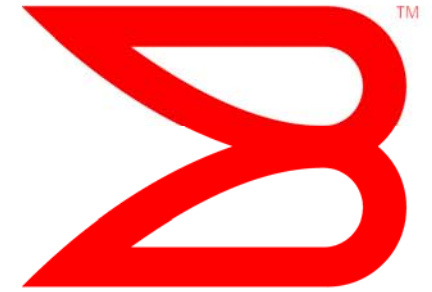


BROCADE



Pathing in the FCoE Adapter

Does it help to use Mapped Network based MAC addressing in FCoE or is the NIC's "Burnt-in" MAC sufficient?

T11/07-655v0

John L. Hufferd
Robert Snively

Two Conflicting Assertions regarding Pathing within the HBA/ASIC: Which is correct?

1. Network Based MAC address helps the routing (pathing) of storage vs. other Ethernet traffic within an HBA ASIC
 - Adapter uses a Mapped Network-based MAC address to discriminate between normal Ethernet flow and Storage flows, or between different Storage Flows then:
 - Adapter coding and construction will be simpler and operation will be more secure.

OR

2. The use of a Single “Burnt-in” MAC address provides robust and efficient handling of FCoE traffic
 - Adapter uses standard processing of one or two “Burnt-in” MAC addresses to discriminate between normal Ethernet flow and Storage flows, or between different Storage Flows then:
 - Adapter coding and construction will be simpler and operation will be more secure.



Assumptions used regarding the number of Logical FC interfaces to the Host from each Adapter

We will be assuming that a Physical adapter may support 100s of FC sessions

- Can and should be accomplished via normal NPIV approaches

May have many NPIV instantiation per Logical FC connection (VN_Port)

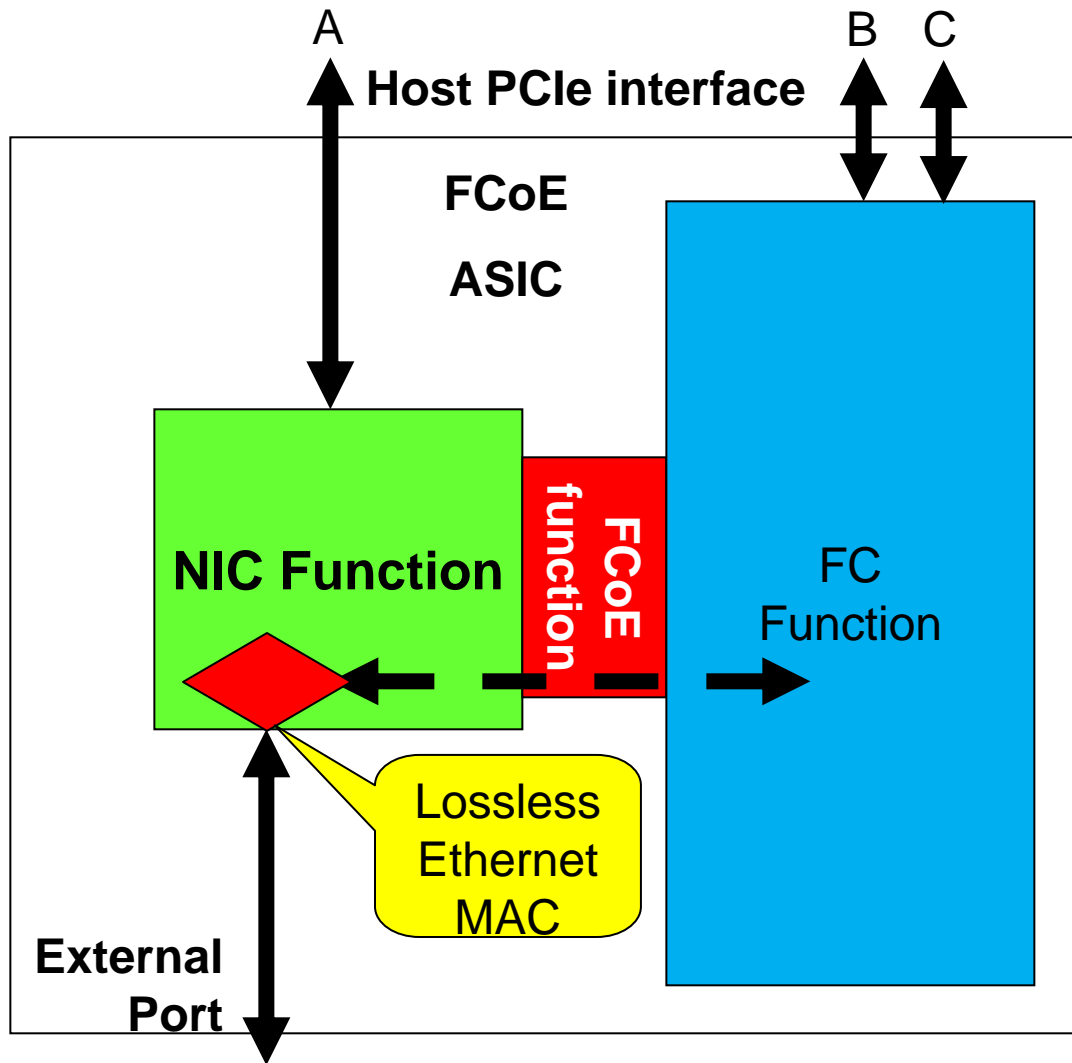
- This will permit 100s of logical FC Fabric Logins (via FDISC)

Each Logical FC connection may have separate MAC addresses or may share a common MAC address

