Arteris FlexNoC IP

AN INTRODUCTION

XINYU LI

Application Engineering Manager
Xinyu.li@arteris.com

Arteris IP – The Leading SoC Integration IP Company

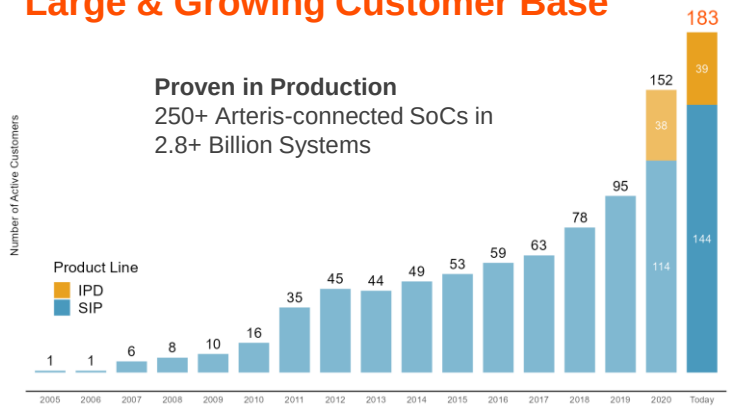
FOUNDED IN 2003 ; HEADQUARTERS IN SILICON VALLEY

Data is current as of 30 November 2021

Continuous Technology Innovation

FlexNoC®	2010	Main interconnect, 2nd gen
FlexWay™	2010	IP subsystem interconnect
FlexPSI	2013	All-digital interchip link
FlexNoC Resilience	2014	Resilience for ISO 26262
FlexNoC Physical™	2015	Links to physical SP&R
Ncore®	2016	Cache coherent interconnect
PIANO®	2017	Automated timing closure
CodaCache®	2018	Independent last level cache
AI Package™	2019	Machine learning interconnect
Ncore 3	2020	CHI & ACE cache coherency

Large & Growing Customer Base



Global Presence

Top Semis use Arteris IP

Publicly Disclosed Customers

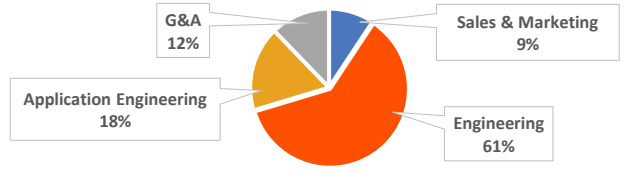
Achronix, Canaan, MaxLinear, picocom, supergate, 炬芯 Actions, Corerain 眸芯科技, MegaChips, rebellions_, Texas Instruments, Dream Chip, METAX 沐曦集成电路, RECOGNI, Altera now part of Intel, Enflame 燧原科技, Microchip, Renesas, UHNDER, Analog Inference, ESWIN, mobileye An Intel Company, Rockchip 瑞芯微电子, arbe, flexlogix AI + eFPGA, mobilint, SAMSUNG, ULTIOHIP, ASICLAND, FURIOSA, Montage Technology, SEMIDRIVE 芯驰科技, UNISOC, AutoChips, GCT, Morningcore 辰芯科技, SEQUANS, 東京大学 The University of Tokyo, Autotalks, 国科 GOKE, Movidius an Intel company, siengine 芯擎科技 Technology, AXIADO, HAILO, 神州龙芯 Montage Technology (Shanghai) Co., Ltd., SiFive, V-SILICON, Baidu 百度, Hisense, NationalChip, Silicon Mobility, 壁仞科技 Biren Technology, Horizon Robotics, NEUREALITY, SiMa.ai, 瀚博半导体 Vastai Technologies, BITMAIN, indie, nextchip, socionext, BLACK SESAME TECHNOLOGIES, INNOSILICON, NTT Electronics, Socle Faraday Technology Group, vayyar, BlueOcean smart, intellifusion 云天励飞, 新岸线 NUFRONT, SPEEDATA, BOSCH, INUITIVE, nuvoTon, ST life.augmented, VeriSilicon, CALTERAH, KYOCERA, NXP, StarFive 赛昉科技, Cambricon 寒武纪科技, Lynxi 灵汐科技, octasic, 希姆计算, VITEC

In Leading Systems

Connected by Arteris Ecosystem

arm, SYNOPSYS, tsmc, SAMSUNG, cadence, ISO 26262, exida, RESILTECH, CXL Compute Express Link, CCIX

Interconnect Technology Think Tank



Active NoC IP Customers – Public

Automotive & Transportation

- Major Auto Tier-1
- Major ADAS System Maker
- Major Automotive EV OEM
- Leading Auto EV Supplier
- Major Automotive OEM
- Major FPGA Company

AI / Machine Learning

- Stealth AI Company
- Major AI Company
- Major Memory Vendor
- Major Internet Company
- Horizon Robotics
- AI Research Institute

IoT, CE & ASIC

- DTV-STB SoC Maker
- MCU Maker
- Major Surveillance Camera OEM
- Japan Camera OEM
- Major System OEM
- Japan System OEM
- Large Drone Maker
- GPU Vendor
- IoT SoC Maker
- Home Appliance Company
- Smartphone OEM
- GPU Company
- Stealth CE Maker
- Very Large SoC Maker

5G and Mobility

- Stealth 5G Company
- Very Large SoC Maker
- Mobile Telecom Company
- Telecom Company
- Smartphone OEM
- Major 5G Infra Vendor
- 5G Infra Vendor

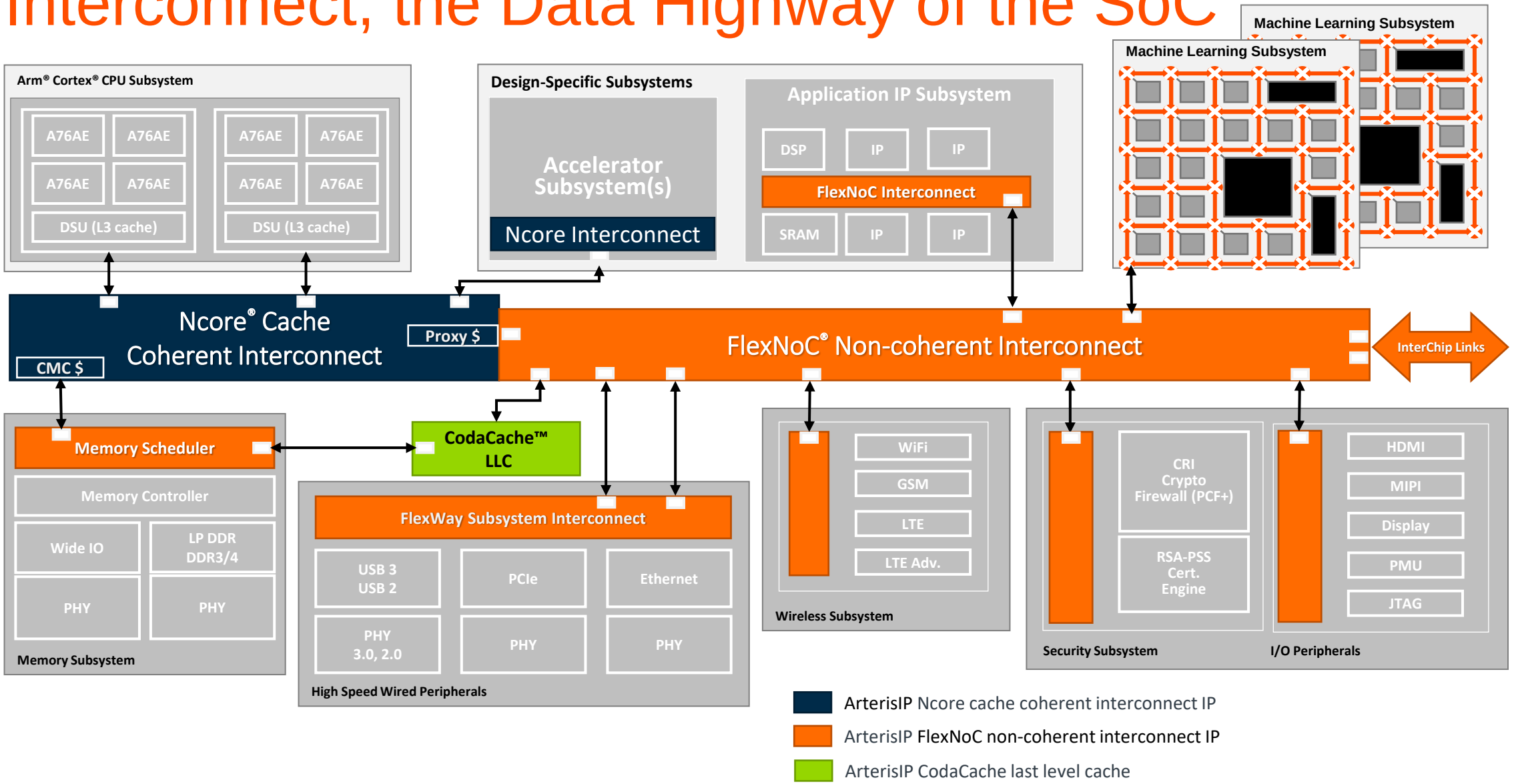
Server, SSD, Networking & Automation

- Enterprise Networking
- Major SSD Vendor
- Major Internet Company
- Cloud and Edge Servers
- Imaging Company
- Major FPGA Company
- Major Industrial OEM

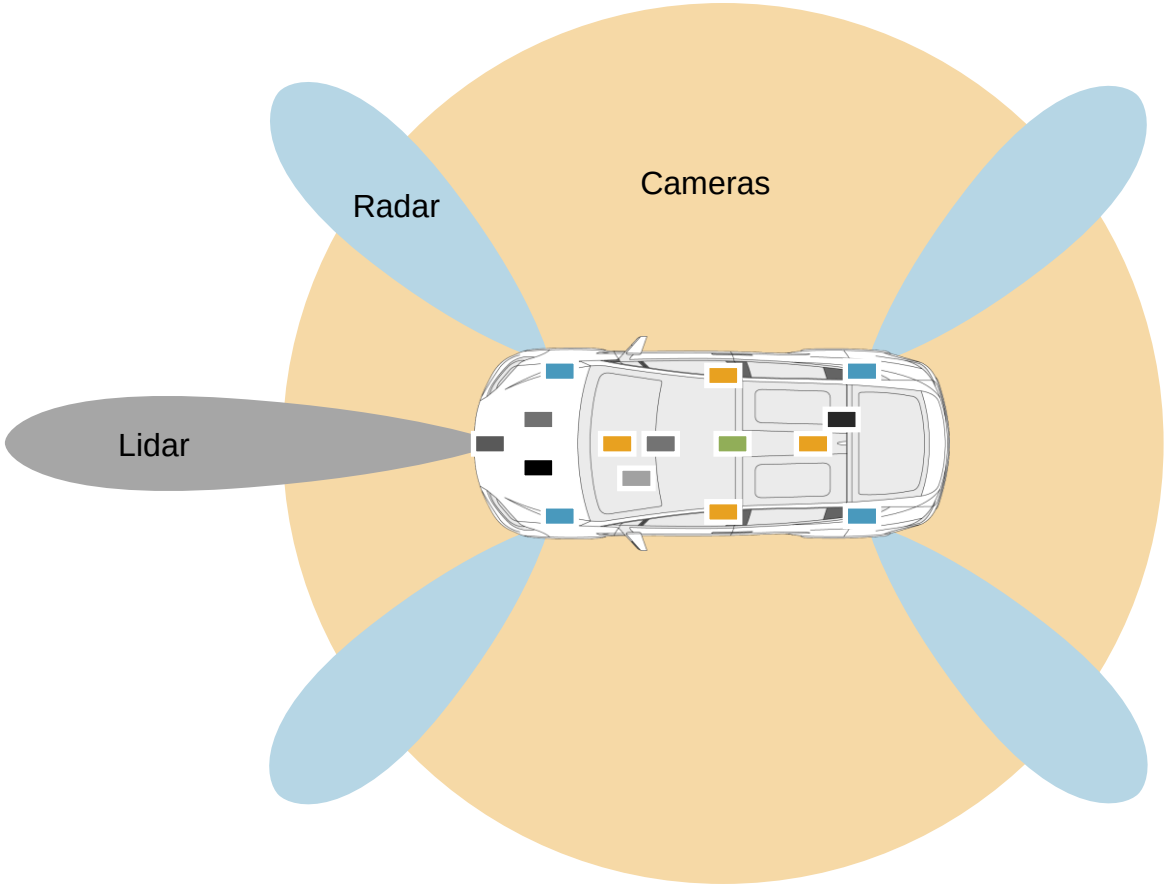
NXP is a trademark of NXP B.V.

**Logos and customer names include Arteris IP users as of 30 November 2021*

Interconnect, the Data Highway of the SoC



Arteris IP Focused on Automotive & Machine Learning



Source: Arteris IP

Vision Camera (4-16) **(4)**	Mobileye, Toshiba, Texas Instruments, nextchip
ADAS / Machine Learning (1-4) **(2)**	Mobileye, Not Public, NXP, STMicroelectronics, Dream Chip, Cambricon, Movidius, AutoChips, SemiDrive
Dashboard / HUD **(2)**	NXP, Renesas
Infotainment **(1)**	NXP, Not Public, Texas Instruments
Radar and/or LIDAR (4-6) **(6)**	Not Public, NXP, arbe Robotics, vayyar
Chassis Control **(4)**	NXP
Engine Control **(2)**	STMicroelectronics
V2X / V2I / WAN Modem (2-3) **(3)**	Sequans, Not Public, Autotalks

Notes: Numbers in parentheses are the number of "complex" SoCs per function. Logos and company names are publicly announced Arteris customers as of 1 July 2019.

What is an ASIL (Automotive Safety Integrity Level)?

