Hardwired Networks on Chip in FPGAs to Unify Functional and Configuration Interconnects

Kees Goossens 1,2
Martijn Bennebroek 3
Jae Young Hur 2
Muhammad Aqeel Wahlah 2

1 Research, NXP Semiconductors
2 Computer Engineering, Delft University of Technology
3 Philips Research

overview

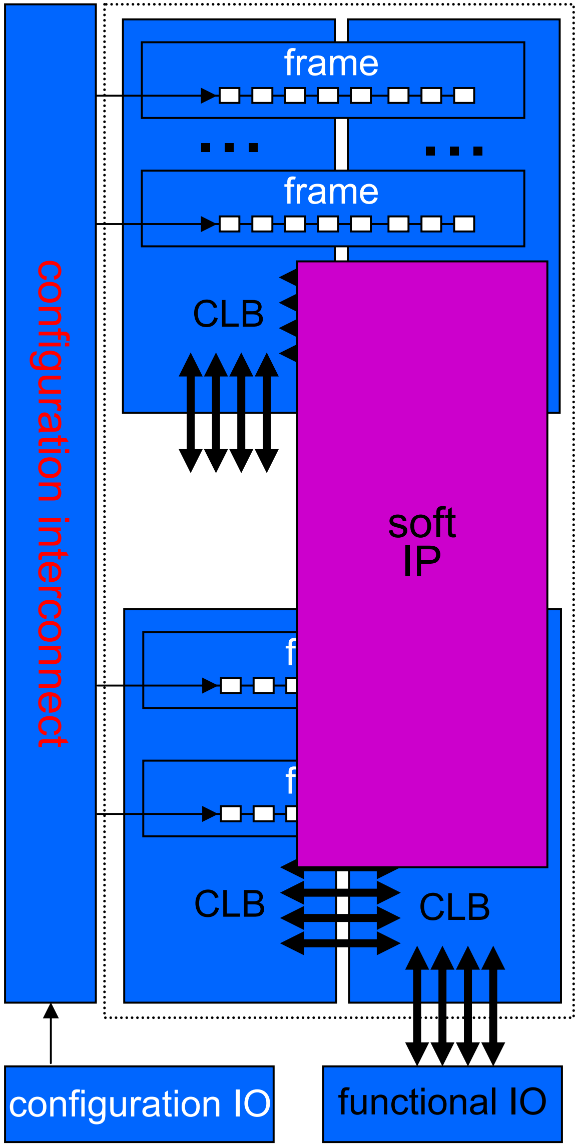
- conventional FPGA
  - separate hard configuration interconnect and soft NOC
- proposed FPGA
  - single hard NOC for configuration, programming, data
- hardware architecture
- boot procedure
- performance : cost analysis
- conclusions

configuration: loading bitstream
programming: writing MMIO registers

conventional FPGA

- soft IP are configured in configurable logic blocks (CLB)
- CLBs are interconnected with switch boxes (shown as fat arrows)

- dedicated configuration interconnect to access frames (minimum unit of reconfiguration)
- frames and CLBs are orthogonal (22 frames cover 16 CLBs in Virtex4)
- hence minimum coherent unit of reconfiguration is 22 frames



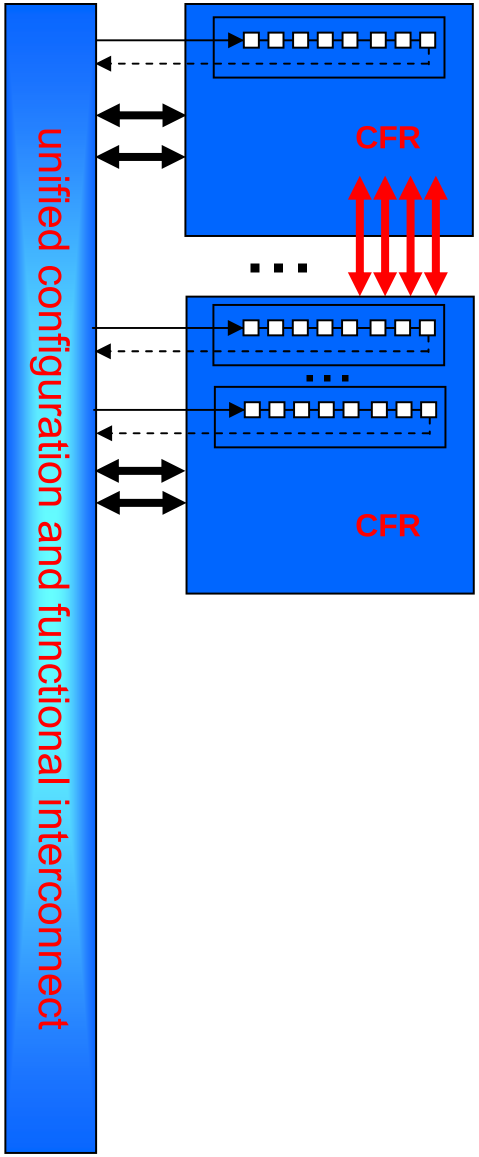
proposed FPGA

- configuration and function region (CFR)
  - cf. Virtex4’s 22 frames
  - but can be orthogonal and independent as in current FPGAs
  - interconnected as in current FPGAs (see red arrows): no “tiles”
- NOC unifies & connects to

⟷ 1. functional data ports of IP

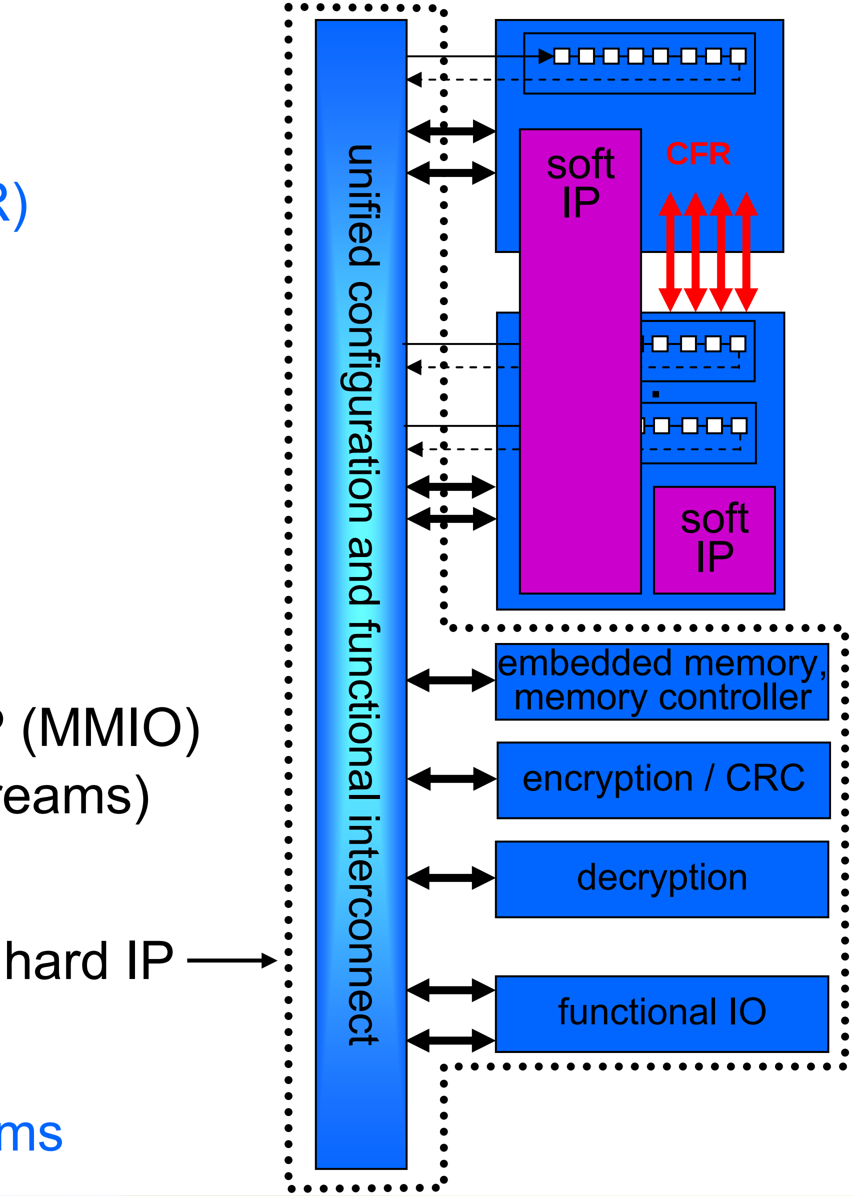
⟷ 2. functional programming ports of IP (MMIO)

