Ring Controller

HAS: High-Level-Architecture-Specifications
MAS: Micro-Architecture-Specifications

Tzahi Peretz (tzahiperetz44@gmail.com)
Shimi Haleluya (shimi.haleluya@campus.technion.ac.il)

Advisor : Amichai Ben-David (amichaibd@gmail.com)

TABLE OF CONTENTS

Figures

Tables

Revision History

Rev. No.	Who	Description	Rev. Date
0.5	Tzahi , Shimi	Initial seed	9 October
1.0	Tzahi , Shimi	Inserted Block desciripitons + use cases	31 October

Table 1 - Revision History

Glossary

Term	Description
ISA	Instruction Set Architecture. (such as X86, ARM, RISC-V etc.)
IO	Input & output.
IP	Intellectual Property. In this case, RTL building block that can be consumed.
Pipeline	Common Way to parallel and utilize Hardware https://en.wikipedia.org/wiki/Instruction_pipelining
RISC	Reduce Instruction Set Computer. (Unlike CISC -Complex Instruction Set Computer) https://en.wikipedia.org/wiki/Reduced_instruction_set_computer
RISC-V	A relatively new open and free ISA. (comparable to intel X86, ARM) https://en.wikipedia.org/wiki/RISC-V
Standard interface	Functional characteristics to allow the exchange of information between two systems
Word	32-bits of data - 4 Bytes. The size of an integer in RV32I ISA.
RC	Ring Controller
F2C	Fabric to Core
C2F	Core to Fabric
LOTR	Lord of The Ring – name of the top project
Round trip	Number of cycles that takes to a request return to the core .
I/F	Interface

Table 2 – Glossary

1 General Description

The RC (Ring Controller) IP is one of the main building blocks in the LOTR project.
A ring network is a network topology in which each agent connects to exactly two other agents, forming a single continuous pathway for transactions through each agent – a ring. Transactions travels from agent to agent, with each agent along the way handling every transaction.
This architecture ensures memory coherency by dividing the shared memory address such that each agent manage a dedicated and exclusive memory range. In this way, coherency is guaranteed, because only the agent itself is accessing its own memory.

1.1 Project Background – LOTR

- **LOTR** – 'Lord Of The Ring'
  The top-level of the project - also known as the Fabric.
  The LOTR connects all the tiles to each other in a ring architecture.
- **LOTR_TILE**
  Connects and pairs each GPC_4T (core) to a RC.
- **RC** – 'Ring Controller'
  The arbitration logic for Core & Fabric transactions.
  The RC maintains the ordering coherency, starvation & Dead Lock prevention.
  Contains the C2F& F2C Buffers to arbitrate the current Fabric<->Core interface.
- **GPC_4T** – 'General Purpose Compute unit with 4 HW threads'
  A RV32I compatible RISCV Core. Capable of running 4 HW threads.
  Contains Data-Memory & Instruction-Memory (I_MEM & D_MEM)

LOTR Block Diagram

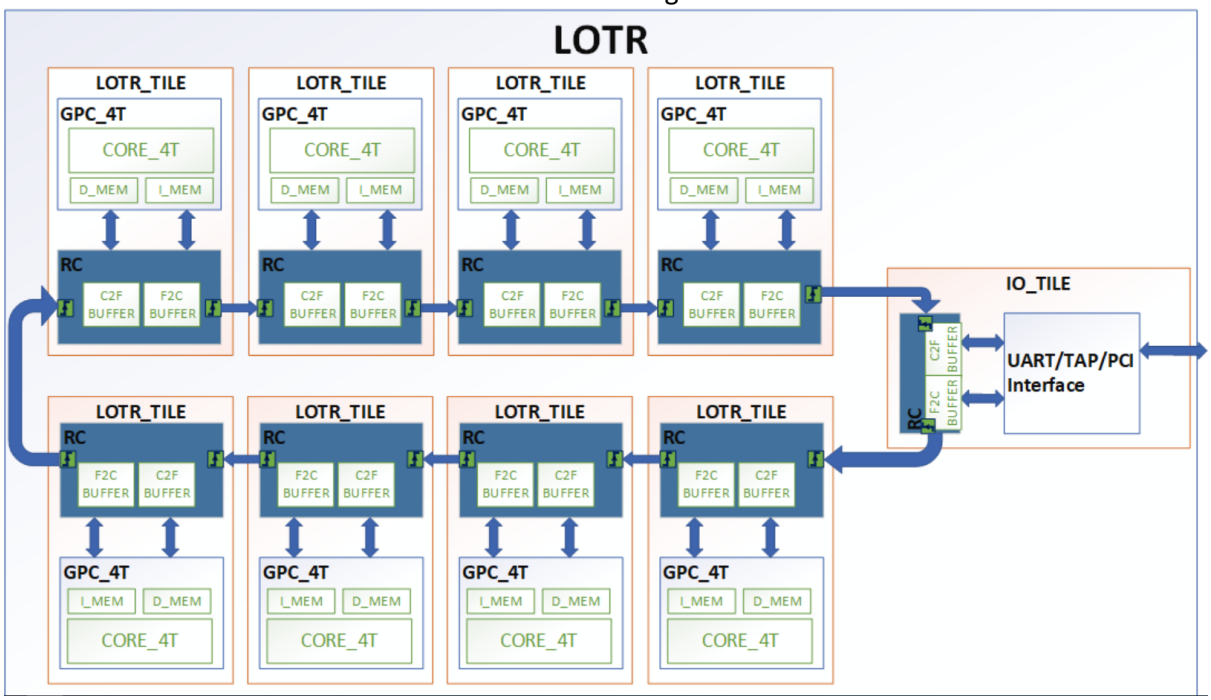

Figure 1 – LOTR Diagram

1.2 RING CONTROLLER BACKGROUND

The Ring Controller is the logic that connects the core to the fabric.
The RC processes and schedules read and write requests between its paired core and the fabric, by using two buffers.

1) F2C - Fabric 2 Core.
2) C2F - Core 2 Fabric.

