

Ilia
greenblat@mac.com
+972-54-4927322

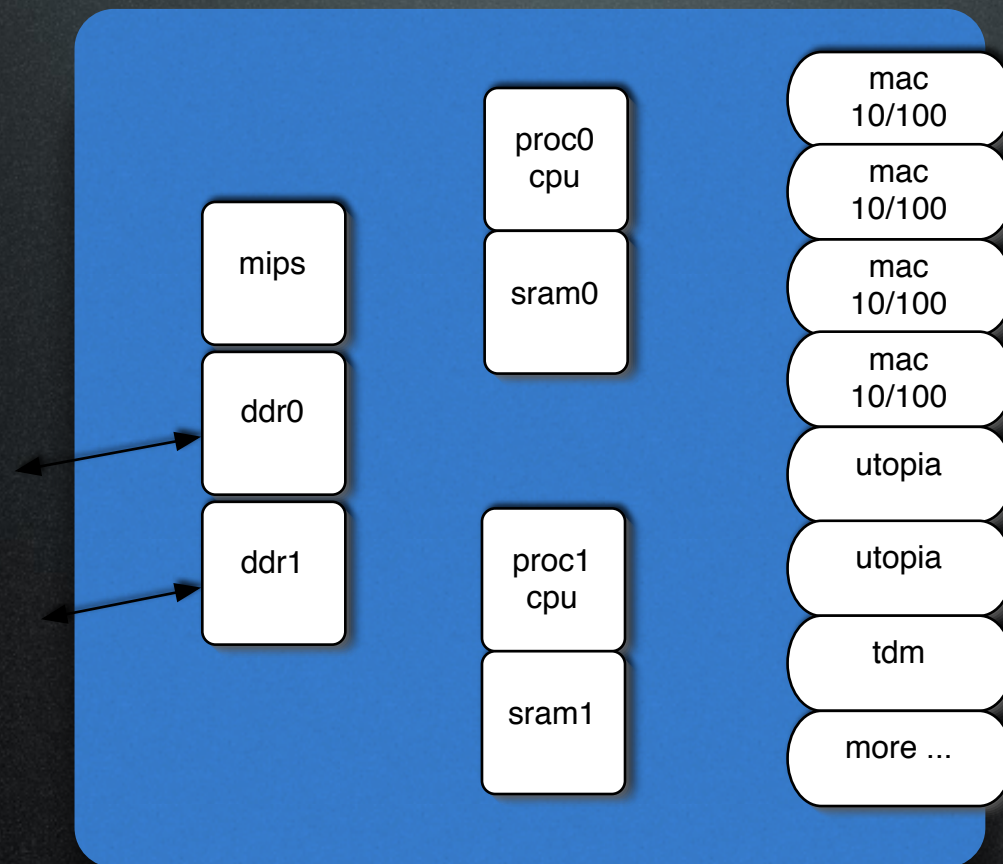
presentation version, not suitable for printing
unless you sell ink.

History

- July 2000: Design center in Israel implements network processor chip.
- The chip built from scratch works fine after 24 month, manufactured in 0.18 TSMC. Year later the center is closed. The chip & technology abandoned.
- Special feature of Chip was that instead of modules hanging on busses, it used interconnect fabric built around rings topology.
- At that time only Sonics NOC was on the market and it was quite young.

Chip overview

supermarket of diverse comm peripherals
glued and driven by software



CPU0 and CPU1 (custom made processors did mostly data plane, while HOST/Mips/Arm did control plane.

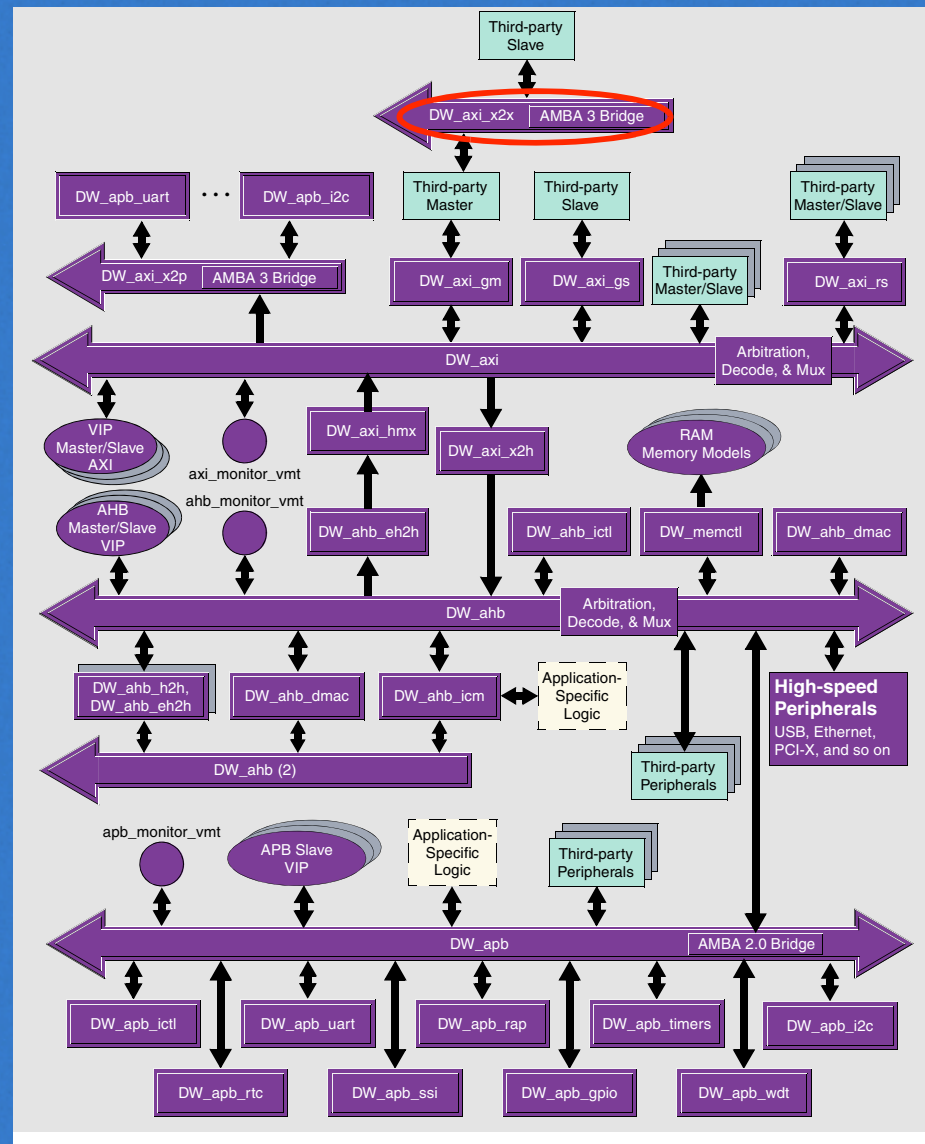
Challenges

- Glue together numerous peripherals into manageable system.
- Hit moving target of application / customer requests / market trends.
- Reduce the cost of changes by doing truly modular design.
- The chip serves software, make the software job manageable.
- Survive the H/W verification.
- Survive P&R and the rest of backend.
- Make clocking / power / test easy.
- Facilitate system modelling for performance studies.

Solution?