- Any of these MAC addresses can be “Burnt-in” or MAPPED

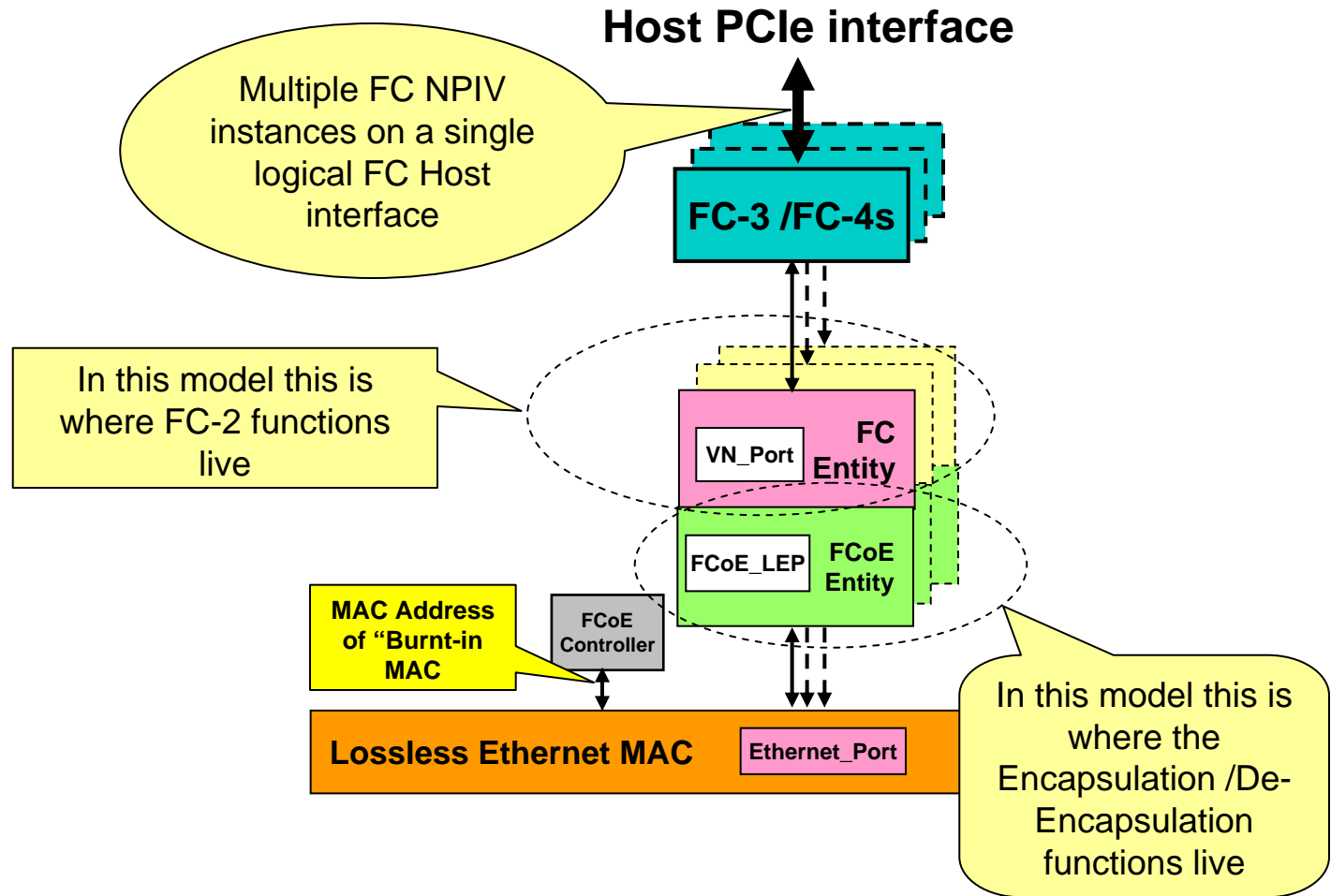


Functions of an FCoE ASIC

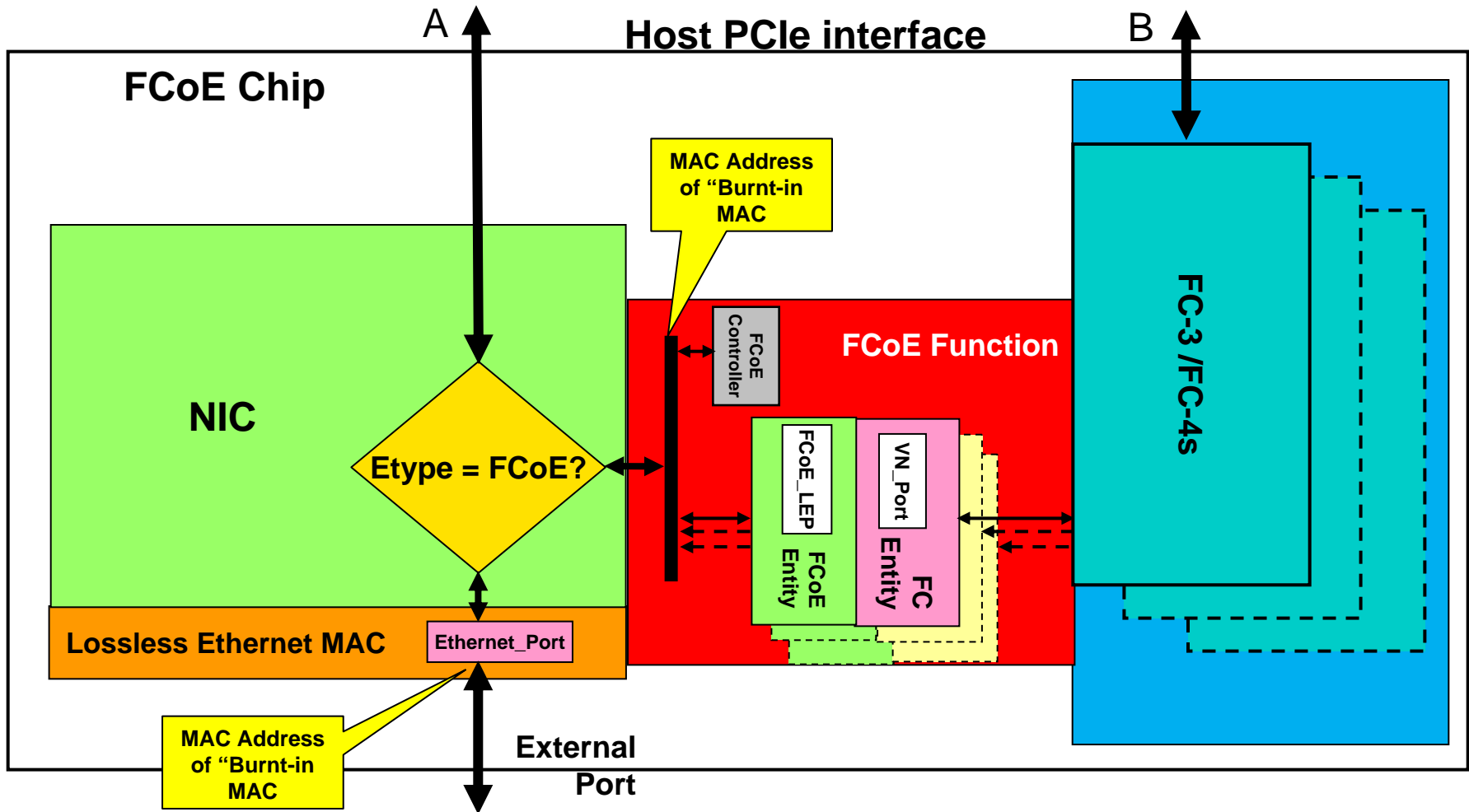


- Has a Normal NIC interface (A) to the Host
- Has one or more Normal FC interfaces (B,C) to the Host