Buffers & Transactions Types:

1. F2C Buffer: Each entry may hold different type of transaction:
   - RD/WR – request from fabric to core – The requests originated in a different core.
   - RD_RSP – returns the data from current core to the fabric.
2. C2F Buffer: Each entry may hold different type of transaction:
   - RD/WR – request from current core to fabric.
   - RD_RSP – return the data from the fabric to current Core.

RC has 2 interface Blocks:

1. **Fabric:** requests & responses from/to the Fabric to/from the relevant RC buffer.
   - RingReqIn : Input from Fabric that **may** RD/WR request this Tiles Memory.
   - RingRspIn : Input from Fabric that **may** RD_RSP to this Tiles original request.
   - RingReqOut : Output to Fabric that **might of** originated in a request from this Tile.
   - RingRspOut : Output to Fabric that **might of** originated in a response from this Tile.
2. **Core:** requests & responses can move from/to the RC buffer to/from the core.
   - C2F_Req : Input from Core to C2F Buffer – Targeting the RingReqOut interface.
   - F2C_Rsp : Input from Core to F2C Buffer – Targeting the RingRspOut interface.
   - C2F_Rsp : Output to Core from C2F Buffer – Originated in RingRspIn interface.
   - F2C_Req : Output to Core from F2C Buffer – Originated in RingReqIn interface.

Arbitrate requests & responses toward the Fabric:

- **RingReqOut: SelRingReqOutQ501H** - Arbitrate between the candidates:
  1. RingReqIn - Ring input request that are not targeting the current Tile.
  2. C2F Buffer - Core requests to Fabric.
  3. Bubble/NOP - In order to avoid overloading traffic onto the RingReq Channel.
- **RingRspOut: SelRingRspOutQ501H** - Arbitrate between the candidates:
  1. RingRspIn - Ring input Responses that are not targeting the current Tile.
  2. F2C Buffer - Core responses to Fabric. (wants to return to the origin core RD request)
  3. Bubble/NOP - In order to avoid overloading traffic onto the RingRsp Channel.

Arbitrate requests & responses toward the Core:

- **F2C_Req:** using the F2C_SelRdCoreQ502H control bit.
- **C2F_Rsp:** using the C2F_SelRdCoreQ502H control bit.

Towards the Core transaction may come only from the F2C/C2F buffers.

1.2.1 RC Block Diagram

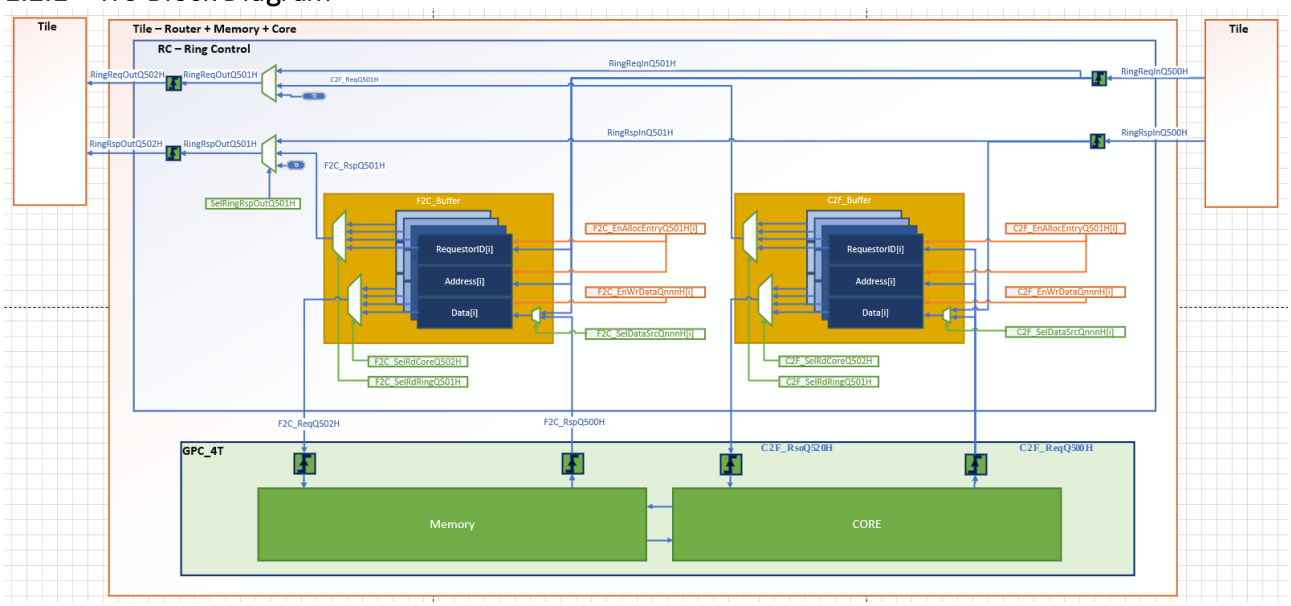

Figure 2 – RC Diagram

2 BLOCK INTERCONNECT STANDARD INTERFACES

In this LOTR Fabric all building blocks are connected using the fallowing Standard Interface:

Name	Size	Optional	Description
Valid	1	No	Valid transaction indication
Opcode	2	No	Request opcode - Enumerate value coding: RD = 2'b00 , RD_RSP = 2'b01 , WR = 2'b10 ,WR_BCAST = 2'b11
Data	32	No	Write request data
Address	32	No	Request address.
Requestor	10	Yes	{8'b coreID, 2'b Thread<#>} of the request initiator
Stall	1	Yes	When Stall signal is asserted, the agent won't accept new requests.

The transactions are single cycle, no credit loop or multi cycle "pumps" of data.
This will ease the integration process in the project, will allow new Ips to connect to the fabric.

Note:
If the 8 MSB bits of address equal to 0xFF, it means that the transaction is write broadcast.

3 TOP LEVEL INTERFACE

In the LOTR Fabric there are 2 Main channels:

- Request Channel: Allows RD/WD request to move from Tile to Tile until it finds its destination.
- Response Channel: Allows RD_RSP to move from Tile to Tile until it finds its destination.