	When	ASIL B	ASIL C	ASIL D
SPFM Single Point Fault Metric	Operating	> 90 %	> 97 %	> 99%
LFM Latent Fault Metric	Key-on	> 60 %	> 80 %	> 90 %
FIT Failure in Time	Operating	< 100	< 100	< 10

Definitions

Single Point Fault Metric (SPFM) - % coverage by safety mechanisms

Latent Fault Metric (LFM) - % coverage by safety mechanisms of multi-point faults

Failure in Time (FIT) - # of expected failures in one billion hours (114,155 years)

Ramifications

Hardware protection in SoC interconnect (rules of thumb)

- **ASIL B** = fault detection (ECC/parity, SW)
- **ASIL C/D** = unit duplication for key logic

Built-in Self Test (BIST) and **checkers** required for HW safety mechanisms!

Certificate / Certificat
Zertifikat / 合格証

TI 1309037 C002

exida hereby confirms that the:

TDA3X ADAS SoC

Texas Instruments, Inc.
Richardson, TX - USA

Has been assessed per all relevant requirements of:

ISO 26262 : 2011

and meets requirements providing:

Systematic Integrity: ASIL B

Safety related function:
The SoC supports the execution of safety-related software in a multi-core architecture.

Application restrictions:
The end-user of this product must strictly adhere to all instructions in the Safety Manual for the device.

The manufacturer may use the mark:

Revision 1.2 August 3, 2017
Surveillance Audit Due
August 1 , 2020

ANSI Accredited Program
ISO/IEC 17065
PRODUCT CERTIFICATION BODY
#1004

exida Certification

Evaluating Assessor

Certifying Assessor

TDA3X ADAS SoC

Certificate / Certificat / Zertifikat / 合格証

TI 1309037 C002

Systematic Integrity: ASIL B

Systematic Integrity: ASIL B
The product is a Safety Element out of Context (SEooC) per ISO 26262:2012-10. The development project, as documented by Texas Instruments, has met the relevant ASIL B design, verification and validation requirements of ISO 26262:2011, parts 2, 5, 7, 8 and 9, as guided by 26262:2012, part 10; and the functional safety management requirements per ISO-26262:2011-2. The product development was also found to comply with requirements of Annex F of IEC 61508-2 for integrated circuit development process.

Random Hardware Integrity and Component Failure Rates
End users are required to calculate failure rates for the part, to be used in item/element safety calculations, with an FMEDA tool. The Texas Instruments supplied tool is delivered with the product, and provides a means to calculate the failure rates based on specific application use and user-provided diagnostics.

Structure of the SoC and Safety (Integrity) Mechanisms
Safety Integrity of the part may be enhanced by utilizing some on-chip diagnostics (e.g., ECC, parity) and by configuring the operation of the chip to provide software based diagnostics. Enhancement may also be gained through diverse and/or redundant design, using external hardware and/or the multiple processors on the IC, as guided in the Safety Manual.

The use of the I/O parts and communications parts is constrained by the Safety Manual. If a use of the product in a safety application is to deviate from that suggested or required by the Safety Manual, the end user is required to consult Texas Instruments for appropriate advice.

The following documents are a mandatory part of certification:

Assessment Report:
TI Q1309-037 R005 V1R1 ISO 26262 Assessment Report VL-28

Safety Manual:
Safety Manual for TDA3x Automotive Vision Safety Critical Applications Processors

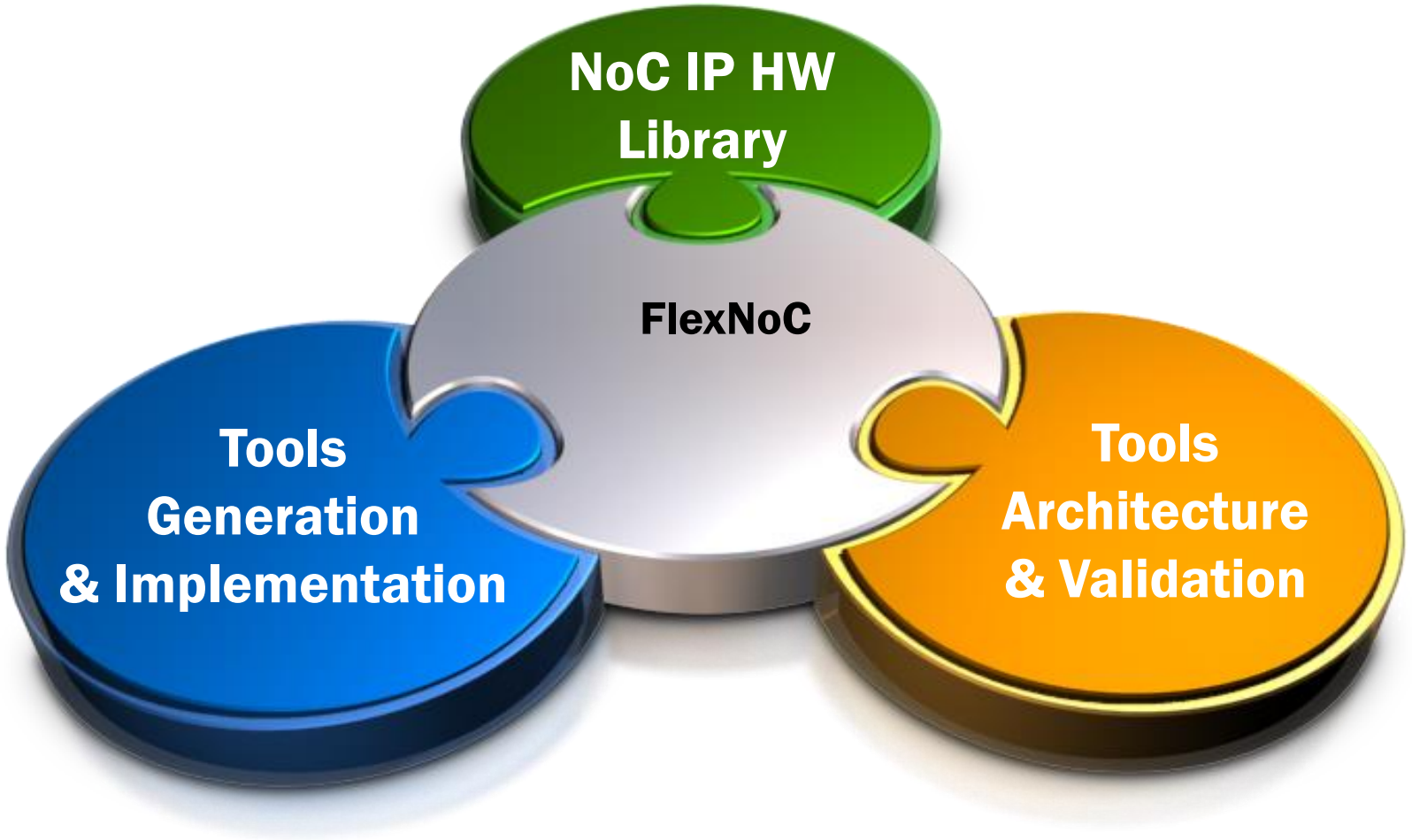
80 N Main St
Sellersville, PA 18960

T-103, V2R1

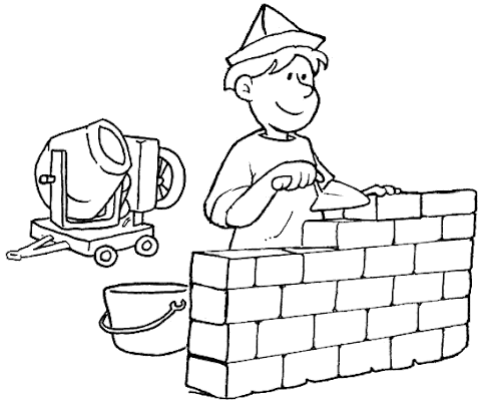
Products and Technology

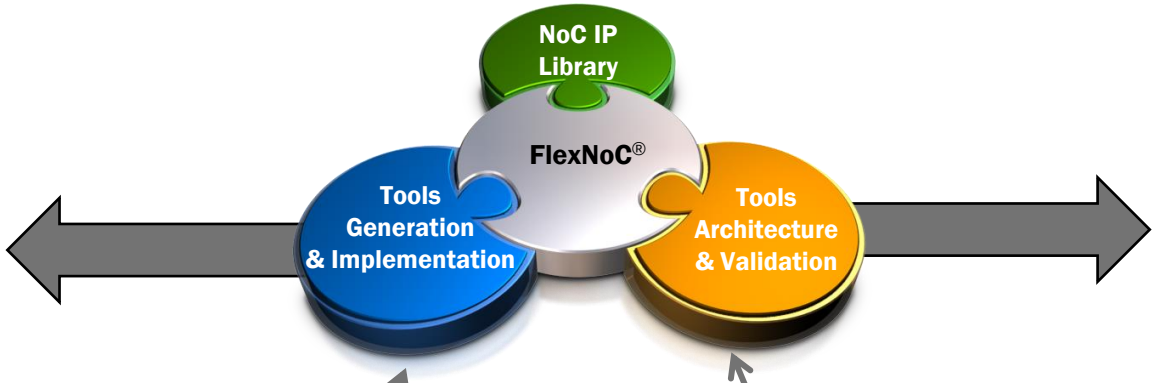
- Proven technology used in billions of SoCs
- 1-2 new products per year addressing SoC innovation
- 4-6 product enhancement releases/yr. for customer support
- Mature quality processes for trouble free deployment
- Deep technical interconnect IP expertise