- FCoE functions not seen by the Host
- FCoE functions perform the Encapsulation and De-encapsulation
- And instantiates a VN_Port

Model of the VN_Port with a Single Logical FC interface

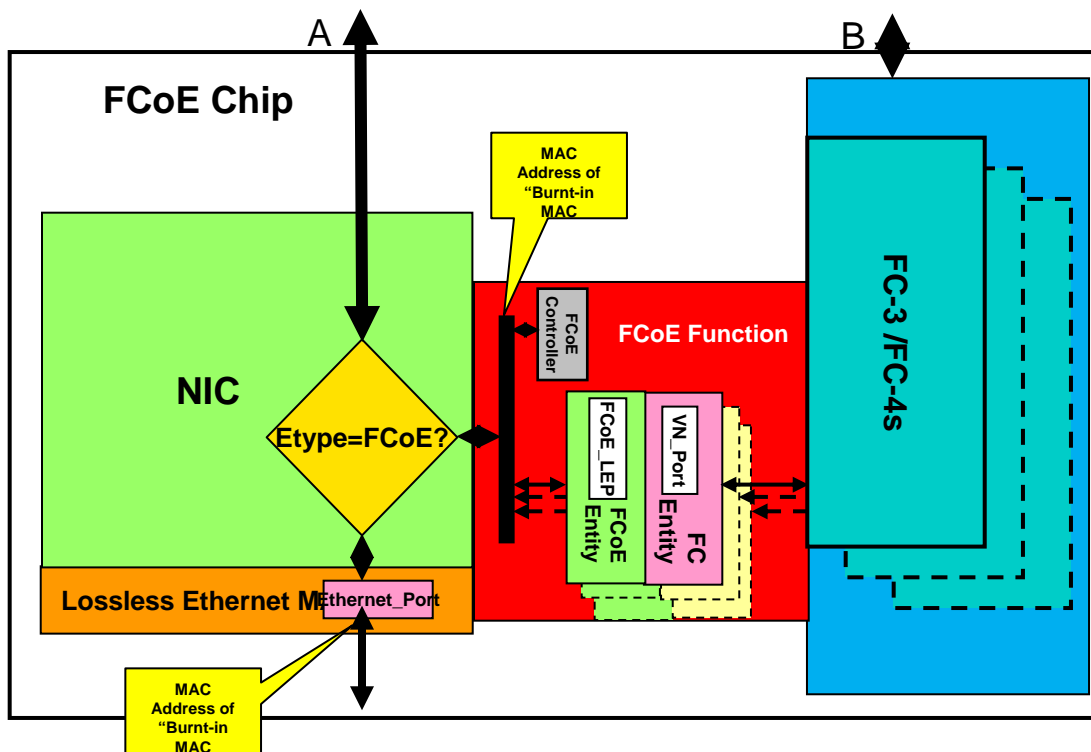


FCoE ASIC with Host NIC interface and FC interface (showing the Model)



