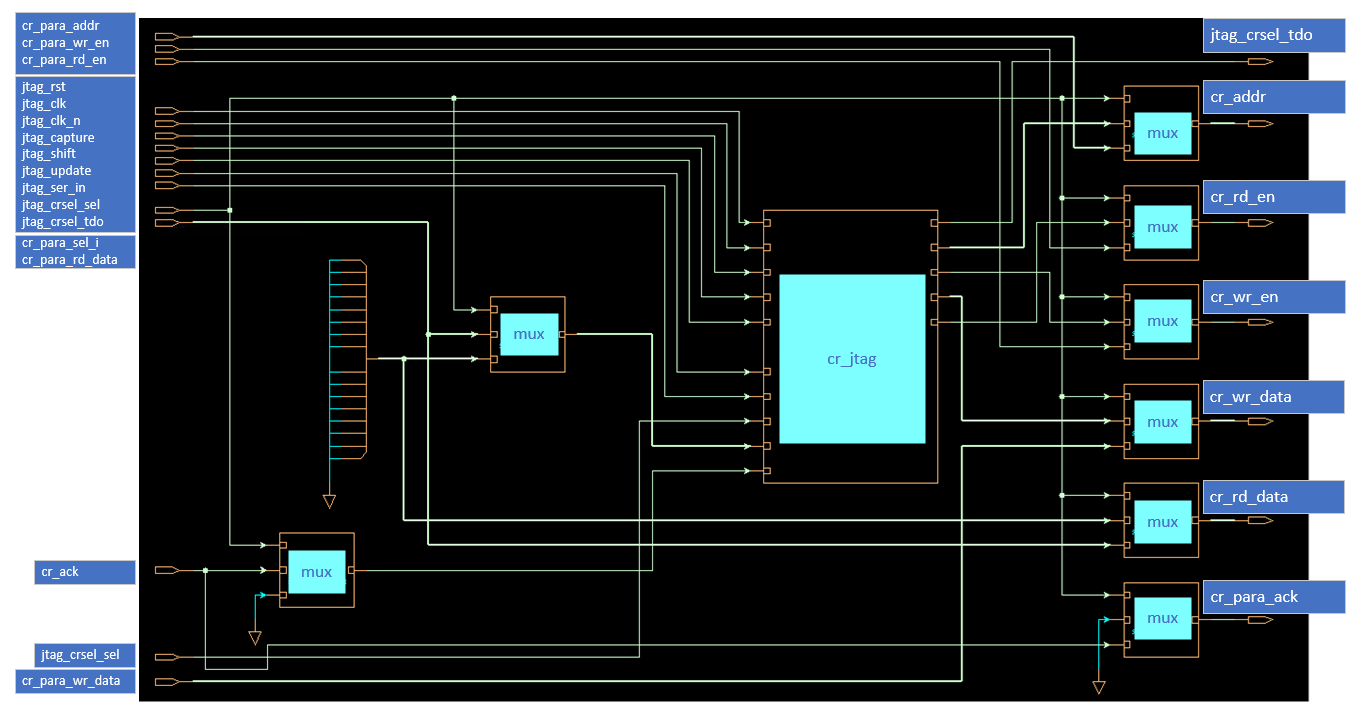


The interal structure of **cr\_dec\_pcs\_raw\_cmn/cr\_dec\_pcs\_raw\_aon\_cmn/cr\_dec\_pcs\_raw\_aon\_cmn2** modules are the same as shown in FIG,

After latching,the **cr\_addr** generated **cr\_sel** and **cr\_bank\_sel** by the decoding function **f1** and **f2**. In which **cr\_bank\_sel\_int** is the register bank chip select signal,**reg\_sel = cr\_addr[4:0]** is the number of registers in every modules. The **cr\_sel\_int** is the one-hot code represent of **reg\_sel**. The **cr\_bank\_sel** not be used, **cr\_sel** connected to the every correspond modules through three different buses,when **cmn\_creg** module instanted into **cr\_dec\_pcs\_raw\_cmn/cr\_dec\_pcs\_raw\_aon\_cmn/cr\_dec\_pcs\_raw\_aon\_cmn2** modules.

