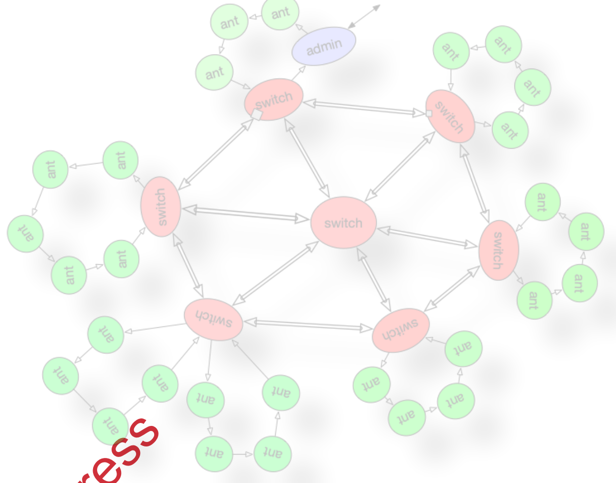
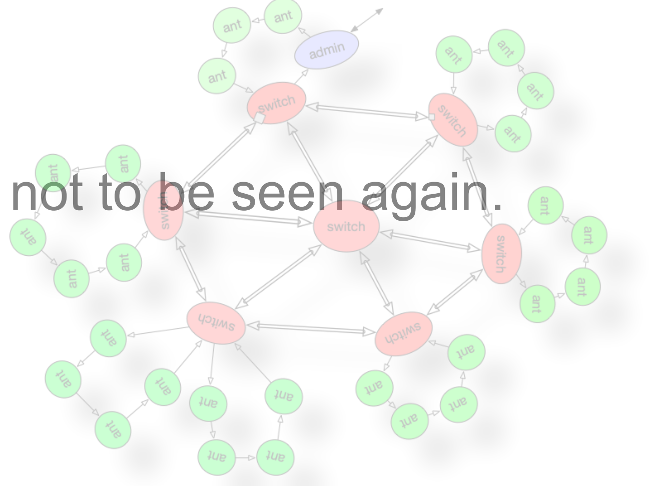


Chip wide net

for the rest of us



Never again transaction goes into the NOC, just not to be seen again.

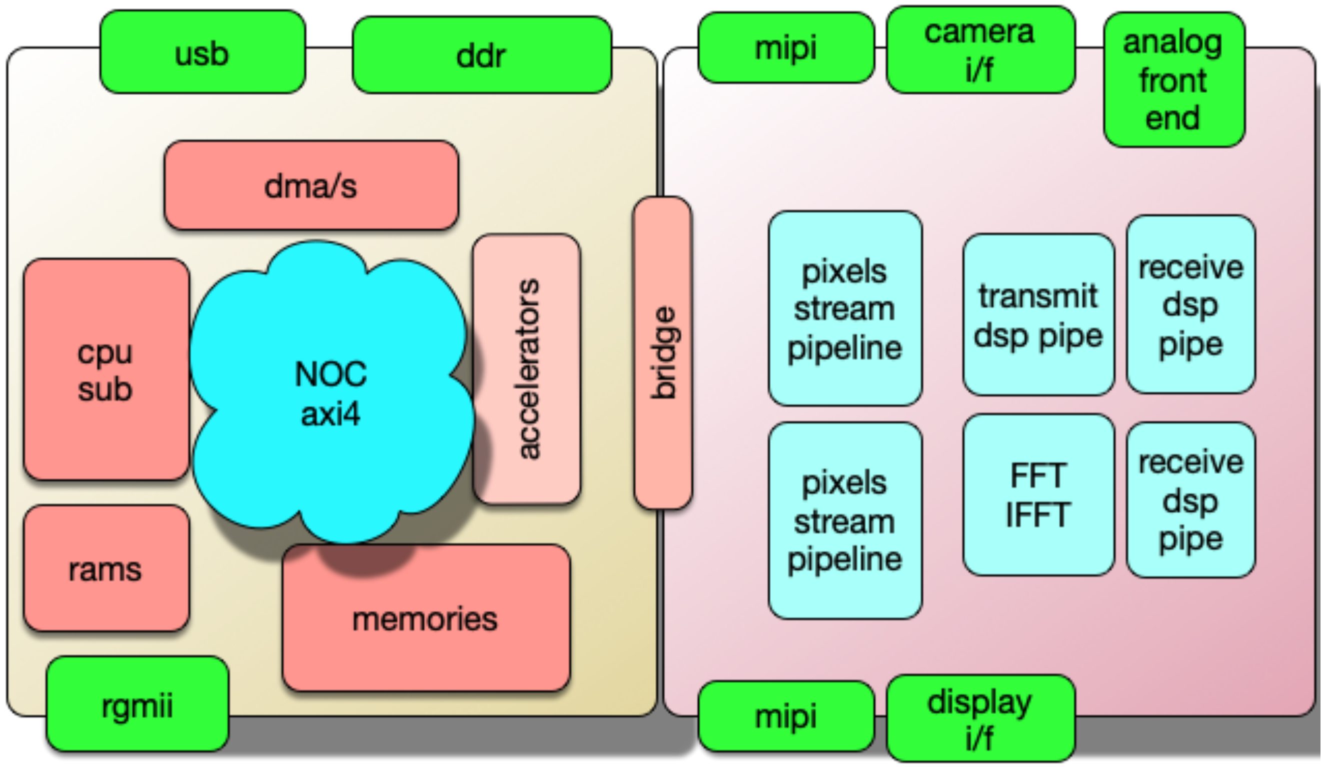
Ilia greenblat@mac.co
+972-54-4927322

ChipEx2019 May 13, 2019

start with - Why?

typical chip (my perspective)

left side SOC



right side Specific

mostly of the shelf modules , assembled by NOC

Mipi, Camera, Display, Modem in-house developed, sewn by “hand”.

What is wrong with this? What can be made simpler?