⇄ 3. configuration ports of soft IP (bitstreams)



proposed FPGA

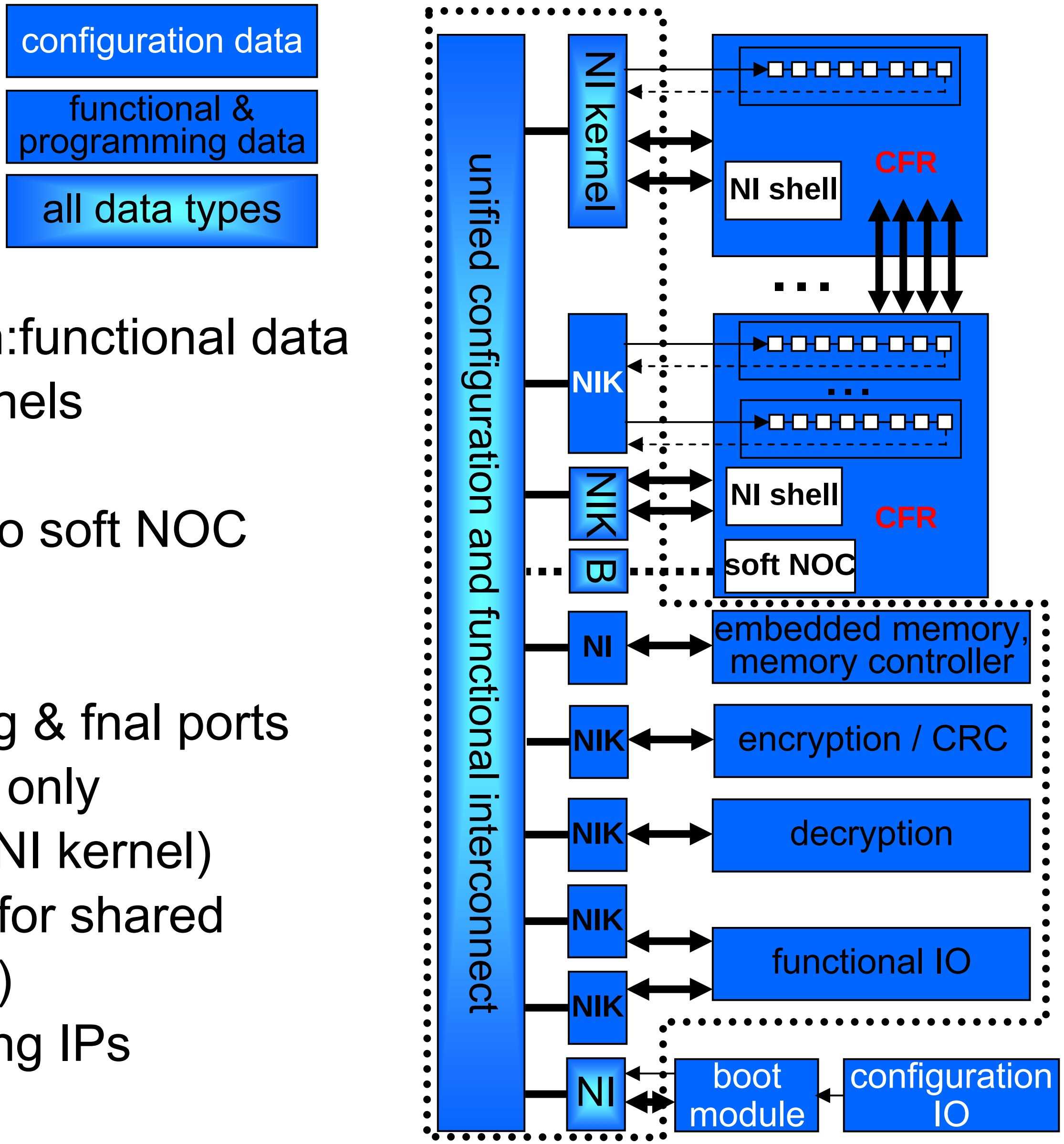
- configuration and function region (CFR)
  - cf. Virtex4’s 22 frames
  - but can be orthogonal and independent as in current FPGAs
  - interconnected as in current FPGAs (see red arrows): no “tiles”
- NOC unifies & connects to
  1. ⟷ functional data ports of IP
  2. ⟷ functional programming ports of IP (MMIO)
  3. ⇄ configuration ports of soft IP (bitstreams)
- 1 & 2 for both soft & hard IP
- NOC is partially hard & soft
- enables conversion of data ⇔ bitstreams



unified configuration and functional interconnect
soft IP
CFR
soft IP
embedded memory, memory controller
encryption / CRC
decryption
functional IO
hard IP