#### creg\_ctl

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| cr\_para\_sel\_i | input | cmn |  | APB OR JTAG SELECT |
|  |  |  |  |  |
| cr\_para\_addr[15:0] | input | cmn |  | CR Parallel Interface |
| cr\_para\_wr\_en | input |  |
| cr\_para\_wr\_data[15:0] | input |  |
| cr\_para\_rd\_en | input |  |
| cr\_para\_rd\_data[15:0] | output |  | cmn |
| cr\_para\_ac | output |  |
|  |  |  |  |  |
| jtag\_rst | input | jtag\_ctl |  | JTAG Register interface |
| jtag\_clk | input |  |
| jtag\_clk\_n | input |  |
| jtag\_capture | input |  |
| jtag\_shift | input |  |
| jtag\_update | input |  |
| jtag\_ser\_in | input |  |
| jtag\_crsel\_sel | input |  |
| jtag\_crsel\_tdo | output |  | jtag\_ctl |
|  |  |  |  |  |
| cr\_addr[15:0] | output |  |  | Internal CREG interface |
| cr\_wr\_en | output |  |
| cr\_wr\_data[15:0] | output |  |
| cr\_rd\_en | output |  |
| cr\_rd\_data[15:0] | input |  |  |
| cr\_ack | input |  |  |



The **creg\_ctl** module include **creg\_jtag** module,**cr\_addr\_mux** module,**cr\_wr\_data\_mux** module,**cr\_wr\_en\_mux** module,**cr\_rd\_en\_mux** module,**cr\_para\_rd\_data\_mux** module,**cr\_para\_ack\_mux** module,**cr\_jtag\_rd\_data\_mux** module and **cr\_jtag\_ack\_mux** module.

The function of the **cmn\_creg** module is switching **CR Parallel Interface**(default interface) and **JTAG Register interfac** to generate **Internal CREG interface** which process controled by **cr\_para\_sel\_i** signal.

The function of the **creg\_jtag** module is changing **JTAG Register interface** into **cr interfac**.

The function of the **cr\_addr\_mu** module is switching **cr\_jtag\_addr** and **cr\_para\_addr** to generate **cr\_addr** which process controled by **cr\_para\_sel\_i** signal.

The function of the **cr\_wr\_data\_mux** module is switching **cr\_jtag\_wr\_data** and **cr\_para\_wr\_data** to generate **cr\_wr\_data** which process controled by **cr\_para\_sel\_i** signal.

The function of the **cr\_wr\_en\_mux** module is switching **cr\_jtag\_wr\_en** and **cr\_para\_wr\_en** to generate **cr\_wr\_en** which process controled by **cr\_para\_sel\_i** signal.

The function of the **cr\_rd\_en\_mux** module is switching **cr\_jtag\_rd\_en** and **cr\_para\_rd\_en** to generate **cr\_rd\_en** which process controled by **cr\_para\_sel\_i** signal.

The function of the **cr\_para\_rd\_data\_mux** module is switching **0** and **cr\_rd\_data** to generate **cr\_para\_rd\_data** which process controled by **cr\_para\_sel\_i** signal.

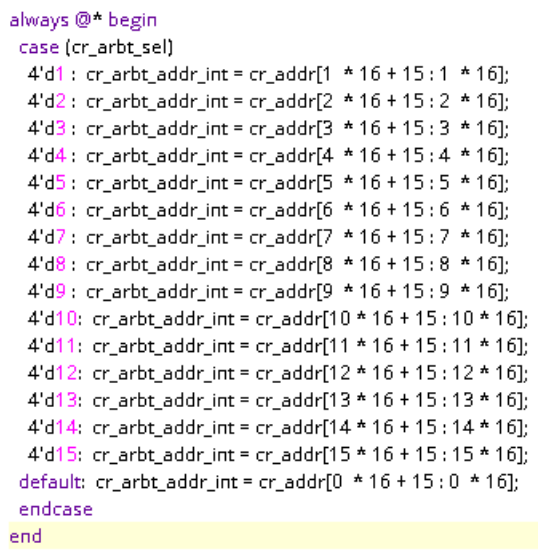
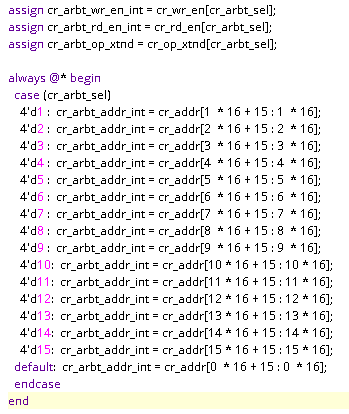
The function of the **cr\_para\_ack\_mux** module is switching **0** and **cr\_ack** to generate **cr\_para\_ack** which process controled by **cr\_para\_sel\_i** signal.