NOC ip and axi4

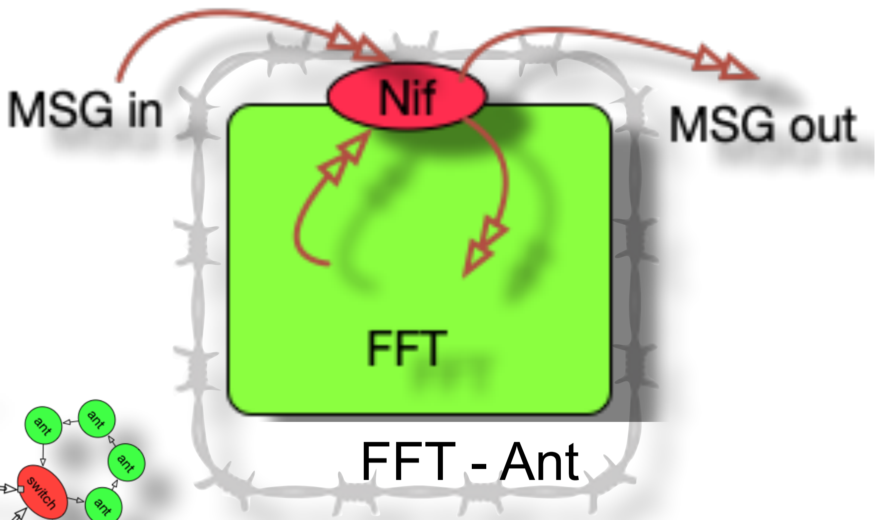
* commercial NOCs are black boxes: what goes in - not necessarily comes out.
* NOCs solve partial problem: additional mechanisms are needed: especially in the "Specific" part. no interrupts, no power, no scan, no bist.
* piling more stuff: variations like AXI4Stream , AXI4Lite.
* AXI4 does not specify how to turn individual modules into coherent connected system.
* Is overkill for many systems. Scary for most designers.
* commercial NOCs disappearing = being swallowed by a bigger fish.
* adHoc connectivity solutions present formidable verification challenges.

If these are not your problems - You are either working in a great company, or You are a manager, or not in the chip design business.

The approach…

First define the network, protocols, layers and ingredients.
Network will shuffle messages.
and finally add interface to modules.

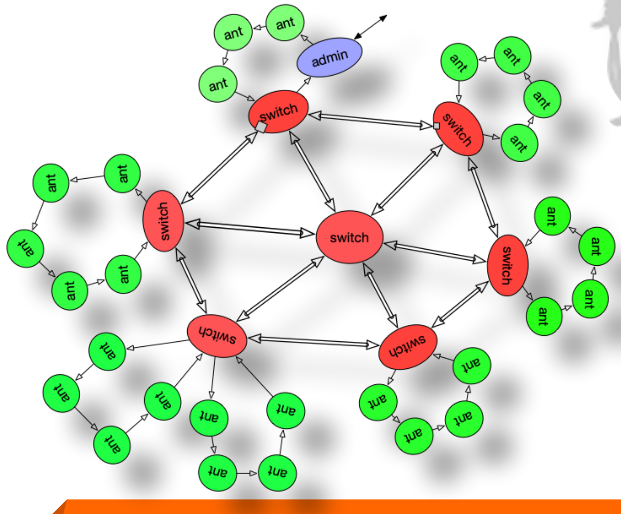
Network is assembled in "LEGO" -like fashion. From standard building blocks.



FFT - Ant

Special module "NIF" is embedded into every "ANT" - working functional module.

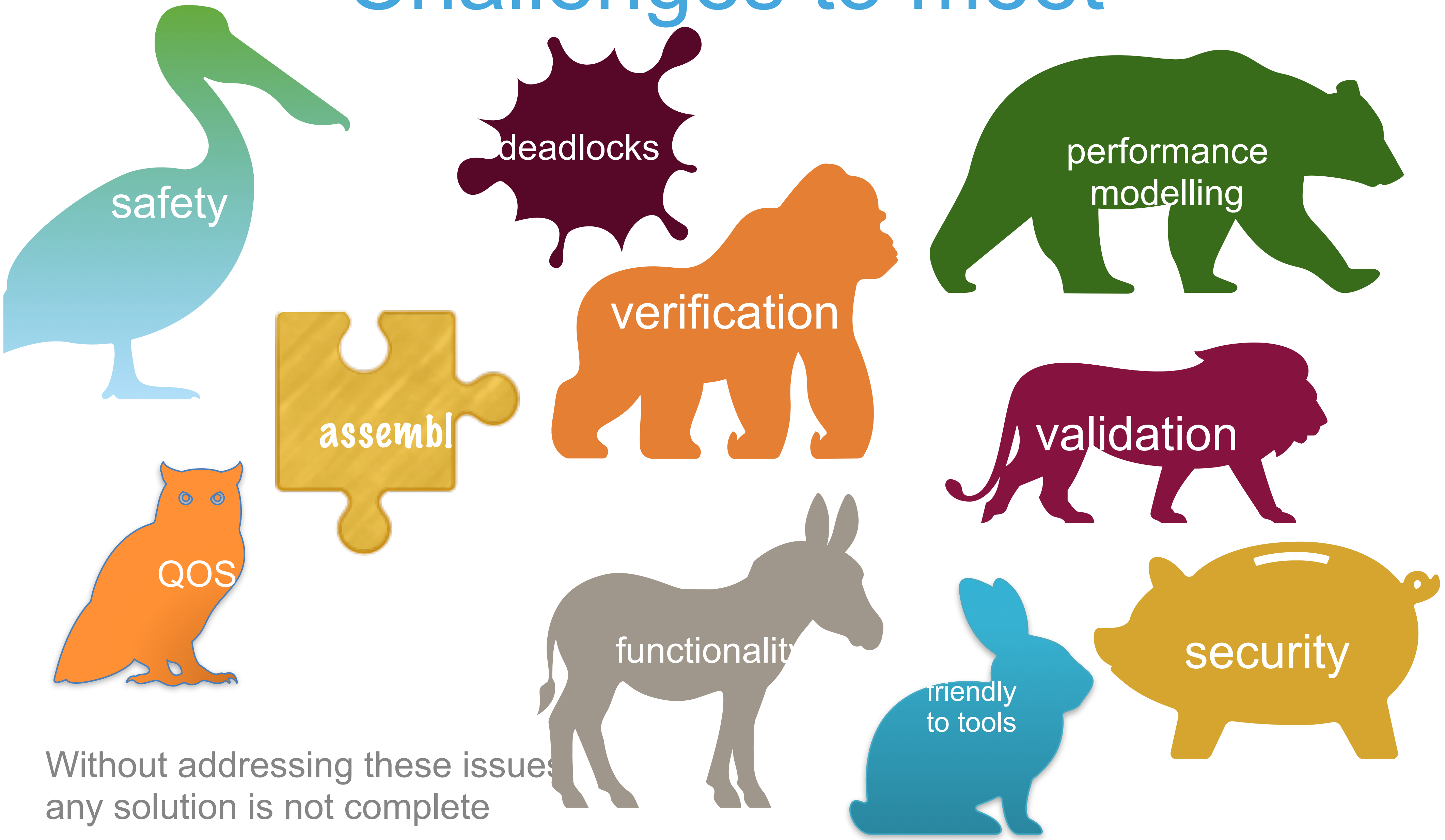
One node on the network becomes the manager and monitor.



Make the network carry ALL the traffic in the design, including video frames, dsp pipes, interrupts, errors, validation, clocking, power, scan, bist. Ideally no wire between modules besides the net

common NOC approach is to define AXI4/APB/AHB/ AXI4lite/AXI4Stream first and then build NOC around it.