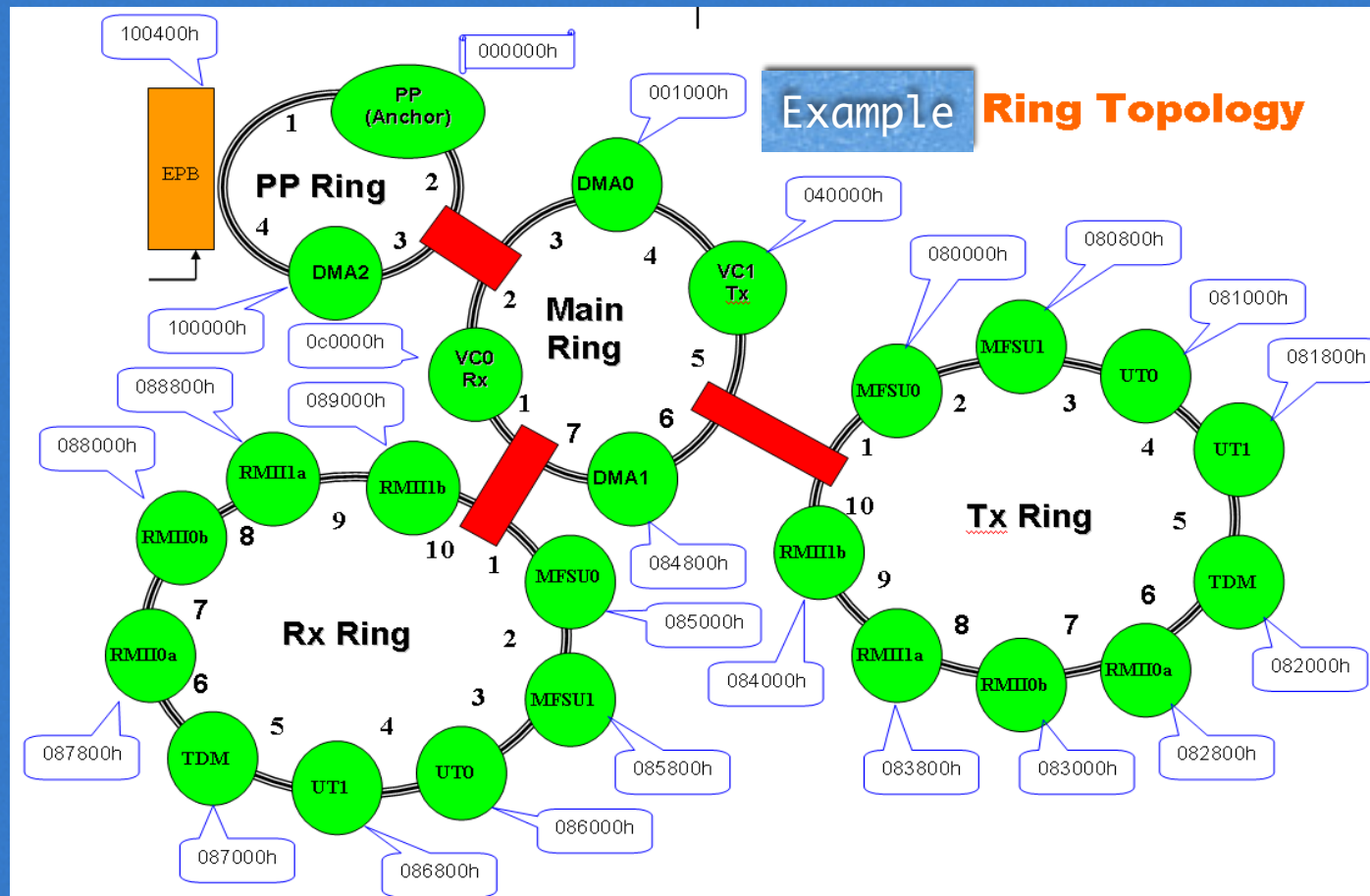
AXI -
industry
standard
fabric.

The purpose of
complexity is to
lock in the poor
user forever.

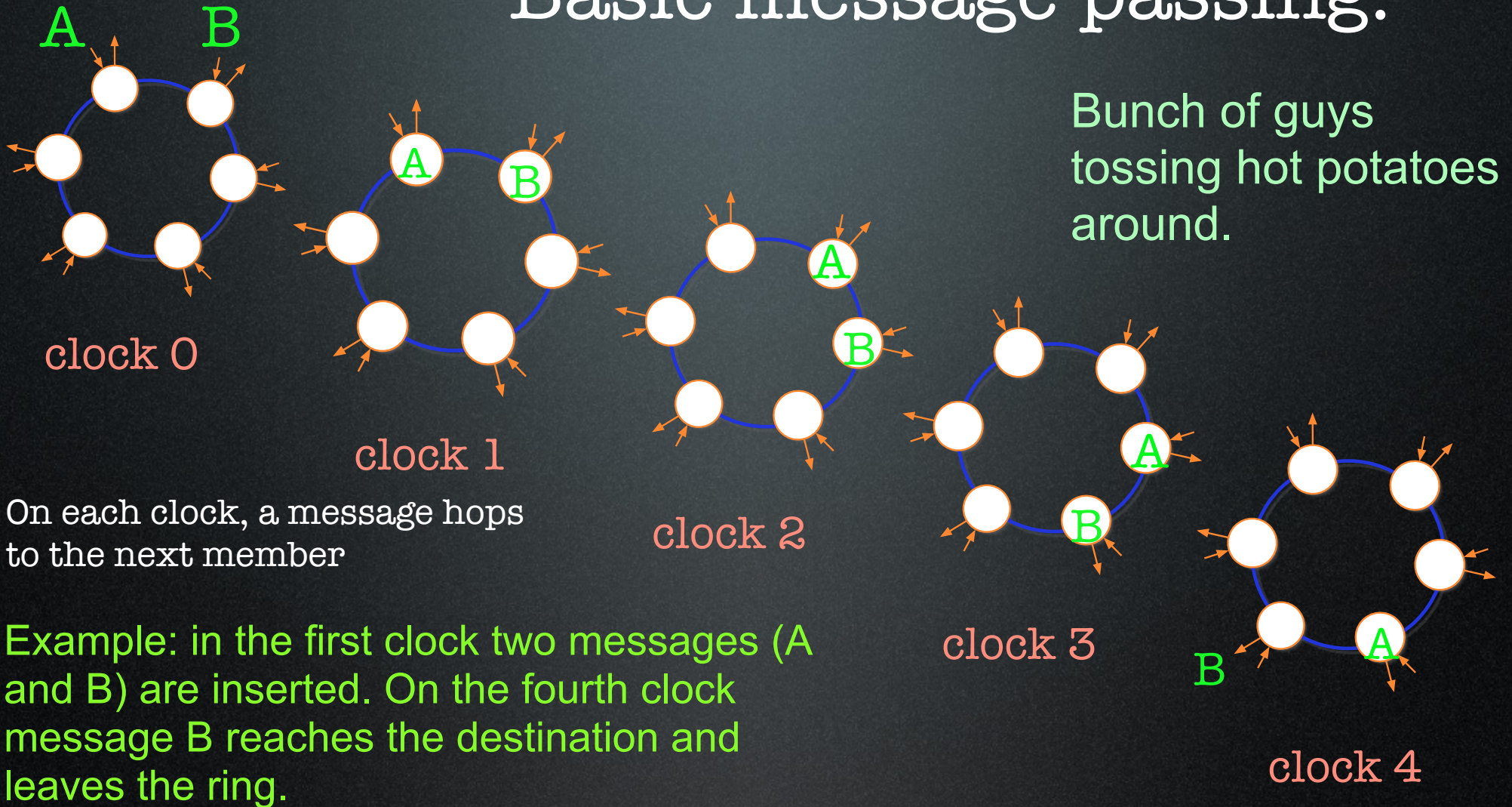


Solution:

The Rings Chip topology

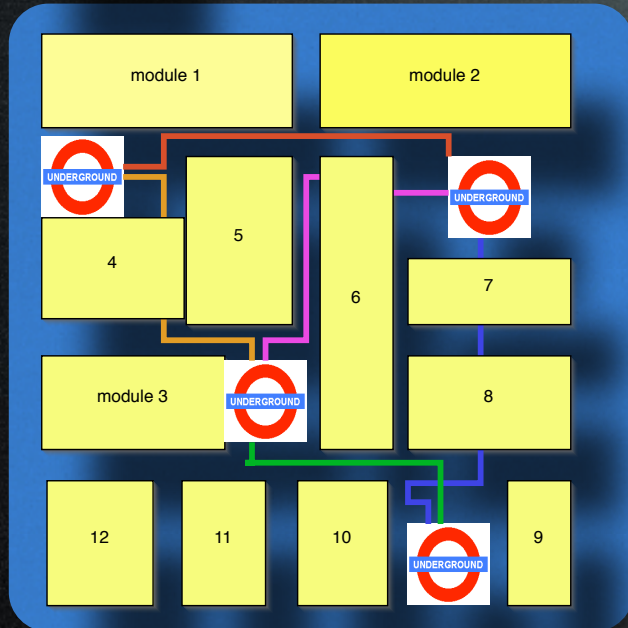


Basic message passing:

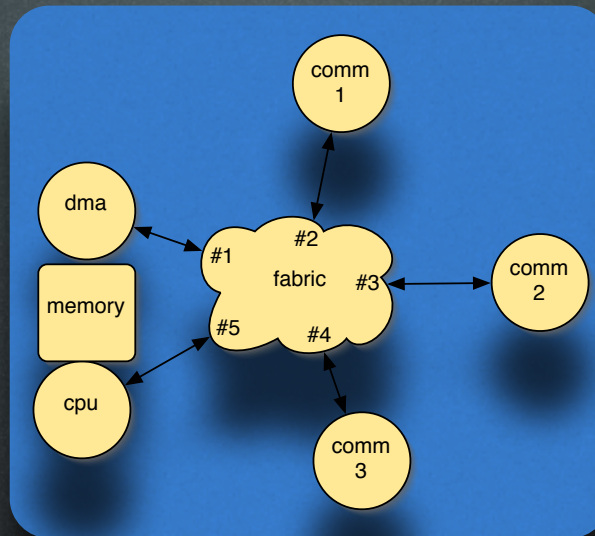


Each member examines the address field of the message. If the message is for itself - it is consumed, otherwise it is passed downstream (next clockwise member).