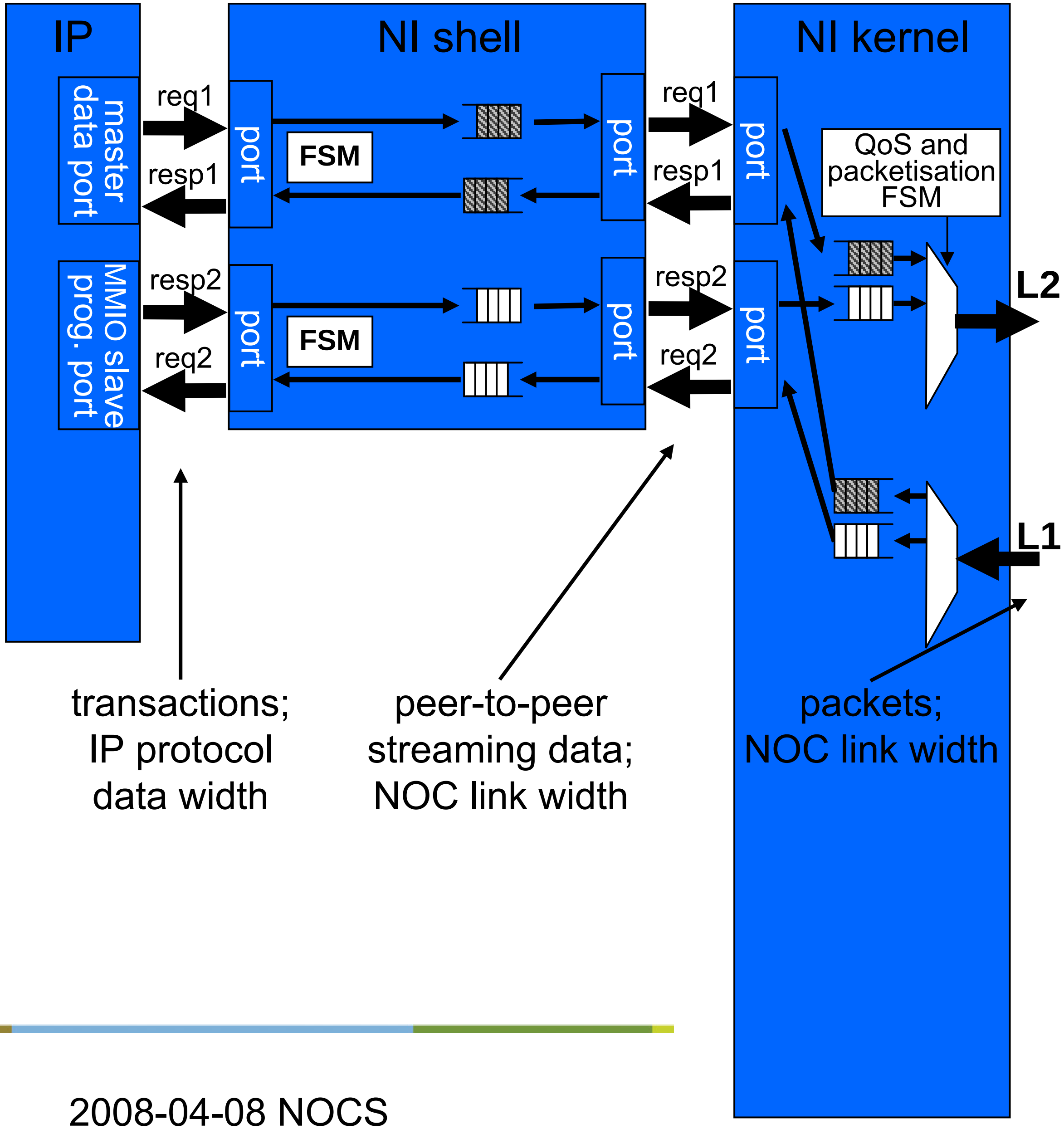
proposed FPGA

- more details on
  - soft:hard balance
  - mixing of configuration:functional data on one or more NI kernels
  - place of boot module
  - possibility of bridging to soft NOC
- shown:
  - NI kernel to both config & fnal ports
  - NI kernel to conf ports only
  - bridge to soft NOC (= NI kernel)
  - hard NI kernel & shell for shared memory IP (memories)
  - hard kernel to streaming IPs (en/decryption & IO)



conventional IP & network interface (NI)

- IP has data and control (MMIO) ports
- both are connected to the NOC: unified data & IP programming interconnect



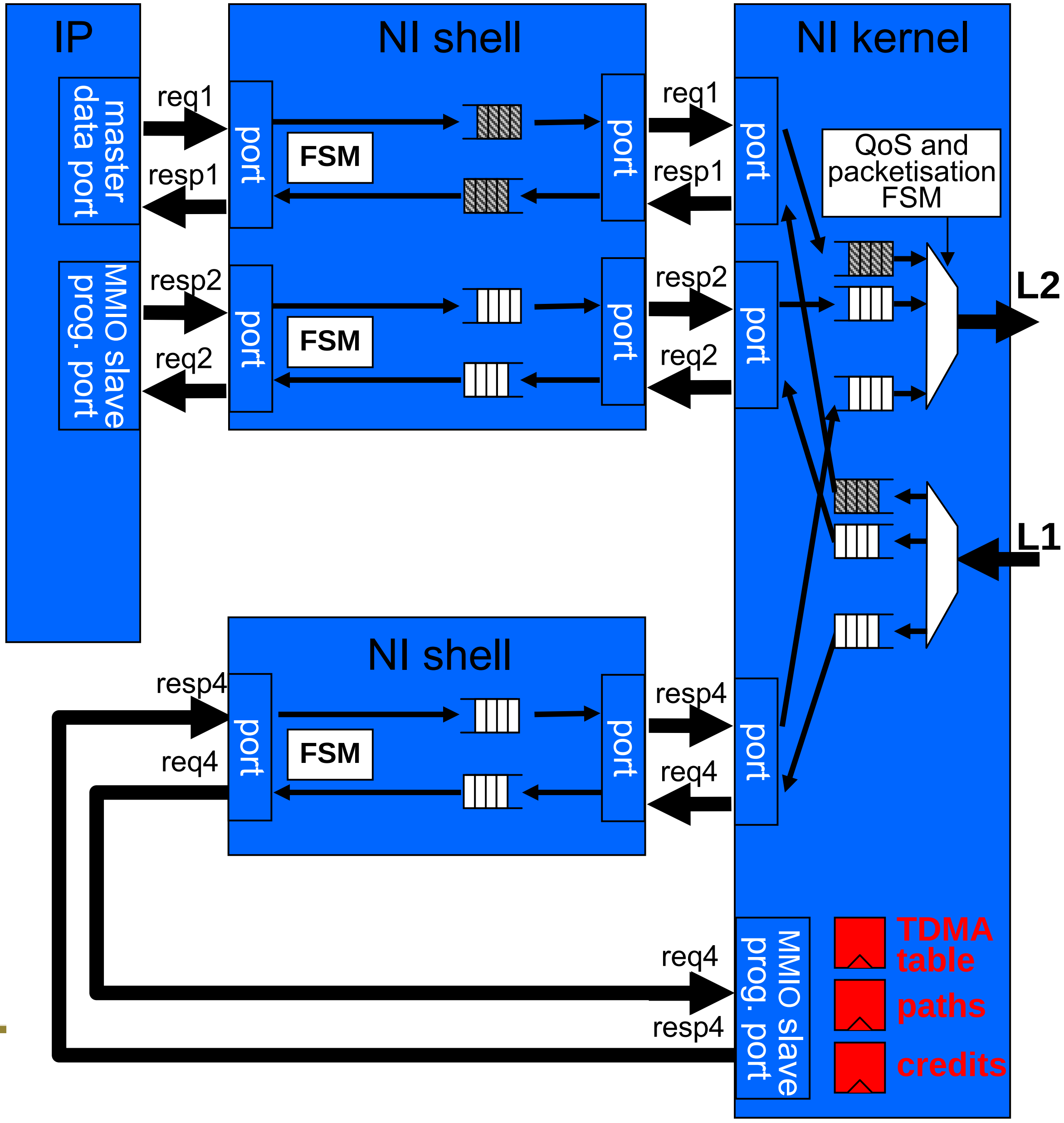
transactions; IP protocol data width

peer-to-peer streaming data; NOC link width

packets; NOC link width

conventional NI

- only NIs of NOC are programmed
- using the NOC itself
- unified IP & NOC programming