Logic of the FCoE chip

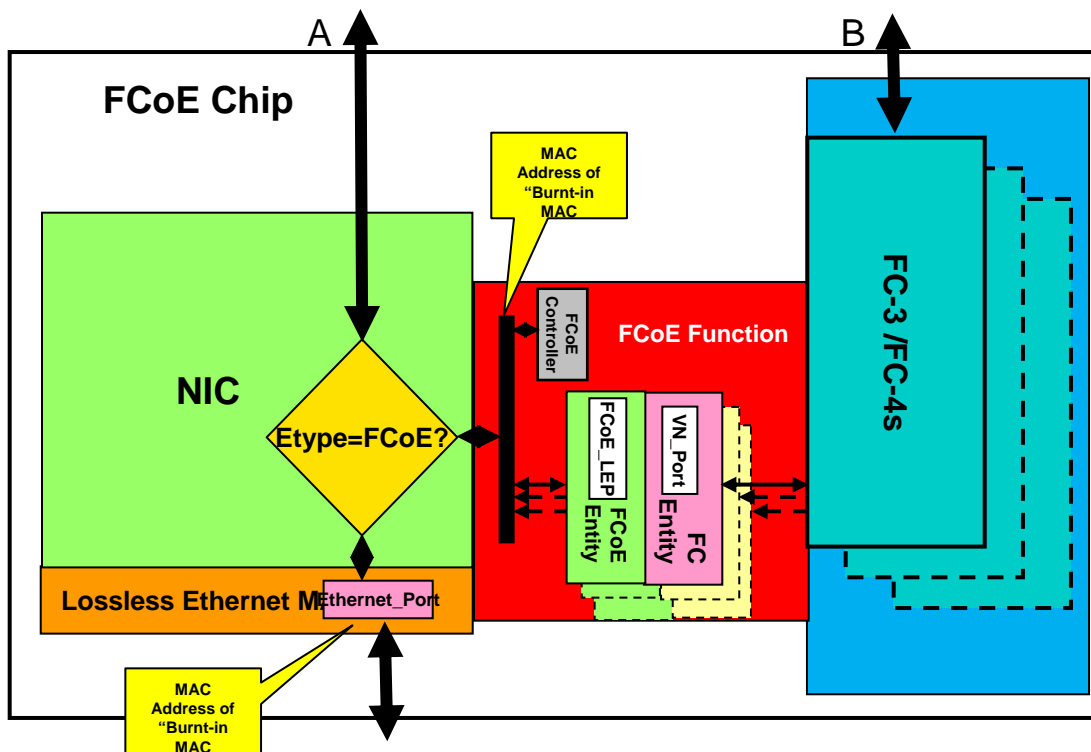
1. The traffic is 1st Routed based on Ethertype
2. FCoE Controller sends Discovery requests (not covered here), & Instantiates the FCoE/FC Entities)
 - Using the “Burnt-in” MAC address



2. The FCoE VN_Port is instantiated as a path to the FC2/FC-3/FC-4 Stack
3. The instantiated VN_Port path will either
 - Continue to use the “Burnt-in” MAC address
 Or
 - Dynamically build a new MAC address from the FC_ID returned from the FLOGI response (but this approach requires understanding of FLOGI Response Frames)
3. The FCoE Entity must strip off the FCoE & Ethernet Headers and Pass FC frames to the FC Entity (FC-2) then to FC-3/FC-4 stacks (the FC2/3/4 is normal FC)

Burnt-In Or Mapped MAC address

1. In this case the FCoE controller operates with the "Burnt-in" MAC Address
 → & the Ethertype test is done in the NIC



2. For a Single Logical FC connection (VN_Port)
 → **Only one MAC Address needed; the original one**
3. All incoming NPIV instantiations
 → use a common function to strip off the FCoE & Ethernet Header and
 → Send the Message to the FC2, & FC3/FC-4 stack handler
4. All outgoing NPIV instantiations
 → Use a common function to add the FCoE & Ethernet Header and
 → Send out with Burnt-In MAC Address as Source
5. **A MAPPED MAC address does not help & requires FLOGI response knowledge**



Can also Route internally based on “Burnt-in” MAC Addresses

A MAC could have 2 “Burnt-in” MAC Addresses

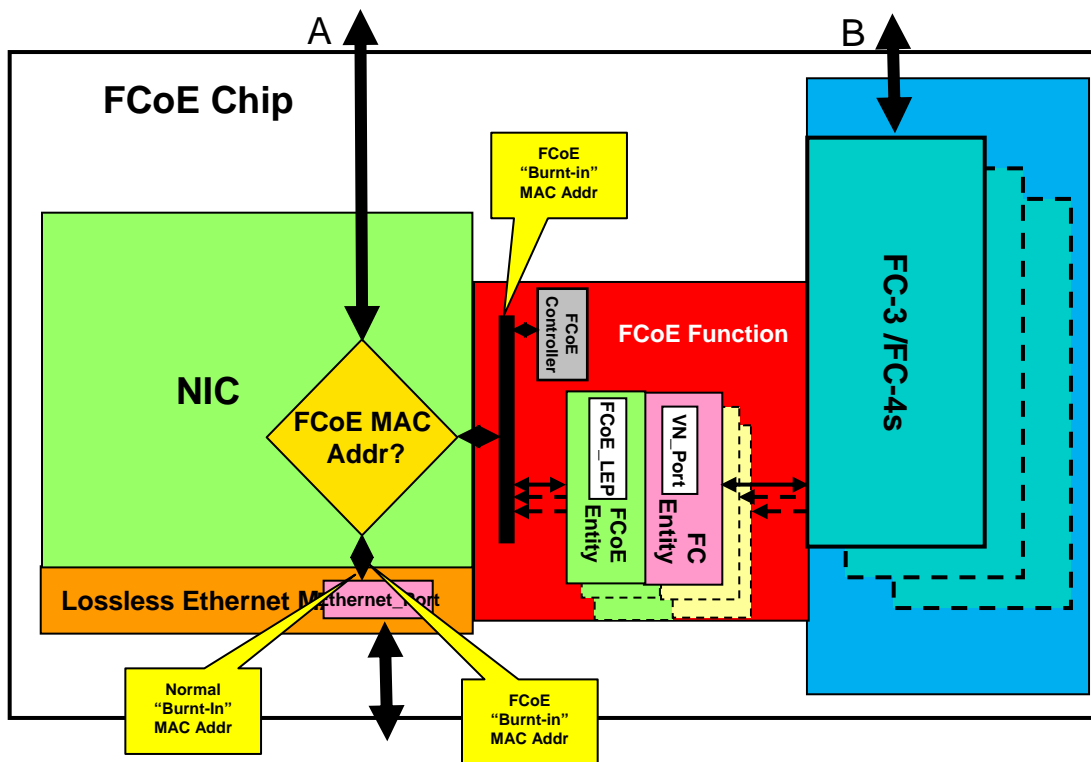
This permits Port Binding in the OS and Software FCoE implementations (perhaps via a VM) to operate through the same NIC with the HW FCoE