The function of the **cr\_jtag\_rd\_data\_mux** module is switching **0** and **cr\_rd\_data** to generate **cr\_jtag\_rd\_data** which process controled by **cr\_para\_sel\_i** signal.

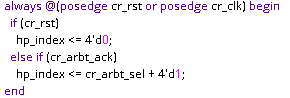
The function of the **cr\_jtag\_ack\_mux** module is switching **0** and **cr\_ack** to generate **cr\_jtag\_ack** which process controled by **cr\_para\_sel\_i** signal.

#### reg\_arbt

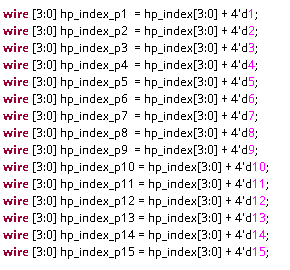
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| cr\_clk | input |  |  | Clock and Reset |
| cr\_rst | input |  |  |
|  |  |  |  |  |
| cr\_req[15:0] | input |  |  | CR input interfaces  (Combine the 16 cr interfaces into a 256-bit bus input, the address is the reg map address) |
| cr\_addr[255:0] | input |  |  |
| cr\_wr\_data[255:0] | input |  |  |
| cr\_wr\_en[15:0] | input |  |  |
| cr\_rd\_en[15:0] | input |  |  |
| cr\_op\_xtnd[15:0] | input |  |  |
| cr\_ack[15:0] | output |  |  |
| cr\_rd data[15:0] | output |  |  |
|  |  |  |  |  |
| cr\_arbt\_addr[15:0] | output |  |  | CR output interface(cr\_interface output after switch,the address is the reg map address） |
| cr\_arbt\_wr\_data[15:0] | output |  |  |
| cr\_arbt\_wr\_en | output |  |  |
| cr\_arbtrd\_en | output |  |  |
| cr\_arbt rd\_data[15:0] | input |  |  |



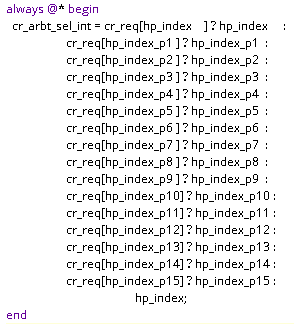
As shown above, **cr\_arbt\_sel (cr\_arbt\_sel\_int)** determines which **cr interface** is selected.



As shown above,if the current interface is selected,the value of **hp\_index** is the current selection value plus 1(**cr\_arbt\_sel**+1).



The above code indicates that **hp\_index\_p1...hp\_index\_p15** is the value of **hp\_index** plus 1..15, that is, the 16 interfaces is cyclically shifted left arrange from low to high based on the current be selected interface value.



Above code is the **cr\_arbt\_sel\_int** value generated process. Based on **hp\_index**, search for the position of **cr\_req[x]** where 1 first appears from low to high in the 16 interfaces(hp\_index->hp\_index\_p1->hp\_index\_p2...->hp\_index\_p14->hp\_index\_p15) ,the search process follows a round-robin selection pattern. If 1 is found, that interface is selected(**cr\_arbt\_sel\_int=hp\_index\_px**), if **cr\_req=0** (no request occurs) , **cr\_arbt\_sel\_int=hp\_index** (the current selected interface number value plus 1).

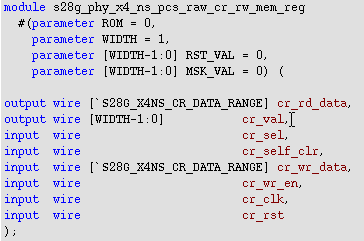
In summary, the function of the **reg\_arbt** module is to determine which of the 16 **CR input interfaces** is used as the **CR output interface** according to the value of **cr\_arbt\_sel**, and **cr\_arbt\_sel** is generated by **cr\_req** decoding. Specific process is the process of looping to the left to search the first be requested interface based on the currently selected interface.If cr\_req=0 (no request occurs), then cr\_arbt\_sel\_int=hp\_index (that is, the current selection path value plus 1).