Arteris FlexNoC – A complete product

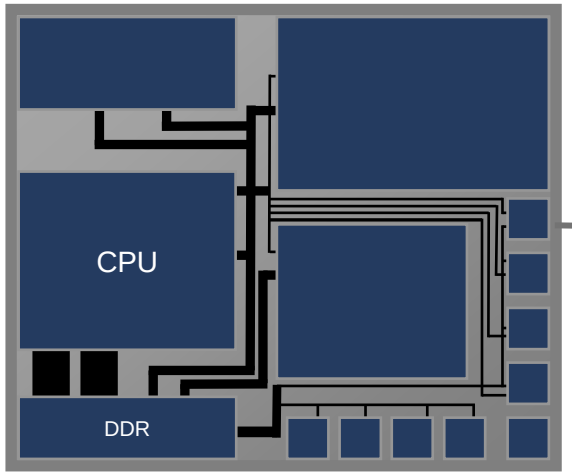


Chip designer and System architect

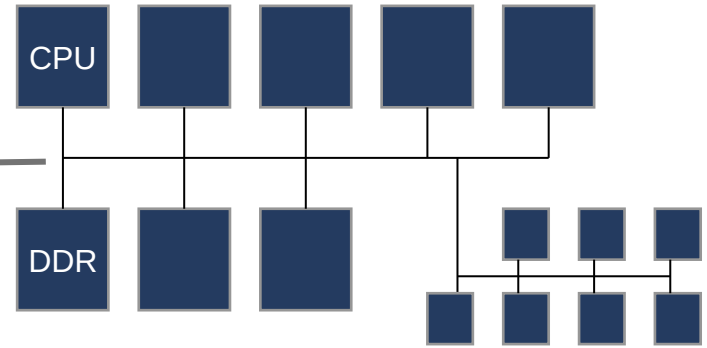




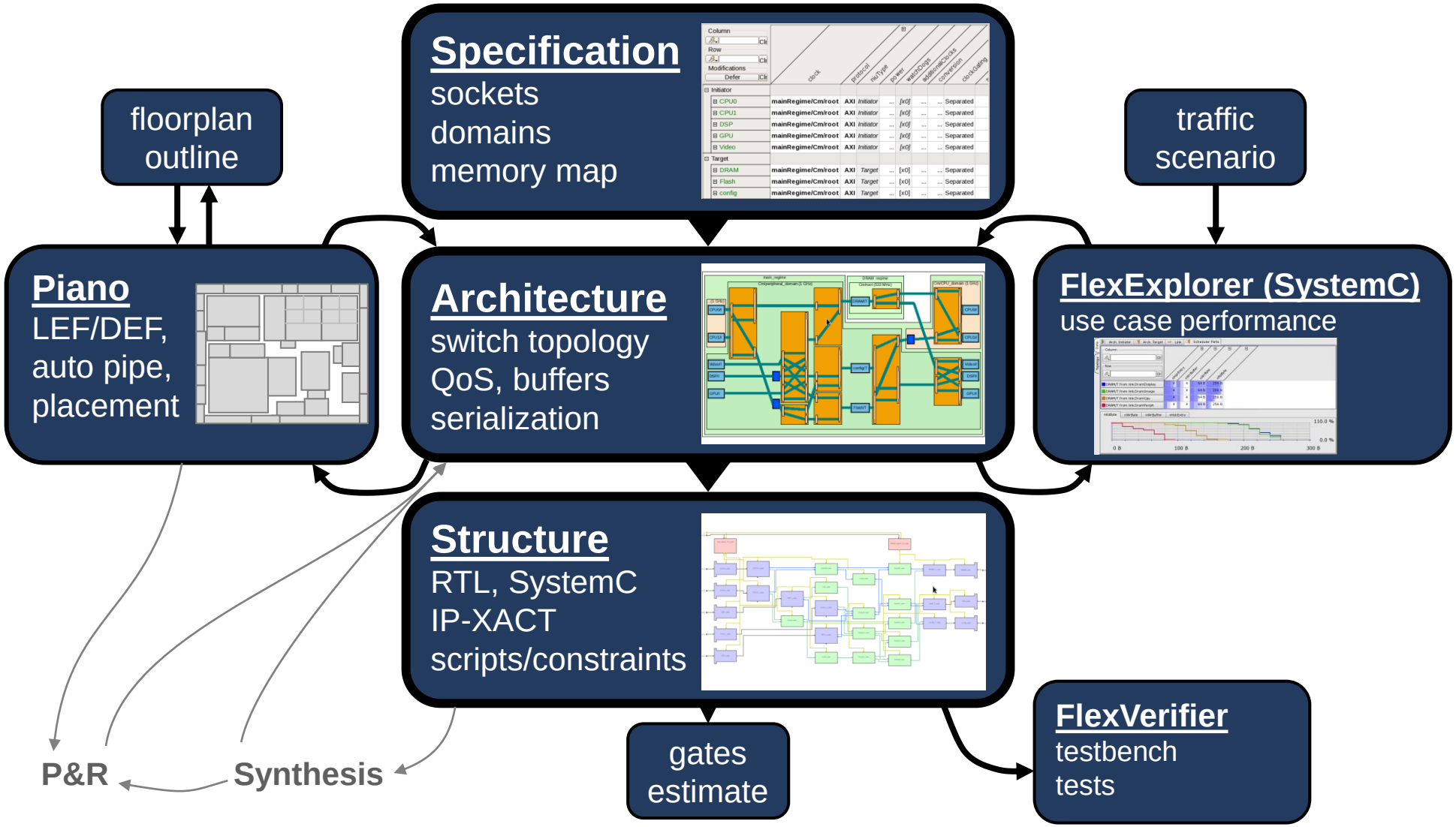
DRCs mW mm² MHz protocols latency MBps



- Wire density
- Power
- Timing closure



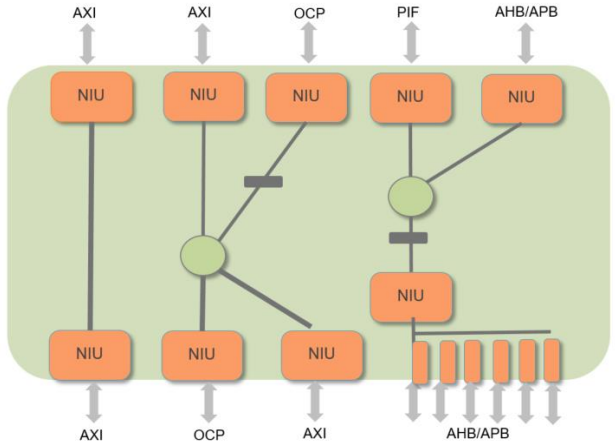
FlexNoC tool flow



Arteris NoC Technical Advantages

BASED ON 12 YEARS SUMMARY OF SOC BENCHMARK RESULTS FLEXNOC VS. OTHER

- **50% Less Wires**
  - Packetization
  - Packet Serialization
- **30% Less Area**
  - Simple Packet Transport Layer
  - Configure Area of CDC
- **Low Power**
  - Power Management Feature
  - 3 Levels of Clock Gating
- **Flexibility**
  - Multi-Protocol + Legacy support
  - Fully configurable architecture
  - PPA trade off, 0 latency path
  - Easy pipelining
- **Scalability**
  - 100+ IPs
  - “Lego Box” NoC IP Library
- **Automation**
  - All in single GUI
  - Development time saving
    - Automatic test bench
    - Modeling platform
    - Area estimation
- **Integration**
  - System Features
    - End-to-end QoS
    - Power
    - Debug
    - DDR scheduling
    - Security
    - Resiliency for Functional Safety



Best Way to Assemble IP Blocks into an SoC

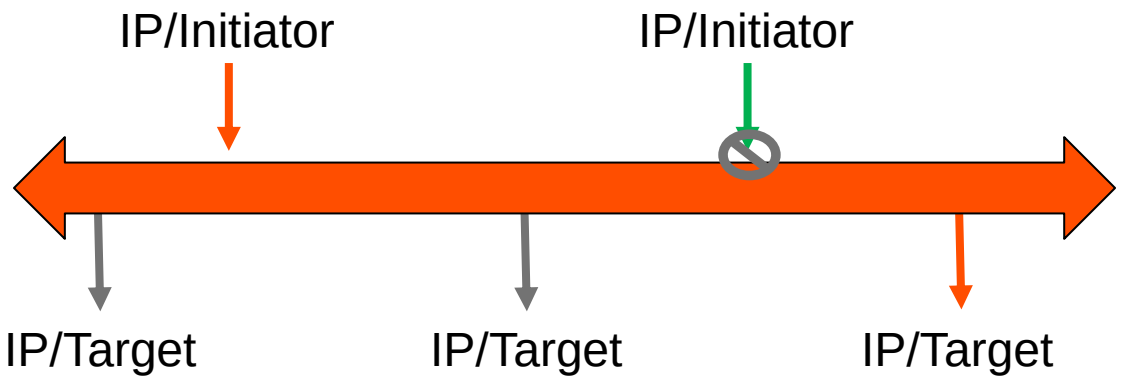
- **Flexibility and Productivity with market leading PPA**
- **Broadest interconnect IP product portfolio**
- **Continuous Interconnect Technology Delivery**
- **Experienced, Responsive Global Support**
- **Quality, Proven Products shipping in billions of systems**

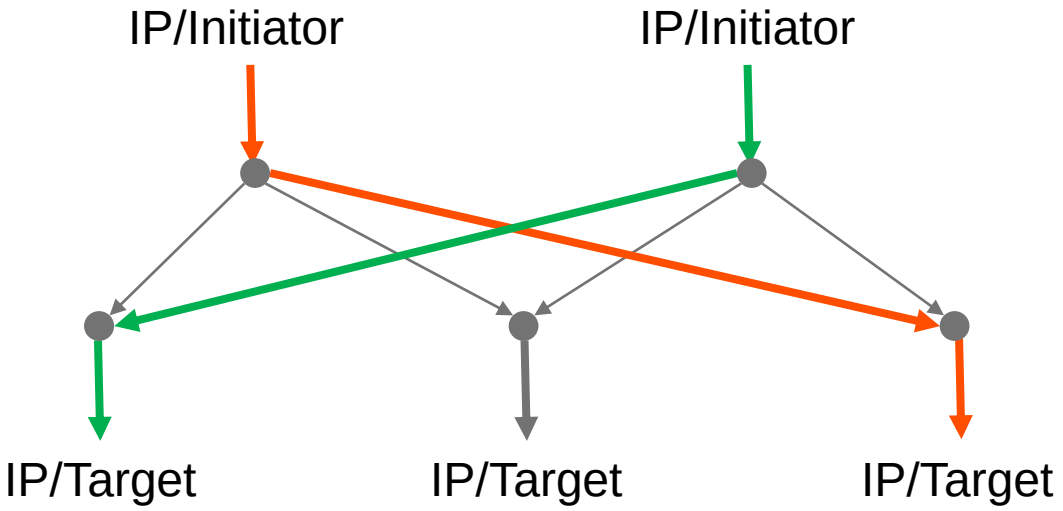
Make more complex, safer systems at lower cost.

NoC architecture overview

WHAT'S A NOC?

Bus or Network?



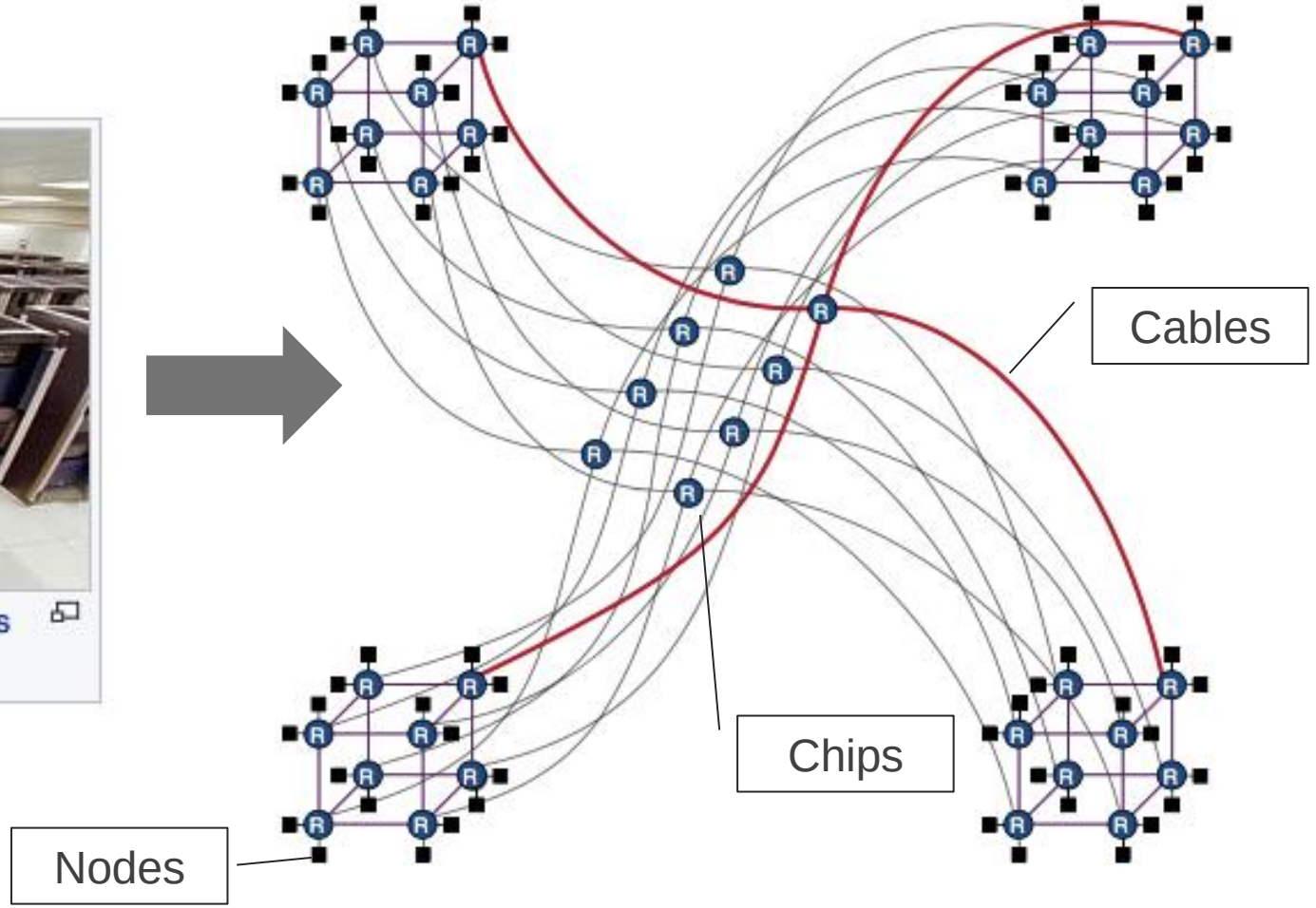


- **Buses** are:
  - Single, shared communication resources
    - Everything is serialized
  - Broadcast from selected source to all destination
  - Arbiter selects which source owns the bus
  - Address decode selects which destination shall listen to request
  - example: PCI
- **Networks** are:
  - Point-to-point communication resources
    - Can have multiple communications in parallel
  - Switches manage traffic
    - Routing to destination
    - Arbitration in case of conflicts

Networks have a long history in computers

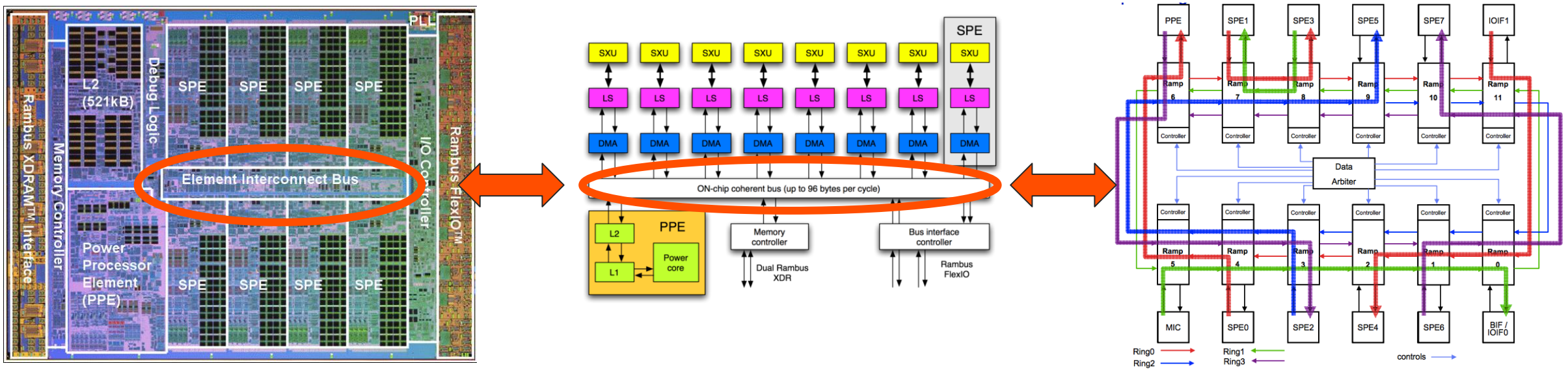
There is a network inside every massively parallel supercomputer (e.g., SGI Origin):

The Origin 2000 known as ASCI Blue Mountain at the Los Alamos National Laboratory in 2001