Because they would both have Ethertype = FCoE

One “Burnt-in” MAC Address for Normal NIC operation

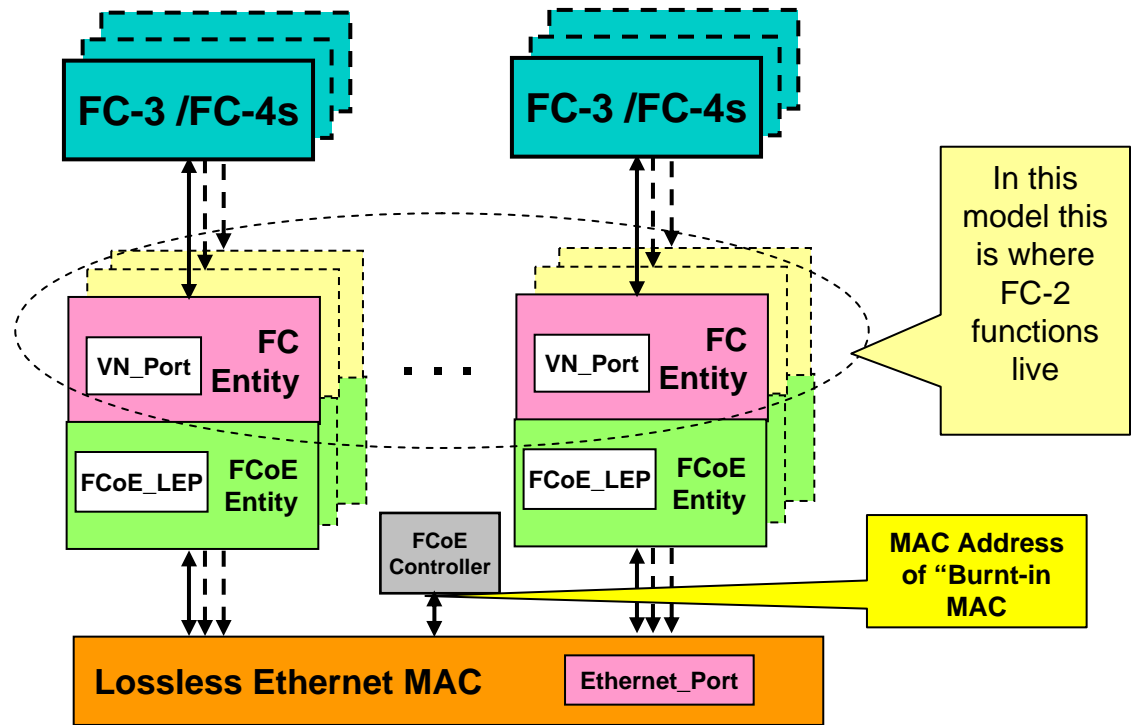
A Second “Burnt-in” MAC Address for FCoE operations

Ethertype = FCoE tests can be done in the NIC or the FCoE controller and then non FCoE Frames with FCoE destination MAC Addresses can be discarded



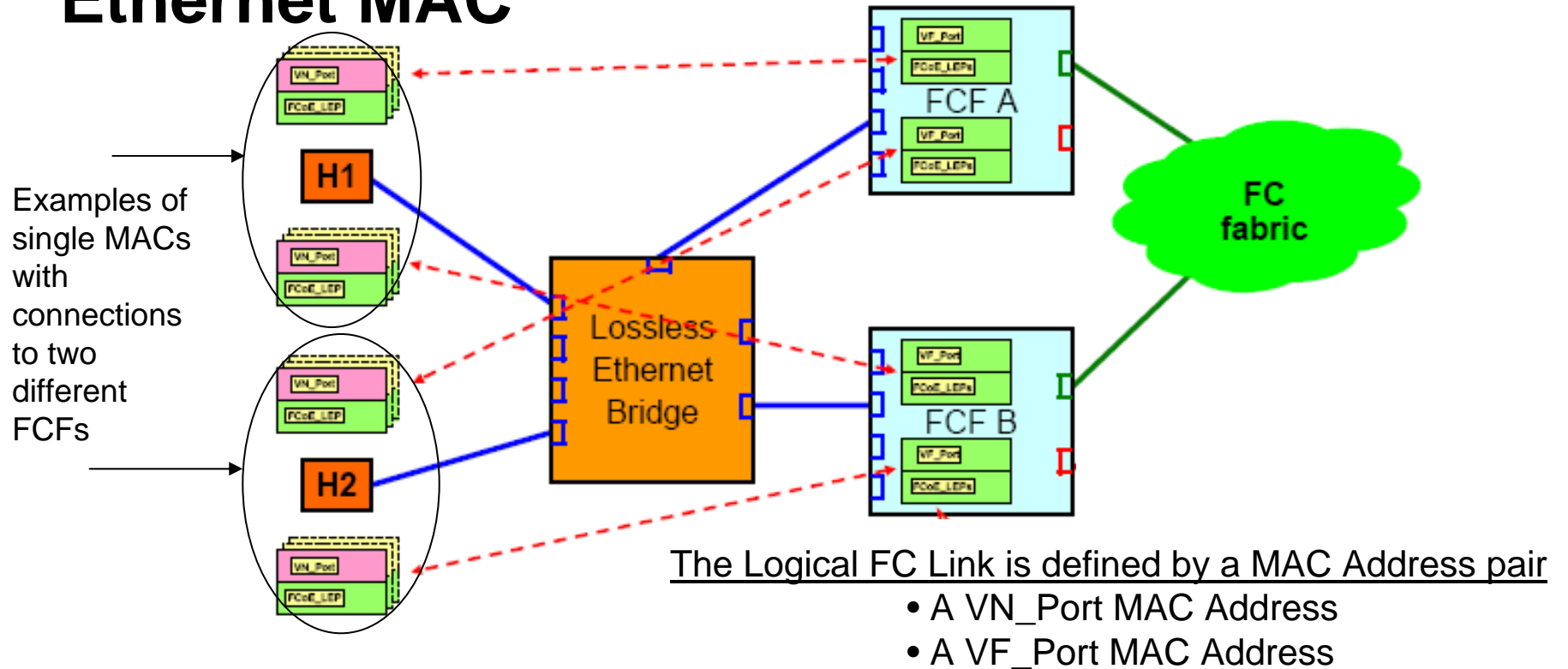
Model of the VN_Port with Multiple Logical FC interfaces