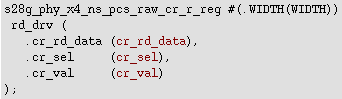
In this design, 4 interfaces are used, so the **reg\_arbt** module realizes the process of converting 4 **cr input interface** parallel input to 1 **CR output interface** serial output.

#### mem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| cr\_c1k | input |  |  | Mem clock |
| cr\_rst | input |  |  | Mem reset |
|  |  |  |  |  |
| cr\_cmn[63:0]\_b[7:0]\_sel[31:0] | input | mem\_creg[3:0] |  | register select in mem |
|  |  |  |  |  |
| cr\_wr\_en | input |  |  | register write enable in mem |
| cr\_wr\_data[15:0] | input |  |  | register write data in mem |
| cr\_rd\_en | input |  |  | register read enable in mem |
| cr\_rd data[15:0] | output |  | Raw PCS lanes | register read data in mem |

The **mem** make up **cmn[63:0]\_b[7:0], b[7:0]** represent 8 mem banks, and **cmn[63:0]** represent 64 mem blocks, each mem block include 32 16bit registers, So if use 4 mem blocks, the memory size is 64（block）\*32（register）\*4（bank）=8192 registers (16bit). The memory unit is the register and the specific operation is as follows.





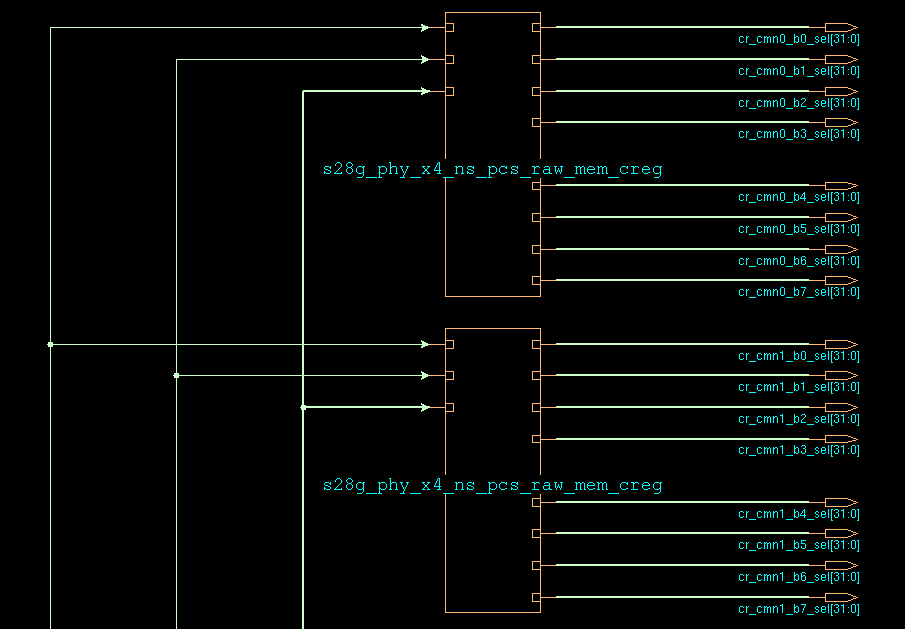
#### mem\_arbt

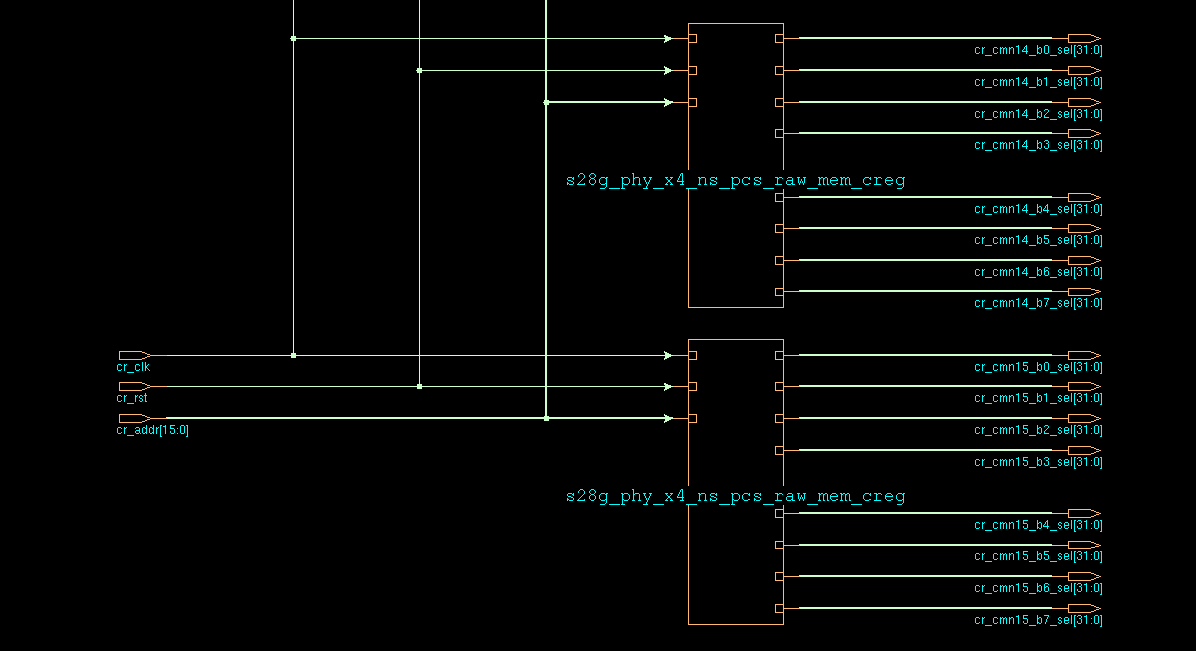
The interface and function of the **mem\_arb** module is the same as **reg\_arbt** module.The differnent is the input and output interface of **mem\_arb** module is data and address of **mem**.

#### mem\_reg0

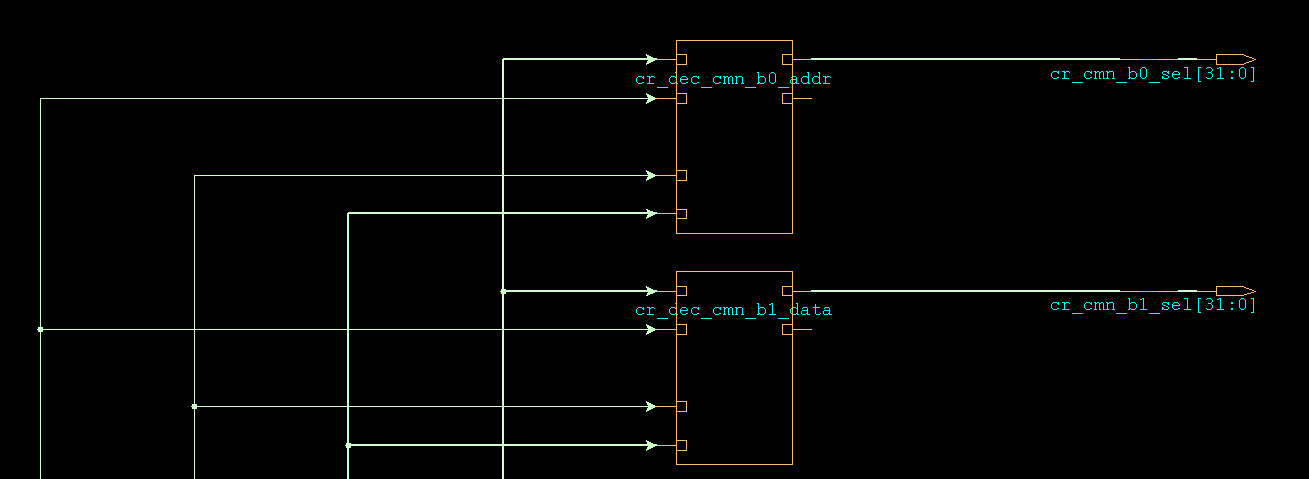
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| cr\_ck | input |  |  | mem clock |
| cr\_rst | input |  |  | mem reset |
| cr\_addr | input | Raw PCS lanes |  | mem address |
|  |  |  |  |  |
| cr\_cmn[15:0]\_b[7:0]\_sel | output |  | mem | register select in mem |