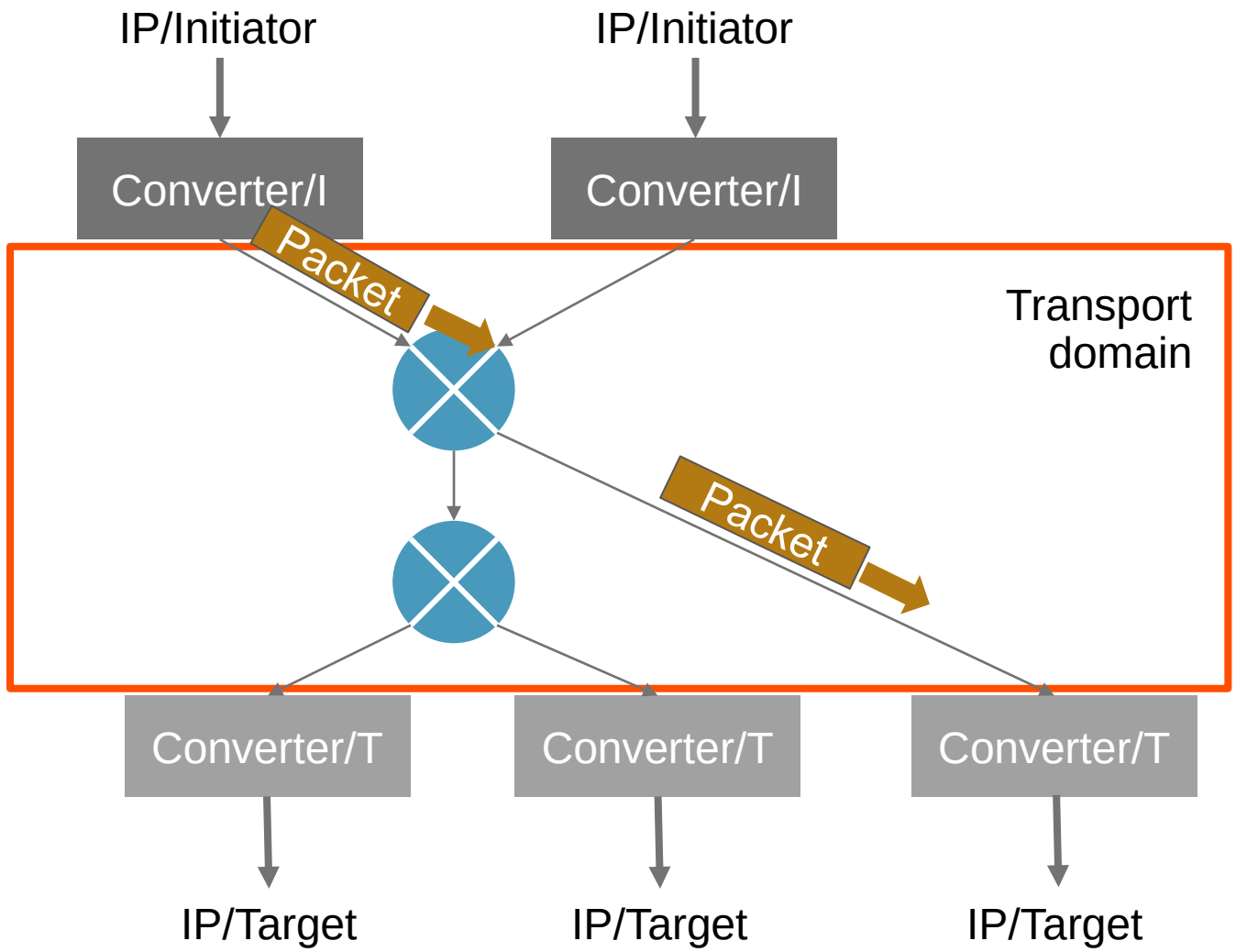
Network-on-Chip

A network contained inside an IC, built with digital logic instead of chips, cables, cabinets, etc.



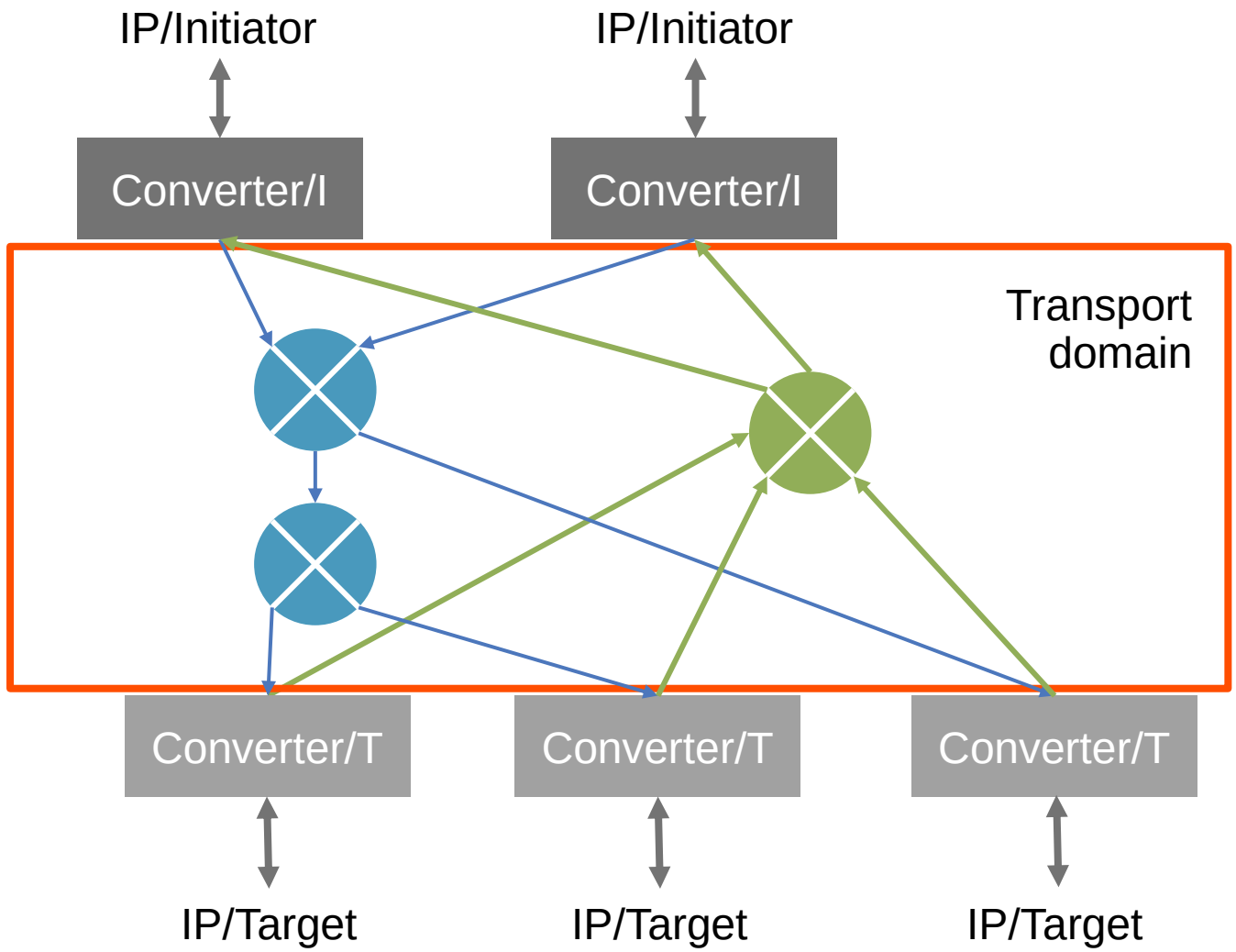
Example: PlayStation 3 Cell processor uses 4 rings to transfer data between processing elements

NoC architecture overview



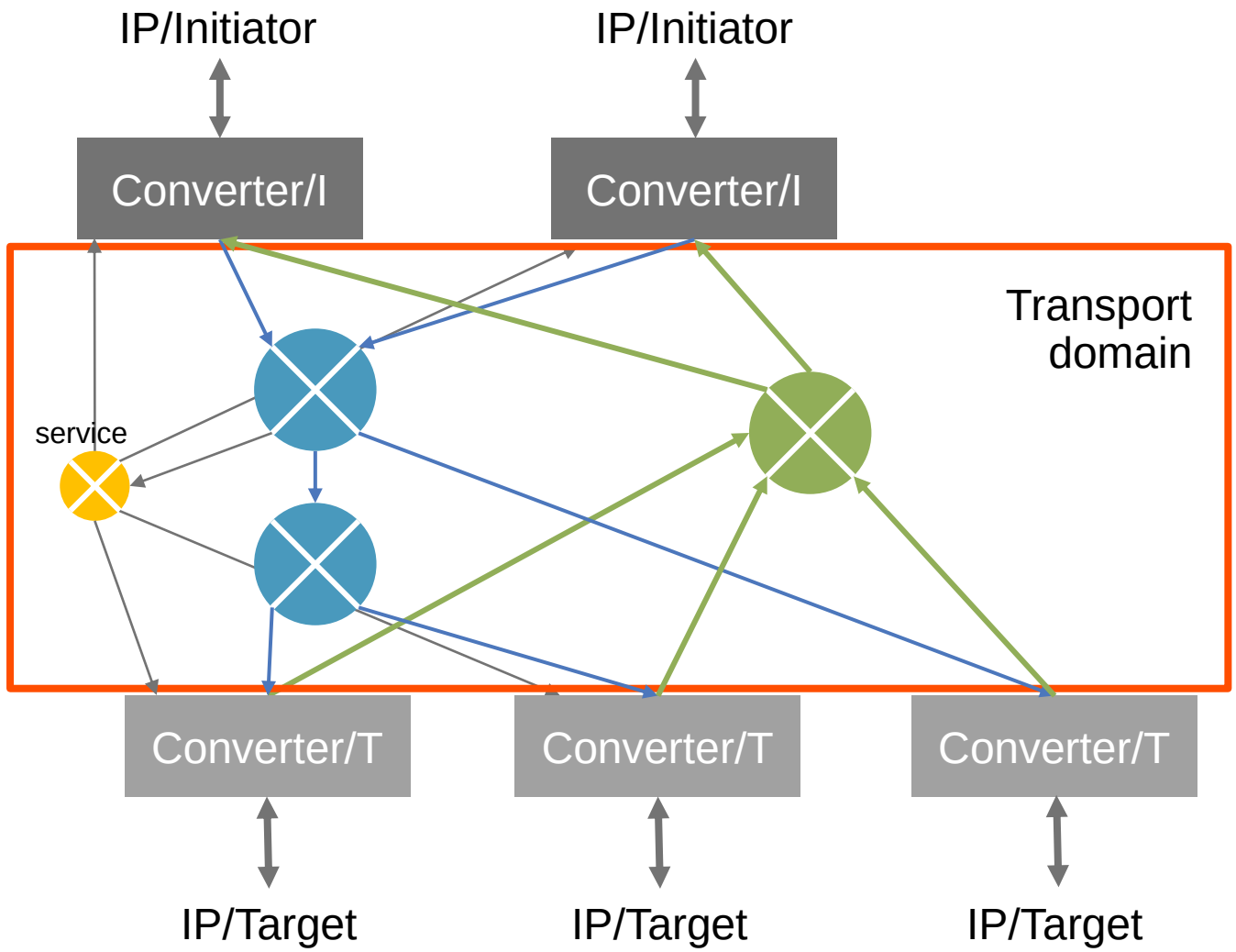
- Network-on-Chip function
  - Transports **requests** from **Initiators** to **Targets** and responses back
  - Initiators: CPU, GPU, DMA…
  - Targets: SRAM, DDR, registers…
- Component breakdown
  - The **transport** is built using **switches**
  - The switch arrangement is the **topology**
  - Often, the **transport uses its own protocol**
    - Fine-tuned for distance spanning, efficiency
    - Transport atoms are called **packets**

NoC architecture overview - 2



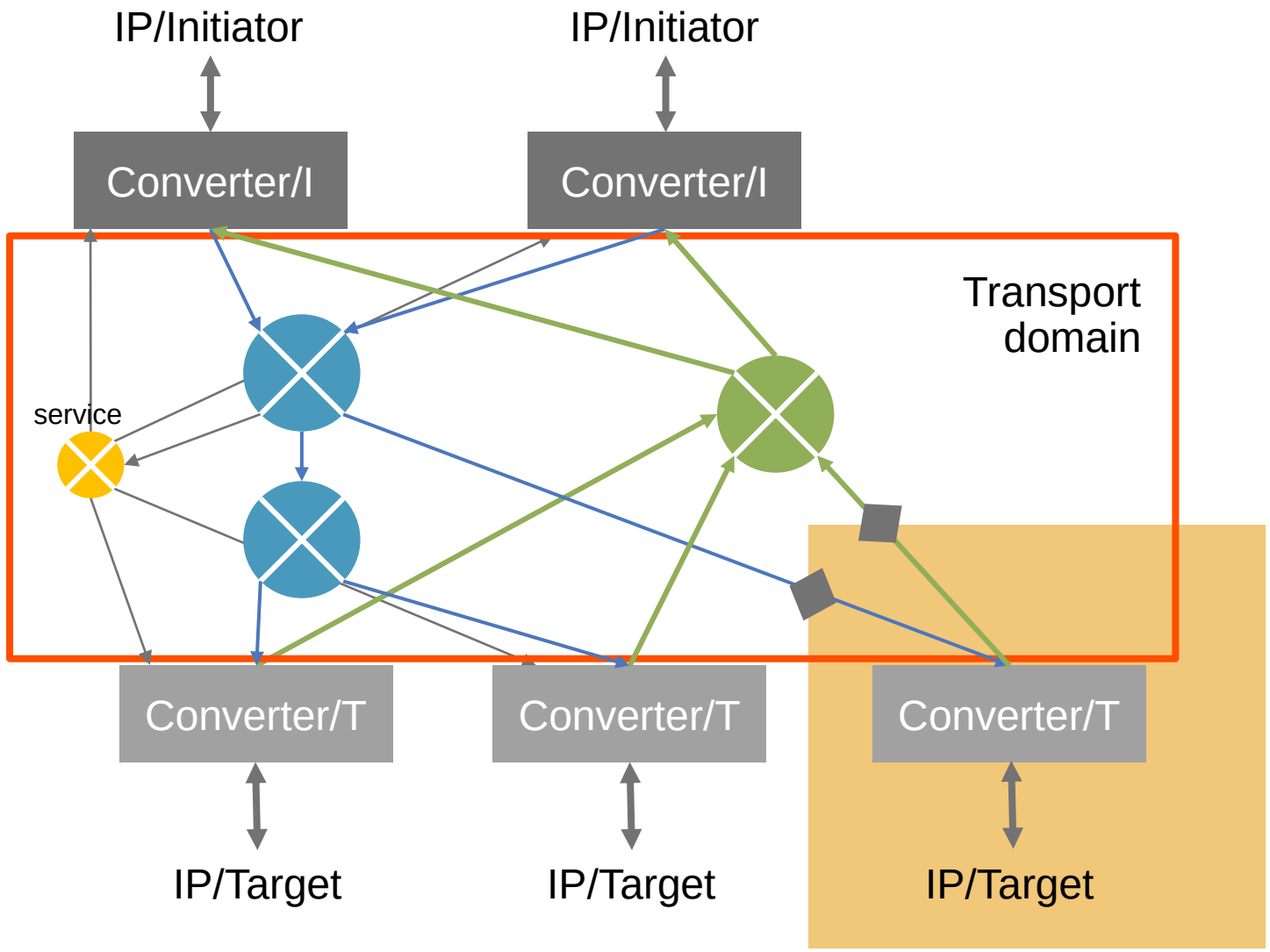
- If transport uses its own protocol
  - Protocol converters exists at the edge
    - Convert IP protocol to transport protocol and vice versa
- Typically transport two major flows of packets
  - Requests packets (I>T)
  - Response packets (T>I)
- Typically on **2 different networks**
  - Avoid circular dependencies = deadlocks
  - Or using Virtual Channels (more on this later)

NoC architecture overview - 3



- Other kind of networks may exist
  - Debug, **Service**, etc.
- Other type of packets may exist
  - Credits, QoS service, etc.