- For each logical N_Port (VN_Port) there is one FLOGI and perhaps 100's of FDISC
- Each VN_Port is seen by the Host as a separate (logical) FC connection
- The number of (logical) FC connections is implementation dependent
- There can be multiple Physical Ethernet Ports, each with multiple logical FC connections (VN_Ports)



- Only one MAC Address is required for all the Logical FC connections (VN_Ports) on a single physical MAC -- see following slides

Multiple Logical FC connections via a single Ethernet MAC



For a logical FC link the FCoE Frames are always sent to and received from a specific FCF's MAC Address

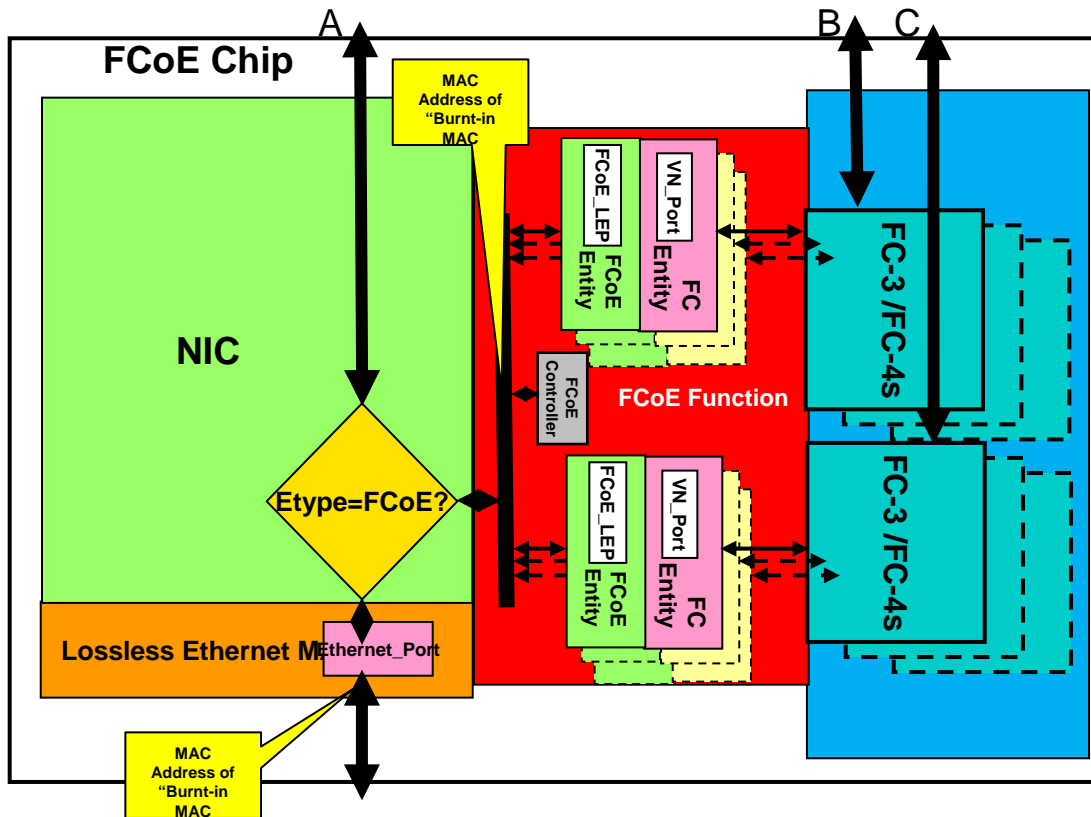
- Therefore, pathing to and from the FC driver is always defined by the MAC Address of the partner FCF

Note: No routing/pathing within the HBA ASIC needs Mapped MAC Addresses

For Multiple Logical FC Interfaces

1. The FCoE controller may continue to operate with the single "Burnt-in" MAC
 - the Ethertype test in this case is still done in the NIC