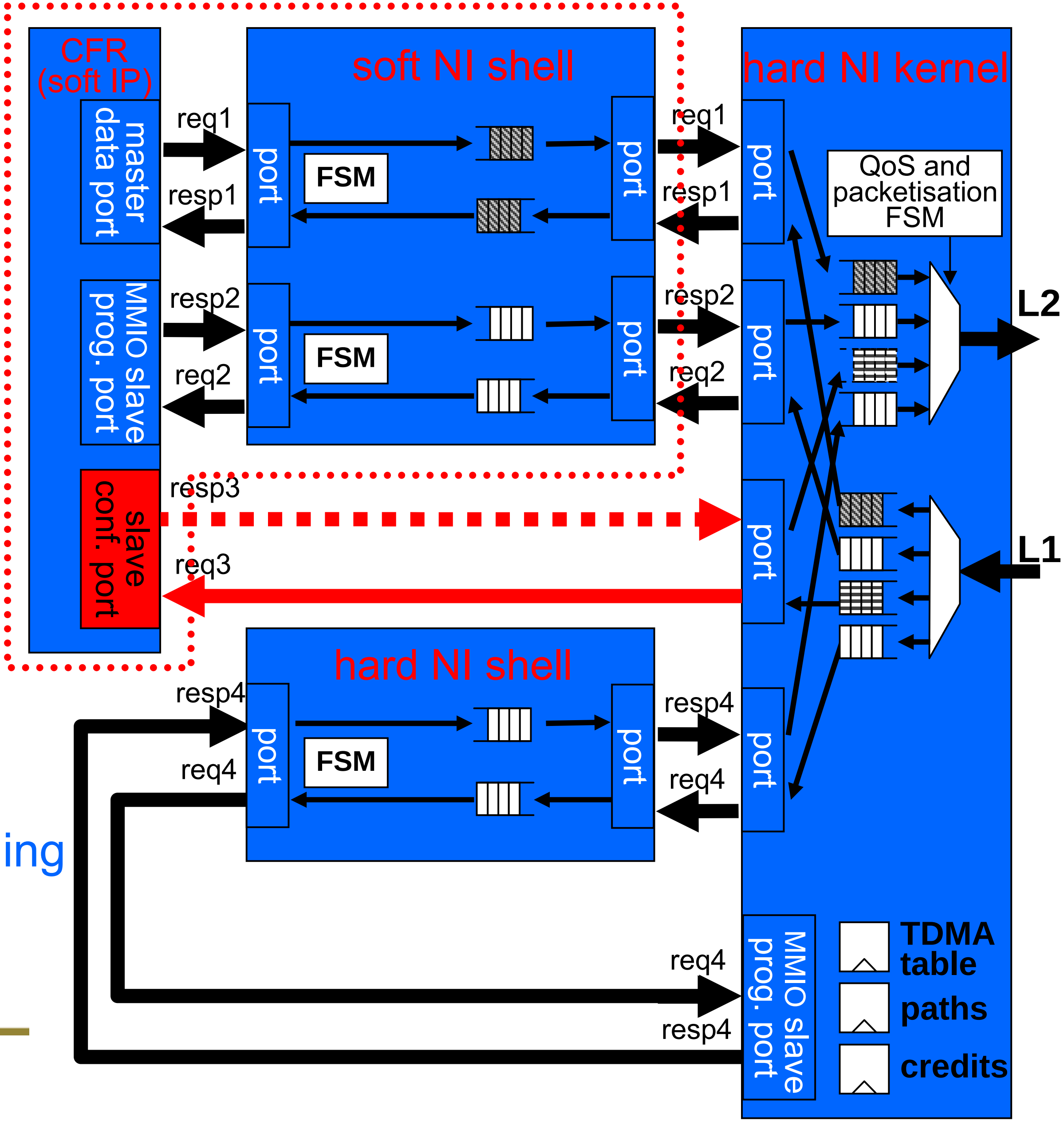


NI in FPGA context

- NI kernel is hard
- IP and its shell can be soft or hard
- hard shell for e.g.: processor, memory, IO

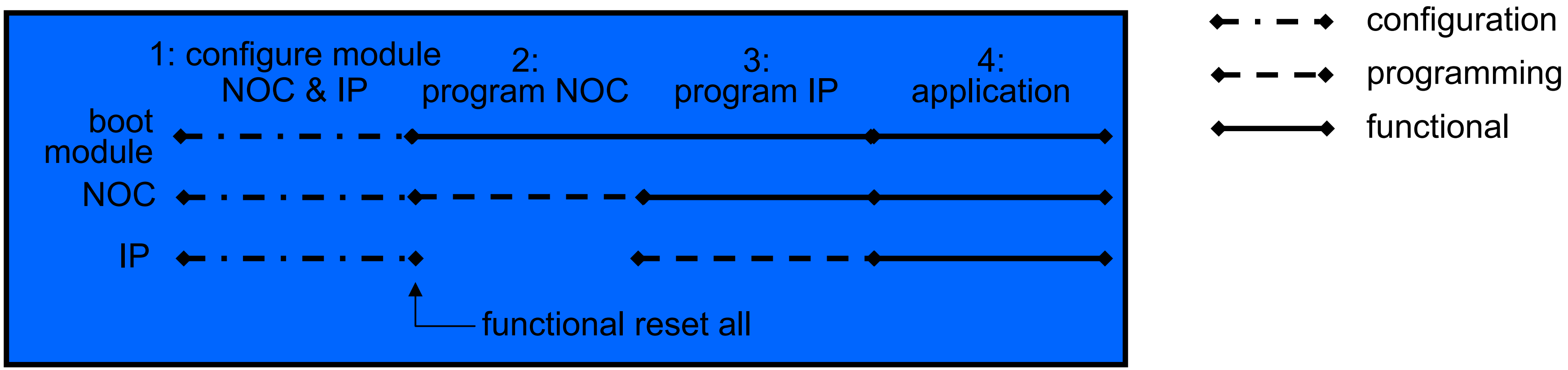
- soft IP is configured by bitstream transported via the NOC

