Ring Controller MAS

Tzahi Peretz

Shimi Haleluya  
instructed by Amichai Ben-David

Revision 1.0

Table of Contents

[1. General description 2](#_Toc492142580)

[1.1 High level definition 2](#_Toc492142581)

[1.2 Top level interface 2](#_Toc492142582)

[1.3 Block diagram 2](#_Toc492142583)

[2. Block A 2](#_Toc492142584)

[2.1 Overview 2](#_Toc492142585)

[2.2 Interface 2](#_Toc492142586)

[2.3 Detailed description 2](#_Toc492142587)

[3. Block B 3](#_Toc492142588)

[3.1 Overview 3](#_Toc492142589)

[3.2 Interface 3](#_Toc492142590)

[3.3 Detailed description 3](#_Toc492142591)

[4. Pipeline 3](#_Toc492142592)

[5. Reset 3](#_Toc492142593)

[6. Bypasses 3](#_Toc492142594)

[7. Power saving 3](#_Toc492142595)

[8. Testing considerations 3](#_Toc492142596)

# General description

The Ring Controller (RC) processes and schedules read and write requests between its core and its fabric ring interface.

The role it plays is receiving requests from fabric and check if this request is relevant to its core by the address space.  
if so, decide if to accept the request if possible (by free buffer entry means) or send it to continue in the ring.

Moreover, send to the ring request from its core.

Taking all the above requests, the RC needs to decide which request to send next to the ring.

The main blocks compounding this unit are:

1. the core to fabric buffer.
2. The fabric to core buffer.
3. The main controller.

our Ring controller needs to be integrated with GPC\_4t core and with LOTR fabric.

## High level definition

1. What are the supported flows?

The supported flows are:

1. Arriving read request relevant to the core address from the Ring.
2. in case of free entry in F2C – insert it to the next free entry in F2C buffer, pending to be sent to core. ( select the next free slot by F2C\_SelWrF2C , and F2C\_EnWrF2C control signal )
3. in case of full F2C buffer – forward the request to the ring to another ring-cycle (cannot stall the ring).(by SelRingOut control signal )
4. Arriving write request relevant to the core address from the Ring.  
   One of F2C buffer entries, is reserved for write requests . hence it is guaranteed that write request can always enter F2C buffer.   
   so ,insert it to the reserved entry in F2C buffer, pending to be sent to core in the next cycle. ( select the next free slot by F2C\_SelWrF2C , and F2C\_EnWrF2C control signal )
5. Arriving broadcast write request from the Ring – check If this broad cast request isn't in C2F.
6. if so - once broadcast request arrives ,insert this request to the write slot in F2C buffer ( as in basic write request) , and also forward this request to ring output .
7. Else, kill this request , and change the state of the buffer entry to FREE .
8. Arriving read response from the Ring – check if the given read request address matches one from the holding read request address's in C2F buffer entries.
9. if so , write the data to the right read request in C2Fbuffer , and change the entry state from READ\_PRGRS to READ\_RDY , pending to be sent to the core .( by C2F\_SelWr, and C2F\_EnWr , and be ready to be sent to the core by C2F\_SelRdCore control signal).
10. Else, forward that read response request next to the ring. (by SelRingOut control signal )
11. Arriving read request from the core.
12. in case of free entry in C2F – insert & save the request in the C2F (by C2F\_SelWr , C2F\_EnWr ) waiting for its read response (the request either will be candidate for sending to the ring ( by C2F\_SelRDRing ) and be held in the buffer for the response).
13. in case of full C2F buffer – send stall signal to the core.
14. Arriving write request from the core.
15. in case of free entry in C2F – insert the request in the C2F (by C2F\_SelWr , C2F\_EnWr ) pending to be sent to the ring ( by C2F\_SelRDRing ) (when it sent to the ring and invalidate the entry).  
    when the write request chosen to be sent to ring output , change the entry state to FREE.
16. in case of full C2F buffer – send stall signal to the core.
17. Arriving broadcast write request from the core - insert the request in the C2F (by C2F\_SelWr , C2F\_EnWr ) pending to be sent to the ring ( by C2F\_SelRDRing ) .   
    the broadcast write request would be saved in the buffer until this specific write request return to this RC again . ( finishes ring loop)

(the request either will be send to the ring and be held in the buffer for the next time it returned).

1. Arriving read response from the core.  
   select the appropriate F2C buffer slot (by F2C\_SelWr ,and F2C\_EnWr ) , insert the data , and change the entry state to RD\_RDY . ready to be selected to be sent to the ring( by F2C\_SelRdRing)
2. How various unit’s blocks interact with each other?

The main controller Is the only block who Contacts the ring input and output.

The main controller also interacts with the other two buffers by listening and writing according to the requests they are holding inside.

The C2F buffer contacts the main controller for sending request it holds to the ring. it also Listening for a receiving read response or broadcast write request from the ring.