NoC architecture overview - 4

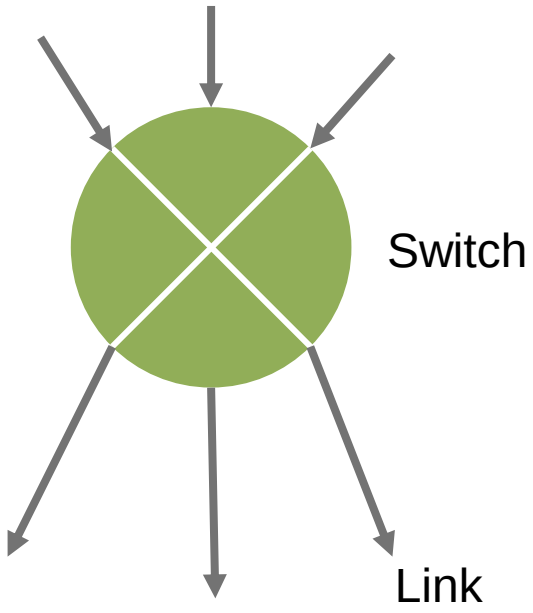


- The NoC can be divided into domains:
  - Power domains:
    - different power supply
    - Can dynamically transition ON<>OFF
  - Clock domains:
    - Synchronous divide of a reference
    - Synchronous subset of a reference (enable)
    - Asynchronous (arbitrary phase and frequency)
    - Mesochronous (same frequency arbitrary phase)
  - The NoC does the **adaptation**

TOPOLOGY

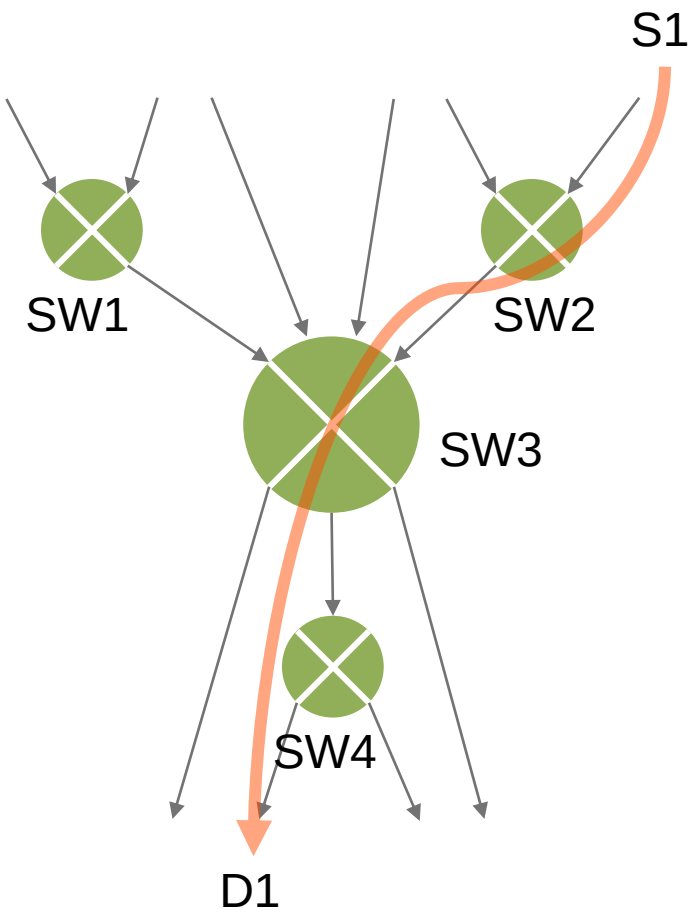
SWITCHES, LINKS, ROUTES

Definitions – Switch & Link



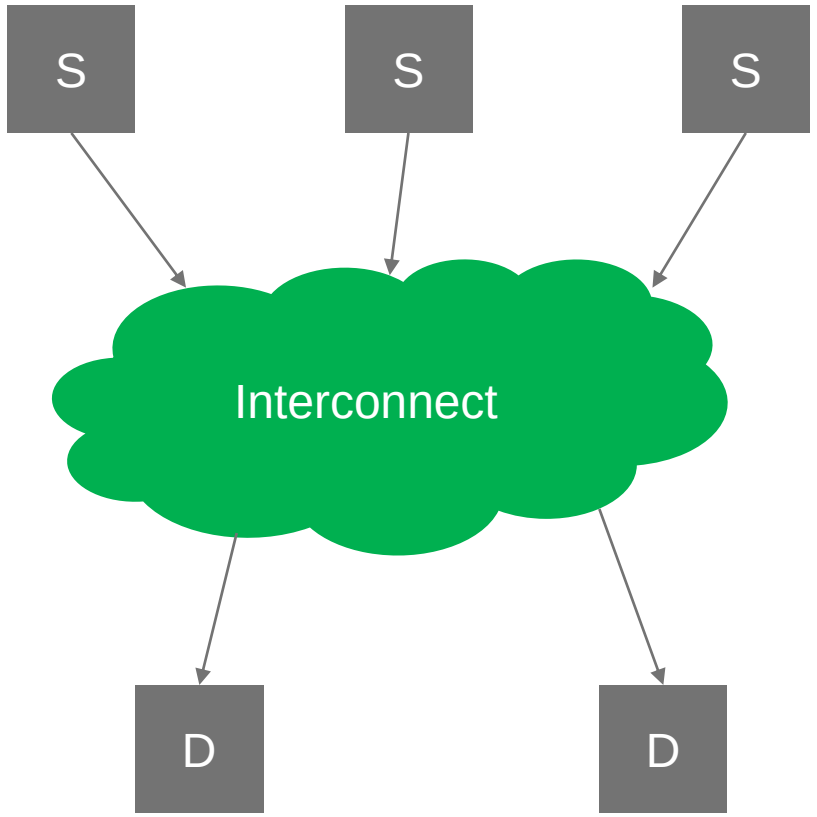
- A **switch** is a logic function that can route the traffic from I inputs to O outputs
  - It performs routing: Mapping a request from its input ports to an output port according to a mapping function
  - It performs arbitration: Resolution of conflicts when multiple input ports need to access the same output port
  - More details on the switch structure later
    - For now it's a black box
- A **link** connects a switch output port to the next switch input port
  - Protocol converters at the edge of the network are connected to switches using links as well
- Switches and links carries **packets**

Definition: Route



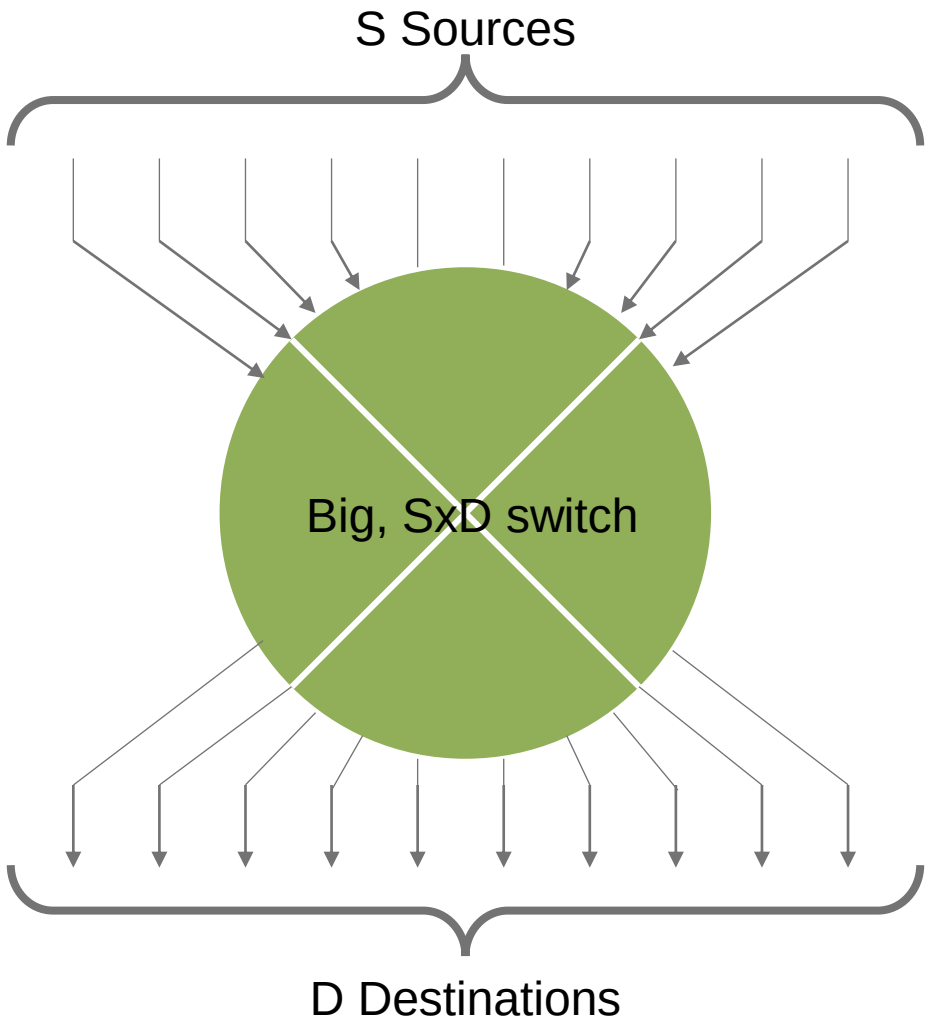
- A **route** is an ordered list of network elements between a source and a destination
- Example:
  - From S1 to D1 the route is {SW2, SW3, SW4)
- The network is performing packet routing:
  - Calculation of a route for each packet wishing to reach a destination
- The list of switches + the routing function = the topology
  - Important consequence: The links are therefore implicit!

How many ways to implement an interconnect?



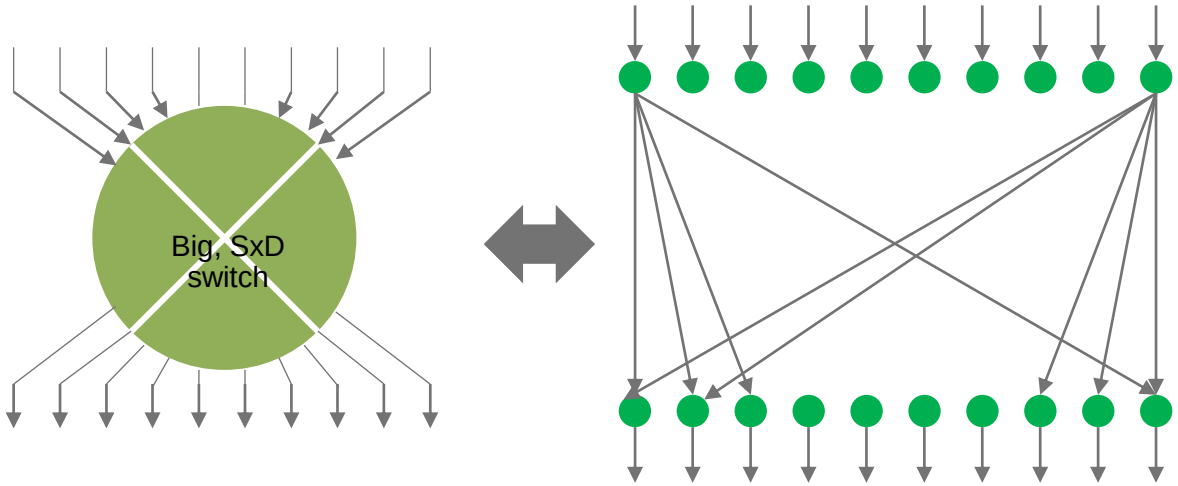
- The problem:
  - We need to connect S sources to D destinations using a set of switches and links
  - Without lack of generality let's assume all S need to communicate with all D
  - What is the optimal interconnect implementation, knowing:
    - The amount of traffic between each S and each D?
    - The location on the chip of all S and all D?
  - Optimal interconnect implementation means minimize:
    - Amount of logic in switches
    - Amount of wiring between switches
- Solving this problem is finding an optimal topology!

Why everything is not just a big switch?