Challenges to meet

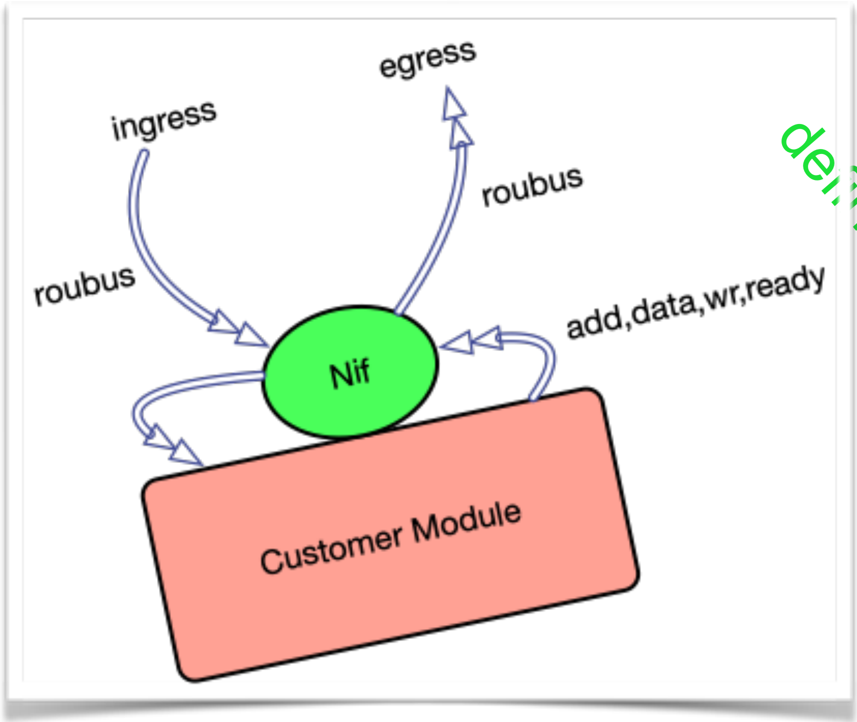


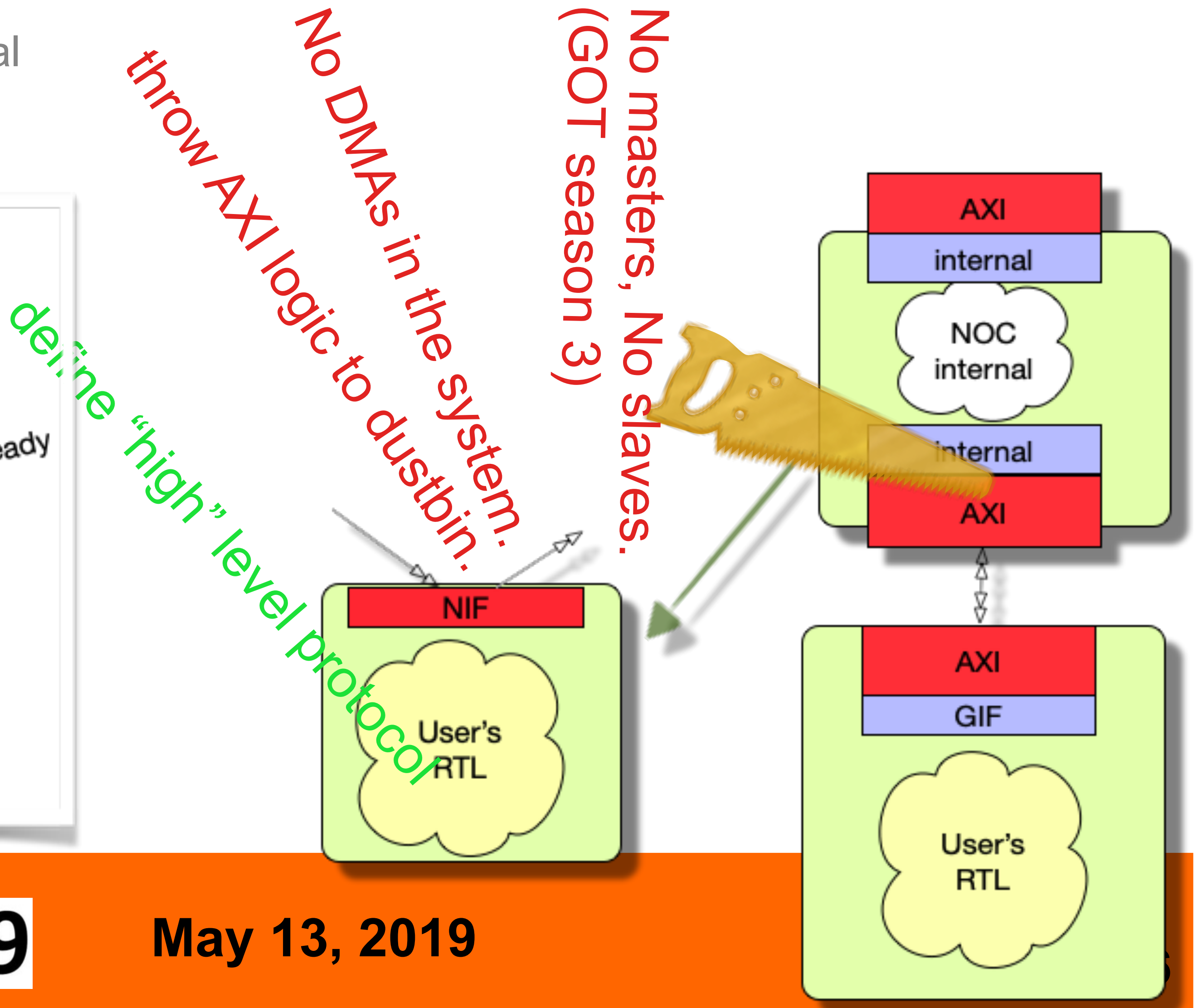
Without addressing these issues
any solution is not complete

The ant

Each customer (functional) module has incoming message bus and outgoing message bus. If incoming bus carries message not addressed to this module, it is just is sent out on the outgoing bus. Otherwise the message is "consumed".