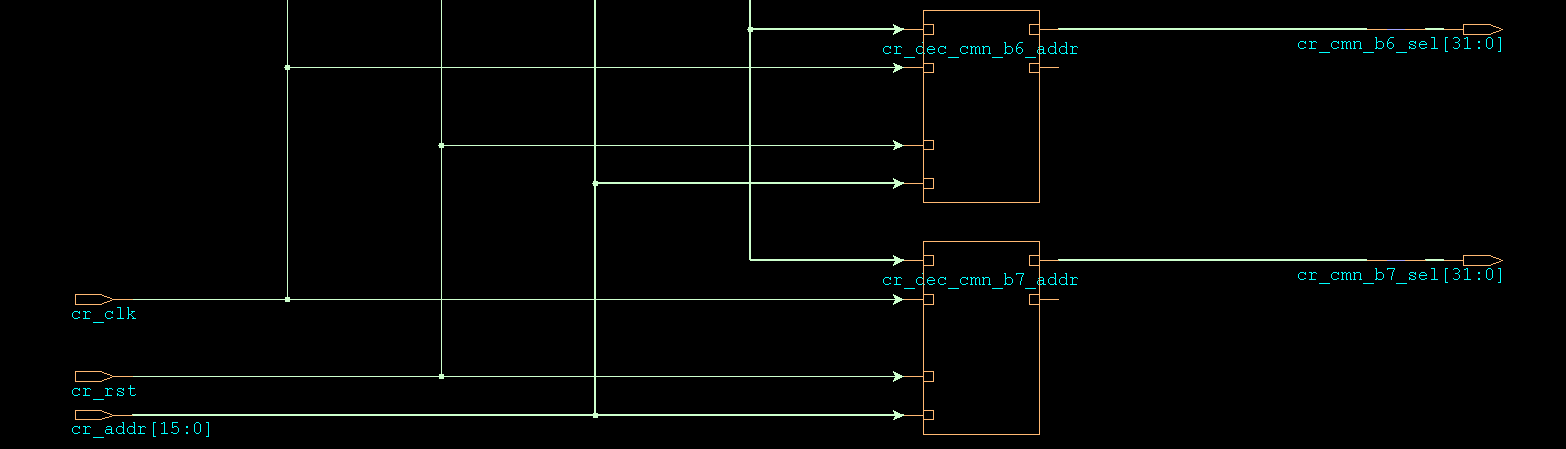
The function of the **mem\_reg** module is to generate the chip select signal of **cmn[15:0]\_b[7:0][31:0]** by decoding **cr\_addr[15:0]** (**cr\_addr** external connection **cr\_mem\_addr**).





The picture above is cmn0\_creg~cmn15\_creg (block select signal generate), and the lower picture is the internal structure of cmnX\_creg (bank select signal generate).





The internal structure of c**r\_dec\_cmn\_bx\_addr** module is the same as **cr\_dec\_pcs\_raw\_cmn/cr\_dec\_pcs\_raw\_aon\_cmn/cr\_dec\_pcs\_raw\_aon\_cmn2** modules.

#### mem\_reg1

The interface and internal structure are the same as above.

Generate **cmn[31:16]\_b[7:0]** chip select.

#### mem\_reg2

The interface and internal structure are the same as above.

Generate **cmn[47:32]\_b[7:0]** chip select.