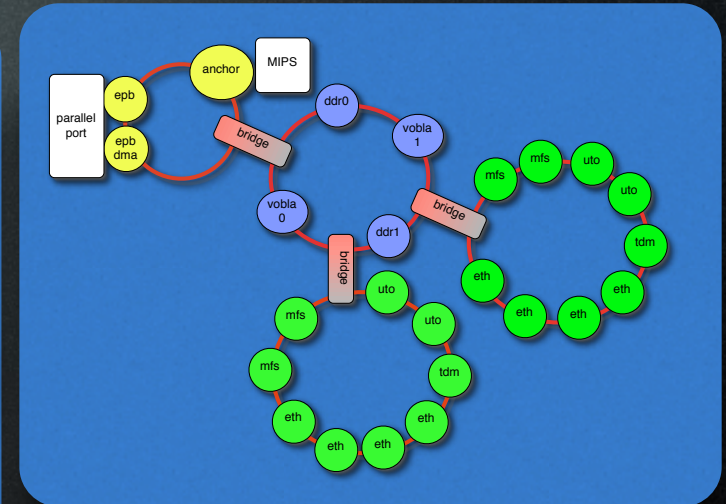
3 Views of fabric



“London Tube”
view



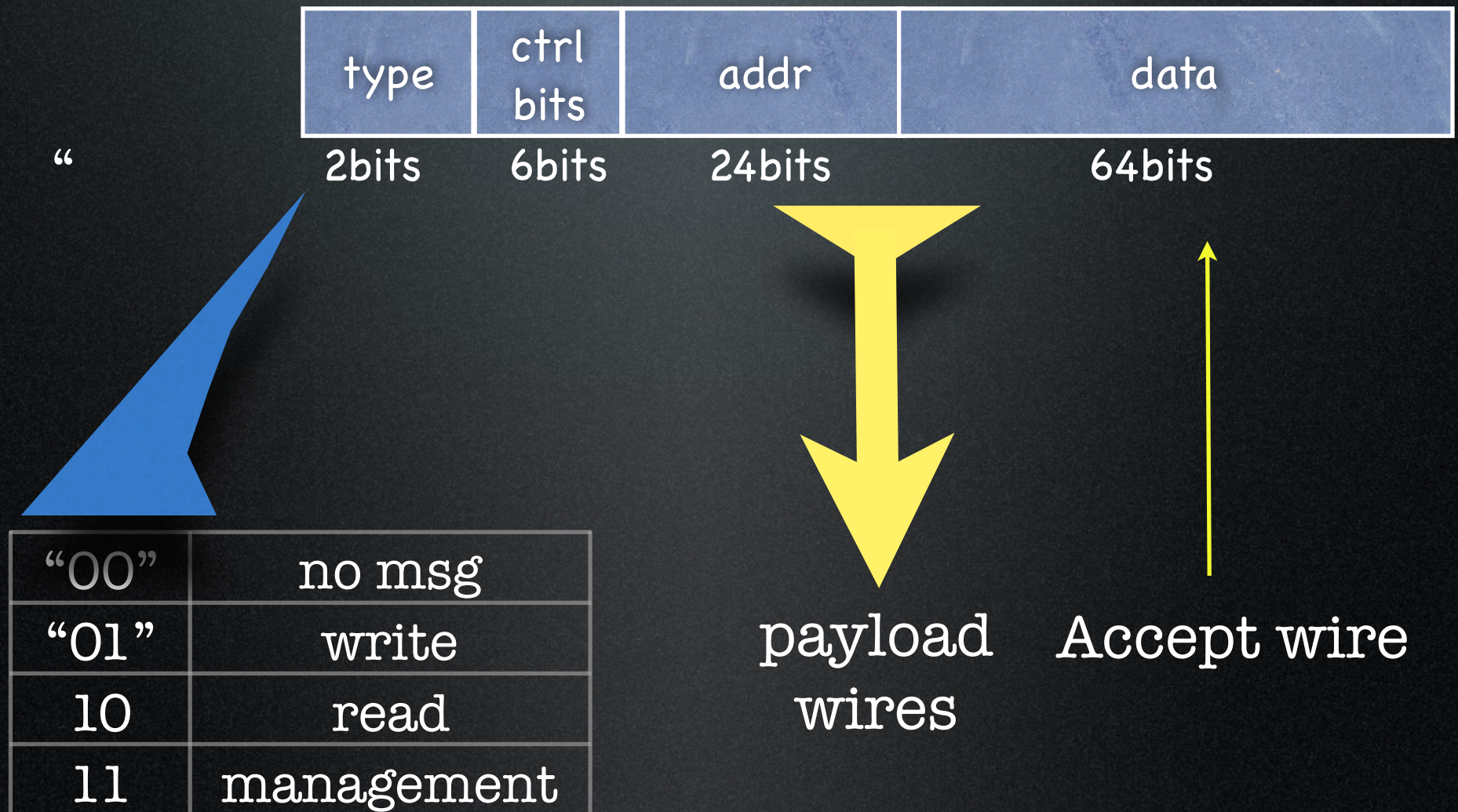
“Fabric is the
center” Second view



“No such thing” view:
no explicit
infrastructure

In the Chip, there was not module called rings.
It made the chip stitching easy.

Message bus



one ring i/f

inventory

self address / state

self address range

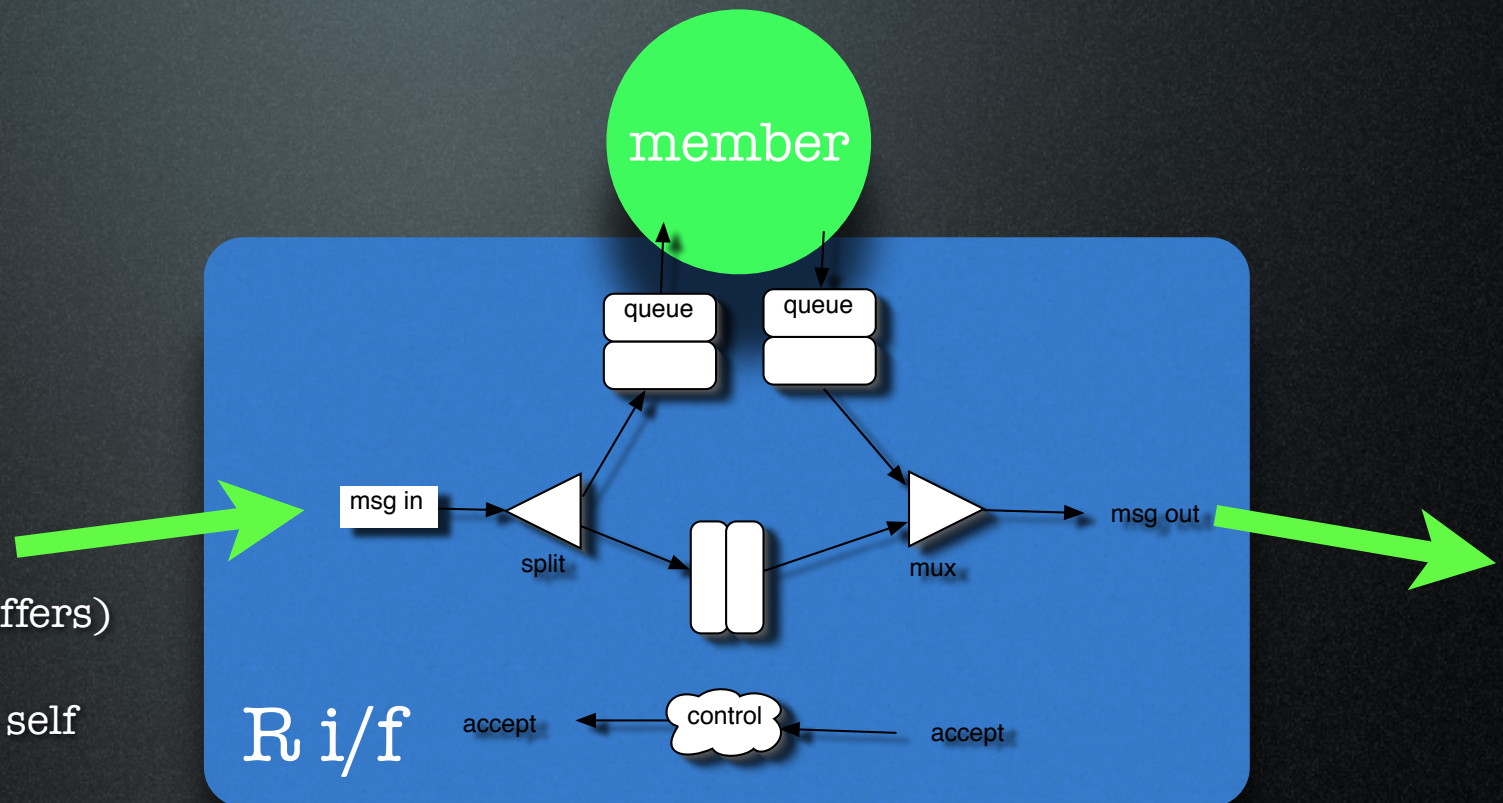
self id (who am i)

