work in progress

working title:
ROUBUS

Chip wide net

for the rest of us

Never again transaction goes into the NOC, just to be never seen again.

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

Motivation

- AXI4 is a bloated behemoth.
- Very hard to design around it and requires complex NOC hardware.
- The industry standard in this case is very problematic and unresolved issues exist. In many scenarios, I don't have an idea how slave/master should react.
- Commercial NOCs have peculiar trait: Transaction goes in and does not arrive anywhere. It is very hard to debug in complex generated rtl of these NOCs.
- Proposed interconnect hardware hopefully is free of these problems. Everything is easily traceable.
- My goal is to design and implement high-perf network-on-a-chip for struggling designer and other humans.
-

Neither masters, nor slaves

* every node on the network is talker and listener. Some modules are more talkers (ethernet Rx) some modules mostly talkers (Tx) and some are both (Rams and Processors)
* The "roubus" carries "messages". Each message is one clock long. There is no history.
* The "roubus" has 3 kinds of messages: Read, Write and Management.
* Write msg carries (mainly): destination address, valid bytes count and payload.
* Read message carries address, return address and number of requested bytes. When this message hits the destination, it triggers sequence of "write" messages. They exploit the return address to be routed back.
* Management messages carry setup info and panic alerts.
* Optionally some links may be asynchronous.

Difference hi-lites

* neither master, nor slave deals with errors. They report errors to network monitor (designated admin) through management messages.
* Design complexity is roughly evenly split between talkers and listeners.
* Each write message has full address and byte count (valid bytes in data). It is routed to the correct destination. No 4k boundary is needed.
* Read messages cause talker to create write messages. Each with the correct destination address (which is the return address). If end of current address space is reached, the talker issues "spillover" read message - it is basically copy of original read message, alas with address and byte count and return address adjusted.
* When listener ant receives read or write message and the address is not in range, it simple passes the message forward.
* Customers can be connected in ring, without any additional modules (except Admin).
* ROUBUS encourages "comprehensive & total" approach. Modules are connected to the chip "EXCLUSIVELY" by ROUBUS. They talk only through ROUBUS. No interrupts, no setup busses, no triggers between modules. No sideband signals. Even ATPG and other testing can be operated through ROUBUS.
* One side effect of this "democracy" is that since every node talks and listens, no DMA in system is needed.
* The network is built by snapping together modules from the "zoo" . Each ant module has embedded network interface instance.

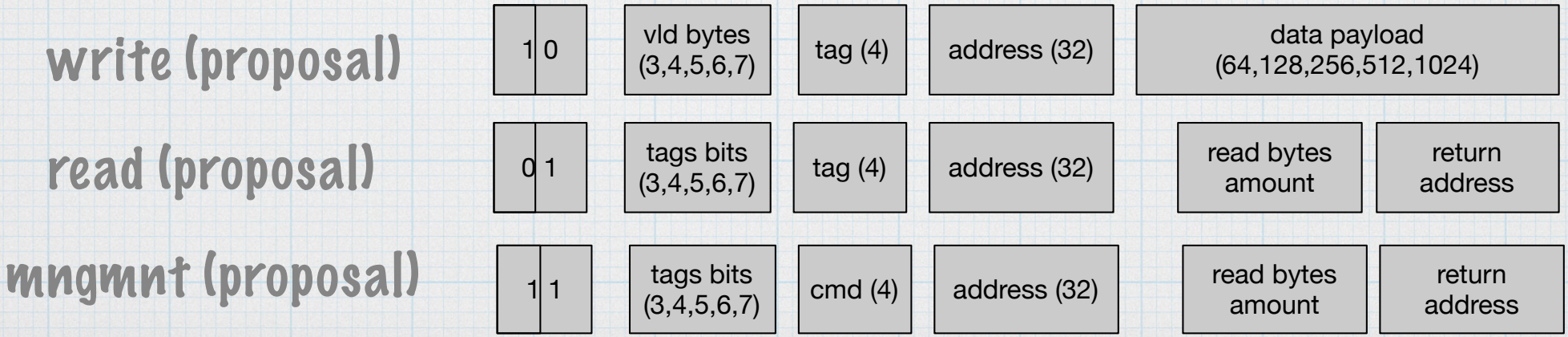
Msg Formats

all messages travel on same "roubus" connection busses. The width of this bus is mainly derived from payload (data) width.
two first bits determine the kind of message:

00 -idle
01 = read
10 = write
11 = mngmnt

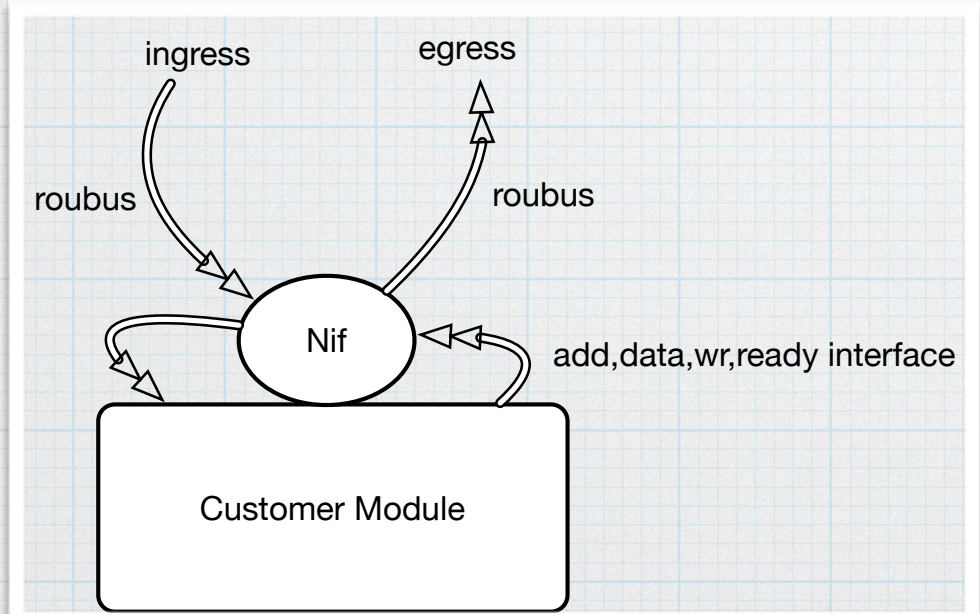
write message includes most fields:
address, data, amount of valid bytes and tag.
Tag may says if it is first/last/thread/incr message.

In read message data bits are used to convey the return address and amount of bytes to return. As well some bits (in tag) that define the max data rate it may send.