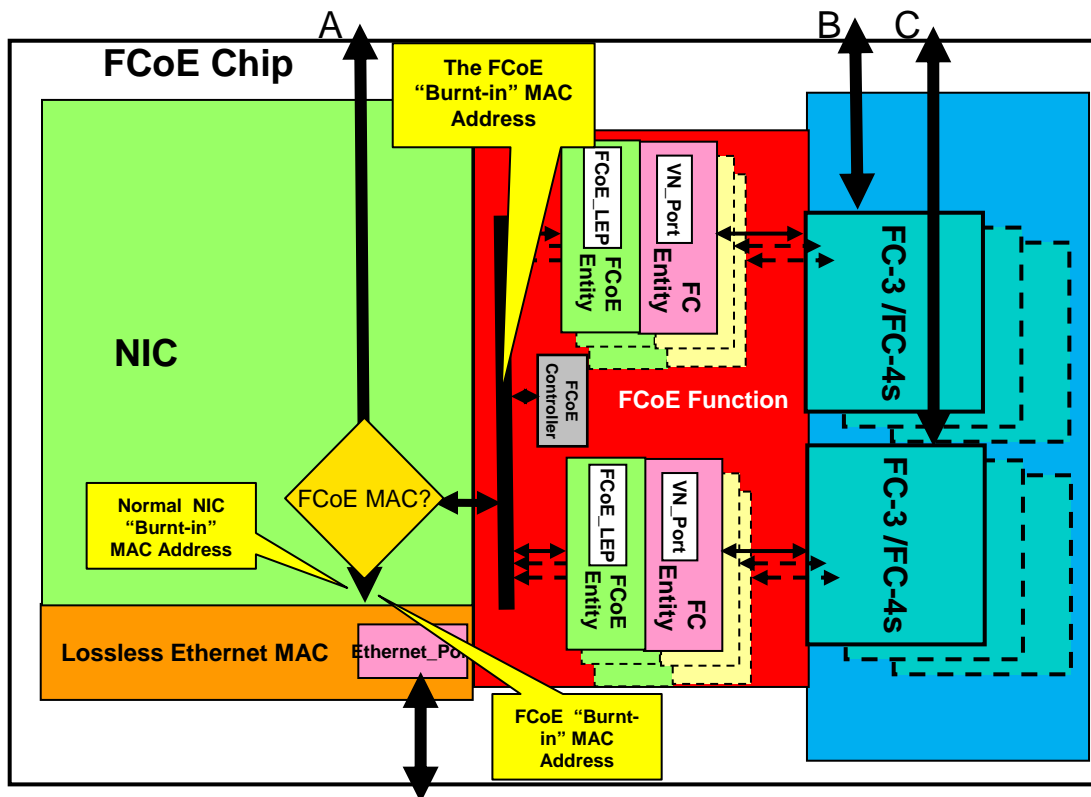
2. For Multiple Logical FC connections
 - Only one MAC Address needed; the original one
3. All incoming NPIV instantiations
 - use a common function to strip off the FCoE & Ethernet Header &
 - Route to the appropriate FC-2, & FC-3/FC4 Stack based on what was in the Source (FCF) MAC Address
4. All outgoing NPIV instantiations
 - Use common function to add the FCoE & Ethernet Header and
 - Send out with Burnt-In MAC Address as Source, & Destination FCF MAC Address



Multiple “Burnt-in” MAC addresses

Used to separate HW based FCoE from other Ethernet Traffic
→ Most NICs come with several “Burnt-in” MAC Addresses

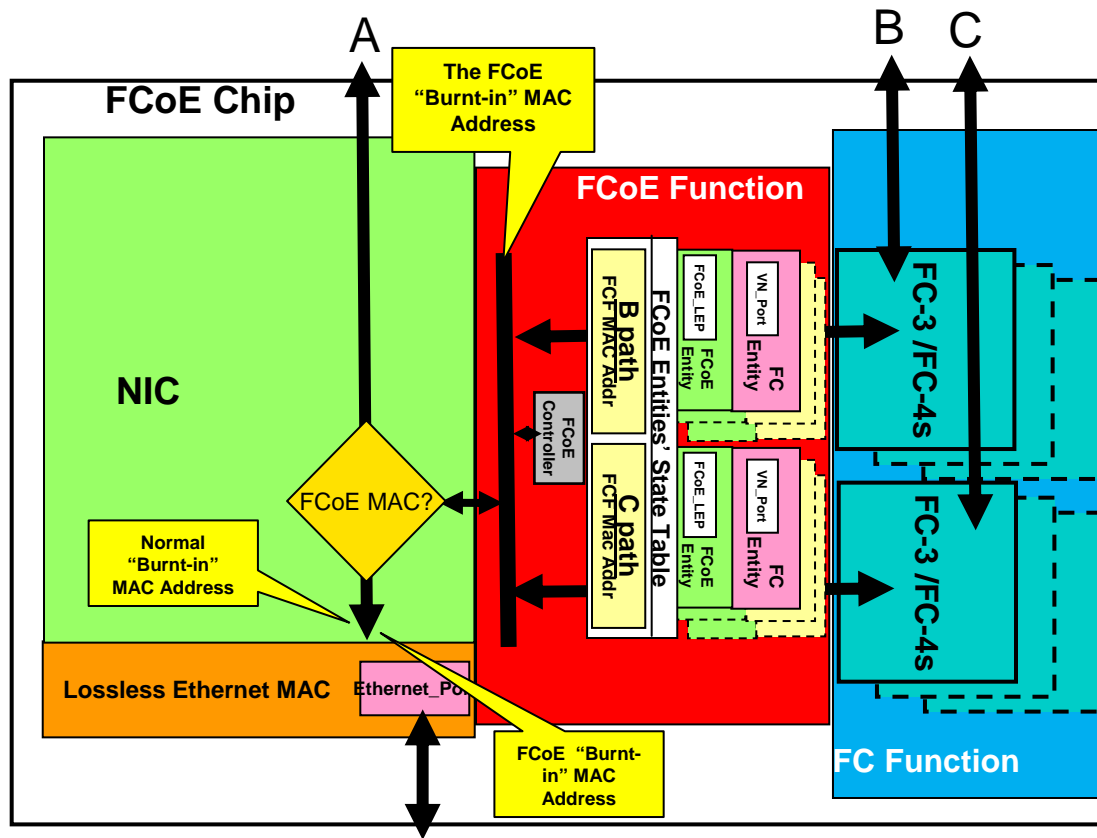
Permits IP, port binding, & Software FCoE to share one “Burnt-in” MAC Address, and HW FCoE to operate on another “Burnt-in” MAC address