queues (6 flexible buffers)

split logic: recognize self address.

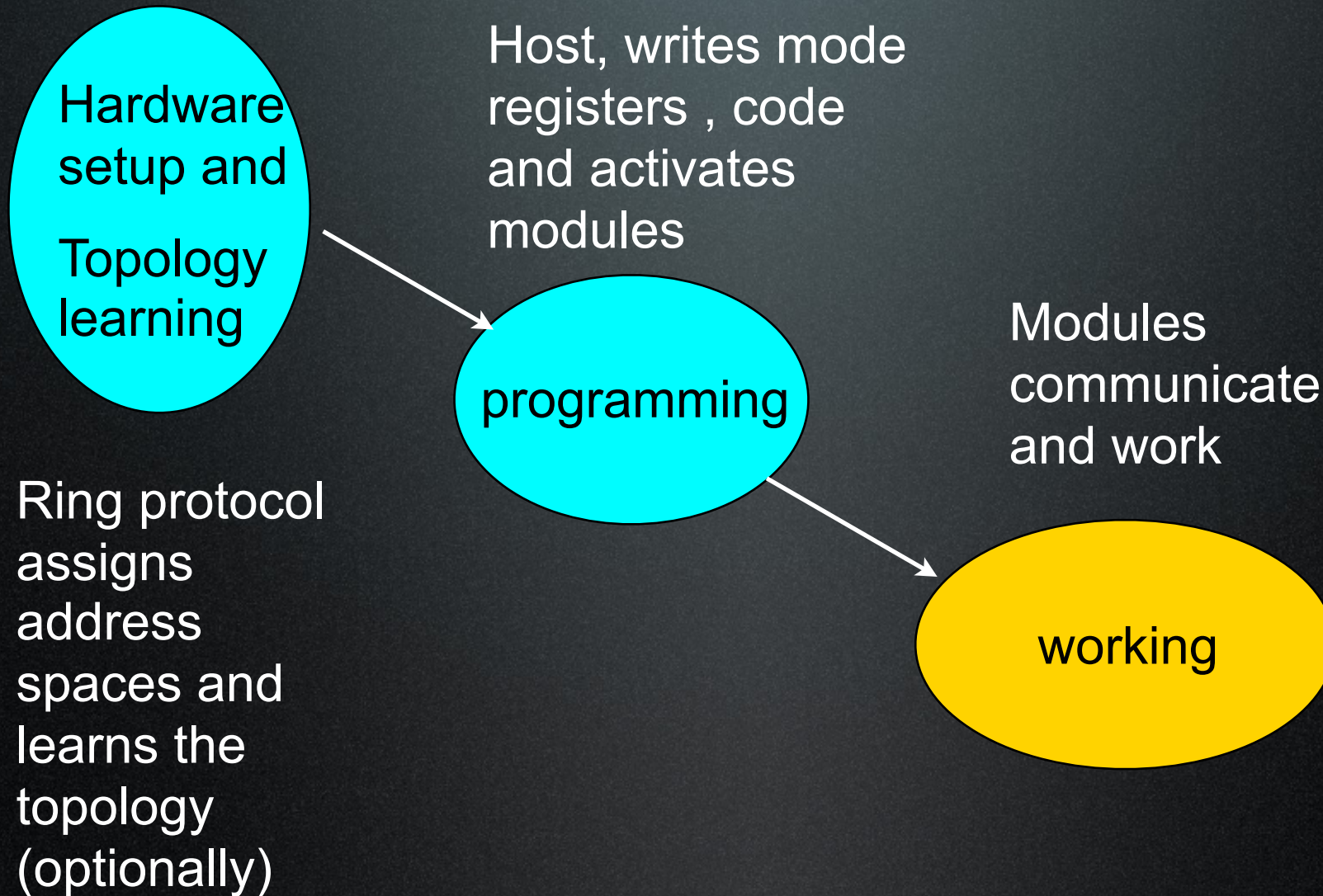
priority out logic: traffic already on the ring has priority.

control register: state, clock enable, test enable, bypass , power management and ...



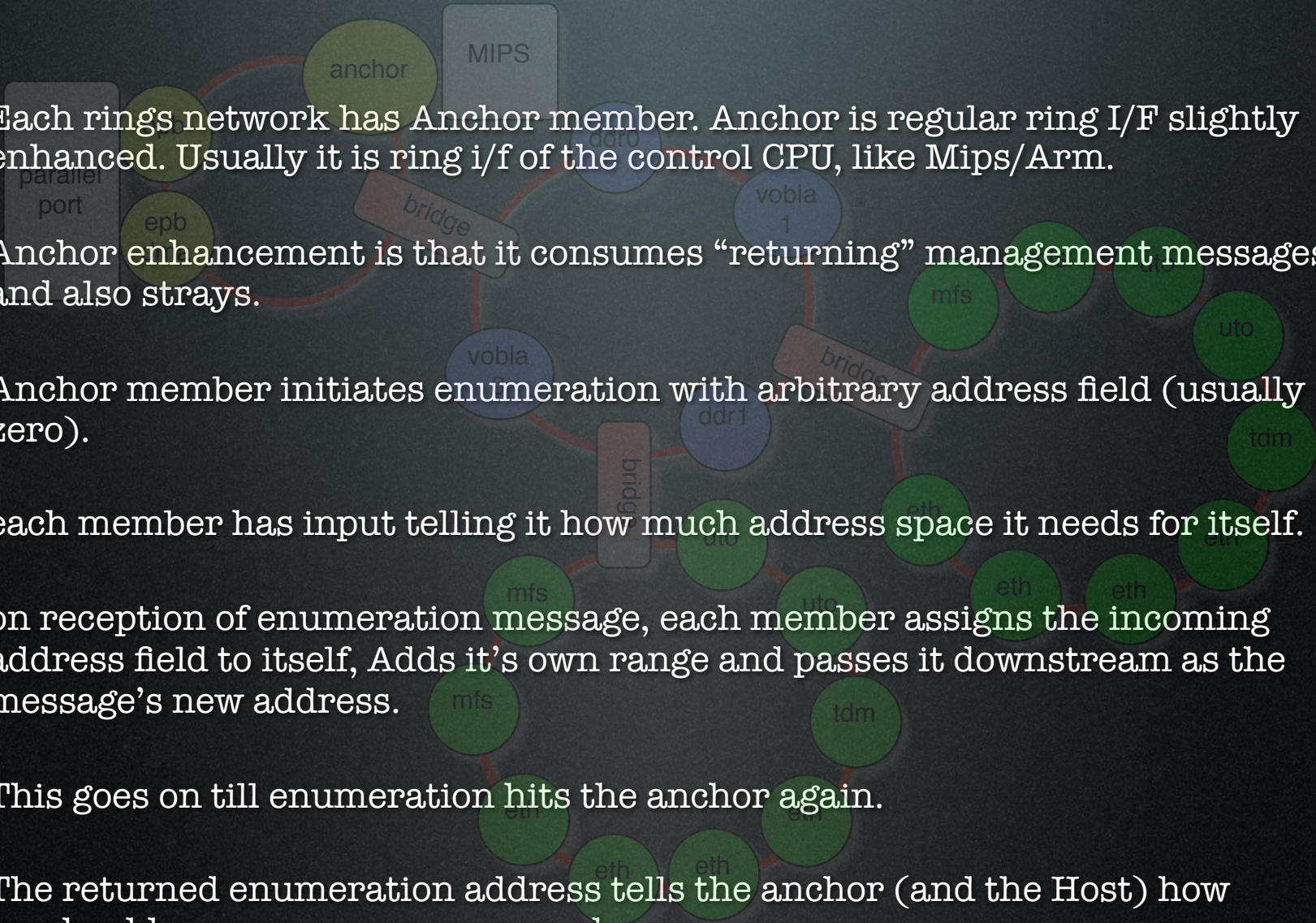
| variable | hw/sw | --- |
|--------------|-------|--------------------|
| self id | hard | specific to member |
| addr range | hard | specific to member |
| self addr | reg | assigned by enum |
| active | reg | assigned by enum |
| is anchor | reg | by enum |
| control bits | reg | by anyone |