- unified interconnect for
  - data
  - NI & NOC programming
  - configuration



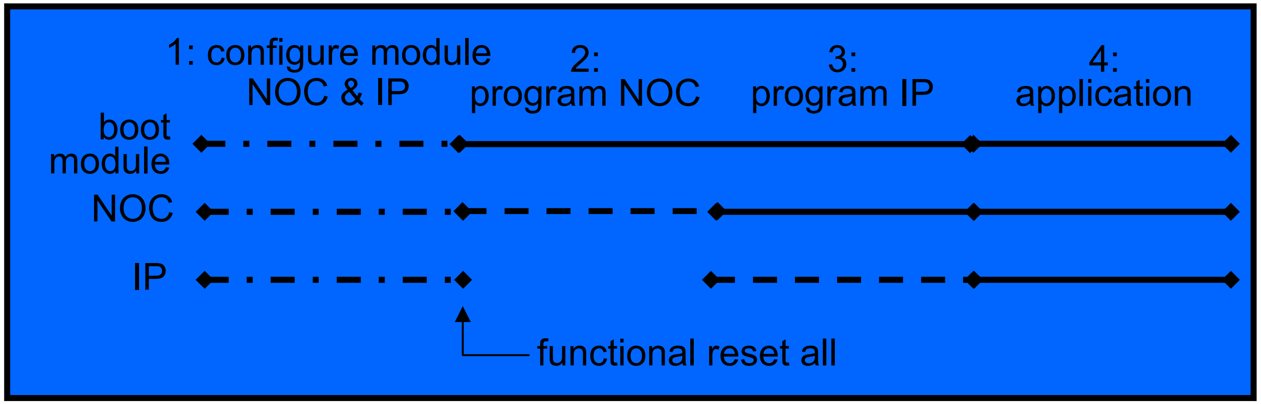
configuration & programming

- conventional

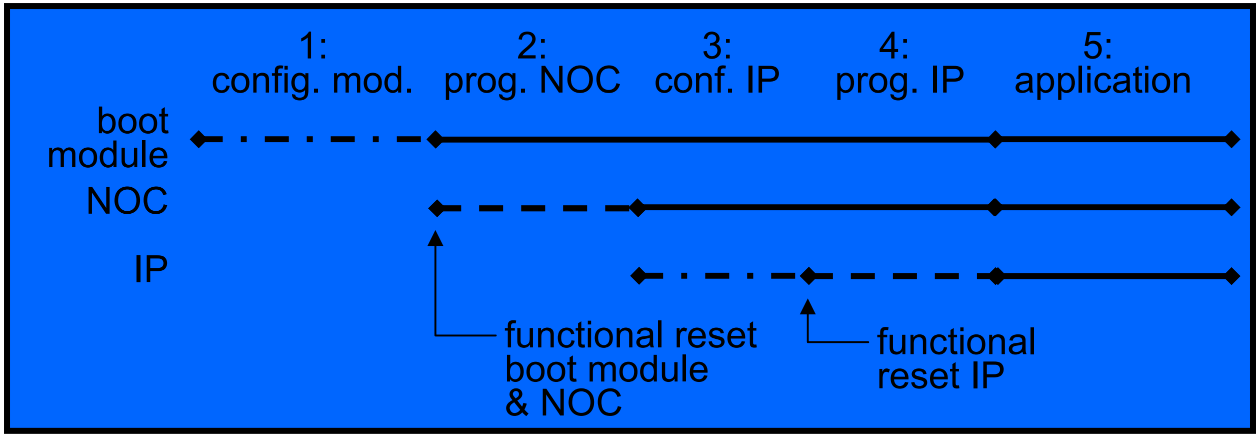


configuration & programming

- conventional



- new



◆- · - ◆ configuration

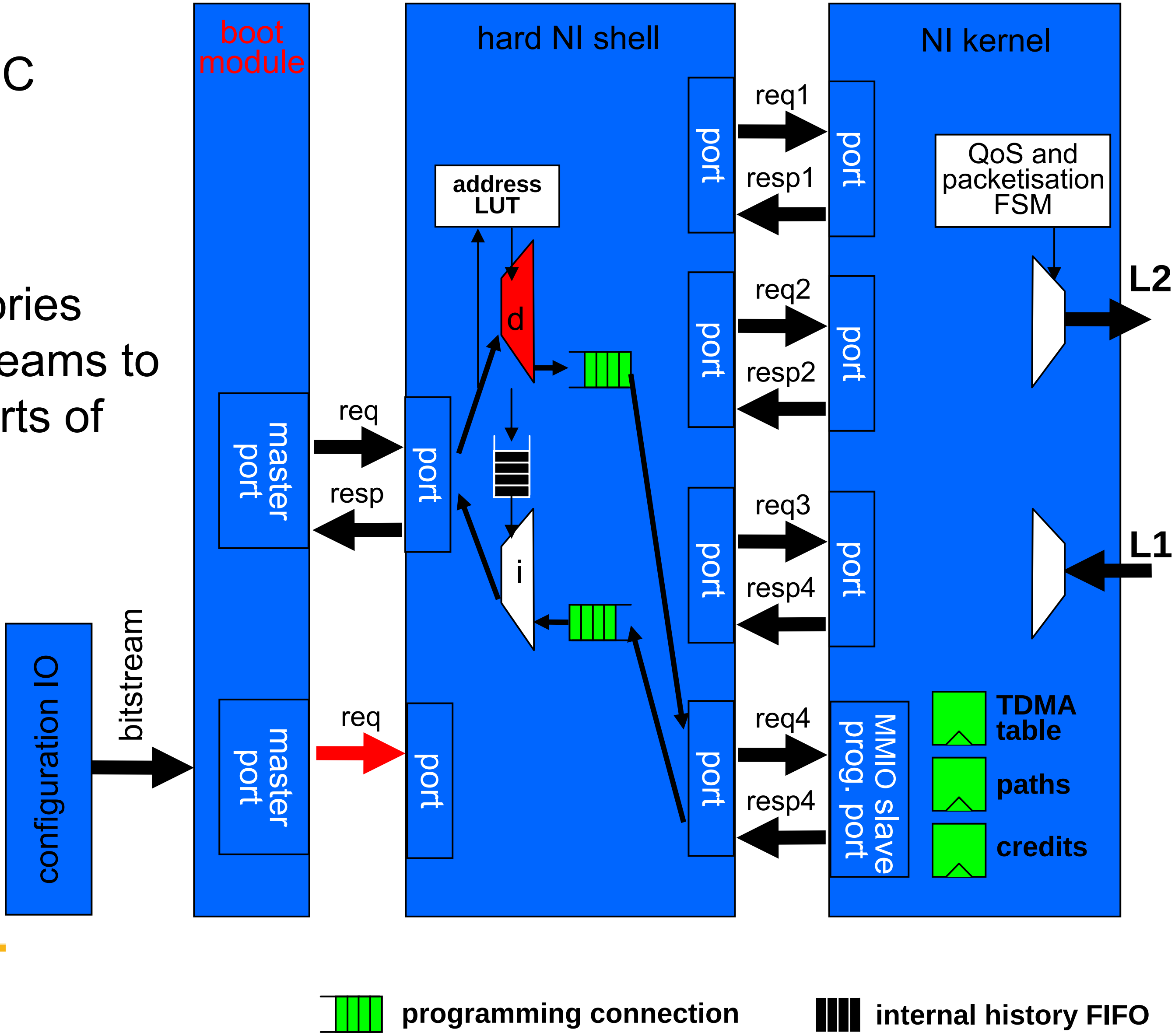
◆- - -◆ programming

◆——◆ functional