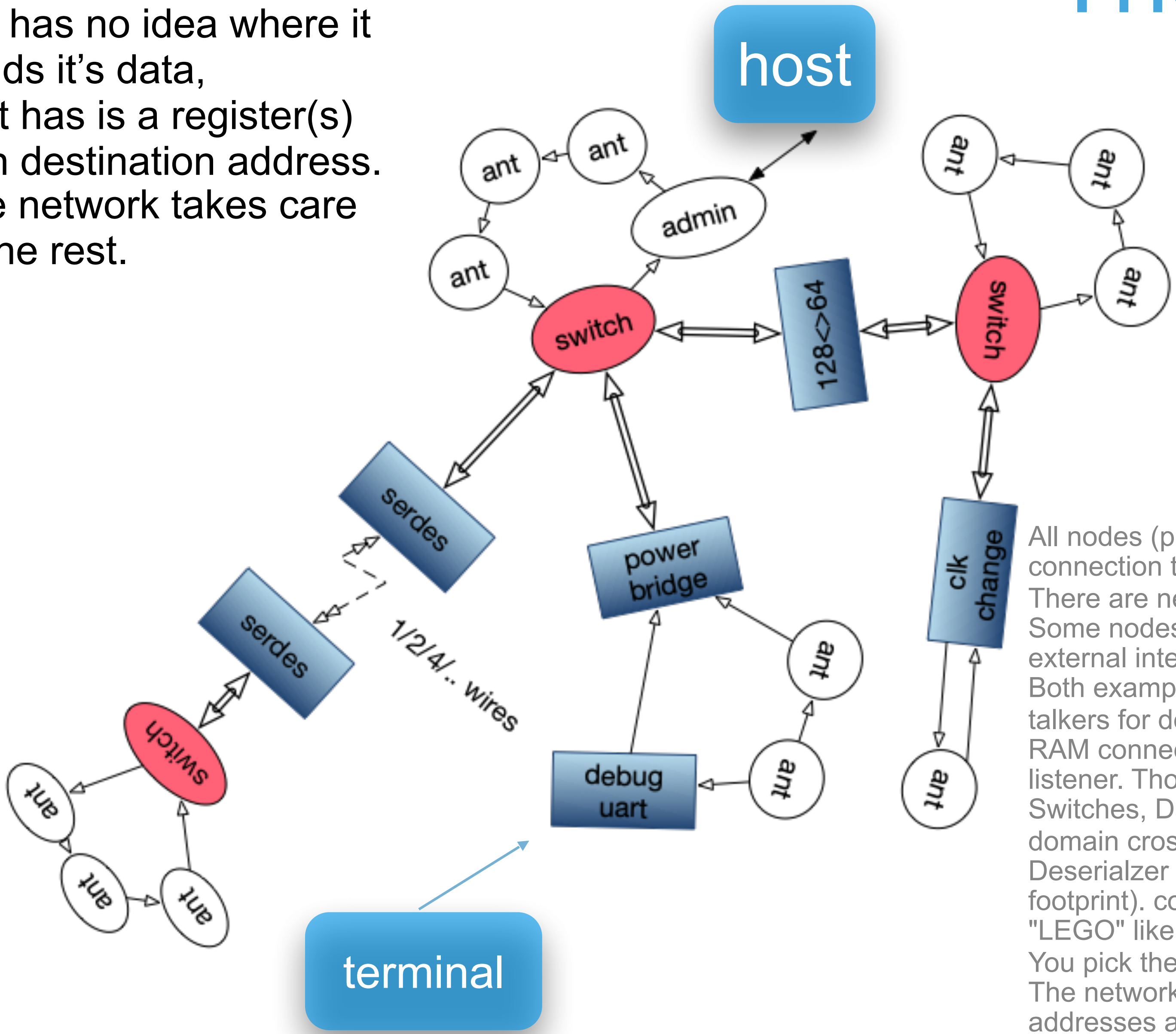
Inclusion of standard rtl into functional module enables central control and monitoring. Simple interface options.





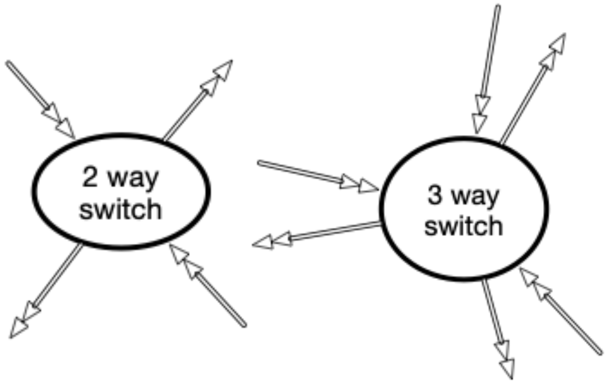
The network

Ant has no idea where it sends it’s data, all it has is a register(s) with destination address. The network takes care of the rest.

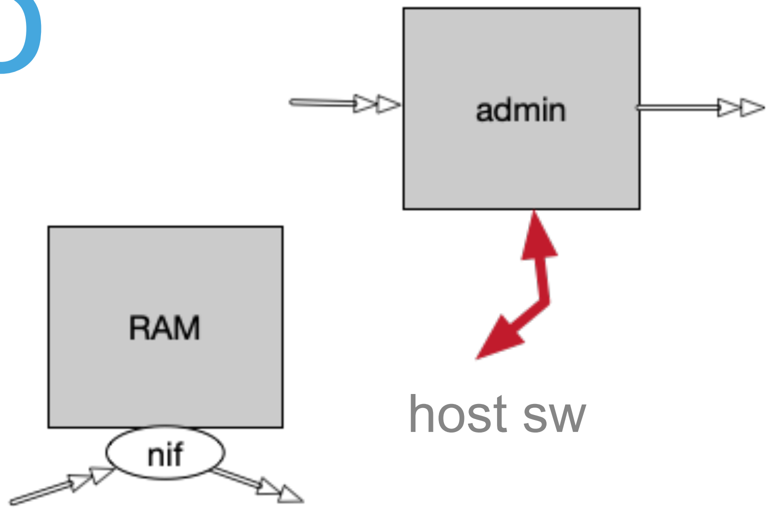


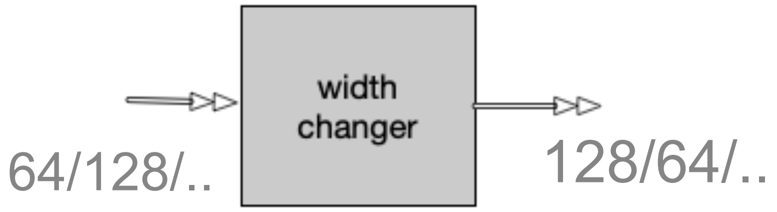
All nodes (processing modules ants) share same connection to the network.
There are neither masters nor slaves.
Some nodes are mostly talkers (like RX-fifo of some external interface) and some mostly listeners (TX-fifo).
Both examples are also listeners for setup info, and talkers for doorbells and buffer 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 Serialiser/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.