This is done to benefit the bandwidth of the memory transactions.
Giving the RD_RSP its own Channel will have better Response time for Core read requests.
This will help with "Idle time" in the cores waiting for the responses.

3.1 FABRIC: REQUESTS & RESPONSES FROM/TO THE FABRIC TO/FROM THE RC BUFFER.

- RingReqIn: Input from Fabric that may RD/WR request this Tiles Memory.
- RingRspIn: Input from Fabric that may RD_RSP to this Tiles original request.
- RingReqOut: Output to Fabric that might of originated in a request from this Tile.
- RingRspOut: Output to Fabric that might of originated in a response from this Tile.

3.2 CORE: REQUESTS & RESPONSES CAN MOVE FROM/TO THE RC BUFFER TO/FROM THE CORE.

- F2C_Req: Output to Core from F2C Buffer – Originated in RingReqIn interface.
- F2C_Rsp: Input from Core to F2C Buffer – Targeting the RingRspOut interface.
- C2F_Req: Input from Core to C2F Buffer – Targeting the RingReqOut interface.
- C2F_Rsp: Output to Core from C2F Buffer – Originated in RingRspIn interface.

3.3 GENERAL INTERFACE SIGNALS:

Name	Size	Direction	Description
QClk	1	Input	Q Clock – a single clock domain.
RstQnnnH	1	Input	Active High Reset
CoreID	8	Input	8 bits unique core identifier.

Table 3 - General Interface

3.4 RC <--> FABRIC INTERFACE SIGNALS:

Name	Size	Direction	Description
Ring --> RC, RingReqIn - Requests from another cores .			
RingReqInValidQ500H	1	Input	Valid transaction indication
RingReqInRequestorQ500H	10	Input	{coreID ,thread number} of the request initiator
RingReqInOpcodeQ500H	2	Input	Request opcode
RingReqInAddressQ500H	32	Input	Request address
RingReqInDataQ500H	32	Input	Write request data
Ring --> RC, RingRspIn - Responses driven by another cores .			
RingRspInValidQ500	1	Input	Valid transaction indication
RingRspInRequestorQ500H	10	Input	{coreID ,thread number} of the request initiator
RingRspInOpcodeQ500H	2	Input	Request opcode
RingRspInAddressQ500H	32	Input	Request address
RingRspInDataQ500H	32	Input	Read response data
RC --> Ring , RingReqOut - Requests from the paired core			
RingReqOutValidQ502H	1	Output	Valid transaction indication
RingReqOutRequestorQ502H	10	Output	{coreID ,thread number} of the request initiator
RingReqOutOpcodeQ502H	2	Output	Request opcode
RingReqOutAddressQ502H	32	Output	Request address
RingReqOutDataQ502H	32	Output	Write request data
RC --> Ring , RingRspOut - Responses driven by the paired cores .			
RingRspOutValidQ502H	1	Output	Valid transaction indication
RingRspOutRequestorQ502H	10	Output	{coreID ,thread number} of the request initiator
RingRspOutOpcodeQ502H	2	Output	Request opcode
RingRspOutAddressQ502H	32	Output	Request address
RingRspOutDataQ502H	32	Output	Read response data

Table 4 - RC <--> Fabric interface

3.5 RC <--> CORE INTERFACE SIGNALS:

Name	Size	Direction	Description
RC_F2C <-- CORE			
F2C_RspValidQ500H	1	Input	Valid transaction indication
F2C_RspOpcodeQ500H	2	Input	Request opcode
F2C_RspAddressQ500H	32	Input	Request address
F2C_RspDataQ500H	32	Input	Read response data
RC _F2C --> CORE			
F2C_ReqValidQ502H	1	output	Valid transaction indication
F2C_ReqOpcodeQ502H	2	output	Request opcode
F2C_ReqAddressQ502H	32	output	Request address
F2C_ReqDataQ502H	32	output	Write request data
RC <-- CORE C2F			
C2F_ReqValidQ500H	1	Input	Valid transaction indication
C2F_ReqOpcodeQ500H	2	Input	Request opcode
C2F_ReqAddressQ500H	32	Input	Request address
C2F_ReqDataQ500H	32	Input	Write request data
C2F_ReqThreadIDQ500H	2	Input	Thread number of the initiator
RC --> CORE C2F			
C2F_RspValidQ502H	1	output	Valid transaction indication
C2F_RspDataQ502H	32	output	Read response data
C2F_RspStall	1	output	Stall signal
C2F_RspThreadIDQ502H	2	output	Thread number of the initiator

Table 5 - RC <--> Core interface

4 MAIN COMPONENTS

4.1 C2F BUFFER

4.1.1 Overview

The buffer depth of C2F is equals to the maximum threads in the GPC_4T unite.
In the LOTR a GPC_4T Agent can send a single read request from each thread – and may only send a new request after the RD Response.
This will allow each thread to allocate an entry in the buffer with a read requests without blocking and Creating backpressure to write requests from other threads.

C2F buffer entry is allocated with a write or read request generated from the core.
In case of read request, the buffer will save a record for the read requests after it was sent to Fabric in purpose to match with the appropriate read response.
This will change the state to READ_READY, and will candidate the entry for C2F_Rsp.

The entry will save a record for write broadcast request, in order to be able to terminate the broadcast transaction once it finishes a ring cycle.