configuration & programming

1. program the NOC

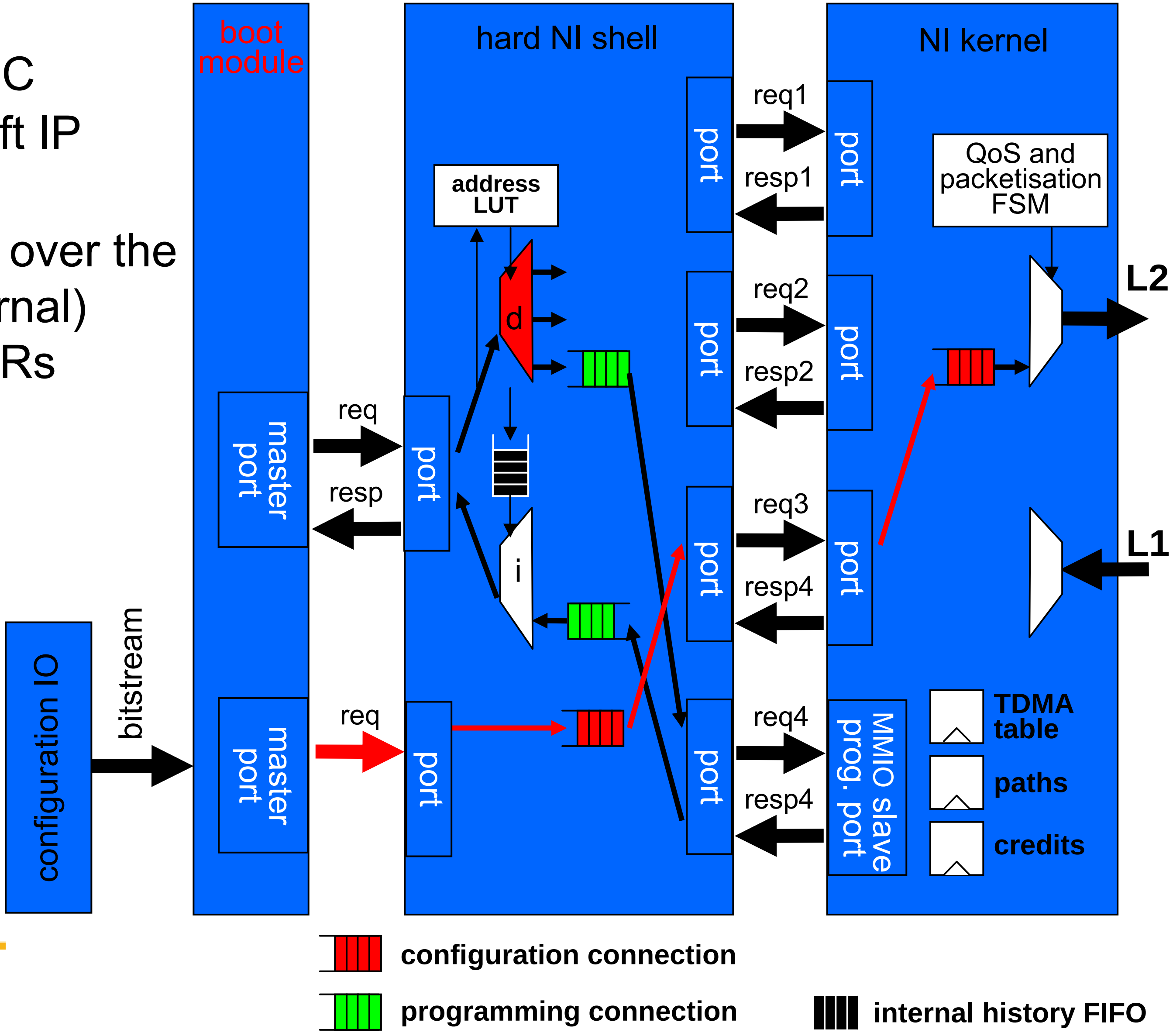
- connect boot module & (external) memories containing bitstreams to configuration ports of soft IP (CFR)



configuration & programming

1. program the NOC
2. configure the soft IP

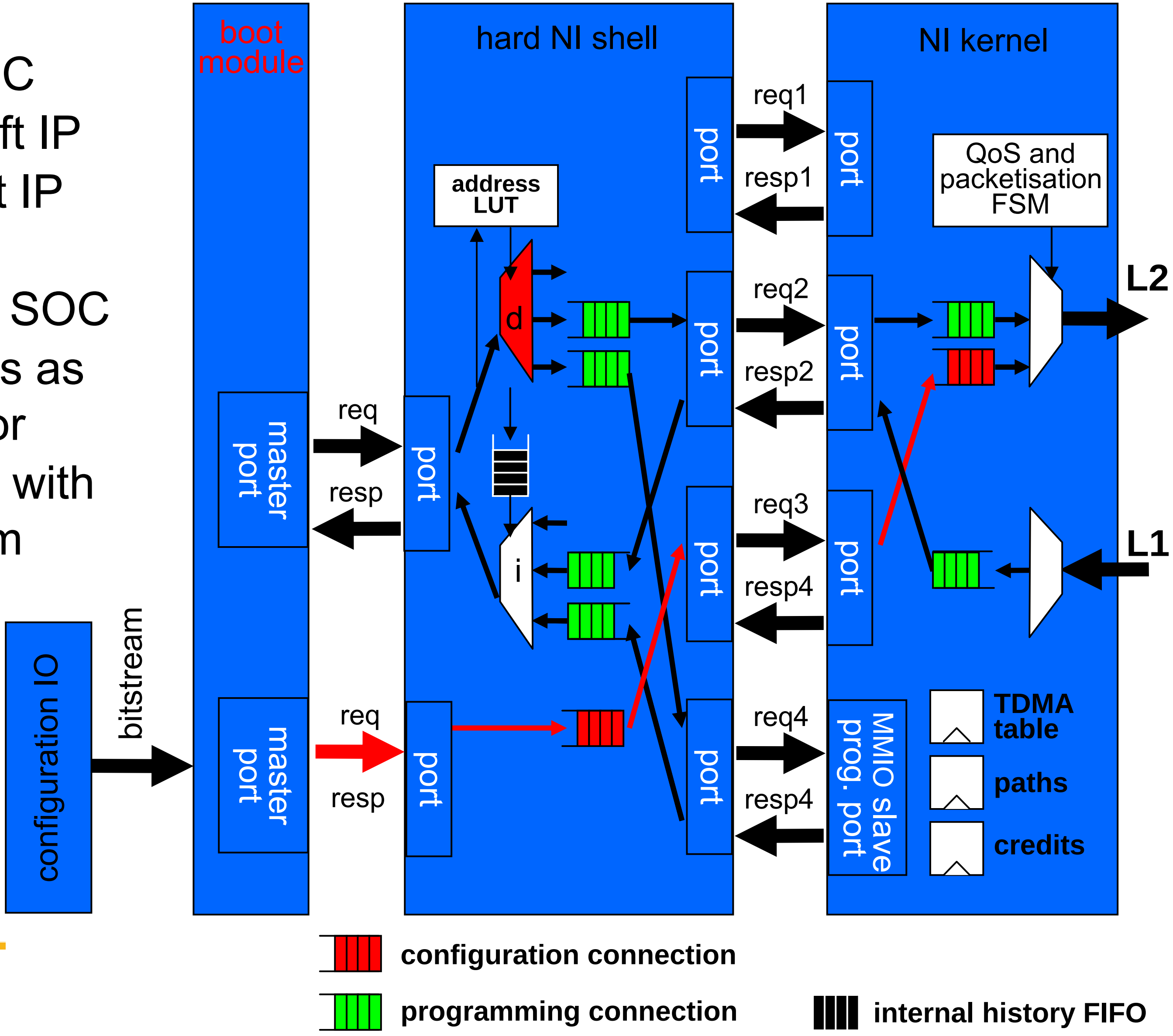
- send bitstreams over the NOC from (external) memories to CFRs