Lego Zoo

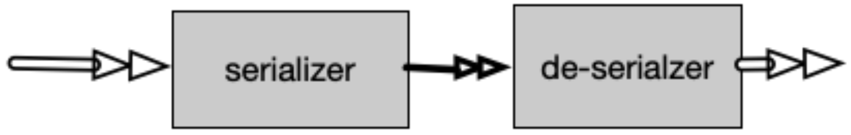


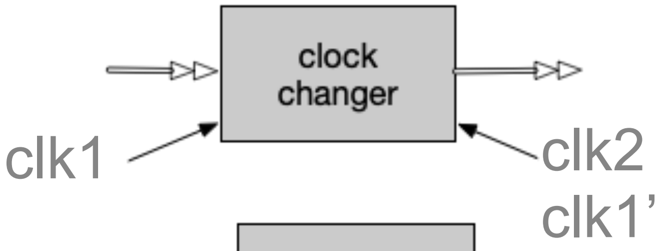
Switches build the network. However for small networks, no switch is essential.
They come in 54/128/256… bits data width

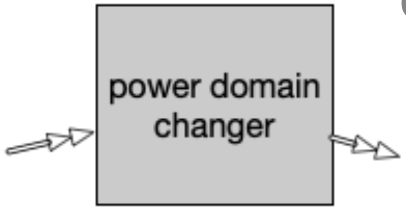


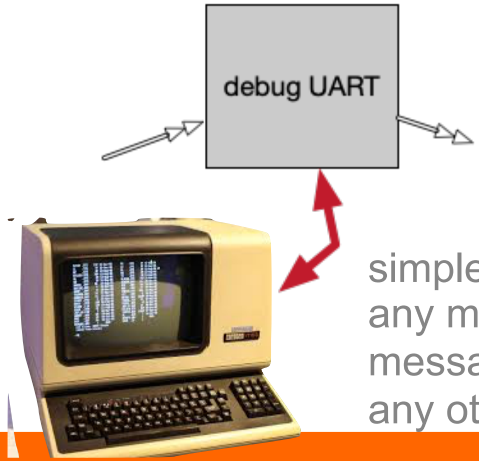


serialisers save number of wires. They come in variety of parallel wires from 1 to 16. They optionally have fast serial clock input. They may be used for inter or intra chip connectivity.





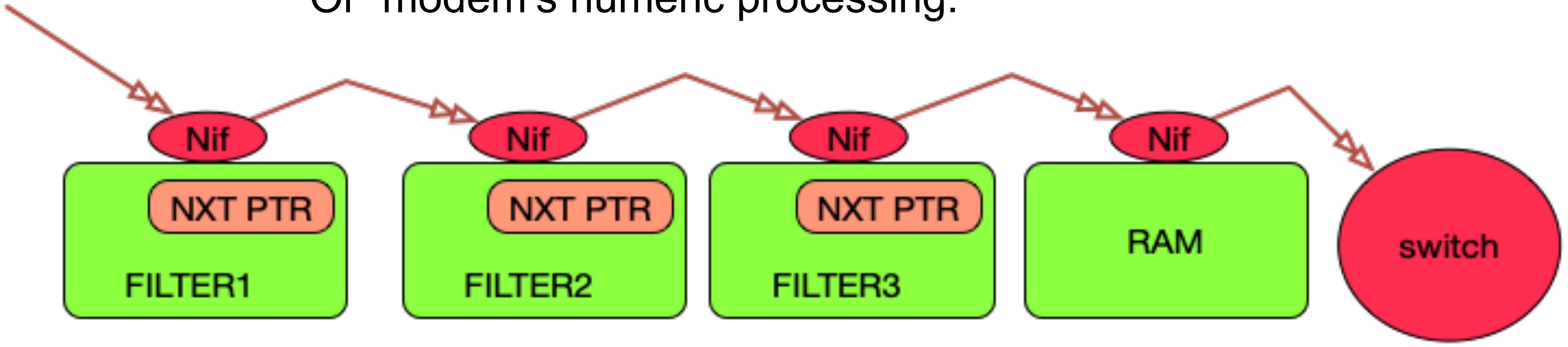




simple character sequence can send any messages. and display any message received. Can impersonate any other module.

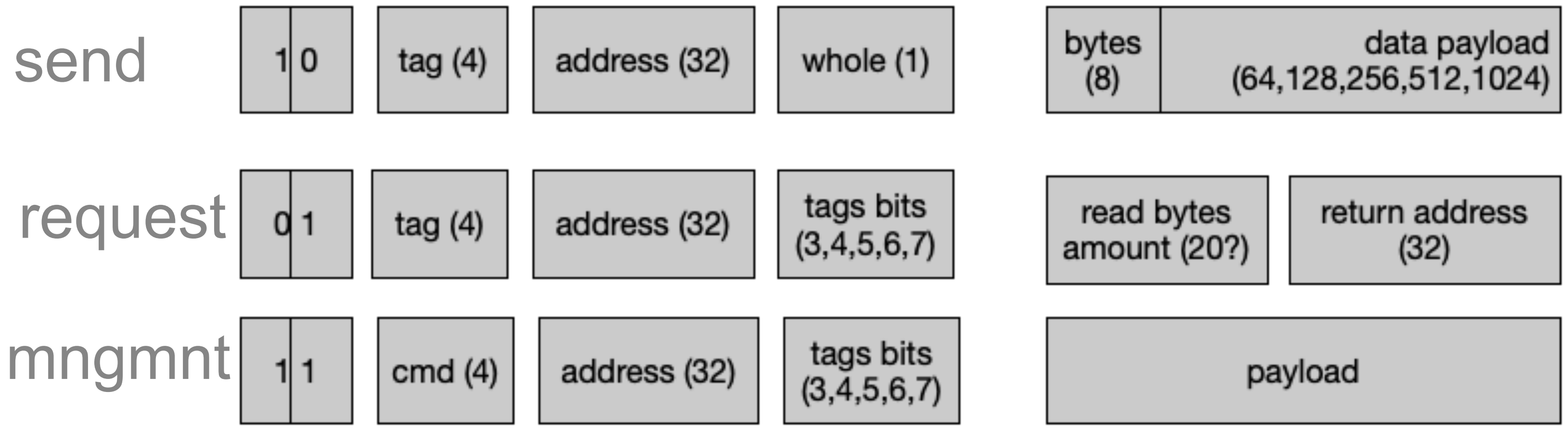
The sequence

As example, each filter is part of video frame pipeline.
Or modem's numeric processing.



Each module has programmed pointer where to send the results. Bypassing a filter , or changing their order is trivial. Also Ram can be used as temp buffer. Or not.