Moreover ,the buffer contacts the core, listening for new request from it and if needed send it a stall signal.

The F2C buffer contacts the main controller for sending requests it holds to the ring and also contacts it for receiving a new requests getting from the ring input.

This buffer also Contacts the core for sending the request its holds to execute .

* 1. What are the interfaces between the internal blocks?

The main controller have an input and output interface with each of the buffers (including data and control signals) . so that the main controller can send a request for the relevant buffer, and also gets requests from both of the buffers , and decide which one to deliver to the ring output.

The size of each one from the ports mentioned above is size of request.

## Top level interface

Core Interface

Table

Description automatically generated

Ring Interface

Table

Description automatically generated

General interfaces



## Block diagram

# C2F Buffer

## Overview

Address the following questions:

1. What’s the purpose of this block?

Contain the write and read request generated from the core .

Also ,this block save a record for the read requests in purpose to get the appropriate read response .  
Also , saves a record for write broadcast request , in order to be able to terminate the broadcast transaction.

1. How it interacts with neighboring blocks?

Main controller :

The buffer exposing all of his entries addresses&opcodes to the main controller , in order to allow the main controller to find if there is a matching read response/write broadcasts .

C2F buffer gets read responses from the ring , and attach the data to relevant entry in the buffer , by C2F\_SelWr ,and C2f\_EnWr control signals .

C2F buffer choose the next in-order request candidate to be sent on the ring , by C2F\_SelRdRing control signal .

Core :

C2F buffer get from the core read and write requests , by C2F\_SelWr ,and C2f\_EnWr control signals .

C2F buffer send to the core read response , that he receive from the main controller , by C2F\_SelRdCore control signal.

C2F buffer can inform the core that it cannot receive more requests , by raising the stall bit .

1. What are the main structures in this block? What size these structure has

the buffer contains THREADS\_NUM number of lines .(default value 4)

Each entry in the buffer is in the form :

{32'b address ,32'b data, 2'b opcode, 2'b ThreadID, 3'b state, 1'b valued bit}

* 1. How these structures are being utilized? De/Allocation

Once a request arrives from the core ,it would be saved in one of the empty entries , by C2F\_SelWr ,and C2f\_EnWr control signals .

If the buffer is full , the buffer will raise the stall signal .

When a request sent to the ring output (selected by C2F\_SelRdCore and by SelRingOut) , its buffer entry will change its status –

In case write request, change the state to FREE .

In case read request, change the state to READ\_PRGRS .

In case write broadcast, change the state to WRITE\_BCAST\_PRGRS

When a read request that already sent ,receiving its appropriate response , change the state to READ\_RDY .

In every cycle , the buffer sends to the core the next in-order read response it holds , by C2F\_SelRdCore and then change the state to FREE .



## Interface

C2F input  
C2F buffer hold a data mux for every entry in the buffer . each mux is feeded by the Ring input(RingInputQ501H) and by the core(C2F\_Req500H) requests .  
mux selector is C2F\_SelWr control signal , which determine its value ( 0 for core request, 1 for ring request) according to the following principle :  
if the slot is FREE , set C2F\_SelWr to 0 , else if the slot state is READ\_PRGRS , set C2F\_SelWr to 1 , otherwise the C2F\_SelWr value is not usefull(DON’T CARE) .   
moreover , each entry has C2F\_EnWr enable signal . which determines , if indeed to write the data from the mux output , according to following principle :   
for read responses from the ring , if the entry state is READ\_PRGRS , compare the addresses . if True , set C2F\_EnWr to 1 ( allow to write the data in this slot ,from the read response) .   
for requests from the core ,if this entry is the next free entry (that going to write into it) set its C2F\_EnWr enable to 1 .

* Next Free entry is the index of the entry which is candidate to be written .   
  TODO -> ENAHNCE ABOUT HOW TO DETERMINE THE NEXT FREE

C2F output  
C2F buffer hold 2 muxes in his output . one that directed toward the core , and one toward the ring output . each mux is feeded by all C2F buffer entries.  
the mux directed to the core controlled by C2F\_SelRdCore control signal , and its role is to choose one of the entries that has READ\_RDY state.   
the mux directed to the ring controlled by C2F\_SelRdRing control signal , and its role is to send the right candidate to the ring , considering ordering issues, and prioritized requests(broadcast) .

# F2C buffer

## Overview

What’s the purpose of this block?

Contain the write and read request generated from the fabric .

Also ,this block save a record for the read requests in purpose to get the appropriate read response .

How it interacts with neighboring blocks?

Main controller :

The buffer exposing all of his entries addresses to the main controller , in order to allow the main controller to find if there is a matching read response coming from the core .

F2C buffer get from the ring read and write requests , by F2C\_SelWr ,and F2C\_EnWr control signals .