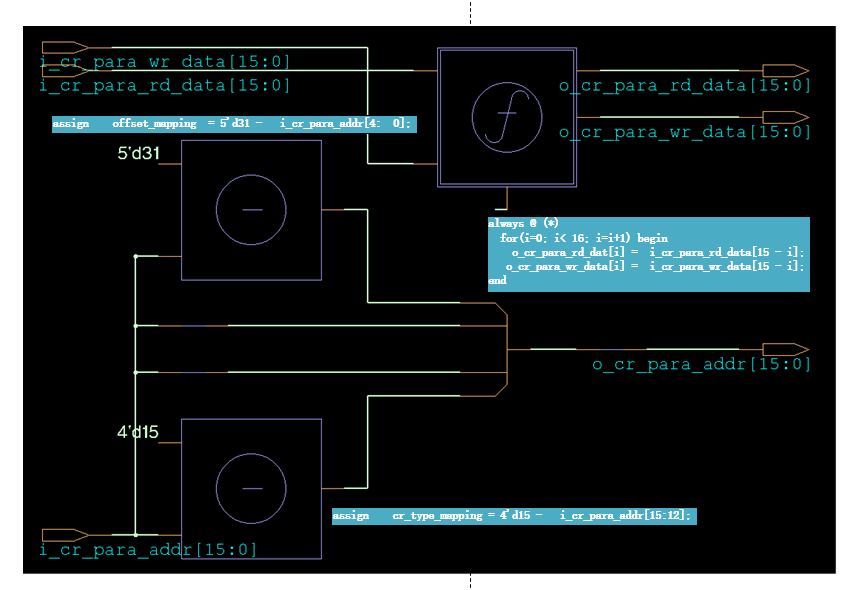
#### mem\_reg3

The interface and internal structure are the same as above.

Generate **cmn[63:48]\_b[7:0]** chip select.

#### reg\_mapping

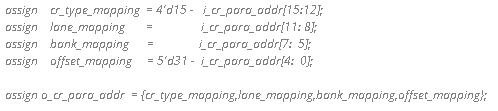
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| i\_cr\_para\_addr[15:0] | input |  |  | cr\_para\_add before mapping |
| i\_cr para\_rd data[15:0] | input |  |  | cr para\_rd data before mapping |
| i\_cr para wr\_data[15:0] | input |  |  | cr para wr\_data before mapping |
|  |  |  |  |  |
| o\_cr para addr[15:0] | output |  |  | cr\_para\_add after mapping |
| o\_cr para rd data[15:0] | output |  |  | cr para\_rd data after mapping |
| o\_cr para\_wr\_data[15:0] | output |  |  | cr para wr\_data after mapping |



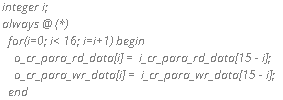
As shown in the figure, the main function of **reg\_mapping** is completed the mapping operation of **cr\_para\_add/cr\_para\_rd\_data/cr\_para\_wr\_data**.

The specific function as below,

Address mapping function:



Data mapping function:



#### Bootloader

##### Interface description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| cr\_clk | input | cmn | bootloader | control register bus clock |
| cr\_rst | input | cmn | bootloader | control register bus reset |
|  |  |  |  |  |
| sram\_bypass\_i | input | cmn | bootloader | control signal when asserted,bypasses the sram interface |
|  |  |  |  |  |
| sram\_bootload\_bypass\_i | input | cmn | bootloader | control signal when asserted,bypasses the sram bootloading function. |
| sram\_init\_done\_in | input | cmn | bootloader | sram\_init\_done input to bootloader signal indicating that the sram has been initialized by the boot loader in the raw\_pcs |
|  |  |  |  |  |
| sram\_init\_done\_out | output | bootloader | cmn | sram\_init\_done output from bootloader signal indicating that the sram has been initialized by the boot loader in the raw\_pcs |
| cr\_mem\_rd\_en | output | bootloader | mem | memory read enable read enable control to read data from memory |
| cr\_mem\_addr['S28G\_X4NS\_CR\_DATA\_RANGE] | output | bootloader | mem | memory address bus address port for the memory |
| cr\_merm\_rd\_data['S28G\_X4NS\_CR\_DATA\_RANGE] | input | mem | bootloader | memory read bus read data port fot the memory |
|  |  |  |  |  |
| sram\_wr\_en | output | bootloader | sram | sram write enable write enable control to write data to sram |
| sram\_addr['$28G\_X4NS\_SRAM\_ADDR\_RANGE] | output | bootloader | sram | sram address bus address port for the sram |
| sram\_wr\_data['S28G\_X4NS\_CR\_DATA\_RANGE] | output | bootloader | sram | sram write bus write data port fot the sram |

##### Interal structure



Bootload\_flow\_control:generate **cr\_mem\_ld\_done** and **sram\_bootload\_en** to control the bootload flow.

Mem\_control:generate related signals to read data from the mem.

Sram\_control:generate related signals to send data to the sram.