4.1.2 Block diagram

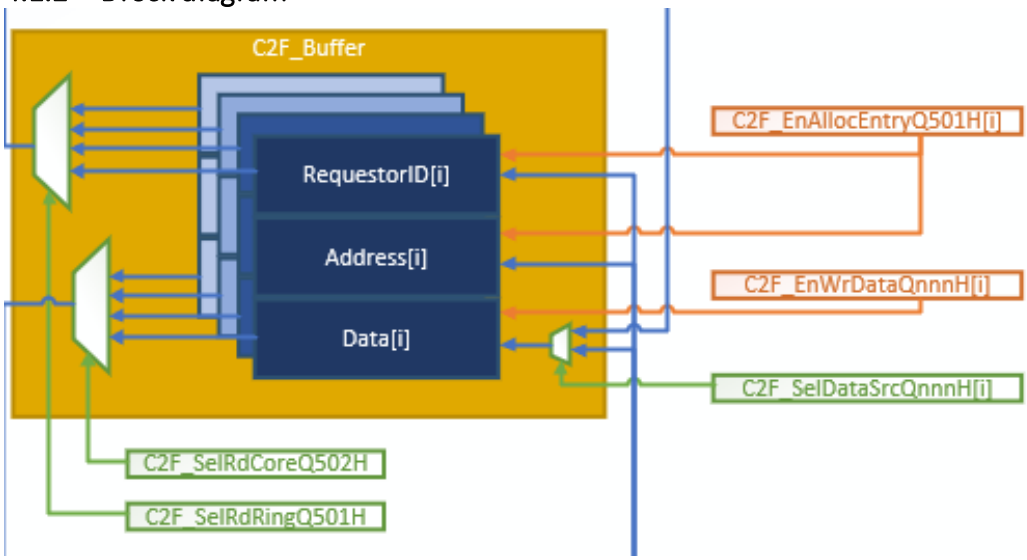

Figure 3 - C2F Buffer

4.1.3 C2F Interfaces

From Core **'C2F_ReqQ500H'**: RD, WR, WR_BCAST – Will allocate entry (C2F_EnAllocEntryQ500H).
From Fabric **'RingRspInQ501H'**: RD_RSP – Will Update the Data and change state to READ_RDY.
Buffer to Fabric **'C2F_ReqQ501':** RD, WR, WR_BCAST – Arbiter win (C2F_SelRdRingQ501H).
Buffer to Core **'C2F_RspQ502':** RD_RSP – Arbiter win (C2F_SelRdCoreQ502H).

4.1.4 C2F Control Signals

The buffer's behavior is defined by assertion/de-assertion of its control signals:

- **C2F_SelDataSrcQnnnH:**
  select the source of the input data to the buffer - between data from RingRspIn and C2F_Req.
- **C2F_EnWrDataQnnnH:**
  Enable for writing to the data entry in the buffer. (WR from Core or RD_RSP from Fabric)
- **C2F_EnAllocEntryQ501H:**
  Enable for allocating an entry in the buffer, for incoming transaction.

4.1.5 C2F Buffer Structure

The number of entries in the buffer is defined by the thread number a single core can run.
Entry structure:

Buffer field	Size	Description
State	3'b	FREE = '000' , READ_RDY = '100' WRITE = '001' , READ_PRGRS = '011' READ = '010' , WRITE_BCAST = '101' , WRITE_BCAST_PRGRS = '110'
Requestor	10'b	{8'b coreID, 2'b Thread<#>} of the request initiator
Address	32'b	Request address.
Data	32'b	WR Data / RD_RSP Data

Table 6 - C2F Entry Structure

4.1.6 C2F Behaviour

- Once a request arrives from the core(C2F_ReqQ500H),it would be saved in one of the empty entries, by EnAllocEntryQ501H control signal.
- If the buffer is full, the buffer will raise the stall signal toward the Core.
- When a request sent to the ring output (selected by C2F_SelRdCore and by SelRing ReqOut), 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

Read Responses:

- An entry that already sent a read (In READ_PRGRS State), receiving its appropriate response, change the state to READ_RDY.
- All the entries in READ_RDY state are candidates to be send to the core.
  The arbitrator (C2F_SelRdCore) Will arbitrate the **oldest in-order** read response it holds.
  Then the Entry will change the state back to FREE.

Read / Write requests:

- Entries in READ/WRITE/WRITE_BCAST are candidates for arbitration to Fabric.
  Winner will be **Oldest In-order.** The Next State of the buffer entry:
    - READ -> READ_PRGRS
    - WRITE -> FREE
    - WRITE_BCAST -> WRITE_BCAST_PRGRS

4.2 C2F CONTROL SIGNALS

Signal	Width[bits]	Description
C2F_EnAllocEntryQ501H	C2F_entries	Enables the allocation of C2F entries. The signal's assertion ensure that the first free entry will be allocated ,when a valid request arrives from the core .
C2F_EnWrDataQnnnH	C2F_entries	Enables data changing of C2F entries. The signal will be asserted when C2F_EnAllocEntryQ501 asserted, or when valid response arrives back to the buffer .
C2F_SelDatarcQnnnH	C2F_entries	Select the data source for each entry . 0: Core(C2F_Req) , 1: Ring(RingRspIn) .
C2F_SelRdCoreQ502H	log2(C2F_entries)	Selects an entry toward the core. Determined by MRO
C2F_SelRdRingQ501H	log2(C2F_entries)	Selects an entry toward the ring. Determined by MRO