Rings life cycle



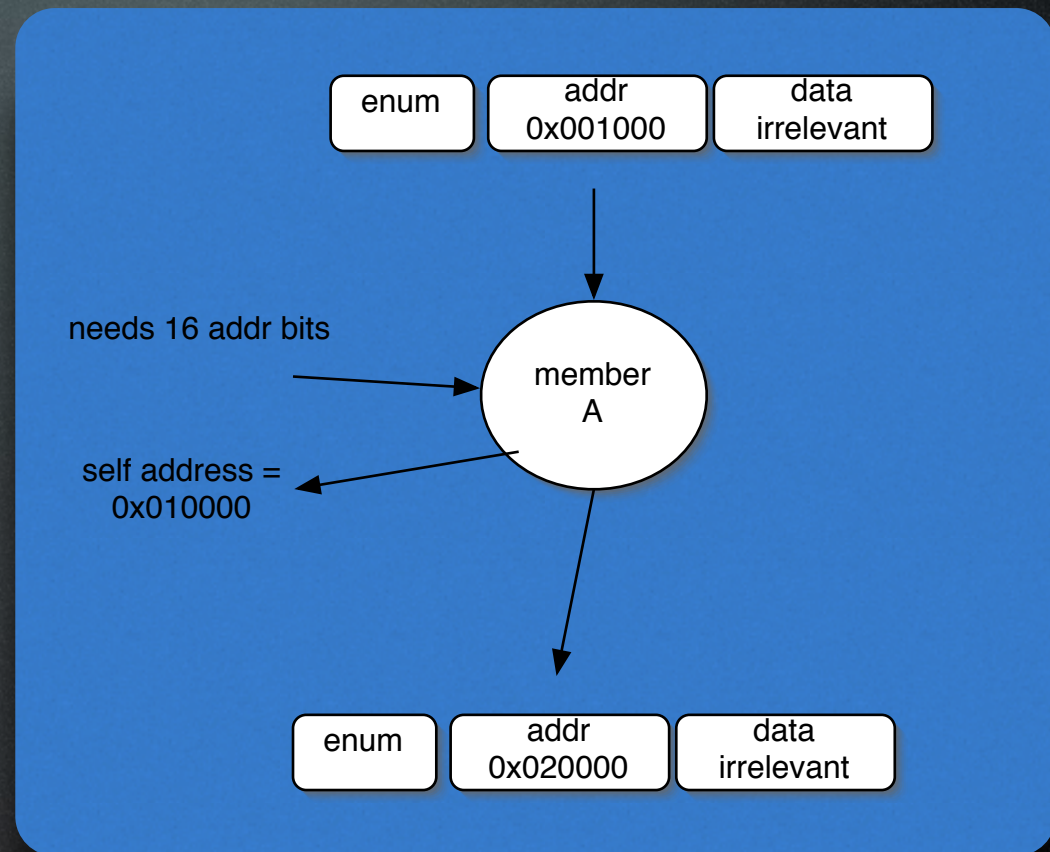
Enumeration algorithm

- Each rings network has Anchor member. Anchor is regular ring I/F slightly enhanced. Usually it is ring i/f of the control CPU, like Mips/Arm.
- Anchor enhancement is that it consumes “returning” management messages and also strays.
- Anchor member initiates enumeration with arbitrary address field (usually zero).
- each member has input telling it how much address space it needs for itself.
- on reception of enumeration message, each member assigns the incoming address field to itself, Adds it's own range and passes it downstream as the message's new address.
- This goes on till enumeration hits the anchor again.
- The returned enumeration address tells the anchor (and the Host) how much address space was consumed.



