Debug, Test, and Security Services on Networks on Chip

Kees Goossens
NXP Semiconductors Research &
Computer Engineering, Delft University of Technology

overview

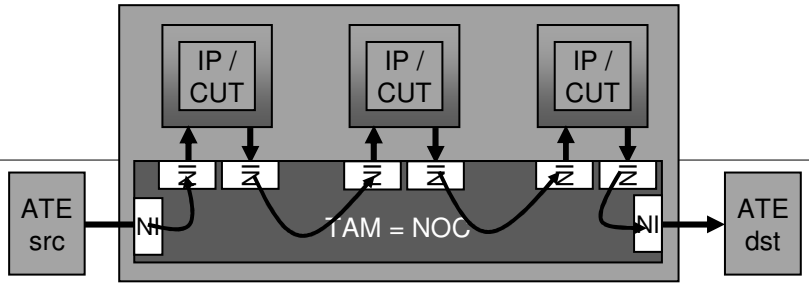
- NOC are at the centre of SOC
- good place to monitor & control the SOC

- for example for:

1. test
2. (communication) performance management
3. debug
4. security

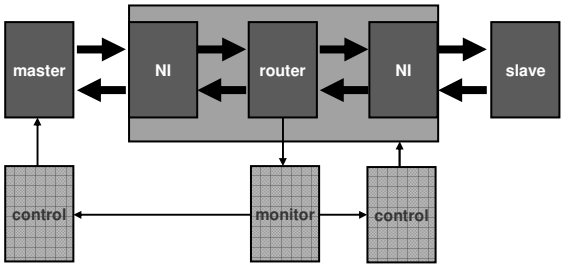
NOC to transport test patterns & responses

- TAM = interconnect ∧ interconnect = NOC ➔ TAM = NOC
- use guaranteed latency (zero-jitter) connections for virtual TAM wires
- NOC is unchanged; test is just another functional mode
- IP test schedule is unchanged: re-use
- new test wrapper: converts between the functional & test modes
  - e.g. AXI & streaming test mode



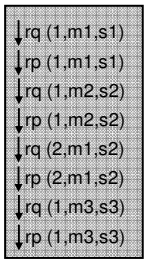
NOC for performance monitoring & control

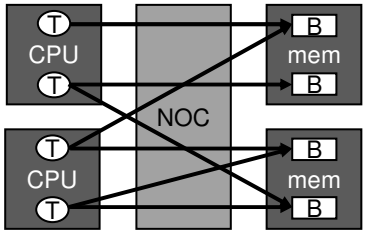
- monitor NIs, routers, IP
  - link utilisation, buffer fillings, credits, etc.
  - hardware is relatively cheap
- control
  - stream to trace memory, or off-chip
  - to a controller, centralised or decentralised
  - control NI, routers, IP



NOC for debug: communication-centric

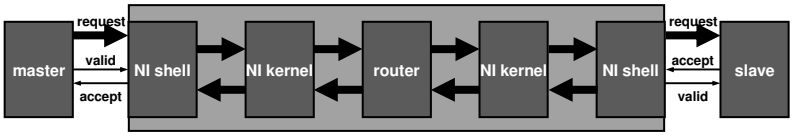
- SOC complexity resides in the interactions between IP blocks / tasks
- the interconnect is the locus of these interactions
  - older interconnects serialise all transactions
  - latest interconnects allow split, pipelined, concurrent transactions: no unique SOC communication trace
  - analyse SOC behaviour by controlling its communication (transaction ordering)





NOC for debug: transaction-based

- transactions are the common interface between IP / processor instructions and interconnect transactions
- signal groups (command, write data, read data, ...)
- valid/accept (valid/ready) handshake



- basic concept: intervene in the valid/accept handshake

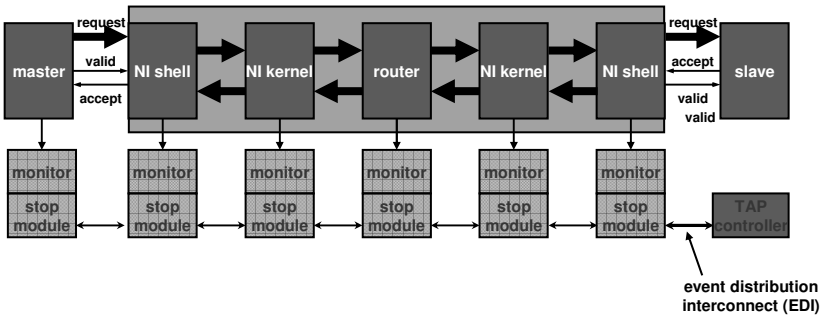
NOC for debug

1. program monitors with event triggers
2. monitors or TAP controller generate event
3. event is distributed to network interfaces
4. finish ongoing handshakes
5. mask accept/valid on selected connections
6. check if NOC activity has ceased, otherwise force a stop
7. read out debug data

- can implement breakpoints, single stepping, etc.