configuration & programming

1. program the NOC
2. configure the soft IP
3. program the soft IP

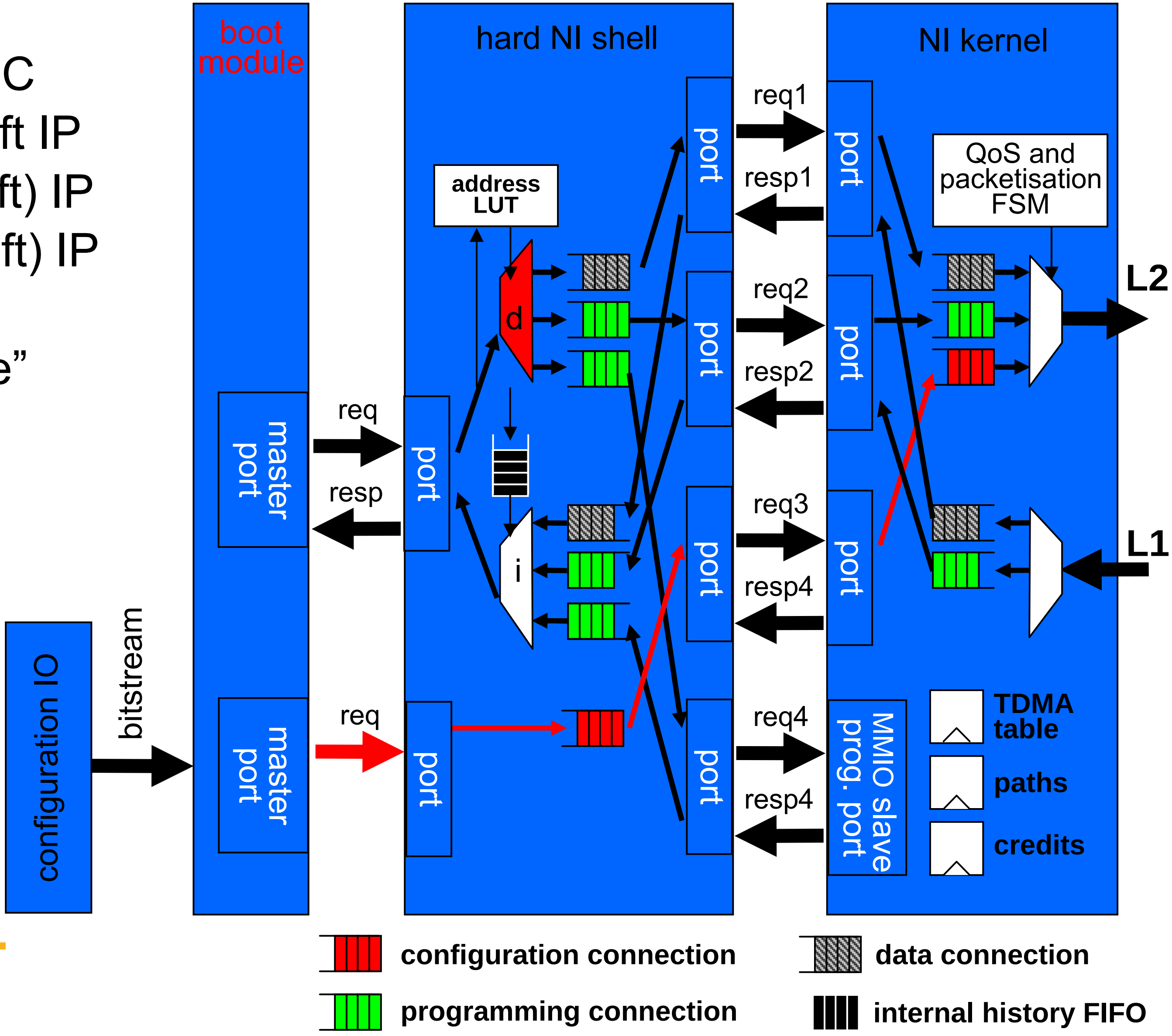
- “normal boot” of SOC
- boot module acts as control processor
- reprogram NOC with connections from boot module to IP MMIO programming ports
- program IP



configuration & programming

1. program the NOC
2. configure the soft IP
3. program the (soft) IP
4. send data to (soft) IP

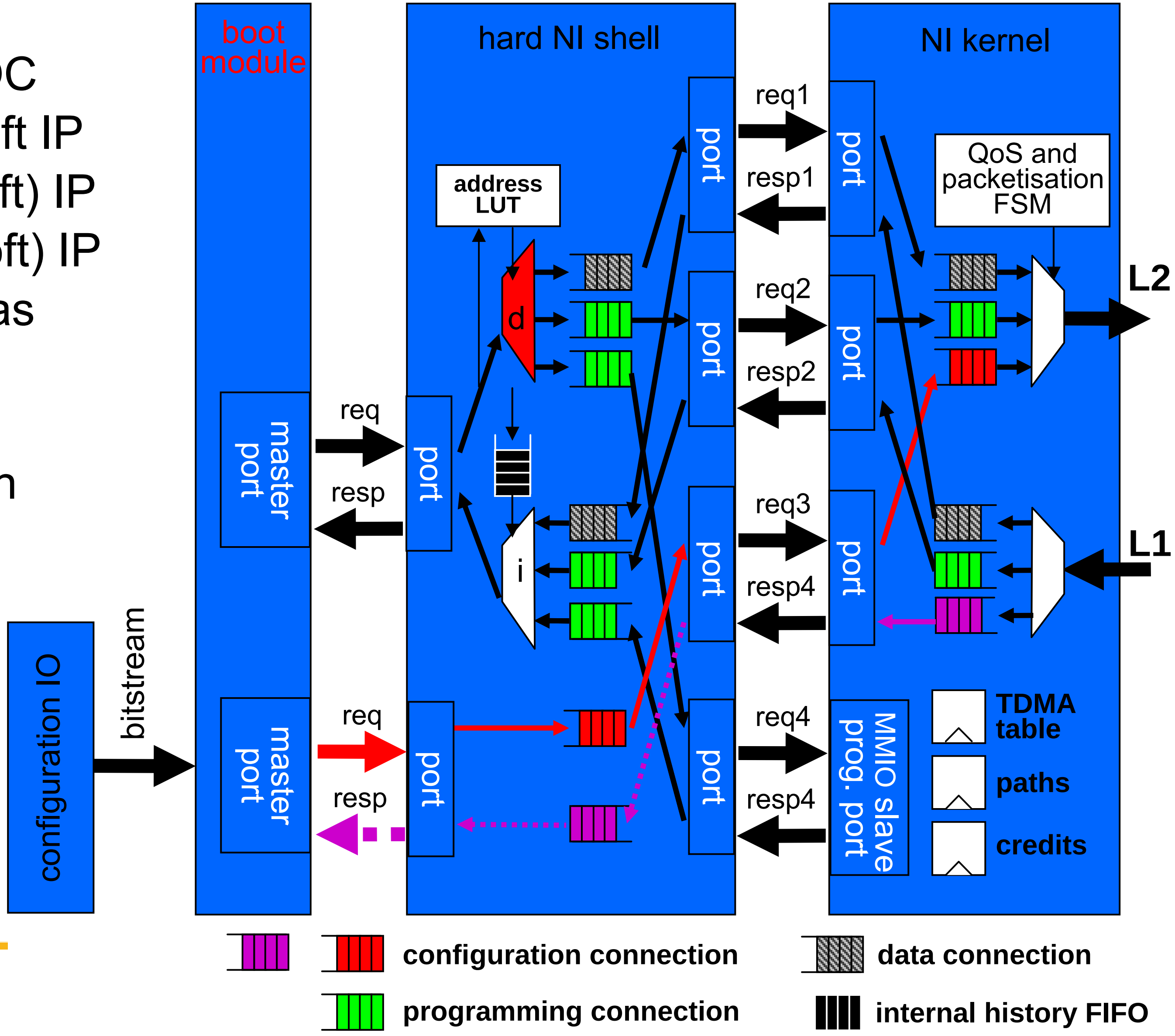
- “functional mode”



configuration & programming

1. program the NOC
2. configure the soft IP
3. program the (soft) IP
4. send data to (soft) IP
5. use bitstreams as functional data