Enumeration

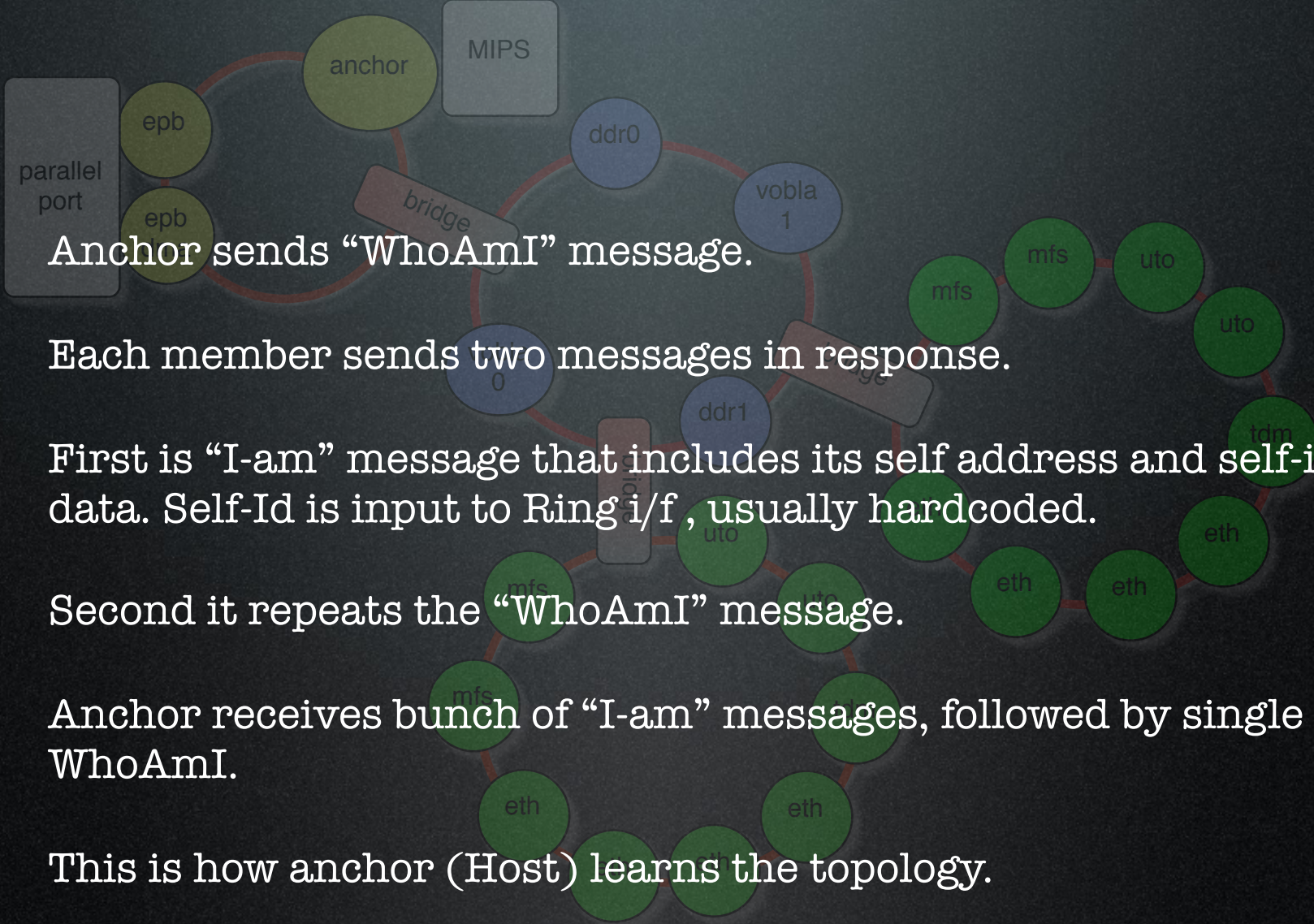
Incoming enum message



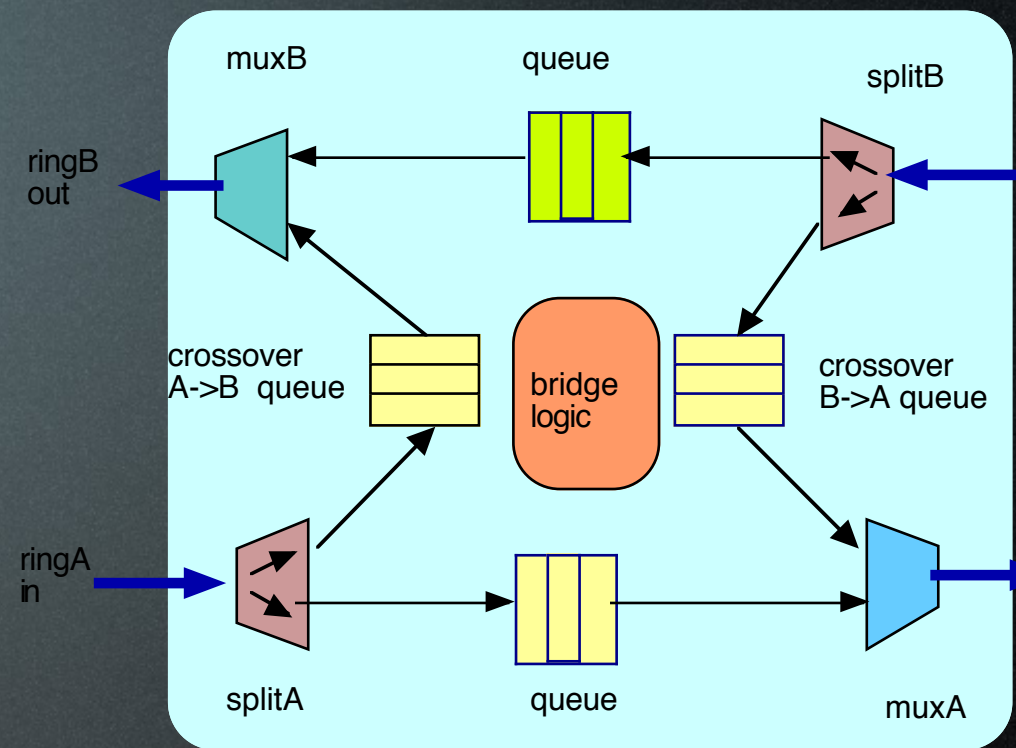
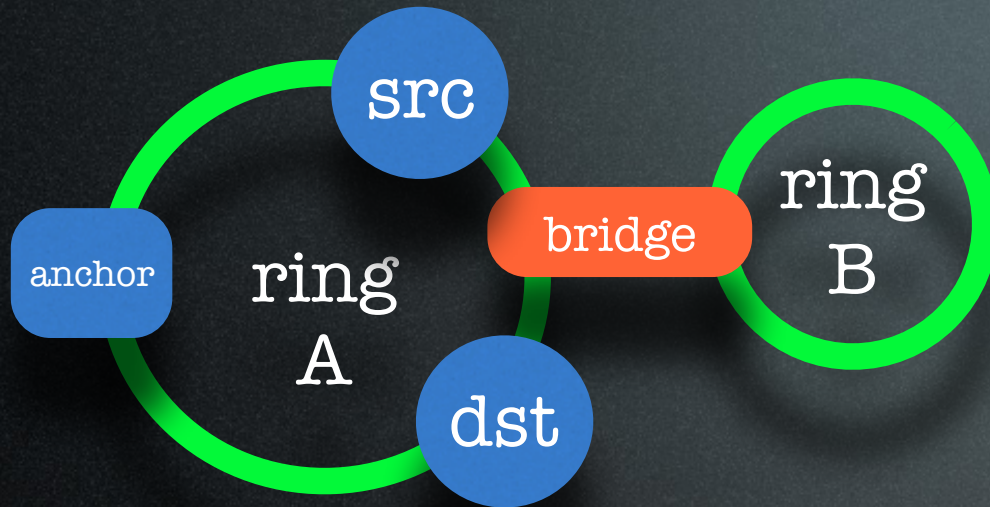
Outgoing enum message

Incoming address is adjusted to fit the self width and the outgoing address is generated to follow. This can create gaps in the address space.

WhoAmI sequence

- 
- Anchor sends “WhoAmI” message.
 - Each member sends two messages in response.
 - First is “I-am” message that includes its self address and self-id in data. Self-Id is input to Ring i/f, usually hardcoded.
 - Second it repeats the “WhoAmI” message.
 - Anchor receives bunch of “I-am” messages, followed by single WhoAmI.
 - This is how anchor (Host) learns the topology.

bridges



| | |
|------------|-----------|
| high limit | far side |
| low limit | near side |
| | activated |

- bridge shortens messages travel distance. Message from “src” to “dst” will not travel ringB.
- in enumeration, bridge records the address of near-end and far-end. It passes management messages without using shortcuts.
- near-end is the end where enumeration message first came.
- both sides of the bridge respond to WhoAmI separately.
- Bridge has built in anti-freeze algorithm (that actually works).

Rings “Protocol” stack

Message based chip forces us to define flows as part of the design and not as part of the verification

L4

Flows

receive flow
transmit flow
dma flow

L3

Transactions

tokens mngmnt
dma request / dma done
Tx or Rx activation

L2

Messages

work messages
management messages

L1

Wires

message bus
back pressure protocol,
RIFS and bridges.

Rings principles

- system built on no-instant gratification (everything takes time)
- totality: only rings allowed. there are no wires coming out of modules, except rings.
- comprehensive: rings carry control and data: interrupts, status, data, clocking, power save modes, debug and test.
- topology based on traffic patterns of the members.
- topology doesn't influence functionality, only performance.
- no masters / no slaves - all members are producers and consumers.
- keep it simple: documentation should be shorter than the code.
- ditch signals and busses - define the chip by transactions and flows. This applies to status registers, Interrupts and other requests and acknowledges.
- each member must consume incoming message in finite bounded number of clocks.
- messages have no sequence context. no memory. Each message is one clock long.
- For any pair of producer+consumer, the Fabric delivers the messages in strict order. Nothing else is guaranteed.

Write message format



Write messages either write into rams, queues or registers, or trigger actions, like interrupts.

Some addresses are defined “sensitive”, when written they trigger action.

Some addresses are actually queues. They collect write messages. Example is serial fifo.

Read message format

| | | | | | |
|-------|--------------|--------|--------|---------------|----------------|
| 10 | ctrl bits | addr | | byte count | return addr |
| 2bits | 6bits | 24bits | 32bits | 8bits | 24bits |

Read message includes return address and how many bytes are requested.

Not all members respond based on the count field.