F2C buffer choose the next read response candidate to be sent on the ring , by F2C\_SelRdRing control signal .

Core :

F2C buffer gets read responses from the core, and attach the data to relevant entry in the buffer , by F2C\_SelWr ,and F2C\_EnWr control signals .

F2C buffer send to the core read and write requests , that he receive from ring, by F2C\_SelRdCore control signal.

What are the main structures in this block? What size these structure has

the buffer contains ROUND\_TRIP\_CYCLES number of lines .

Each entry in the buffer is in the form :

{32'b address ,32'b data, 2'b opcode, 2'b ThreadID, 3'b state, 1'b valued bit}

* 1. How these structures are being utilized? De/Allocation

Once a request arrives from the ring ,it would be saved in one of the empty entries , by F2C\_SelWr ,and F2C\_EnWr control signals .

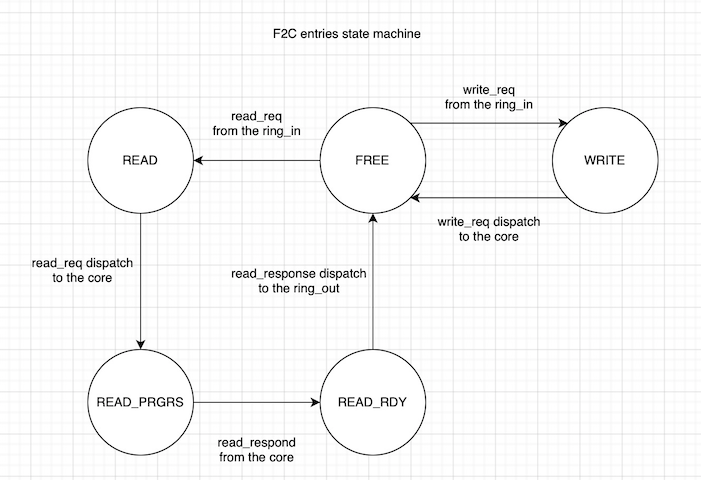
If the buffer is full , and a new read request from the ring arrives, it would be forwarded to the ring output .there is an empty slot in F2C buffer that reserved for write request , so it is guaranteed that write request always will be accepted to the buffer .

In case of read response: When a response sent to the ring output (selected by F2C\_SelRdCore and by SelRingOut) , its buffer entry will change its status to FREE .  
In case of read and write requests : When a request sent to the core (selected by F2C\_SelRdCore ) , its buffer entry will change its status accordingly .

In case write broadcast arrive : first we save it in F2C write slot entry . and also , we forward this request to the ring output .

When a read request that already sent ,receiving its appropriate response , change the state to READ\_RDY .

In every cycle , the buffer sends to the ring the next in-order read response candidate it holds , by F2C\_SelRdCore ,if SelRingOut indeed choose it change the state to FREE .



## Interface

F2C input  
F2C buffer hold a data mux for every entry in the buffer . each mux is feed by the ring input(RingInputQ501H) and by the core (F2C\_Rsp500H).  
mux selector is F2C\_SelWr control signal , which determine its value ( 0 for core request, 1 for ring request) according to the following principle :  
if the slot is FREE , set F2C\_SelWr to 1 , else if the slot state is READ\_PRGRS , set F2C\_SelWr to 0 , otherwise the F2C\_SelWr value is not usefull (DON’T CARE) .   
moreover, each entry has F2C\_EnWr enable signal . which determines , if indeed to write the data from the mux output , according to following principle :   
for read responses from the core , if the entry state is READ\_PRGRS , compare the addresses . if True , set F2C\_EnWr to 1 ( allow to write the data in this slot ,from the read response) .   
for requests from the ring ,if this entry is the next FREE entry (that going to write into it) set its F2C\_EnWr enable to 1 .

* Next Free entry is the index of the entry which is candidate to be written . (determined by ?)

F2C output  
F2C buffer hold 2 muxes in his output . one that directed toward the core , and one toward the ring output . each mux is feed by all F2C buffer entries.  
the mux directed to the core controlled by F2C\_SelRdCore control signal , and its role is to choose one request ,from the entries that has READ/WRITE state.   
the mux directed to the ring controlled by F2C\_SelRdRing control signal , and its role is to send one of the entries that has READ\_RDY state.

# Main Controller

## Overview

What’s the purpose of this block?  
The main controller purpose is to control and manage the transactions traffic moving from and to the RC .

Specified Main Controller roles :   
- Select the ring output by SelRingOut output mux . choose the output according :

|  |  |  |
| --- | --- | --- |
| Signal | Priority | Choose when : |
| RingInput | 1 | F2C is full , and new read request arrives to F2C or the transaction intended for other core .  Or write broadcast request . |
| 0 | 2 | Only if RC ventilation counter reaches 4 . |
| F2C\_Rsp | 3 | There is at least one entry in F2C with READ\_RDY state. |
| C2F\_Req | 4 | There is at least one entry in C2F with read or write state. |

- addressee detector – the main controller check if the request it received from the ring input is intended to its core addresses range .