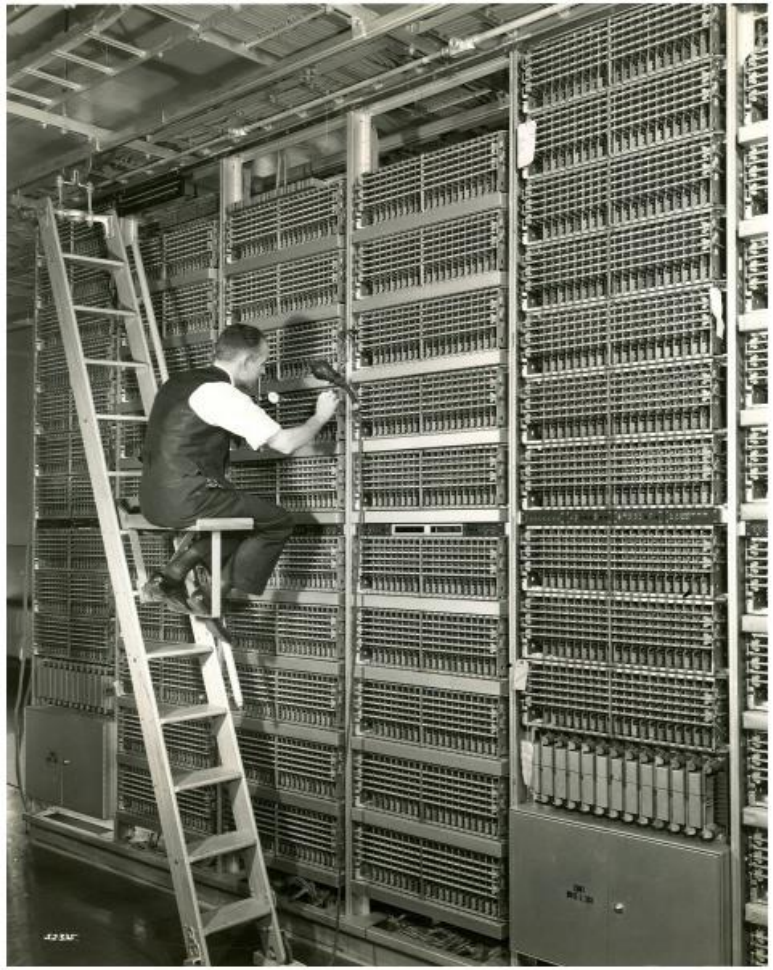
- Simplest solution: Use a S x D switch!
- Problem: It does not scale well!

Cost of a “big” switch



- A S x D switch can be decomposed in:
  - S switches of size 1 X D
    - Also called “splitters”
  - D switches of size S x 1
    - Also called “mergers”
  - S x D wires
- This topology is called a **crossbar**
- Examples:
  - A 10 x 10 crossbar has:
    - 10 switches 1x10; 10 switches 10x1; 10x10 = 100 wires
  - A 20 x 20 crossbar has:
    - 20 switches 1x20, 20 switches 20x1, 20x20 = 400 wires
- The cost of a crossbar is **quadratic**

Big crossbars implementation challenges



#1 crossbar switch, East 30th St., New York, New York, 1938.

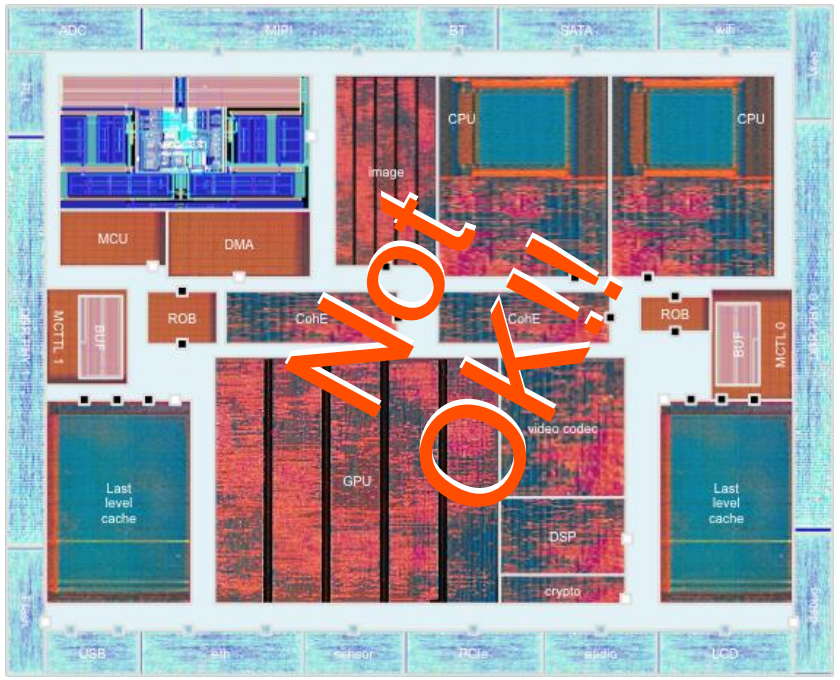
- Big crossbars use a lot of silicon real estate
  - Number of wires increases as S x D
  - Wires → signal integrity buffers → gates
- Crossbars require careful layout
  - Often need to dedicate a location in the middle of the chip

Big crossbars and layouts

Big cross bar can be in the middle

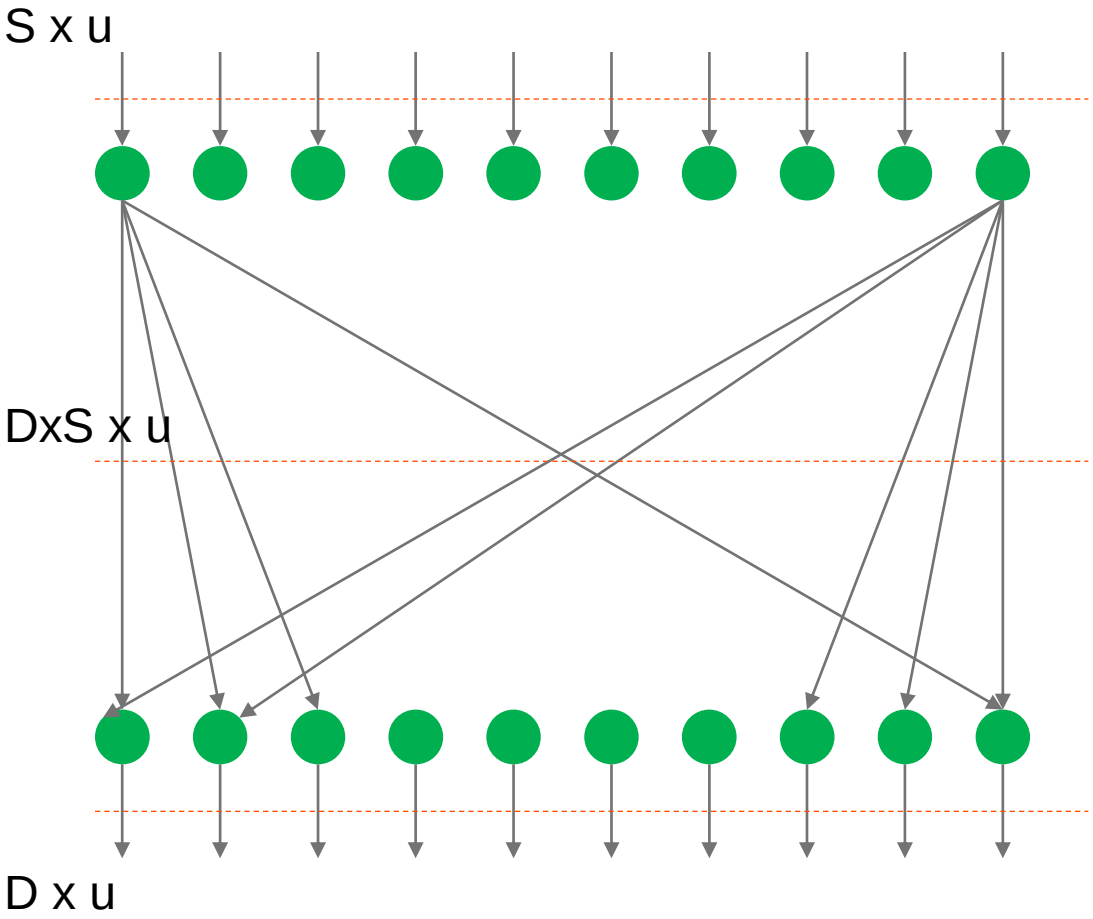
OK (kind of)

There is already something in the middle



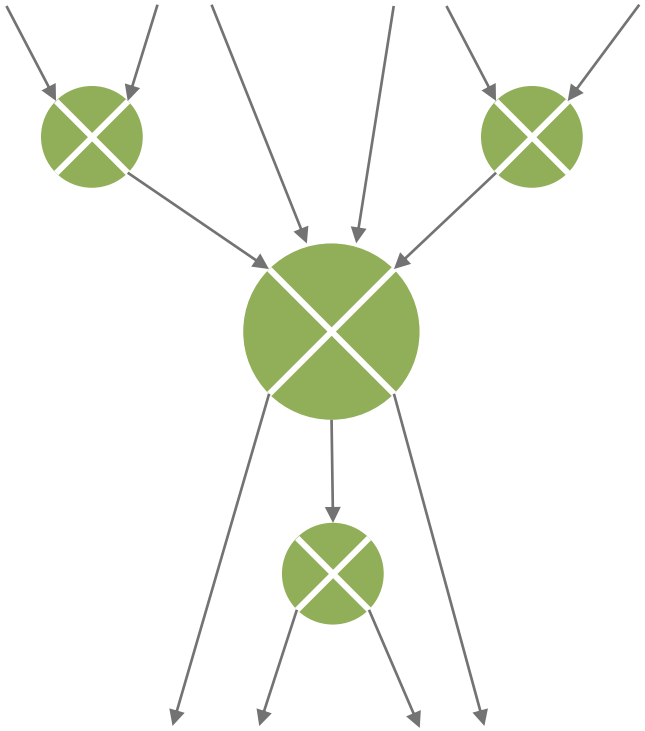
Don’t even try

Big Cross Bar inefficient use of wires



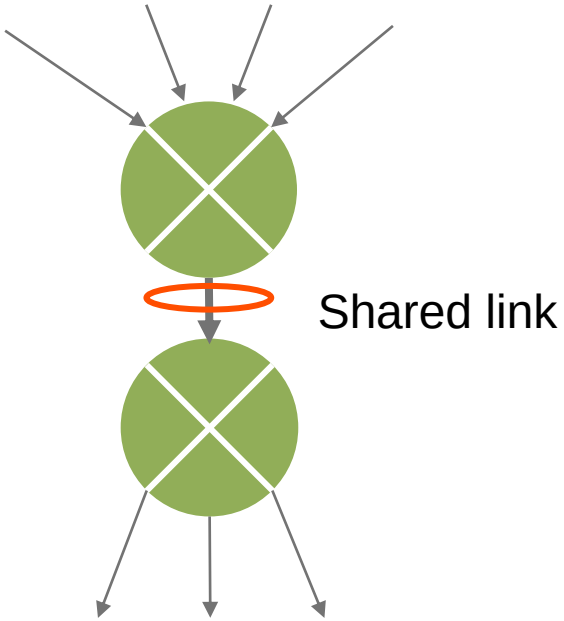
- Cross-section bandwidth: How much bandwidth goes through links when the interconnect is cut “in the middle”
- If we say that each source S and each destination D has the same link width ad run at the same clock
  - Carries a bandwidth of “u”
- Network must transport min(S x u, D x u)
- But cross-section bw is S x D x u
- S x D x u >> min(S x u, D x u)
- Conclusion: A huge majority of crossbar wires do nothing!

Reducing the size of the big crossbar



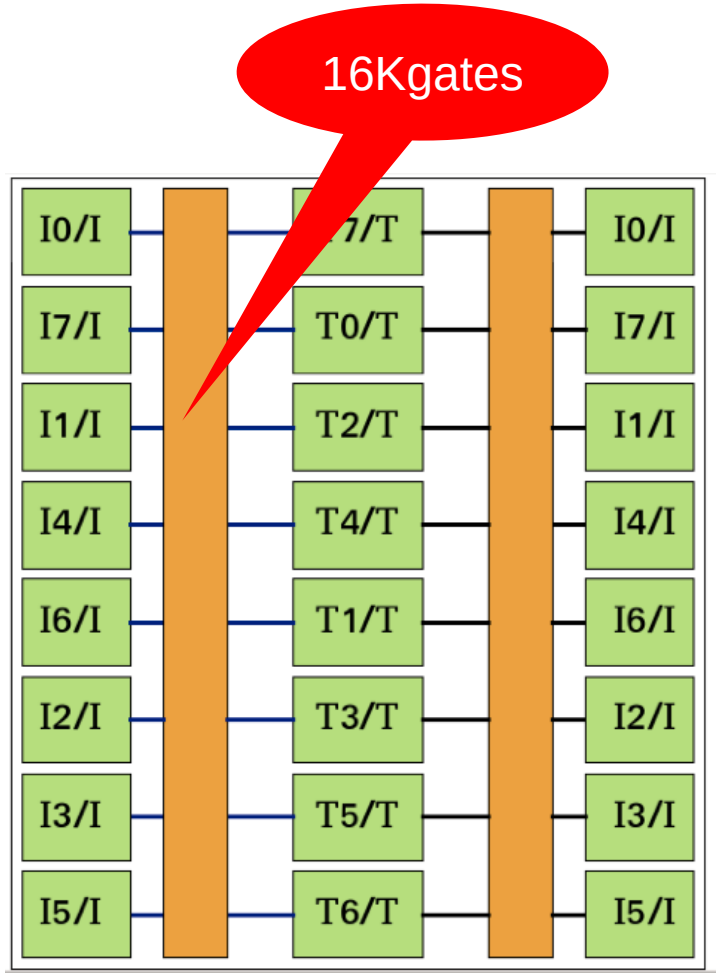
- **Combining traffic from masters as early as possible**
- **Keeping traffic to slaves combined as long as possible**
- Goal: Reduce the dimension of the crossbar
  - Gain is fast since crossbar cost is quadratic
  - A 14x14 crossbar is ½ the size of a 20x20!
    - Combining 6 sources, and 6 destinations, already half the size!
- Of course, reduction in size also comes with a reduction of the maximum throughput
  - Paths that were parallel before becomes serialized
  - This is perfectly acceptable for most applications!