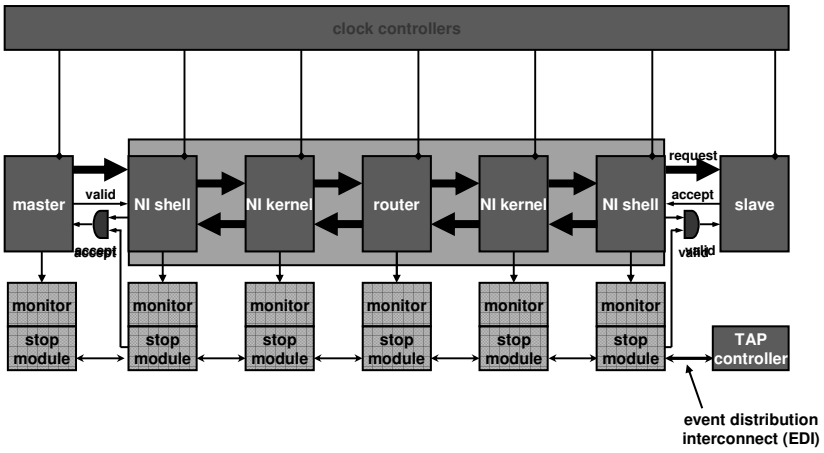
NOC for debug: architecture & operation

- monitor or TAP controller generates event
- event is distributed on broadcast interconnect
  - runs at NOC functional speed
  - follows NOC layout