But only one of the MAC Addresses needed for the HW FCoE since the FCoE controller uses the “Burnt-in” MAC address to perform discovery and then direct the FC frames to the appropriate FC2, & FC3/FC4 stacks based on Ethernet Source (FCF)

Implementation Example (steps 1-9)

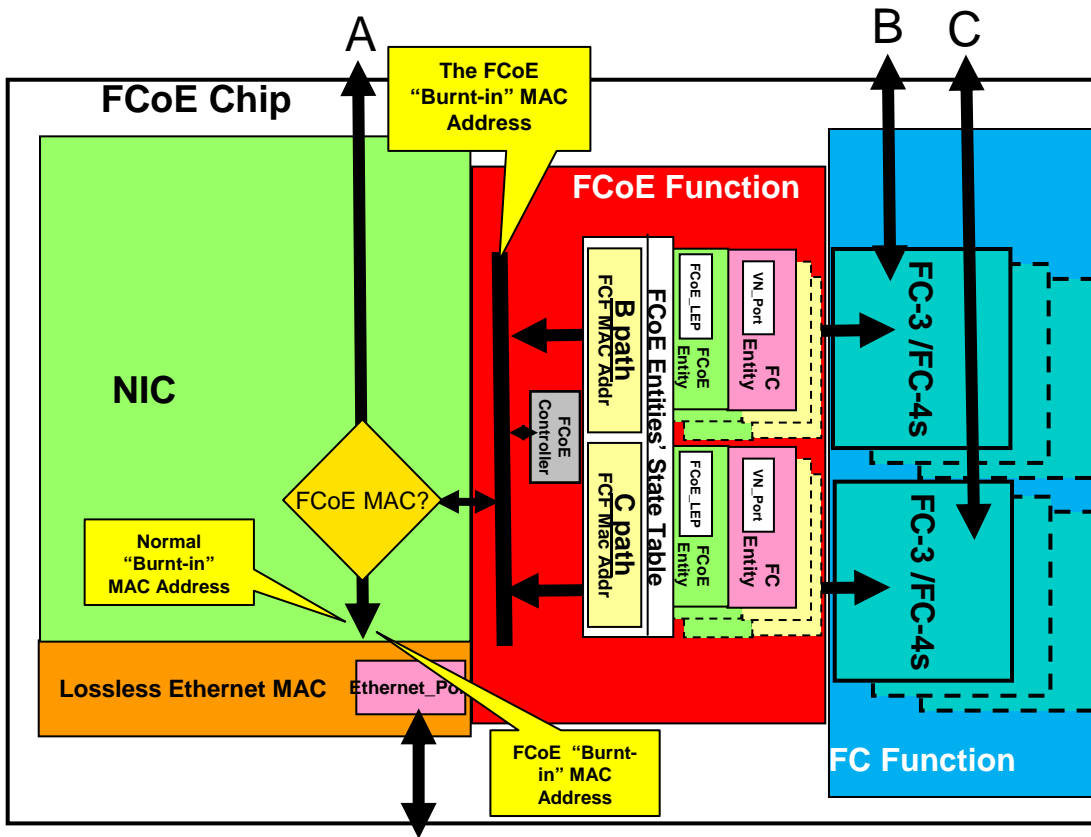
1. FCoE Controller does discovery (process not discussed here)
2. Sorts out Destination FCF MAC Addresses for paths B & C, then keeps that relationship in the State Table
3. Host issues FLOGI on Host interface B (or any other FC outgoing operation)



4. FC Frame passed to the FCoE Entities'
5. FCoE & Ethernet headers/frames created
6. Path B's Destination FCF MAC Addr placed into Ethernet Frame as Destination Addr
7. FCoE "Burnt-in" MAC Addr placed into Ethernet Frame as Source Addr
8. FC Frame sent out using FCoE Ethernet Frame
9. On subsequent outgoing Frames repeat 3-8

Implementation Example (steps 9-14)

9. FLOGI Response FCoE frame (or any other frame) is received and
10. The Source (FCF) MAC Address is used to find the appropriate Path (in this case Path B)
11. FCoE Entity strips off the Ethernet & FCoE Frame & headers
12. The Frame is now a real FC Frame
13. Using the appropriate path (in this case Path B) pass the FC Packet to FC-2 (FC Entity) and on to FC3/FC4
14. On subsequent incoming Frames repeat 9-13



Analysis

Characteristic	Server-provisioned MAC (“Burnt-in”)	Fabric-provisioned MAC (“Mapped”)
Number of MACs per ENode	1-2	100’s
Separation of ENet and FC	ENet and FC separated	ENet MACs derived from FC information
Identification of FCF-MAC	FCF-MAC source address	FCF-MAC source address
Internal routing	Ethertype & FCF-MAC Address	Mapped ENode MAC, Ethertype
ENode internal simplicity	Simple and standard	Simple?, standard? requires extraction of FC_ID by the FCoE Controller and creating more unique MAC Addresses



Summary and conclusions

There seems to be no important value in the HBA for Mapped MAC addresses. In fact:

- Not using Mapped MAC Addresses requires no knowledge of, or extracting information from, the FLOGI Response
- Using “Burnt-in” MAC Addresses makes things very simple

There can be 100’s of NPIV connections supported with a single MAC Address

There can be several Logical FC connections (VN_Ports) sharing a single MAC address each logically connected to a different FCF

Use of “Burnt-in” MAC Addresses are SIMPLE and straight forward