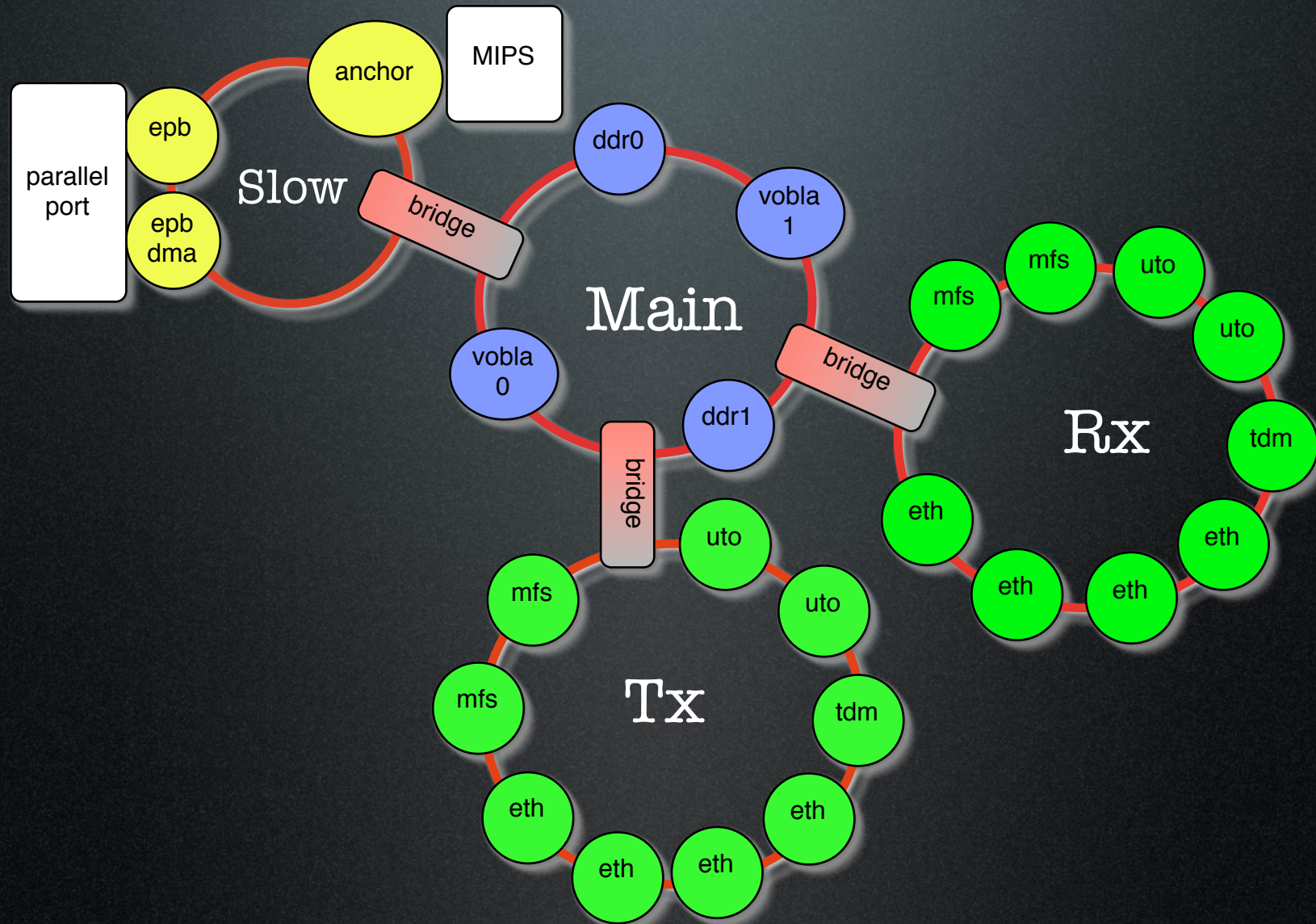
The response to this message is write message addressed to return address.

In case of count field, the response can be sequence of messages.

No instant gratification

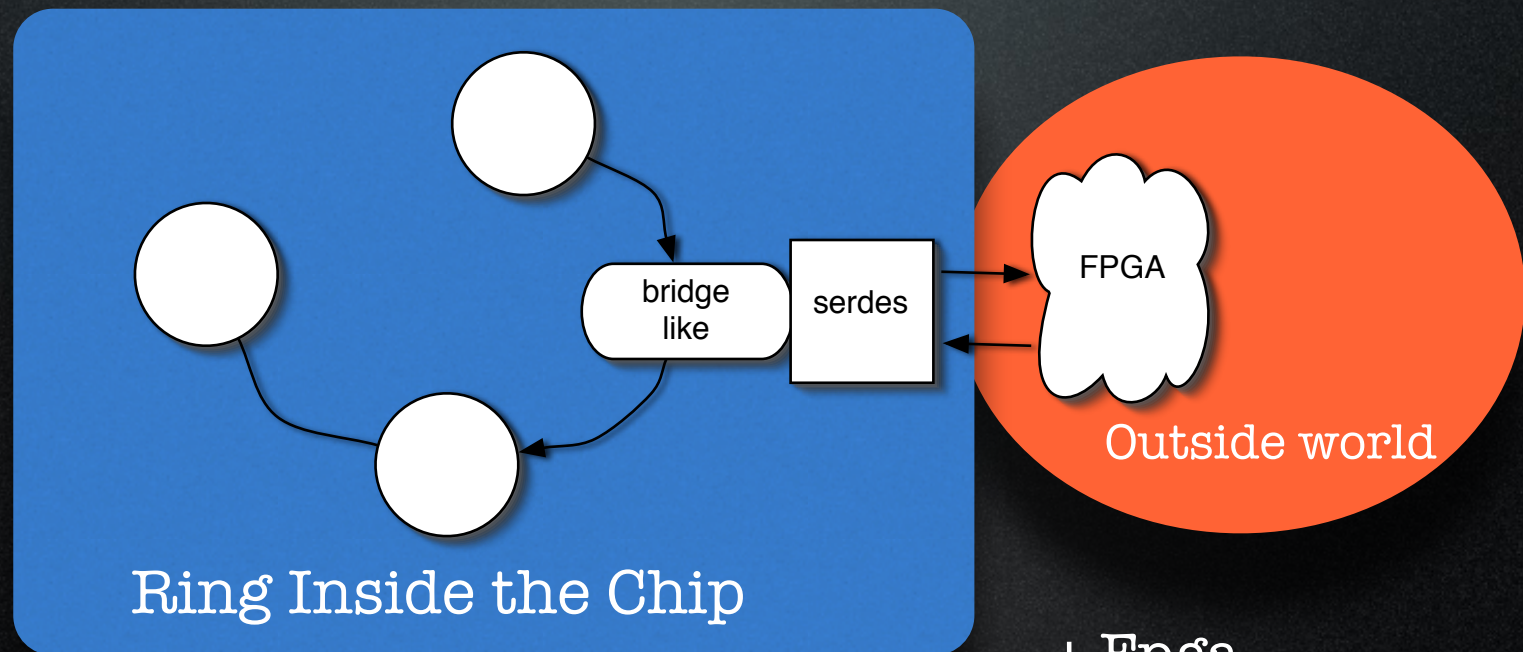
- Reads are expensive for casual use.
- There is no low-level mechanism to know that read ended.
- That's why design employs "Write ahead principle".
- example 1: Each peripheral's status register that is used by CPU is transmitted on each change to agreed location in that cpu space.
- example 2: on limited resources (like dma queue) we put tokens reservation system. This to prevent the queue overrun.

Actual chip topology



Going outside

It was proven on FPGA that external rings are easy.



- + Fpga
- + another T chip
- + compatible asic

The software is not affected much by adding external members.

Cpu compound connects to rings

Setup and writing program is done through the ring.

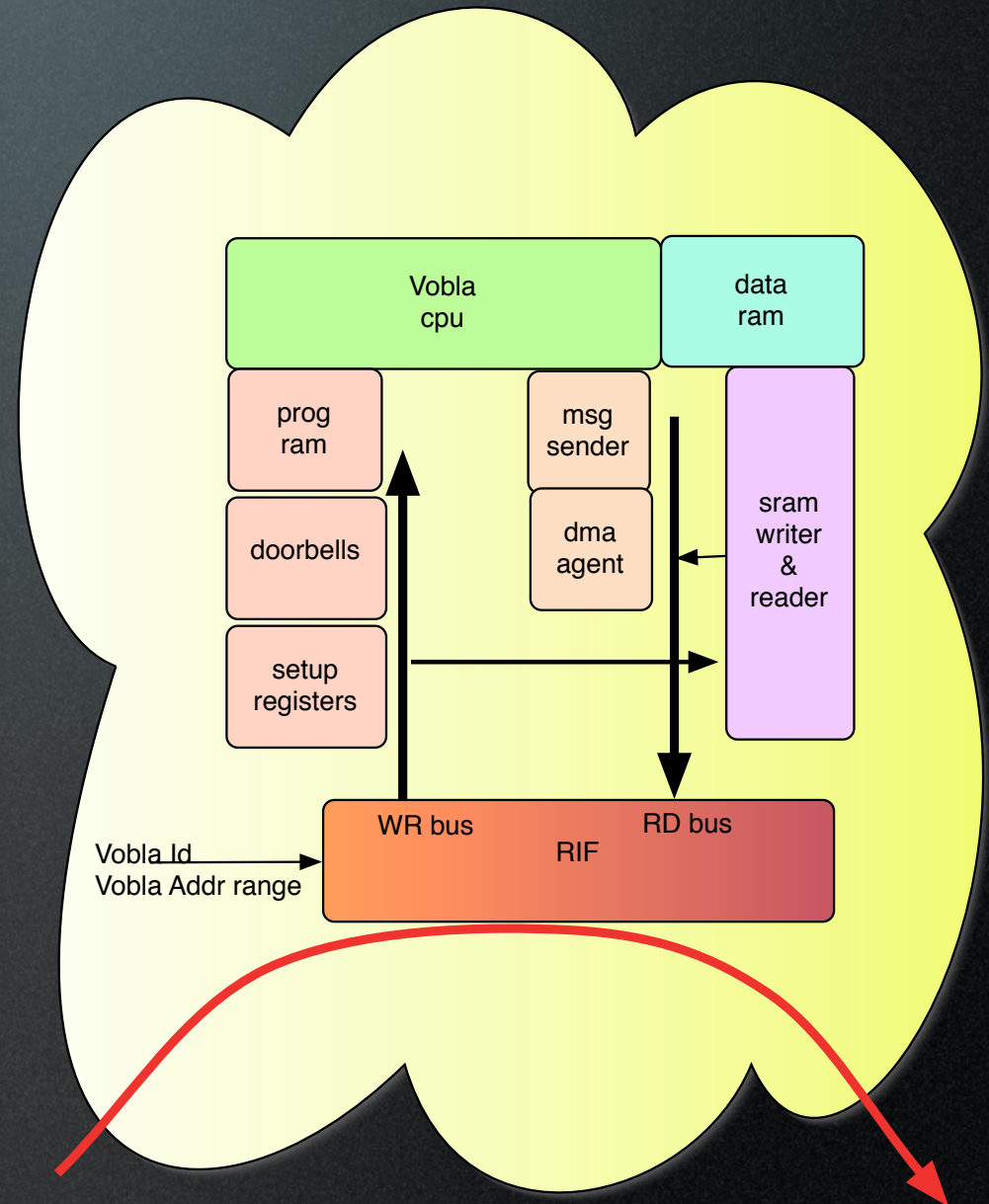
Doorbells are interrupts waking up various tasks.