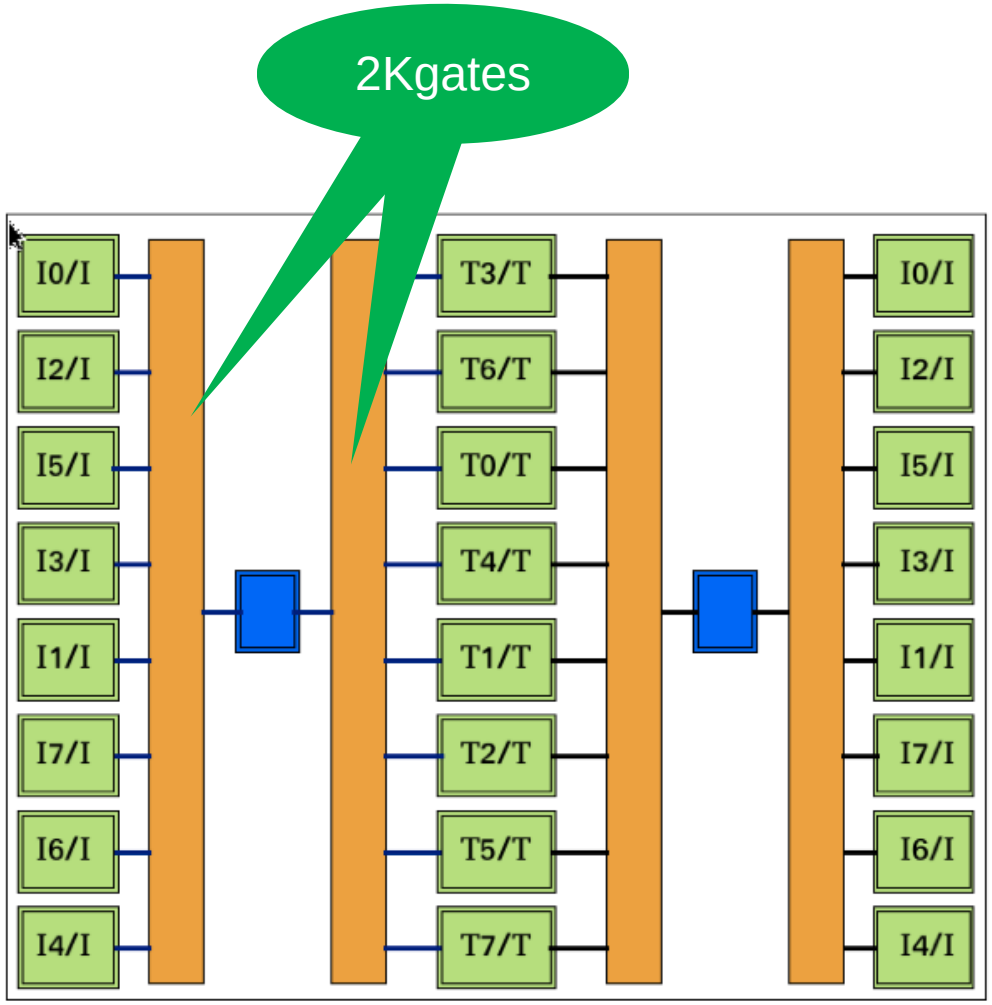
Minimizing the size of the big crossbar



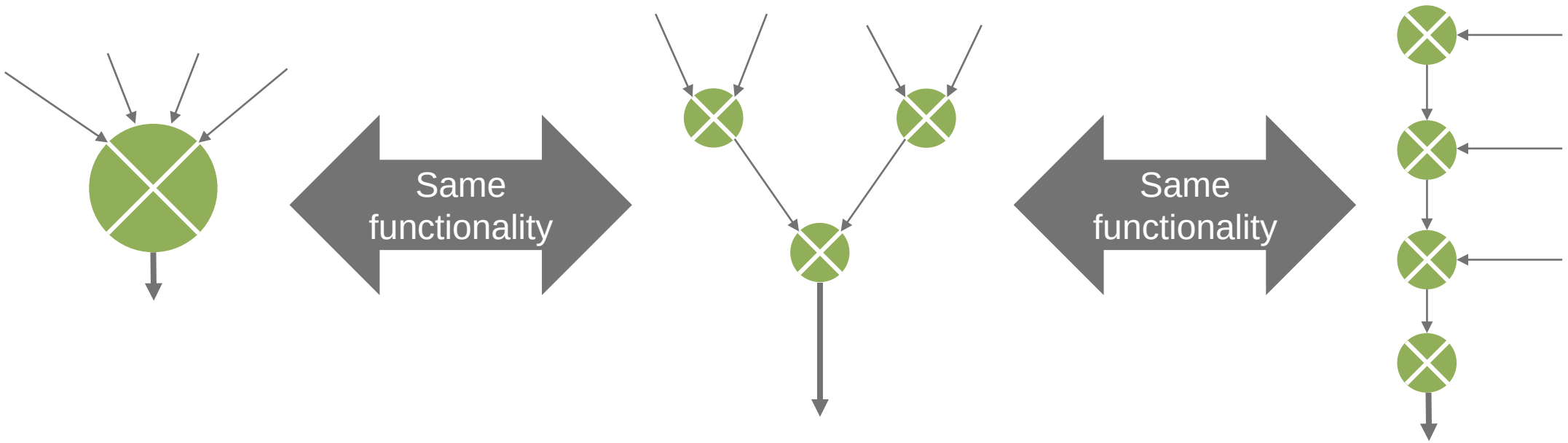
- What's the smallest topology using this approach?
  - Combine all sources into one
  - Distribute the result to all destinations
  - This topology is called a **shared link**
- All packets serialized on a single link
  - Becomes the bottleneck for performance
- In the area/performance tradeoff, it's the smallest area / lower performance ratio

Example


16Kgates
I0/I
I7/I
I1/I
I4/I
I6/I
I2/I
I3/I
I5/I
T0/T
T2/T
T4/T
T1/T
T3/T
T5/T
T6/T
I0/I
I7/I
I1/I
I4/I
I6/I
I2/I
I3/I
I5/I


2Kgates
I0/I
I2/I
I5/I
I3/I
I1/I
I7/I
I6/I
I4/I
T3/T
T6/T
T0/T
T4/T
T1/T
T2/T
T5/T
T7/T
I0/I
I2/I
I5/I
I3/I
I1/I
I7/I
I6/I
I4/I

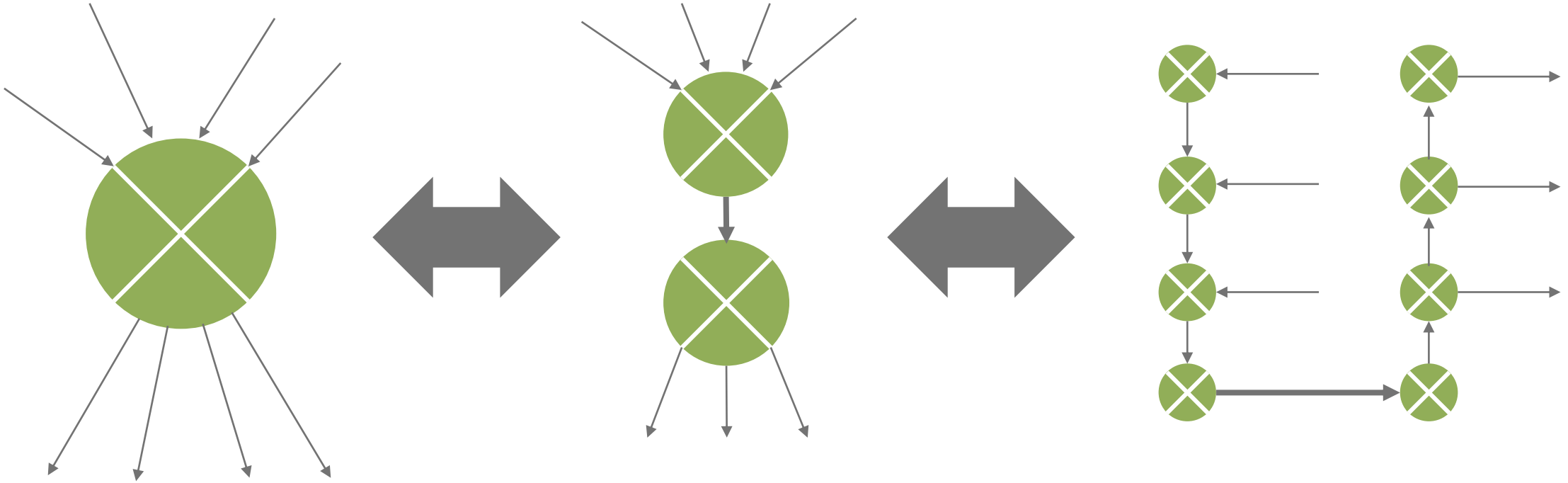
Adaptation to layout – basic idea



Centralized merger:
Everything done in a central place

Distributed merger:
Function spatially distributed → Easier to implement!

Adapt to layout – cont.

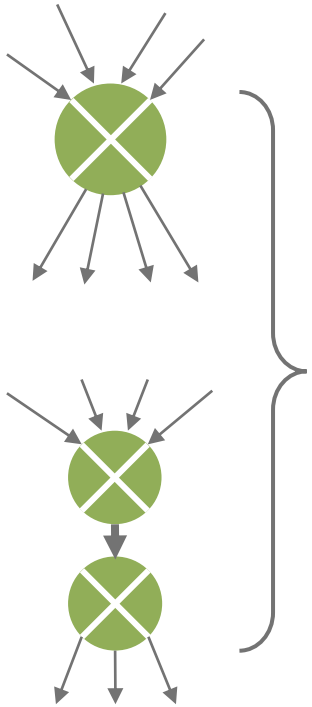


Centralized implementation
Biggest

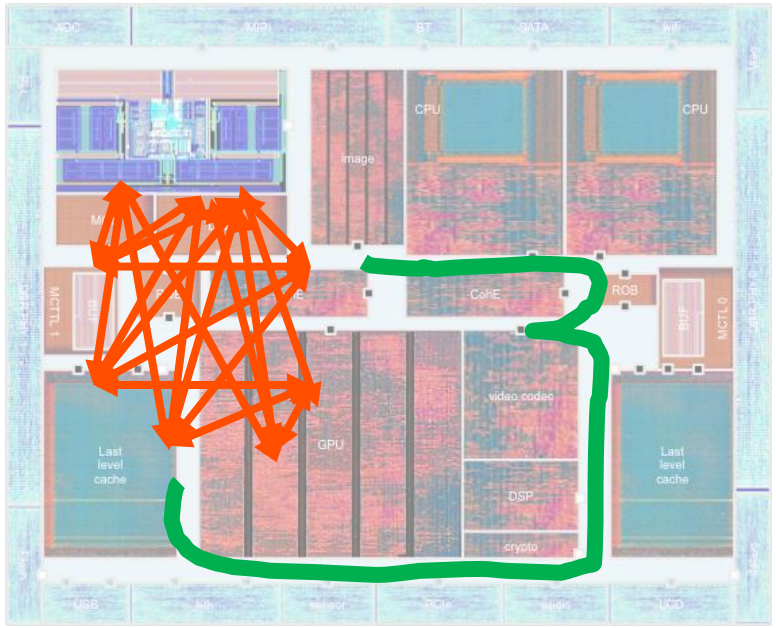
Centralized implementation
Smallest

Distributed implementation

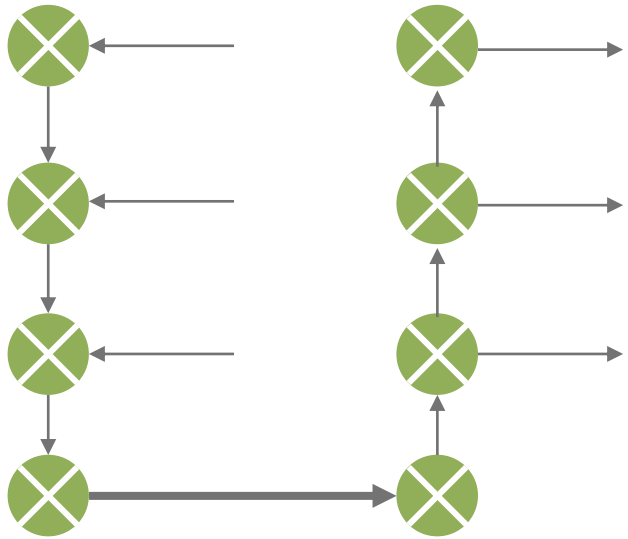
Benefit of distributed topologies



Difficult!