Table 7 - C2F Control Signals

4.2.1 States Machine

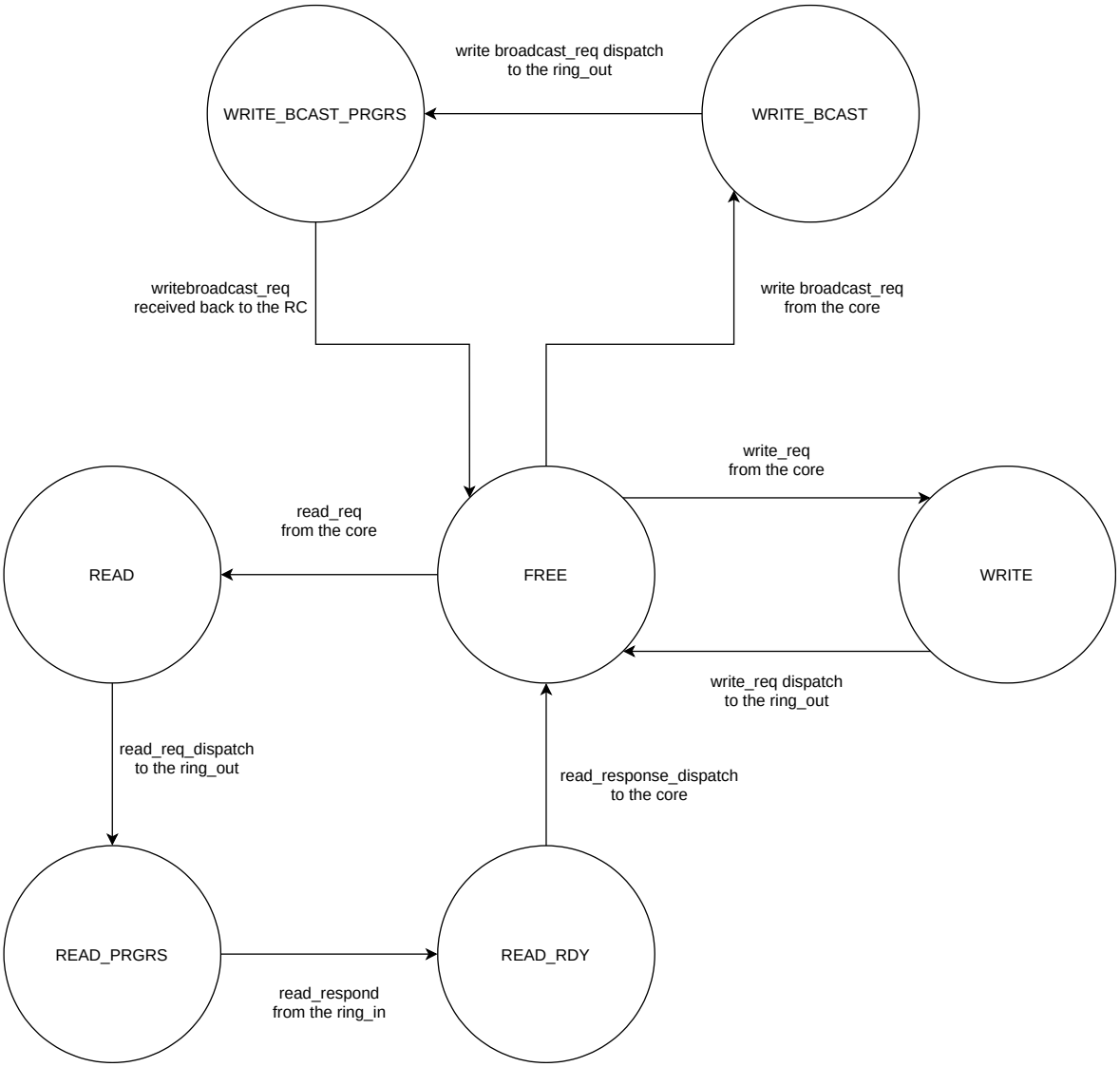

Table 8 - C2F States Machine

4.3 F2C BUFFER

4.3.1 Overview

The Buffer depth of F2C is the "round-trip" RC->Core>RC.
This is to allow multiple reads to access the Cores Memory without blocking all the entries.
Each RD will allocate the entry until the RD_RSP will return from the "round-trip" RC->Core>RC.

F2C buffer allocates entry with WR and RD request generated from the fabric.
In case of read request, the buffer will save a record for the read requests after it was sent to Core in purpose to match with the appropriate read response.
This will change the state to READ_READY, and will candidate the entry for F2C_Rsp.

4.3.2 Block diagram

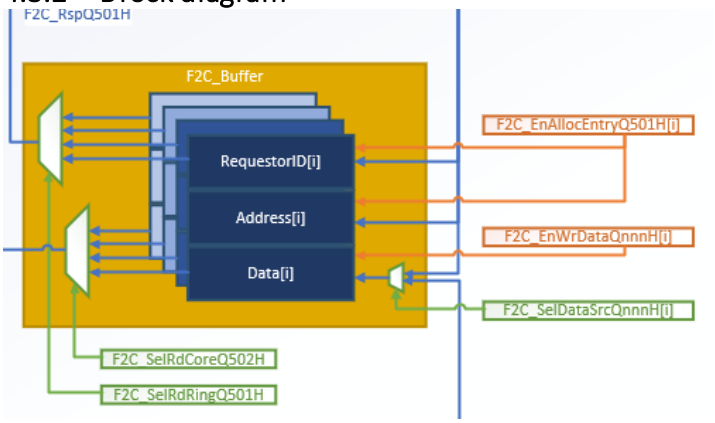

Table 9 - F2C Buffer

4.3.3 Interfaces

F2C buffer get from the RingReqIn interface read and write requests, as well as read responses from the F2C_Rsp(core) .
Transactions from the buffer are getting forwarded either toward the ring through RingRspOut interface , or toward the core through F2C_Req.

From Core **'F2C_RspQ500H'**: RD_RSP – Will Update the Data and change state to READ_RDY
From Fabric **'RingReqIn'**: RD, WR – Will allocate entry (F2C_EnAllocEntryQ501H).
Buffer to Fabric **'F2C_RspQ501':** RD_RSP – Arbiter win (F2C_SelRdRingQ501H).
Buffer to Core **'F2C_ReqQ502':** RD, WR – Arbiter win (F2C_SelRdCoreQ502H).

4.3.4 Control Signals

The buffer's behavior is defined by assertion/de-assertion of its control signals:

- **F2C_SelDataSrcQnnnH:**
  select the source of the input data to the buffer - between data from RingReqIn, and F2C_Rsp.
- **F2C_EnWrDataQnnnH:**
  Enable for writing to the data entry in the buffer.
- **F2C_EnAllocEntryQnnnH:**
  Enable for allocating an entry in the buffer, for incoming transaction.

4.3.5 Structure

Number of entries in the buffer is defined by the number of cycles that takes to a request return to the core.