Message classes



00 -idle, no msg
01 = read
10 = write
11 = mngmnt

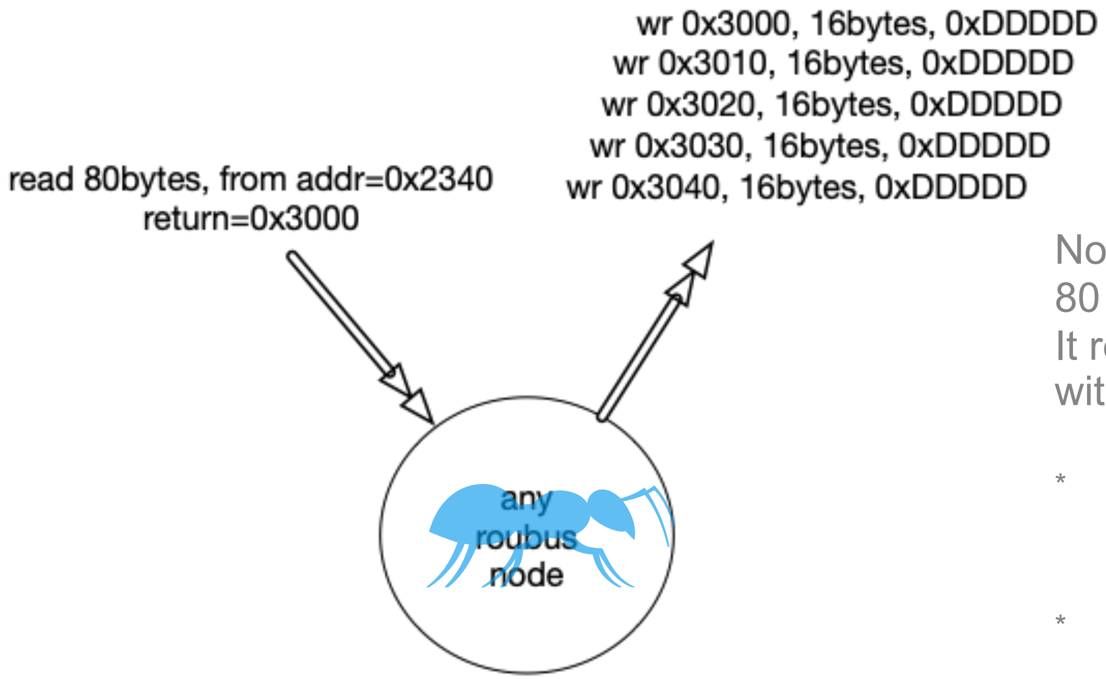
“send data” messages have destination address, data payload and tags.
Tags may signify first in frame, first in line, increment address and such.

“request data” messages have destination address, return address, tags and amount of data requested. Ant that receives these messages, generates in return bunch of “send data” messages.

“management request” is usually sent by manager to initialise the network and query its topology and health.

“management response” is generated by workers to complain about errors or answer to management queries.

reading


wr 0x3000, 16bytes, 0xDDDDD
wr 0x3010, 16bytes, 0xDDDDD
wr 0x3020, 16bytes, 0xDDDDD
wr 0x3030, 16bytes, 0xDDDDD
wr 0x3040, 16bytes, 0xDDDDD
read 80bytes, from addr=0x2340
return=0x3000
any
roubus
node

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.

- the sending node is not necessarily at return address. This is inherent “dma” like functionality.
- there are several tags that go with messages, not shown 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.

* Spill-over option

*No bursts!

network wakeup

ADMIN starts the sequence.

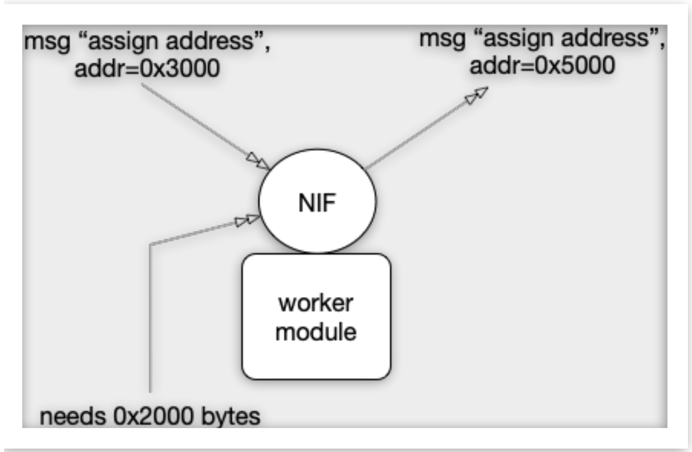
ordering of switches

Each switch assigned an unique id on the spanning tree and port to root.

“who’s your daddy?”

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

run assign address ranges


msg “assign address”, addr=0x3000
msg “assign address”, addr=0x5000
NIF
worker module
needs 0x2000 bytes

Each switch configures it’s initial routing table. Admin may query the network.

fill routing tables in switches

work

Dynamic addressing

Address map of any design is created on each power-up.

And why it is good for You?

WHEN DOING…..

partial test bench

insensitive to uneven chip design advances.

full chip simulation

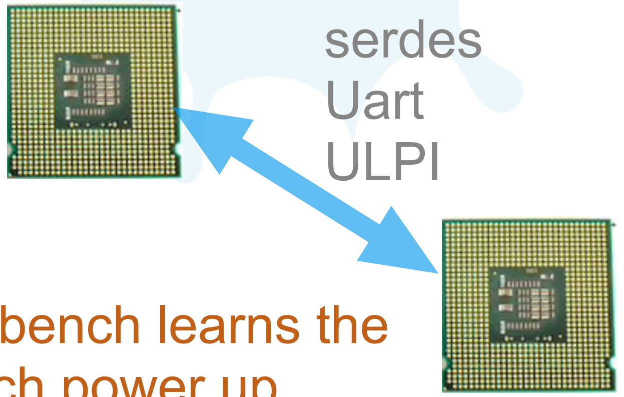
all tests are on the same page- less wasted cycles

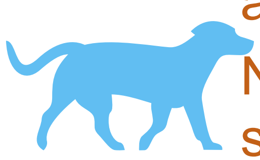
Silicon

self discovery and monitoring is pretty good self test on power up and during lifetime

Multi Chip, Multi FPGA

transparent to software multichip configuration





Software or Test-bench learns the addresses on each power up.

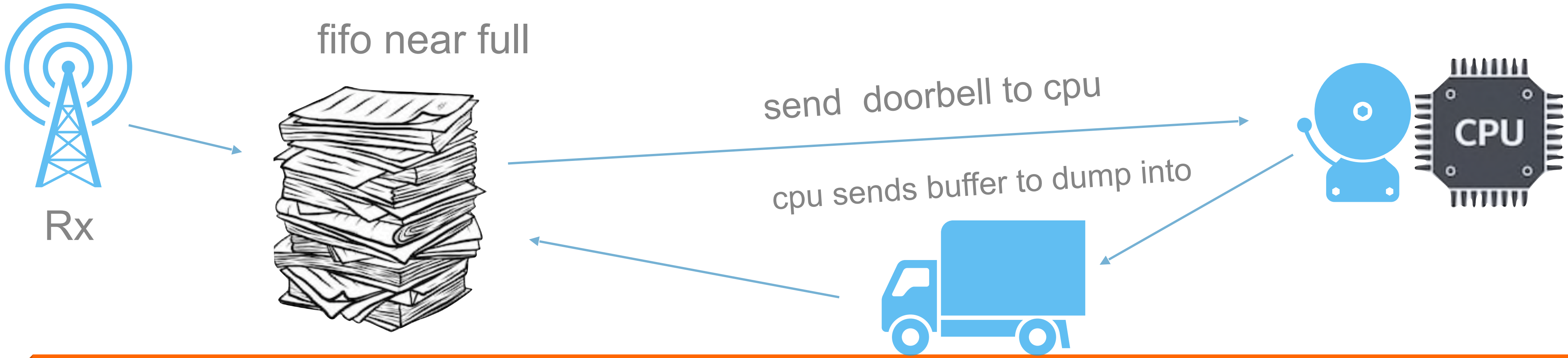
No more chasing its own tale - when something changes anywhere.