- (de)compression
- de/encryption
- synthesis
- optimisation
- etc.



performance & cost

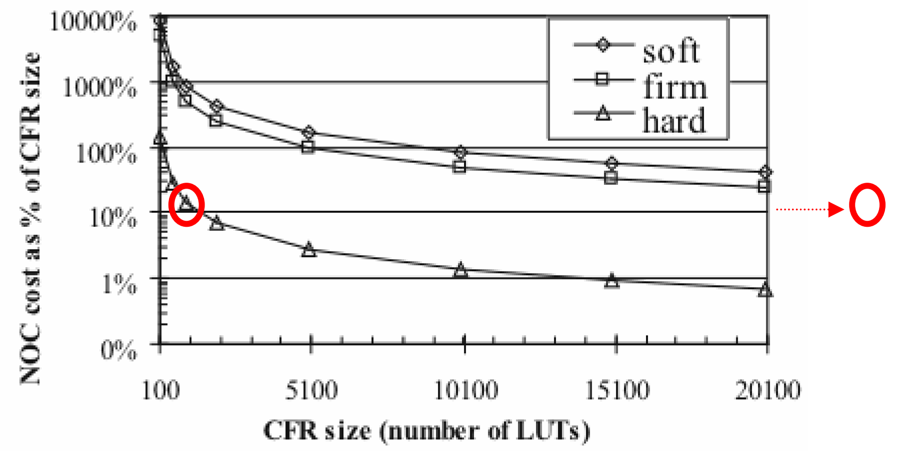
- compare
  - dedicated configuration interconnect & soft NOC, with
  - unified hard NOC
- on
  - area
  - functional speed
  - configuration footprint (bits)
  - configuration & programming time

performance & cost

- essentially, it all depends on
  - area soft:hard ≈ 35:1
  - speed soft:hard ≈ 3.5:1
  - configuration footprint of soft NOC (bitstream) : programming footprint of hard NOC (MMIO registers) ≈ 214:1
- resulting in
  - boot time soft:hard ≈ 1:200
  - functional performance:cost (bit/sec:area) soft:hard ≈ 1:147

performance & cost

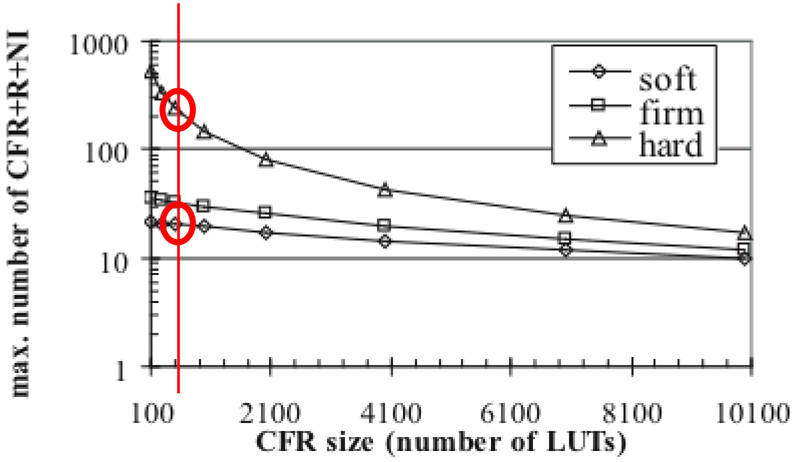
- percentage cost of router + NI kernel against CFR size (# LUTs)
- 10% area cost for interconnect implies CFR of
  ≈ 80,000 LUTs (soft NOC)
  ≈ 1,400 LUTs (hard NOC)



(a) Percentage overhead for varying CFR sizes.

performance & cost

- # CFRs against their size (# LUTs)
- # of CFRs of 1000 LUTs
  ≈ 19 CFRs (soft NOC)
  ≈ 144 CFRs (hard NOC)



(b) Number of IPs on a Virtex-4 for varying CFR sizes.

performance & cost

- speed
  - 118MHz unoptimised soft NOC (90nm)
  - 500MHz optimised hard NOC (130nm)

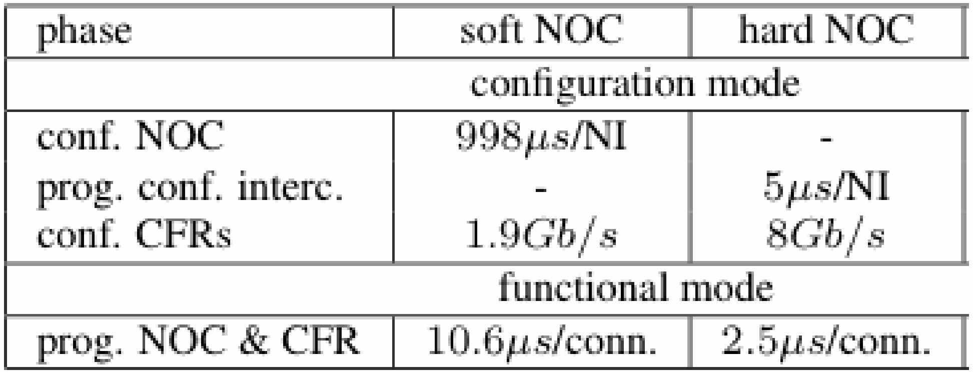
- performance:cost = bit/sec : area
  - 32 bit x 118 MHz / 8.10 mm^2 = 466 = 1 soft
  - 32 bit x 500 MHz / 0.23 mm^2 = 69,565 = 147 hard

performance & cost

- configuration speed
  - 1.9 Gb/s for dedicated configuration interconnect
  - 8 Gb/s for hard NOC
- programming speed
  - 118 MHz soft NOC
  - 500 MHz hard NOC
- configuration footprint for soft NOC
  - 1.8 Mb (8300 LUTs / router+NI)
- programming footprint for hard NOC
  - 2100 bit per connection
- thus
  - 1 ms per NI for soft NOC
  - 10.6 microsec per NI for hard NOC

performance & cost

- boot time
- dedicated configuration + soft NOC
  - configure soft NOC
  - configure IP
  - program soft NOC
  - program IP
  - total
- hard NOC
  - program soft NOC
  - configure IP
  - program IP
  - total



phase	soft NOC	hard NOC
configuration mode		
conf. NOC	$998\mu s$/NI	-
prog. conf. interc.	-	$5\mu s$/NI
conf. CFRs	$1.9Gb/s$	$8Gb/s$
functional mode		
prog. NOC & CFR	$10.6\mu s$/conn.	$2.5\mu s$/conn.

conclusions

- nice idea but:
  - low-level implementation of kernel:shell interface would be nice
    - investigate impact of mapping shell in LUTs
  - performance : cost analysis should be improved
    (it is in favour of soft interconnects)
- advantages of real-time NOC not explained
- “type casting” of bitstreams ⇔ control ⇔ data
  has many cool applications
  - run-time synthesis & modification of bitstreams
    - en/decryption, CRC check, peep-hole / JIT compilation of bitstreams
  - run-time relocation of soft IP
    - e.g. structural test engines,
      re-packing of soft IP on partial dynamic reconfiguration