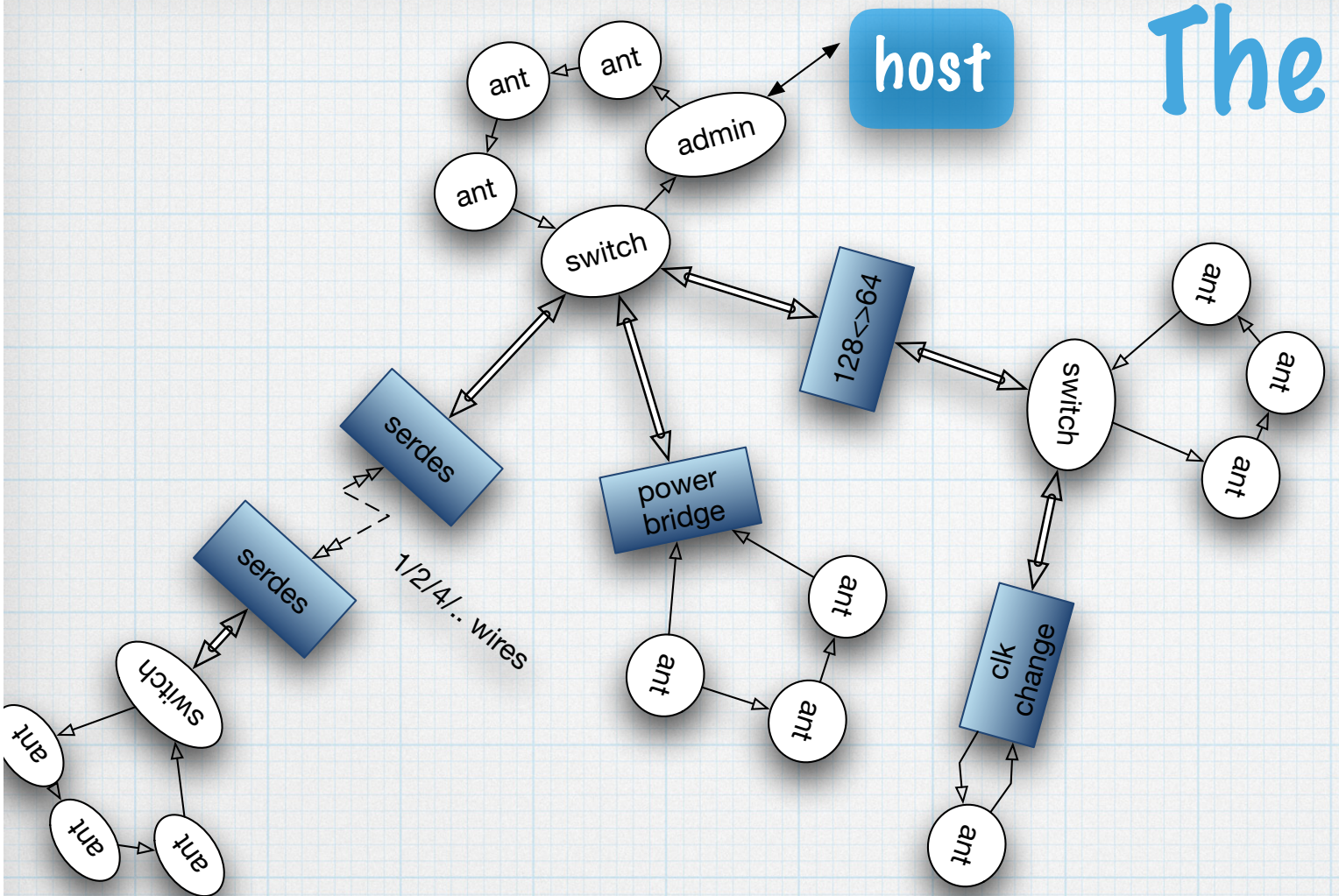
The worker, the ant

Each customer module has incoming message bus and outgoing messages bus.
If incoming bus carries message not addressed to this module, it is just sent out on outgoing bus.
Otherwise the message is "consumed". Which usually means looking at lower address bits to know what to do with it. Each worker module has NIF (noc interface module) embedded into it. It can be 64-128-256 data width (at least for now).



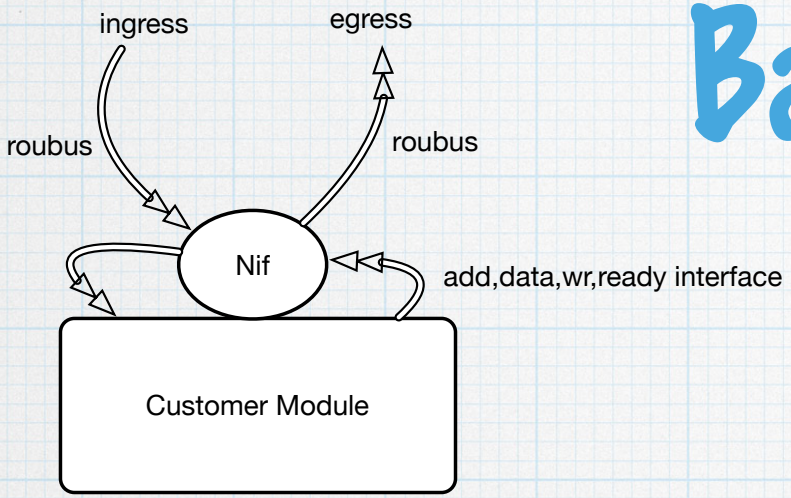
Each Nif also has configuration register. It is written by special management message (either broadcast or address specific) . It has bits like: stop the clock, stop the power, go into test mode and such.