- Addresses comparison , in order to detect the right buffer entry , which waits for the read response data :  
a. between read response from the ring , and C2F entries that holds READ\_PRGRS state.   
b. between read response from the core , and F2C entries that hold READ\_PRGRS state.

- Write broadcast control:  
once a write broadcast request arrives to the RC , the controller would detect whether this request created in his own core . by checkin the C2F buffer , if it already hold this write broadcast request .   
  
- maintain the RC ventilation counter :  
in order to certify progress, and to avoid deadlocks , it is necessary that at least one of every four RC generated requests , the RC would send NOP to the ring , and then reset the counter ( rather than sending a request candidate from his buffers) .   
Also ,this counter would be reset in case the RC send to the ring output invalid request .  
  
- manage the control signals over the F2C and C2F buffers .

How it interacts with neighboring blocks?  
C2F buffer  
- C2F\_SelWr: choose for entry , whether to receive input from the ring , or from the core  
- C2F\_EnWr: choose for entry , whether to write a new request from its input , or not .  
- C2F\_SelRdRing : select the entry that going to be the candidate for dispatch out to the ring. (from entries with READ/WRITE state)   
- C2F\_SelRdCore: select the entry that going to be the candidate for dispatch out to the core. . (from entries with READ\_RDY state)

F2C buffer  
- F2C\_SelWr : choose for entry , whether to receive input from the ring , or from the core   
- F2C\_EnWr : choose for entry , whether to write a new request from its input , or not .  
- F2C\_SelRdRing : select the entry that going to be the candidate for dispatch out to the ring. (from entries with READ\_RDY state)  
- F2C\_SelRdCore : select the entry that going to be the candidate for dispatch out to the core. (from entries with READ/WRITE state)

What are the main structures in this block? What size these structure has  
SelRingOut : 4 to 1 mux   
  
addressee detector   
keep the core’s memory blocks ids .  
get the core ID from the address of the given request.   
compare between both -> if True , perfrom this request . else , forward it to the ring output .   
  
Addresses comparison between read response from the ring , and C2F entries –   
include two sub-units -> first , compare opcodes to detect read response .  
and second , compare address with C2F entries .   
this comparison located between the ring input and C2F buffer .  
  
Addresses comparison between read response from the core , and F2C entries -> first , compare opcodes to detect read response . second , compare address with F2C entries .  
this comparison located between the core input and F2C buffer .  
  
RC ventilation counter  
2 bits counter .

# Pipeline + data path

Core Requests  
RD/WR Request from Core. A Load/Store from a Cores thread is sent to the Fabric .  
The Data Path of Request from Core to Fabric.   
Q500) C2F\_Req500H -> C2F\_NextBufferQnnnH   
Q501) C2F\_BufferQnnnH-> C2F\_ReqQ501H -> RingoutputQ501H   
Q502) RingoutputQ502H

Memory response   
RD\_RSP from Memory with Data.   
A Read request that originated from Ring input to the Local memory - Read Response back to Fabric the Data.  
The Data Path of a Read Response from Local Memory to Fabric.  
Q500) F2C\_Rsp500H -> F2C\_NextBufferQnnnH   
Q501) F2C\_BufferQnnnH-> F2C\_RspQ501H -> RingoutputQ501H   
Q502) RingoutputQ502H

Ring Input  
Ring Input has 3 Data Paths option:

1. Pass through (no Allocation in Buffers)   
Q500) RingInputQ500H   
Q501) RingInputQ501H -> RingoutputQ501H   
Q502) RingoutputQ502H

2. RD/WR Req to Local Memory (Allocating F2C Buffer entry)   
Q500) RingInputQ500H   
Q501) RingInputQ501H -> F2C\_NextBufferQnnnH   
Q502) F2C\_BufferQnnnH -> F2C\_ReqQ502H

3. RD\_RSP Response to a Core Request (Allocating C2F Buffer entry)   
Q500) RingInputQ500H   
Q501) RingInputQ501H -> C2F\_NextBufferQnnnH   
Q502) C2F\_BufferQnnnH -> C2F\_RspQ502H

# Reset

----------- Advanced topics ------------

Work on these items once you have finished the basic definitions above

# Bypasses

Describe unit’s processing pipeline bypasses:

1. What stages can be bypassed? Why?
2. What are the condition to bypass a stage?

# Power saving

1. Specify Power features (clock gating, avoiding excessive toggling, etc.).
2. Specify area trade-off for power, logic, and timing.

# Testing considerations

Describe the validation hooks in the design.

Noteworthy items for validation testplan (potential complicated/weak points in the design that validation should emphasize on)