Entry structure is:

Buffer field	Size	Description
State	3'b	FREE = '000' , READ_RDY = '100' WRITE = '001' , READ_PRGRS = '011' READ = '010' , WRITE_BCAST = '101' , WRITE_BCAST_PRGRS = '110'
Requestor	10'b	{8'b coreID, 2'b Thread<#>} of the request initiator
Address	32'b	Request address.
Data	32'b	WR Data / RD_RSP Data

Table 10 - F2C Entry structure

4.3.6 Behaviour

- **Allocating new entry:**
  A RD/WR/WR_BCAST request arrives from the RingReqIn, it would be saved in one of the empty entries, according to the F2C_EnAllocEntry signal.
  If the buffer is full, and a new **read** request from the ring arrives, it would be forwarded to the ring output through RingReqOut.
  (We assume there is always a free write entry for "back2back" write request.)
- **In case of read response:**
  When a response sent to the ring output through RingRspOut
  (selected by F2C_SelRdRing and by SelRingRspOut), 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:**
  1) We save it in an entry in a regular WRITE state.
  2) We forwards this request to the ring output through RingReqOut.
- When a read request that already sent, receiving its appropriate response from F2C_RSP, change the state to READ_RDY.
- In every cycle, the buffer:
  - sends to the ring the **oldest** read response candidate it holds.
  - sends to core the **oldest** read/write candidate it holds.

4.4 F2C Control Signals

Signal	Width[bits]	Description
F2C_EnAllocEntryQ501	F2C_entries	Enables the allocation of F2C entries. The signal's assertion ensure that the first free entry will be allocated ,when a valid request, that matches the core's ID arrives to the RC.
F2C_EnWrDataQnnnH	F2C_entries	Enables data changing of F2C entries. The signal will be asserted when EnAllocEntryQ501 asserted, or when valid read response arrives to the buffer .
F2C_SelDataSrcQnnnH	F2C_entries	Select the data source for each entry . 0: Core(F2C_Rsp) , 1: Ring(RingReqIn) .
F2C_SelRdCoreQ502H	log2(F2C_entries)	Selects an entry toward the core. Determined by MRO
F2C_SelRdRingQ501H	log2(F2C_entries)	Selects an entry toward the ring. Determined by MRO

Table 11 - F2C Control Signals

4.4.1 States Machine

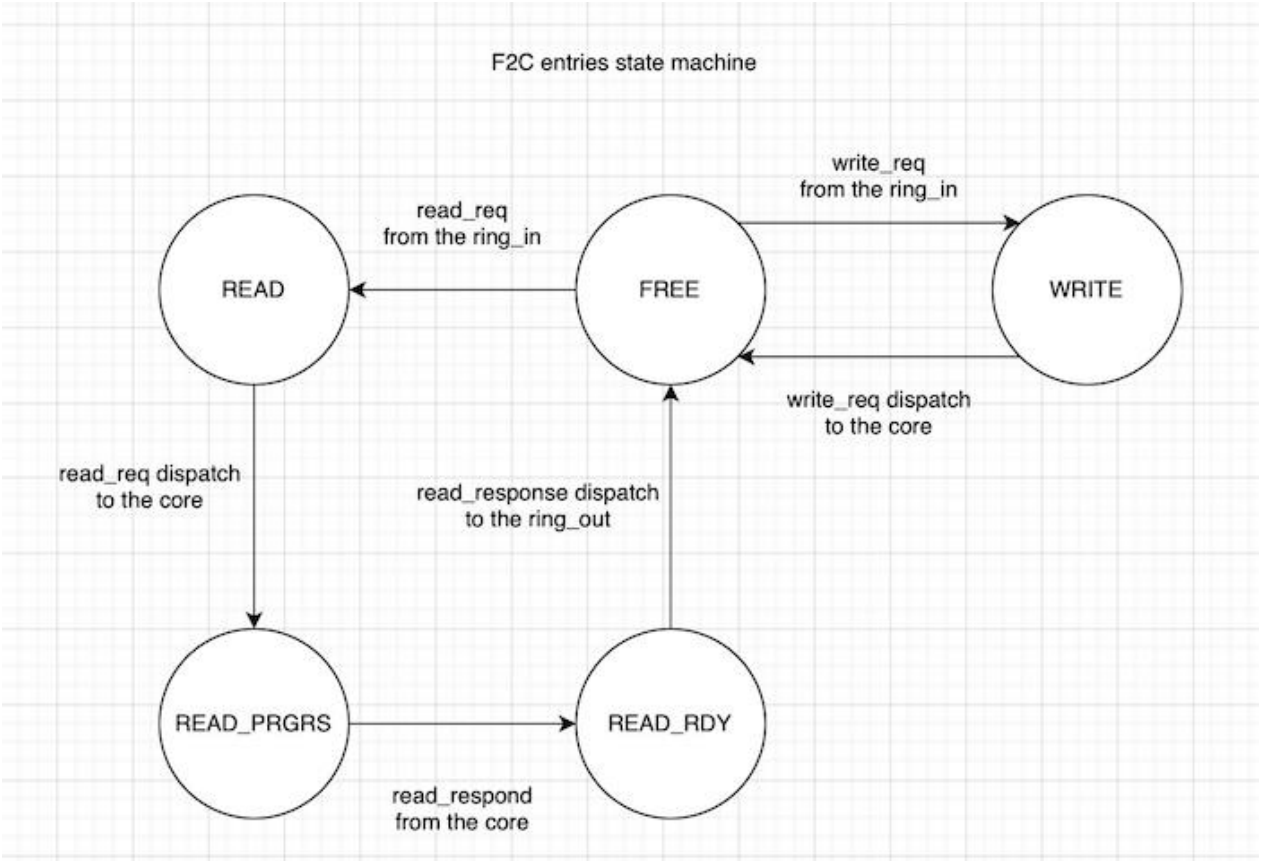

Figure 4 - F2C States Machine

4.5 MRO – MATRIX REMEMBER ORDER

4.5.1 Overview

The MRO is a logic used to remember who is the "oldest" candidate.
The purpose of this component is determine the control signals of its buffer.
For example: C2F_SelRdCore, and C2F_SelRdRing.
This logic will help choose properly which stored transaction (=entry) will get moved toward the core/ring. Considering in-order policy, and transactions<->wire association rules.
In the RC design we are using 2 instances of the MRO, one for each buffer.