The network

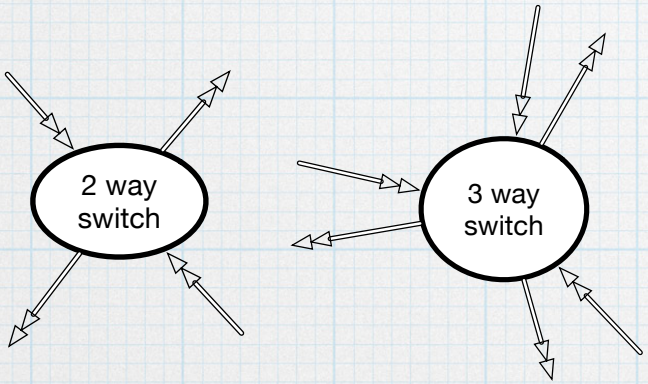


All nodes (processing modules, ants) are connected same to the network.
There are neither masters nor slaves.
Some nodes are mostly talkers (like RX-fifo of some external interface) and some mostly listneners (TX-fifo).
Both examples are also listeners for setup info, and talkers for doorbells and buffer requests.
RAM connected to network is equally talker and listener. Though It talks only when addressed to.
Switches, DataWidth crossers, Clock and Power domain crossers,Off-Chip crossers and Serializer/Deserialzer (for moving on chip long distance in small footprint). constitute the
"LEGO" like zoo of support.
You pick them up from a pile and they snap together.
The network is mostly self-organising in terms of addresses and routing, including inter-chip comms.

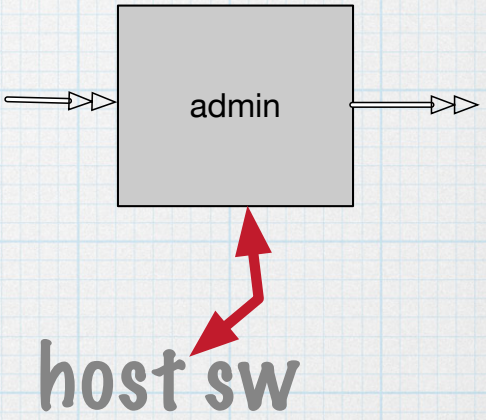
Basic Zoo

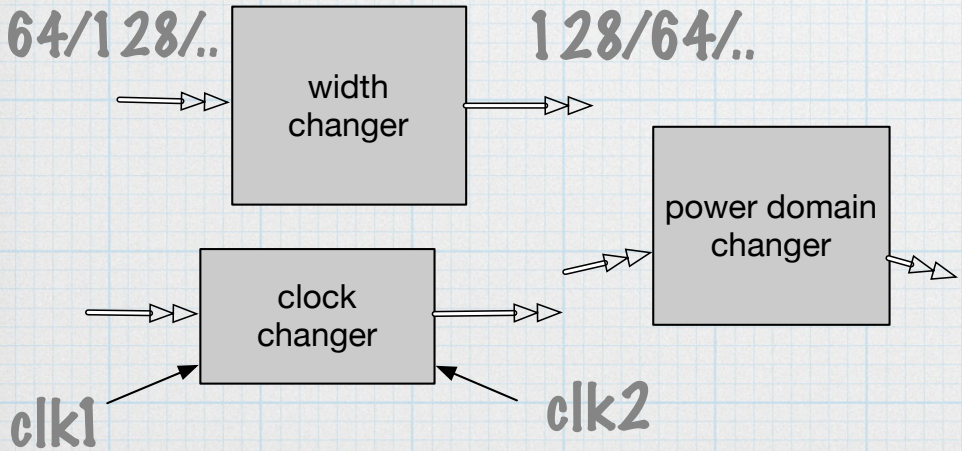


basic Nif : it has ingress message port, egress message port and either simple (ram like) interface to the customer or filtered message interface. Not shown are back pressure signals on all four ports.

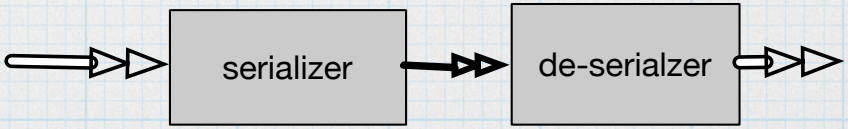


switches build the network. However for small networks, no switch is essential.