Errors & Interrupts

Mapping sideband signals to the Net.

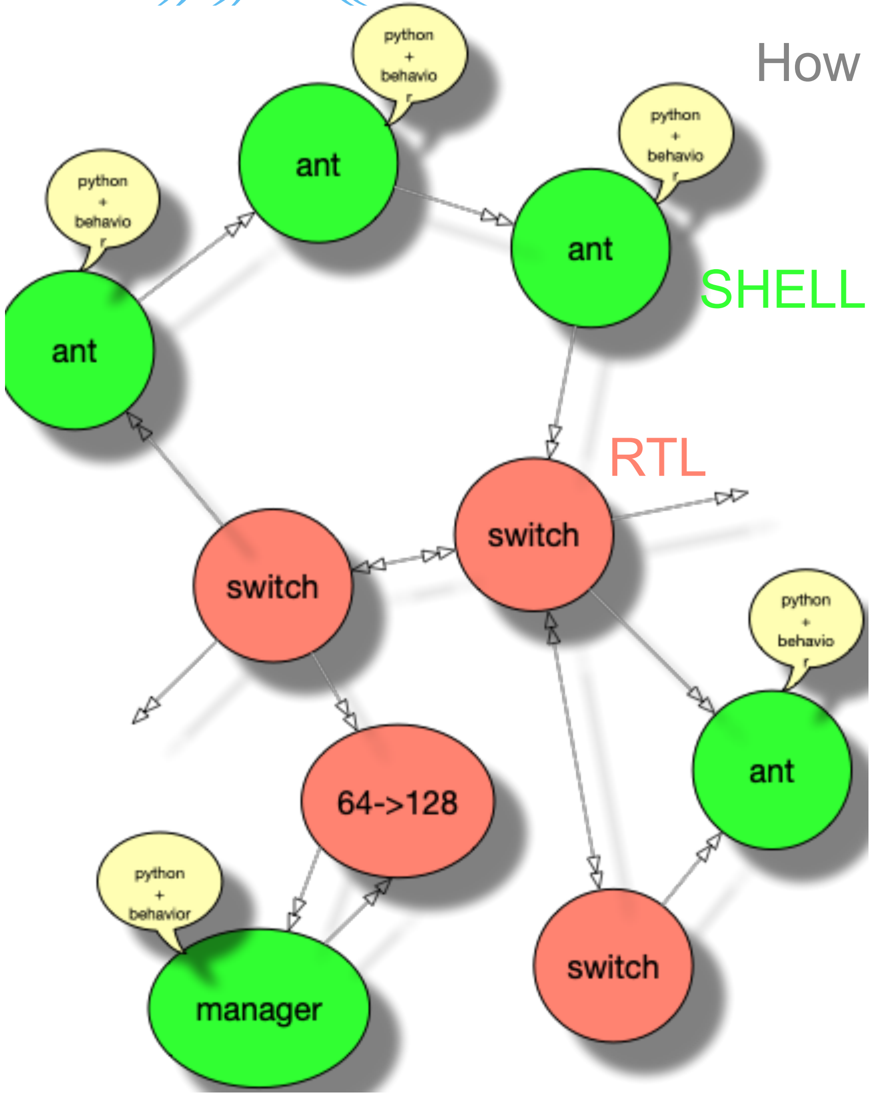
During setup stage of power on sequence, each switch marks the shortest path to manager. Error detected by any agent on the network, are sent as manager response message. It will be automatically routed to the manager node. The data field is stuffed with whatever the sender thinks is relevant.
Switches also catch “stray” messages and send them to manager.

Any module that needs service is programmed with address to ask for service. This address in destination is “sensitive” - detection of writing to it triggers chain of events. For example, transmit fifo nearing emptiness, sends request to fill it. Or same fifo may read from preassigned buffer pool.

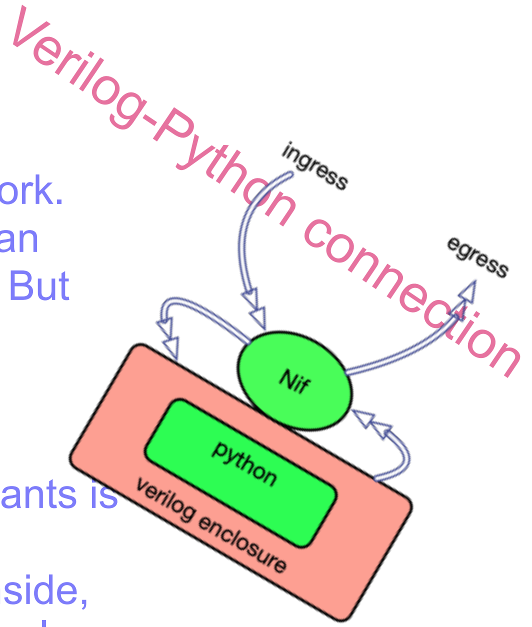


modelling

How to validate performance?



We connect the ants and the switches into netlist of the network. To prove the performance we can use real switches in simulation. But how about the ants?



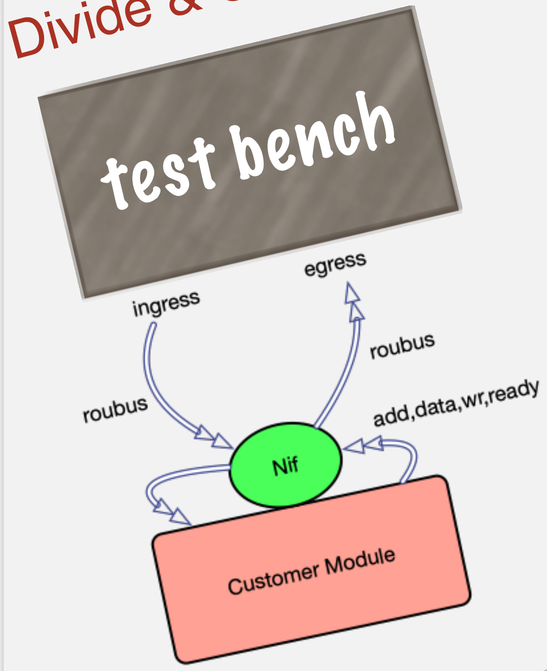
For ants we use verilog-python connection. The organisers of the ants is modelled using python code.
Since all ants, no matter what is inside, look the same from the outside, single python model with adaptable communication habits can fit the bill.
We take one ant shell verilog , stuffed with python filling and configure many copies of it with specific behaviours.
Use Verilator and You have perfect free fast model.