4.5.2 Block diagram

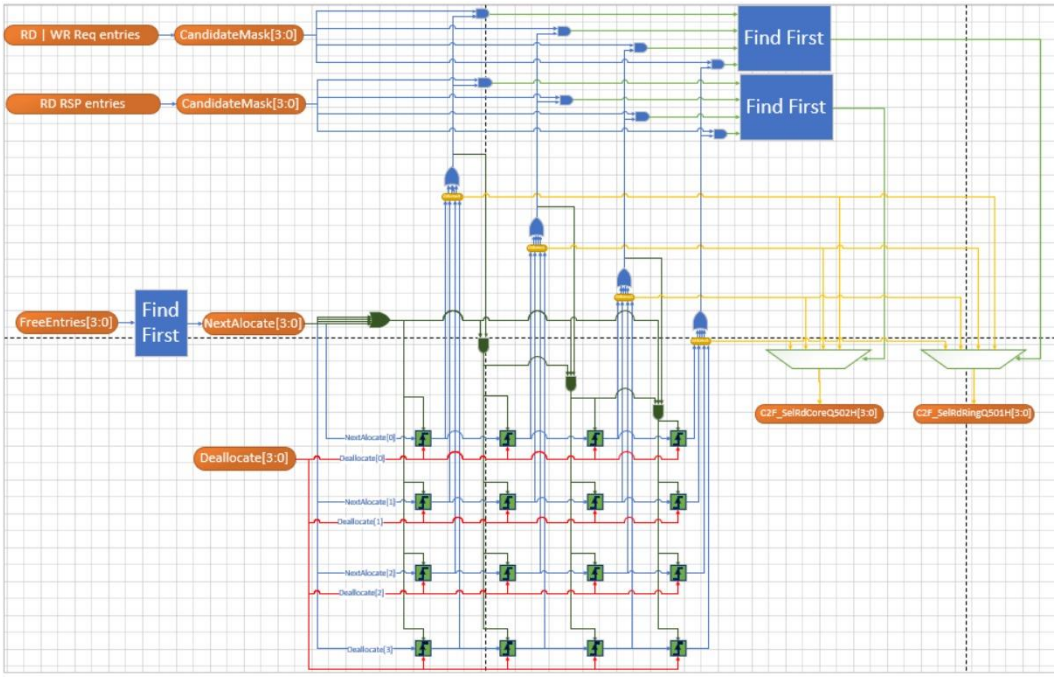

Figure 5 - MRO Diagram

4.5.3 Interfaces

Inputs:

- EnAlloc – is the current NextAlloc signals is valid in the current clock cycle.
- NextAlloc – determined by FindFirst when the given request toward the buffer is valid.
- DeAlloc – indicates for deallocation of an entry in this cycle.
- Mask0 – used to mask a particular command.
- Mask1 - used to mask a particular command.

Outputs:

- SelRdCore(Oldest0) – chosen entry to move from the buffer ,toward the core for example .
- SelRdRingOut(Oldest1) - chosen entry to move from the buffer ,toward the ring for example.

4.5.4 Behaviour

- Each column in the matrix represents the buffer entries states in a particular time. The deepest column is the oldest entries states, the first column (from left) is the most current entries states.
- Once there is a valid new NextAlloc, the writing enable for each FF in the matrix will get asserted, and each column will be stored into the next flops column.
- Once there is a valid new DeAlloc (i.e some entry changed its state to 'FREE') , the appropriate line will get reset .
- In every cycle, Oldest0/1 are determined by using FIND FIRST component on OR gate result from each column in the matrix, after filtering the relevant entries (using the masking mechanism).

4.6 FINAL MUXS

4.6.1 Overview

In RC there are 2 "Final Muxs"

1) RingReqOut: SelRingReqOutQ501H - Arbitrate between the candidates:
   - RingReqIn - Ring input request that are not targeting the current Tile.
   - Bubble/NOP - In order to avoid overloading traffic onto the RingReq Channel.
   - C2F Buffer - Core requests to Fabric.
1) RingRspOut: SelRingRspOutQ501H - Arbitrate between the candidates:
   - RingRspIn - Ring input Responses that are not targeting the current Tile.
   - Bubble/NOP - In order to avoid overloading traffic onto the RingRsp Channel.
   - F2C Buffer - Core responses to Fabric. (wants to return to the origin core RD request)

Each final mux will select which signal will be output to the fabric considering:

1) Traffic overloading on the ring. (Starvation on next Tile)
2) And priority principles. (Coherency/Oldest/quality of serves)

4.7 GLOBAL RC CONTROL SIGNALS

SelRingReqOutQ501H – The selector to RingReqOut
SelRingRspOutQ501H - The selector to RingRspOut

4.7.1 SelRingReqOutQ501H

Signal	Priority	Description	Chose condition
RingReqInQ501	1	The 'raw' input from RingReqIn path.	If our core is not the addressee for the transaction / Write broadcast when the current core is not the initiator.
NOP	2	Invalid dummy transaction	When we counted 4 valid transactions in a row in the this output .
C2F_ReqQ501	3	Transaction initiated from the core	If there is a valid transaction available.

Table 12 - RingReqOut Muxing

4.7.2 SelRingRspOutQ501H

Signal	Priority	Description	Chose when ?
RingRspInQ501	1	The 'raw' input from RingRspIn path.	If our core is not the addressee for the transaction .
NOP	2	Invalid dummy transaction	When we counted 4 valid transactions in a row in the this output .
F2C_RspQ501	3	Read response from our core to the fabric.	If there is a valid transaction available.

Table 13 - RingRspOut Muxing

5 RC PIPE STAGES + DATA PATHS

Signals in the pipeline will be recognized using the suffix **QnnnH**:

'Q' -> name of Clock domain.

'nnn' -> Pipe stage number. Example: 500, 501, 502.

'H' -> Signal is a "positive edge" sensitive. Signal may change when clock transition Low to High.

5.1 RINGREQINQ500H -> F2C_BUFFER

Q500H) A valid transaction arrives from ring.

Q501H) Allocated in one of the F2C entries if there's a free slot.

Q502H) Dispatch to the core, through F2C_Req interface.

5.2 RINGREQINQ500H -> RINGREQOUTQ502H

Q500H) A valid transaction arrives from ring .