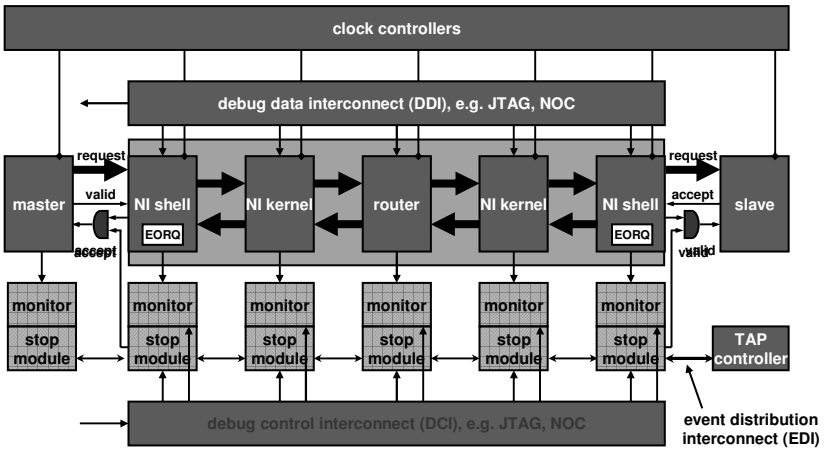
NOC for debug: architecture & operation

- finish ongoing handshakes, then mask accept/valid



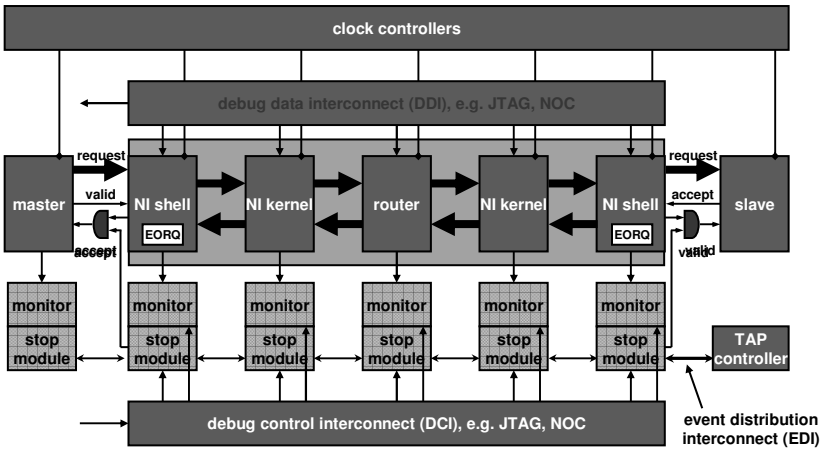
NOC for debug: architecture & operation

- check if NOC activity has ceased, otherwise force a stop

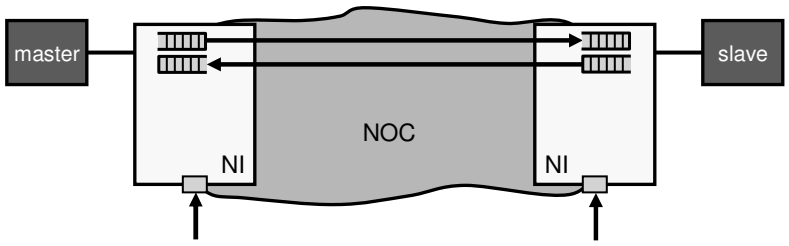


NOC for debug: architecture & operation

- read out debug data



NOC for DRM & security: configuration

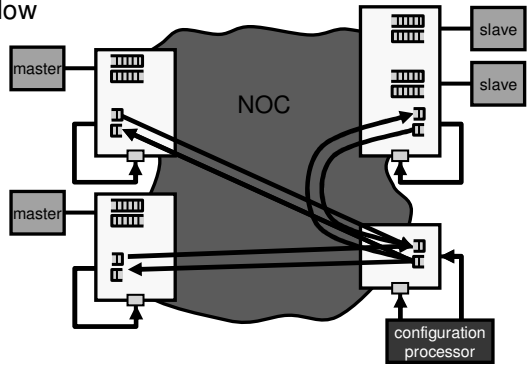


- managed (virtual) point-to-point communication, not broadcast
- connection consists of
  - request & response channels
- channel configuration: write registers in the source NI using memory-mapped IO
- every channel has its own
  - QoS (BE, GT)
  - programmable memory map

NOC for DRM & security

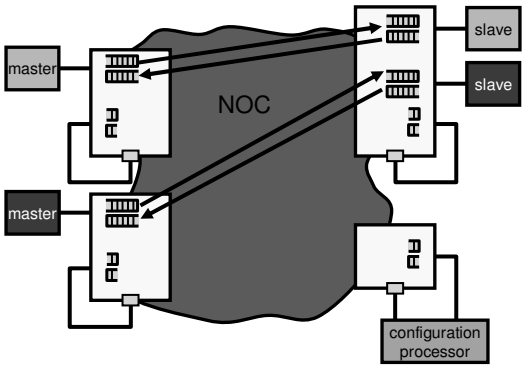
- managed (virtual) point-to-point communication, not broadcast
- bootstrap NOC from 1 secure location
- reconfiguration software runs on (e.g.) embedded processor & is generated by the design flow
- configuration channels & boot processor can be hardwired for extra security

```
do_use_case_1() {
    open_connection (A,B,10MB/s);
    open_connection (C,D, 200MB/s);
    …
} ...
```



NOC for DRM & security

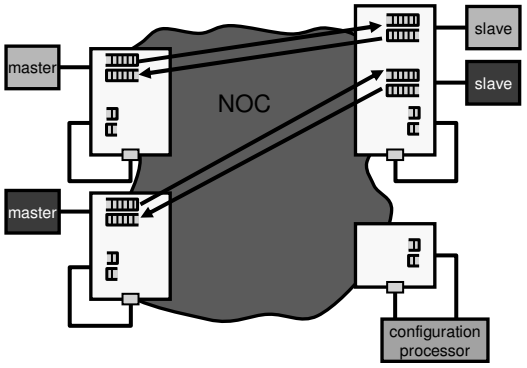
- managed (virtual) point-to-point communication, not broadcast
- bootstrap from 1 location
- configuration channels can be hardwired for extra security
- every channel has its own memory map
  - data access isolation
  - no snooping or data modification



NOC for DRM & security

- managed (virtual) point-to-point communication, not broadcast
- bootstrap from 1 location
- configuration channels can be hardwired for extra security

- by using guaranteed (GT) connections the performance of communications is also decoupled
  - no information leaks through timing analysis
  - no denial of service



acknowledgements

- monitoring [IEEE Comm. Mag. 2003, HLDVT'04, DSD'06, Pastrnak ISCE'06]
  - Calin Ciordas (TUE)
- test [ETS'06, IET Comp. & Dig. Tech.'07]
  - Erik Jan Marinissen (NXP)
  - Alexandre Amory (UFPE)
  - Pengwei Ren (TUD)
  - Ardy van den Berg (TUD)
- debug [NOCS'07, ETS'07]
  - Bart Vermeulen (NXP)
  - Siddharth Umrani (TUD)
  - Remco van Steeden (UT)
  - Martijn Bennebroek (Philips)
- plus the whole Æthereal team
  - esp. Martijn Coenen (NXP), Andreas Hansson (TUE), Jef van Meerbergen (Philips)