Functionality is not affected by topology, only performance.

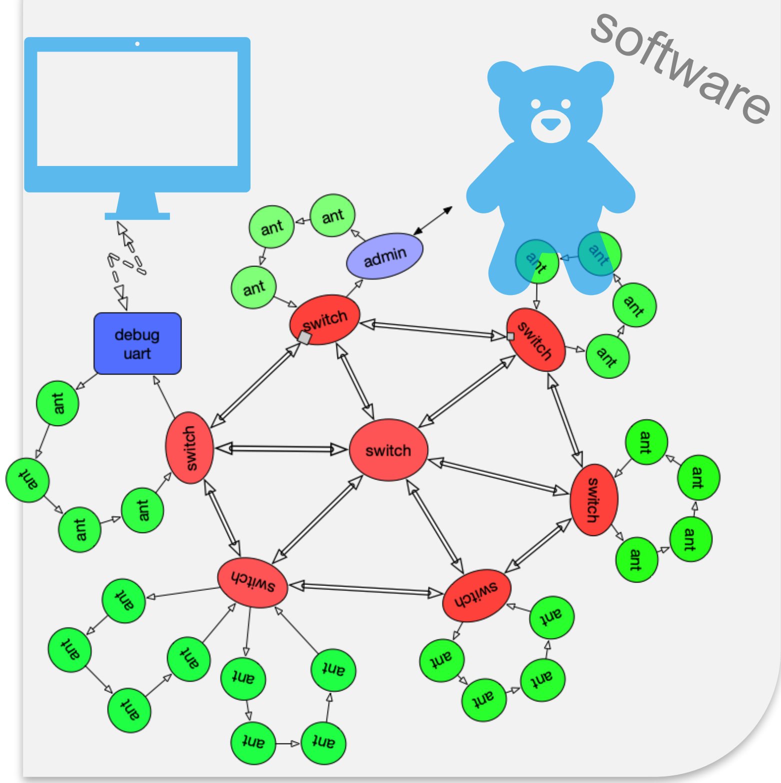
verification & Validation

Each module by itself
Play all scenarios
Cover 90%

Divide & Conquer



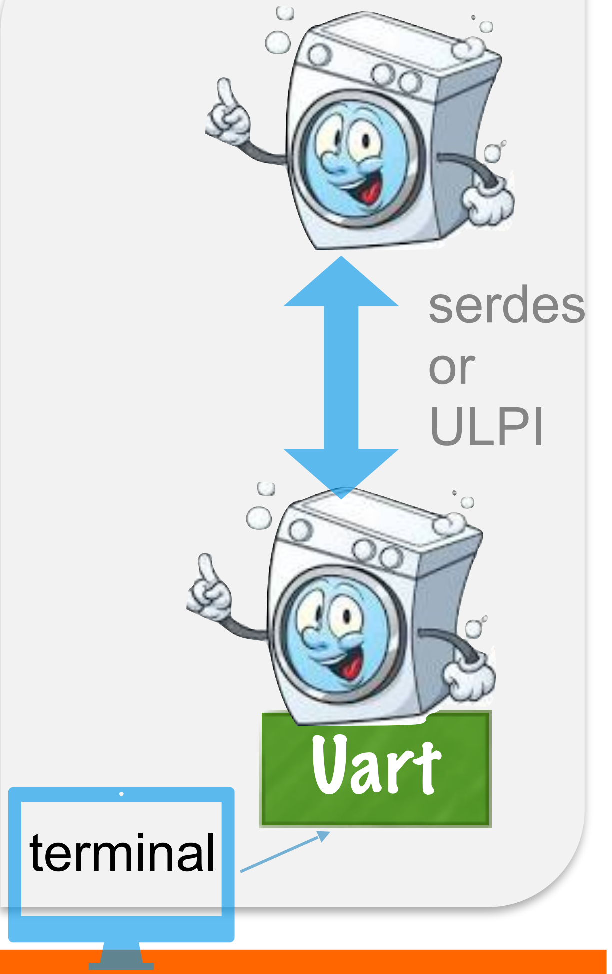
Partial chip / Full chip or Silicon



any node can fake any other node

nodes without RTL implementation can be faked with python

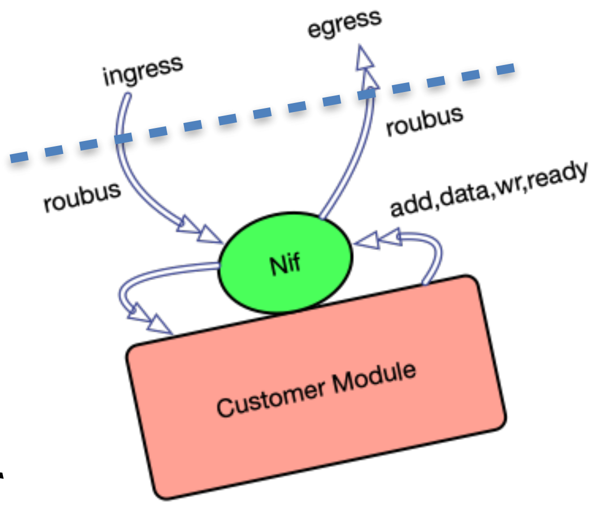
Multi chip / FPGA



what about GDSII?

Are we friendly to: Backend / Layout /Timing closure

All Constraints cut here



Timing always stops at the gate of the Ant
Place&Route blobs may bunch together any number of ants

Network topology can be tweaked up to a few days before tape-out.

To meet chip level timing - reshuffle topology.

LEC may end at module level.

Promises…

- 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, should make the software job manageable.
- Enable adequate level of verification.
- Survive P&R and the rest of backend.
- Make clocking / power / selftest easy.
- Give enough time for system modelling.

not to forget…

security: there are ideas, alas not mine, but plausible. Switches are the policemen. Adding Field in the message.

safety: ECC, CRC, CHECKSUM

cache coherence: when (or if) we need it?

system design: it is different. mental switch needed. requires higher level agreements.

scan, bist, clocking, power : all included.

“comprehensive & total” approach. Modules are connected to the chip “EXCLUSIVELY” by ROUBUS. They talk only through ROUBUS. No sideband signals.

Can All these fine goals be achieved in the same design?
Will it end the world hunger?

Thank You

for RTLs and more clone:

https://github.com/greenblat/vlsistuff.git