## RISCV controller

#### Interface description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| cr\_clk | input | raw\_lane | lane\_riscv | control register bus clock |
| cr\_rst | input | raw\_lane | lane\_riscv | control register bus reset |
| cr\_reg\_ack | input | raw\_lane | lane\_riscv | control register bus Read/Write acknowledgement Acknowledge signal indicating completion of the CR parallel read/write access. |
| cr\_reg\_rd\_data  [15:0] | input | raw\_lane | lane\_riscv | control register bus read data |
| cr\_reg\_addr  [15:0] | output | lane\_riscv | raw\_lane | control register bus address |
| cr\_reg\_wr\_en | output | lane\_riscv | raw\_lane | control register bus write enable |
| cr\_reg wr\_data[15:0] | output | lane\_riscv | raw\_lane | control register bus write data |
| cr\_reg\_rd\_en | output | lane\_riscv | raw\_lane | control register bus read enable |
| cr\_chan\_addr  [3:0] | input | raw\_lane | lane\_riscv | control register channel address. lane0:0;lane1:1;lane2:2;lane3:3; |

#### Register address

The channel field must be set to zero for common registers,so only use the"cr\_chan\_addr"input if addressing a lane register.

## Exteral RAM/ROM

### Exteral RAM

#### Interface description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| sram\_clk | input | cmn | external ram | external ram clock |
| sram\_rst | input | cmn | external ram | external ram reset |
| sram\_addr [`S28G\_X4NS\_EXT\_ROM\_ADDR\_RANGE] | input | cmn | external ram | external ram address bus |
| sram\_rd\_data [`S28G\_X4NS\_CR\_DATA\_RANGE] | output | external ram | cmn | external ram read data bus |
| sram\_wr\_en | input | cmn | external ram | external ram write enable |
| sram\_rd\_en | input | cmn | external ram | external ram read enable |
| sram\_wr\_data [`S28G\_X4NS\_CR\_DATA\_RANGE] | input | cmn | external ram | external ram write data bus |

#### Interal structure



### Exteral ROM

#### Interface description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal name** | **I/O** | **From** | **To** | **Description/Function** |
| rom\_clk | input | cmn | external rom | external rom clock |
| rom\_rst | input | cmn | external rom | external rom reset |
| rom\_addr[`S28G\_X4NS\_EXT\_ROM\_ADDR\_RANGE] | input | cmn | external rom | external rom address bus |
| rom\_rd\_data [`S28G\_X4NS\_CR\_DATA\_RANGE] | output | external rom | cmn | external rom read data bus |

#### Interal structure



##### mem\_reg0

The interface and internal structure are the same as 3.1.2.6.

##### mem\_reg1

The interface and internal structure are the same as 3.1.2.7.

##### mem\_reg2

The interface and internal structure are the same as 3.1.2.8.

##### mem\_reg3

The interface and internal structure are the same as 3.1.2.9.

##### Mem

The interface and internal structure are the same as 3.1.2.4.

## Hardware summarize

cr\_ para\_ addr[15:12]=4'b0101,PHY broadcasts to all PMA lanes.

cr\_para\_ addr[15:12]=4b0110, PHY broadcasts to all RAW PCS lanes.

cr\_para\_ addr[15:12]=4b0111, PHY broadcasts to all RAW PCS AON lanes.

The registers can be operated by firmware/jtag/apb.

Controller operate register flow:

1. Xlane.fsm can get order and operate data through cr[7:0]\_mem\_bus❶ in Interal ROM/Exteral ROM/Exteral RAM;

2. Xlane.fsm/Xlane.riscv complete register operate with reg\_arbt through cr[7:0]\_reg\_bus❷;

3. Reg\_arbt complete operate with registers in every module through cr\_bus❸ ;

4.Xlane.fsm/Xlane.riscv can run firmware from Interal ROM/Exteral ROM/Exteral RAM that be selected by macro define NO\_SRAM\_SUPPORT and NO\_EXT\_ROM\_SUPPORT➍.

5.When the macro NO\_EXT\_ROM\_SUPPORT be defined,can be selected run firmware from Interal ROM/Exteral ROM by setting **rom\_data\_control.➎**

6.Can select fms or riscv to become main controller use the signal of cpu\_mux\_control.➏



# User guide

## 4.1 Configuration Requirements

## 4.3 Registers

# Special Note

# FPGA Verification

# Test Mode Strategy

# Simulation List