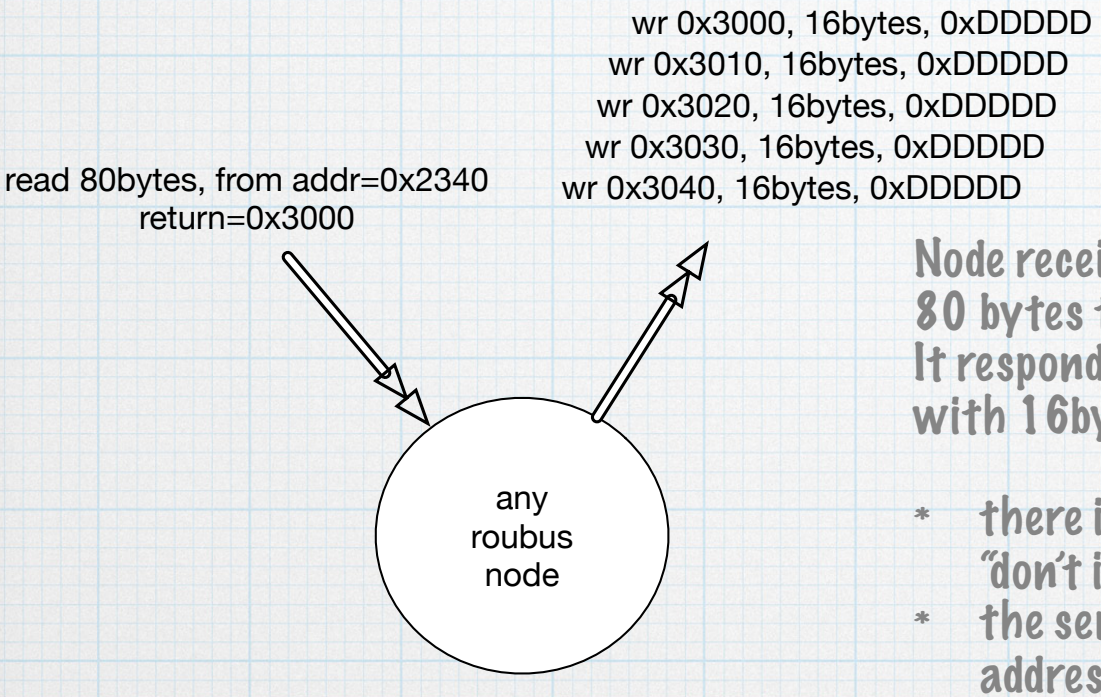




serialisers save number of wires. They come in variety of parallel wires from 1 to 16. They optionally have fast serial clock input. They may be used for inter or intra chip connectivity.



reading



Node receives a read message asking it to send 80 bytes to addr=0x3000.
It responds by sending 5 write messages (each with 16bytes) to incrementing addresses.

* there is an option in read message to tell: "don't increment the address".
* the sending node is not necessarily at return address. This is inherent "dma" like functionality.
* there are several tags that go with messages, not show here.
* It is a system error, if the node doesn't know how to respond to many bytes request. It will send "error" management messages.
* Management response message, originating at any node, automatically routed to admin.

Stages in network morning.

After reset, the host asks the "admin" node to initiate a "discovery" protocol. This is a management message circulating around the network and collecting data about nodes.
It is possible that some nodes reside outside this silicon.
At the second stage of the protocol, nodes assign self addresses .
Third stage, switches build routing tables.
Last stage of initiation, nodes report to "admin" about their status and identity.

Next stage, host through admin, after learning the network, can configure & program all nodes.
If the network covers only one chip, the "learning" becomes actually self test, as good results are known in advance.
In multi chips configuration, only one "admin" is really an admin. An admin becomes "silent" , if instead of being activated by host, it receives a management message.

discovery process, exact management messages and protocol : work in progress.

network wakeup

enumerate switches

here each switch assigned an unique id.

Each ant requests and receives unique address range.
The size of the range is an internal "ant" parameter.

run assign address ranges

Each switch configures it's routing table. During work load, software can alter these tables.

fill routing tables in switches

work

The idea is that most of the setup is done without software interference from admin node.
However software should be able to program anything in the network. Even without running any protocol in advance.
Notice that address assignment is re-done on every wakeup. For simple networks it means more of a self test. Software from admin can query all addresses.