MsgSender and Dma agent are helping CPU to manage resources.

Data Ram can be connected to it's own RIF. In this version it was not needed.

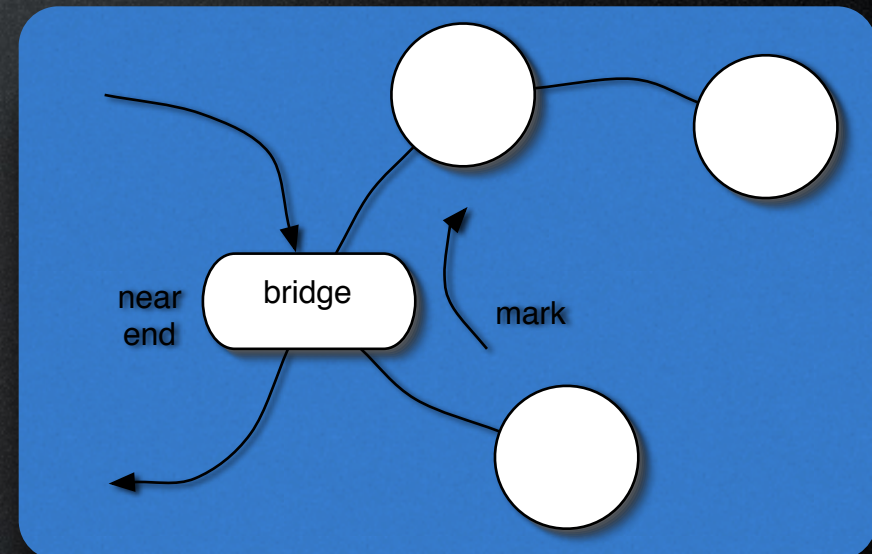
Sram writer can get unaligned access of any width (1 to 8 bytes).

Sram reader can serve up to 8 requests of various byte counts and source/destination addresses. Also unaligned.



Strays: messages to nowhere

- Stray is message that has address of no member.
- The danger is that stray will circulate in the ring till eternity.
- Catching of strays is done by anchor and far-end of bridges.
- Any message passing through far-end is marked.
- if Marked message arrives at far end,
it is declared stray.
- It's type is changed to management
and it ends up at anchor.



Silicon proven Rings

- It was shown that rings can serve as base for real sale-able silicon device.
- No compromises on rings totality were needed.
- Rings provide total visibility of each module. The main cpu can impersonate any module. Great for debug and bring-up.
- Short hops and adaptable topology facilitate high speed design.
- Topology is transparent to software.
- Easy load balance. Based only on address configuration.
- Floorplan affects Rings topology when needed.
- No special problems were encountered when existing or new IP was adopted to rings.
- Synthesis constraints were trivial.

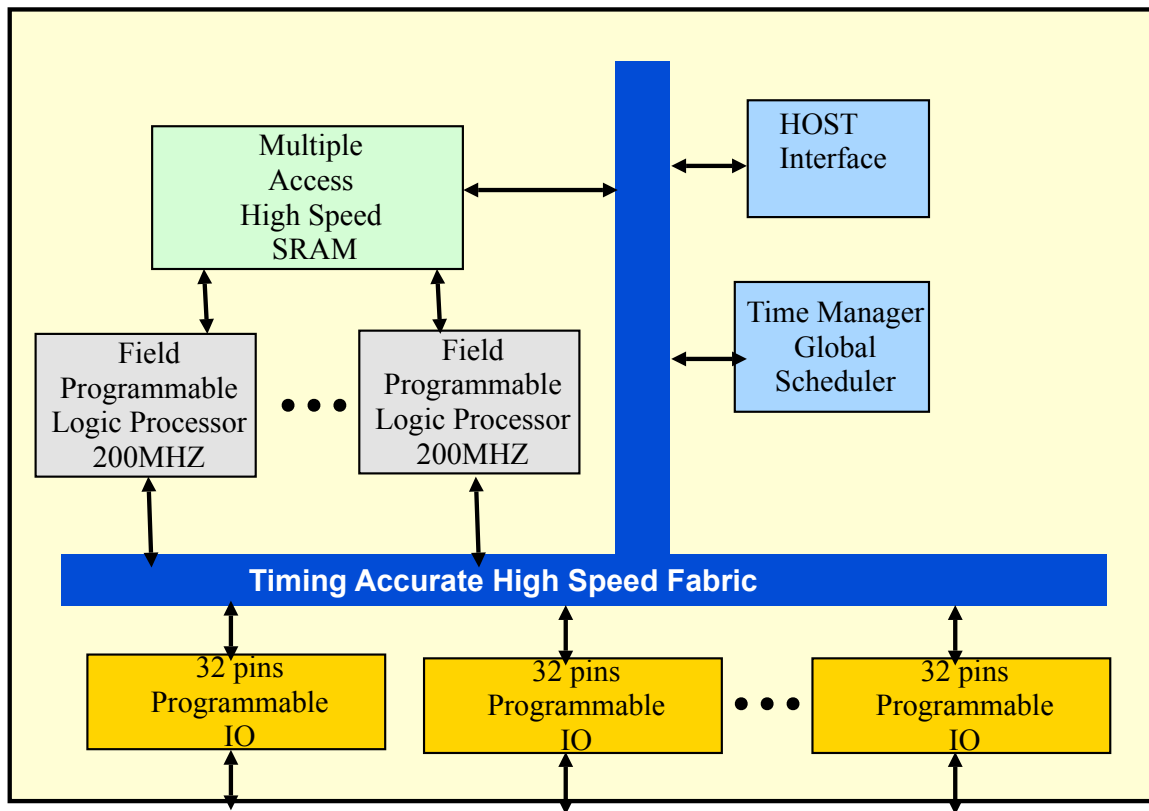
Latency

- Latency is the number of clocks it takes to move data from place to place .
- Latency was/is major PR(public relations) concern.
- It didnt cause any real problems.
- Critical flows were pipelined.

Potential

- Across the chip clock alignment is not mandatory (and in fact pretty stupid). Each member can align itself only. GALS - global async, local sync.
- “Clockless rings” is tempting option.
- Rings can serve as scan chains.
- Rings can serve as production testing infrastructure.
- Rings can carry “reset” and other globals.
- Going external provides software-transparent gluing of many chips.

Original rings driver



This slide from bigger presentation was for non-initiated audience. Rings had to be disguised. In actual implementation numerous rings with renaming were planned. Renaming is like NAT in ethernet. This system may be viewed as very fast Verilog simulator, that obeys real timing constraints. It is useful replacement of FPGA for relatively slow and complex peripherals.

The idea to build rings fabric was developed originally for **Low Cost Field Programmable Logic** device. All resources are running on cluster of rings.