Q501H) Didn't allocated in one of the F2C entries.

Q502H) The transaction continues to the ring through RingReqOut interface.

5.3 F2C_RSPQ500H

Q500H) A valid read response transaction arrives from the core.

Q501H) Data getting written to the appropriate entry in F2C buffer.

Q502H) The read response selected to get dispatched toward the core in RingRspOut interface.

5.4 C2F_REQQ500H

Q500H) A valid read/write transaction arrives from the core.

Q501H) Allocated in one of the C2F entries if there's a free slot.

Q502H) The transaction will be dispatch toward the ring, if selected by SelRingReqOut.

5.5 RINGRSPIN -> C2F_BUFFER

Q500H) A valid response arrives from ring.

Q501H) Data getting written to the appropriate entry in C2F buffer.

Q502H) Dispatched toward the core in C2F_Rsp interface.

5.6 RINGRSPIN -> RINGRSPOUTQ502H

Q500H) A valid response arrives from ring.

Q501H) Data **did not** got written to the C2F buffer.

Q502H) The transaction continues to the ring through RingRspOut interface.

6 ORDERING AND STARVATION RESTRICTIONS

- Request should be sent on the ring in the order they come in from the core.
- Pending read requests in the C2F buffer , would change their state to RDY when a matching read response received , and in the next cycle they can be sent to the ring .
  (* No need for reorder - because it keeps only read commands - read commands doesn't invoke dependcies issues.)
- Pending request in F2C buffer , should be commit in the order they were sent.
- The core thread will not send any request , until it get responded to his first read request.
- When C2F buffer is full, raise a stall signal .
- When the RC gets a transaction that it is not the addresee , the incoming transaction will be forwarded immediately.
- When the RC gets a transaction that it cannot handle, because its relevant buffer is full, the incoming transaction will be forwarded immediately.
- In order to certify progress , and to avoid deadlocks , it is necessary that at least one of every four cycles , would be invalid .
  (* would be certified by that each RC would count the cases when it receives invalid request from the ring input, and it send a valid request to the ring output. when this counter reach 4 , the RC must send a nop .(in that case ,also reset the counter) this counter would be reset in case the RC send to the ring output invalid request.)

7 EXAMPLE TRANSACTION FLOWS

7.1 CASE 1 – WR FROM RING.

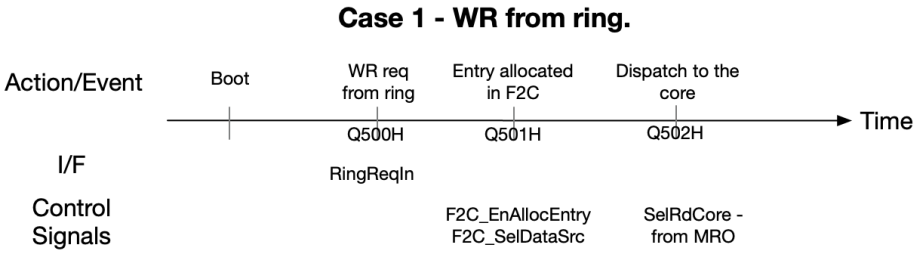

Figure 6 - Case 1 Timeline

7.2 CASE 2 – RD FROM CORE, RD_RSP FROM RING.

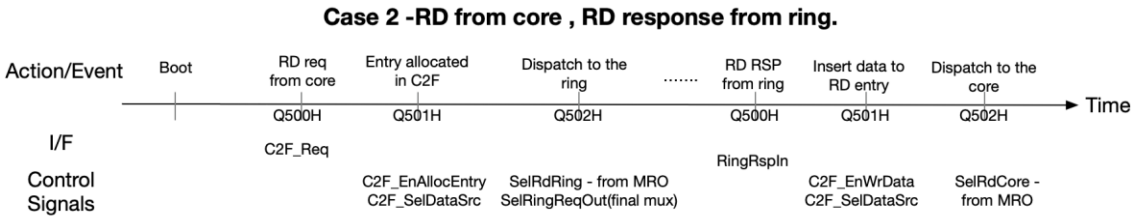

Figure 7 - Case 2 Timeline

7.3 CASE 3 – STALL SIGNAL RAISED.

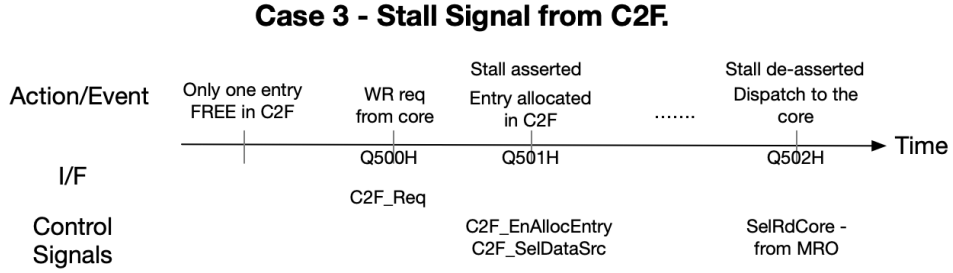

Figure 8 - Case 3 Timeline

7.4 CASE 4 – FORWARDING.

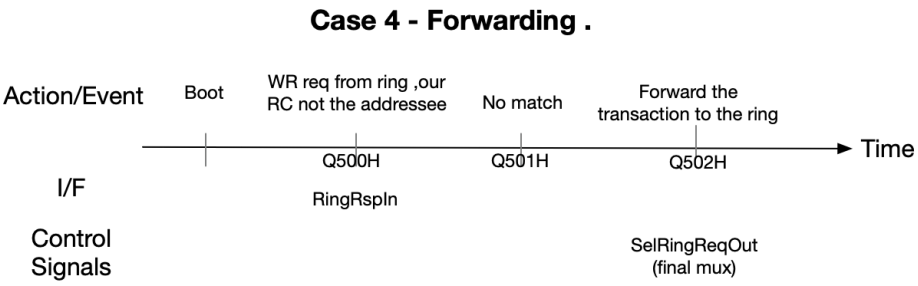

Figure 9 - Case 4 Timeline