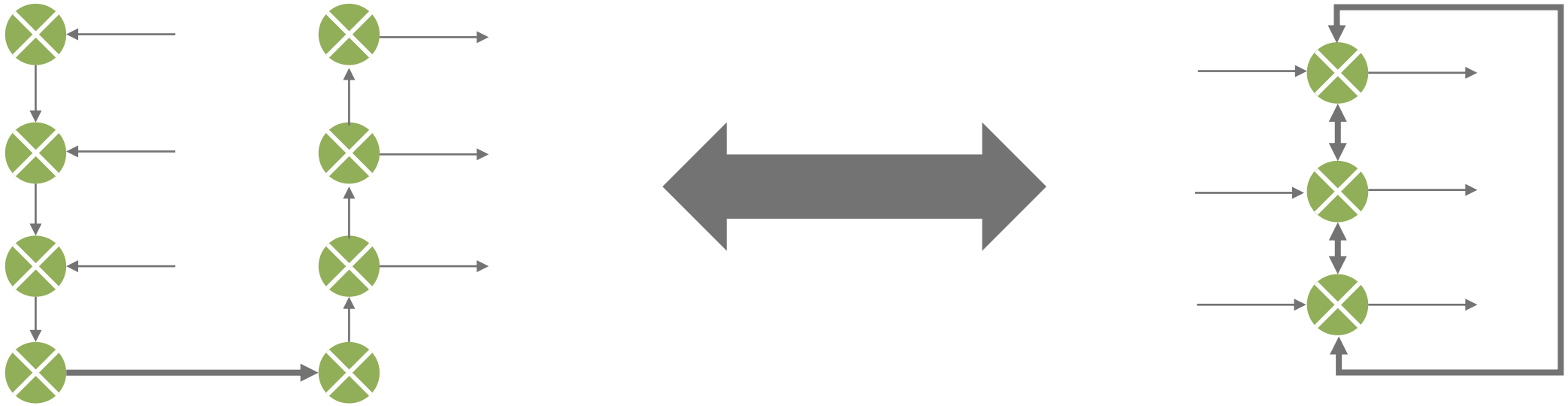


Easy!



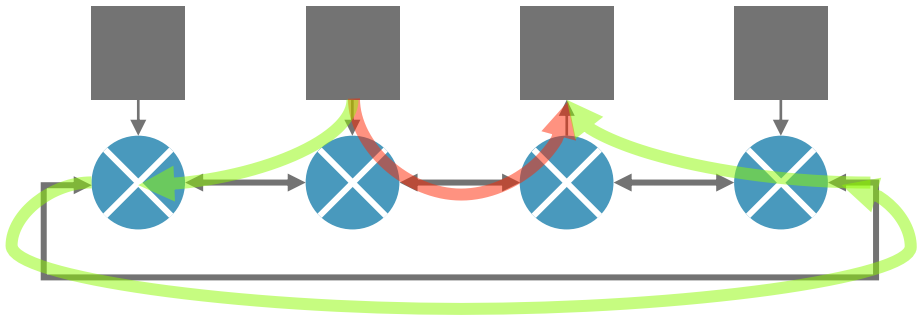
A distributed implementation is well adapted to most layouts

One step further – The ring!



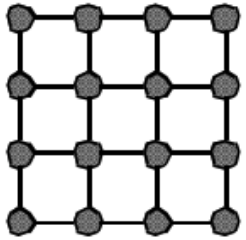
- Often some sources and destinations are close to each other: They can share a switch
- Switches become both sources and destinations of traffic
- To reduce the average distance (in switches) between source and destination: Bi-directional connections between switches
- We have built a ring topology

Concept: Path diversity

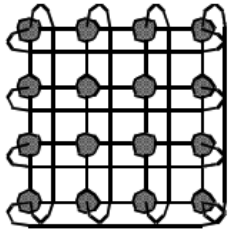


- Rings are a class of topology with more than one possible route between a source and a destination
  - 2 connections between 2 switches: East and West
  - So 2 paths between a source and a destination: Going East first or West first
- When a topology has path diversity, routing can become **adaptive**
  - Selection of a path depends on workload
  - Network performs **load balancing**
  - However, adaptive routing has challenges
    - Variable latencies
    - Deadlocks, livelocks
    - Implementation complexity

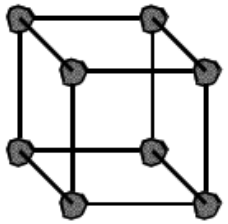
How about Torus, Meshes, Cubes, etc.?



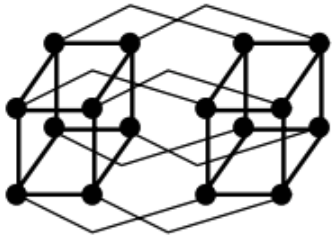
2D Mesh



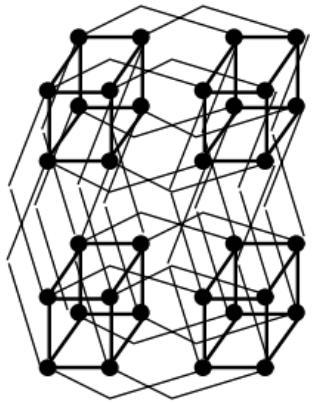
2D Torus



3D Cube



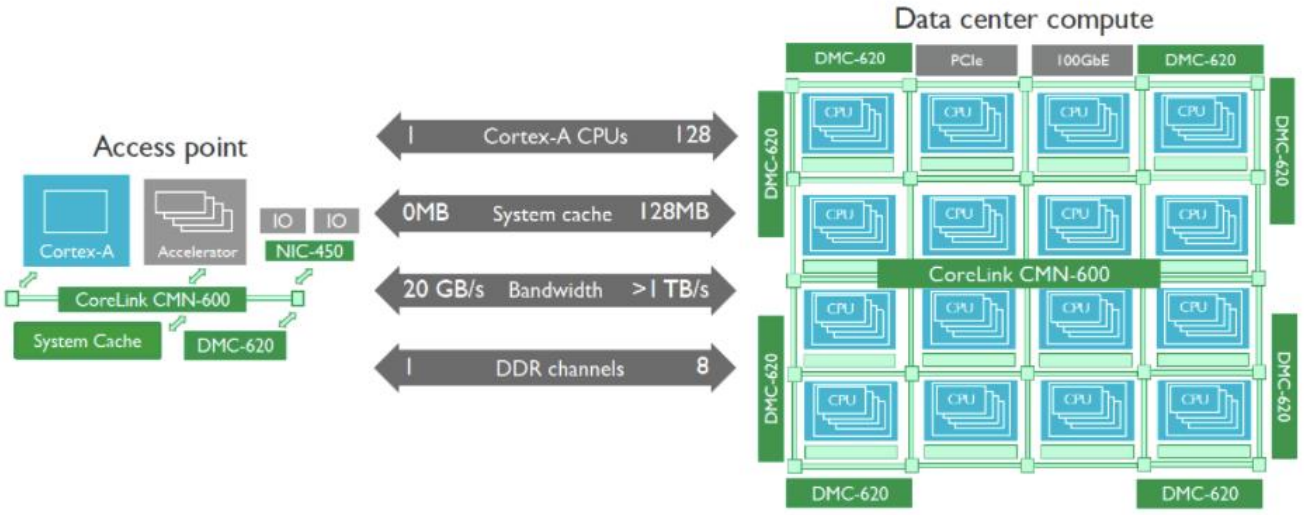
4-D



5-D !

- Regular “classic” topologies:
  - Organization of components on the chip follows a well known 2D arrangement
  - Communication between elements is regular
  - Network can follow the component arrangement!
    - High path diversity
    - Complex routing functions
  - Benefits: Strictly controlled link length can allow very high speed
- However, almost never found as-is in a SoC!
  - 99% of SoCs have irregular communication patterns
  - Most have a Unified Memory Architecture (UMA)
    - Most traffic goes to/from DDR

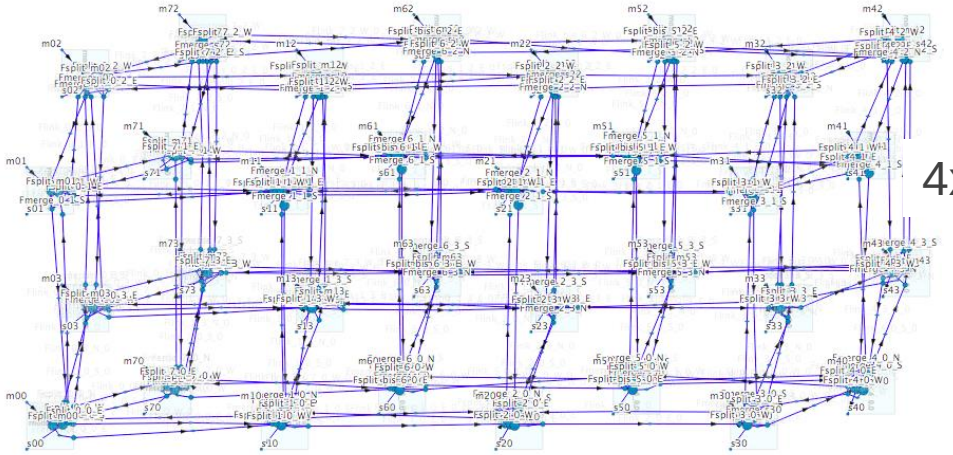
Meshes in the competition landscape



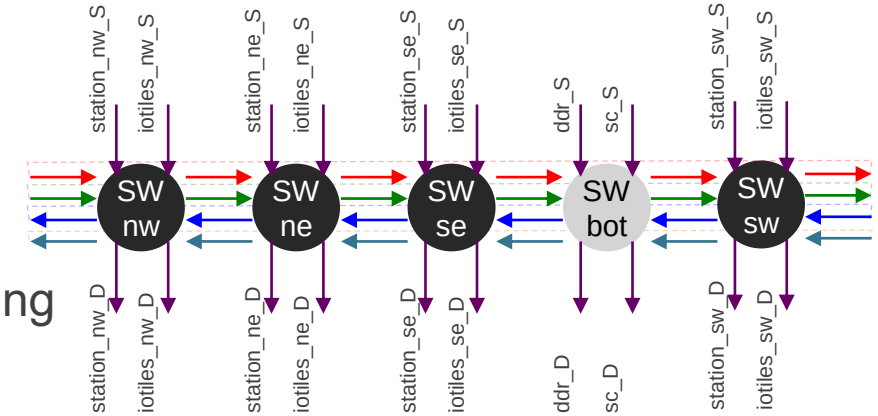
Automated interconnect generation with ARM CoreLink Creator

- **Intel / NetSpeed Orion** is a “degenerated mesh”
  - Routers are placed on a grid
  - Links are adjusted to deal with traffic
- **ARM CMN-600** is a mesh-based cache coherent interconnect

Can Arteris do meshes & rings today?



4x8 Torus

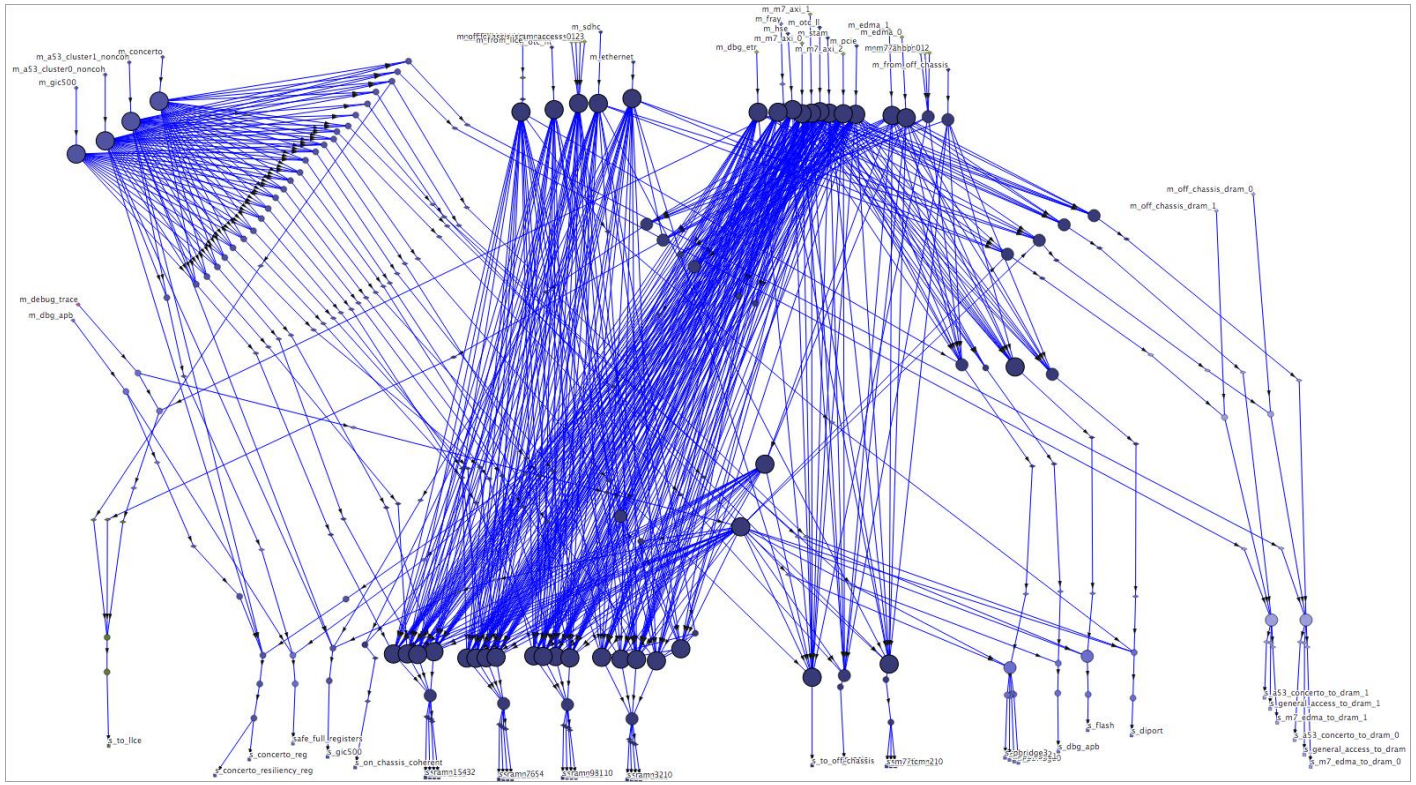


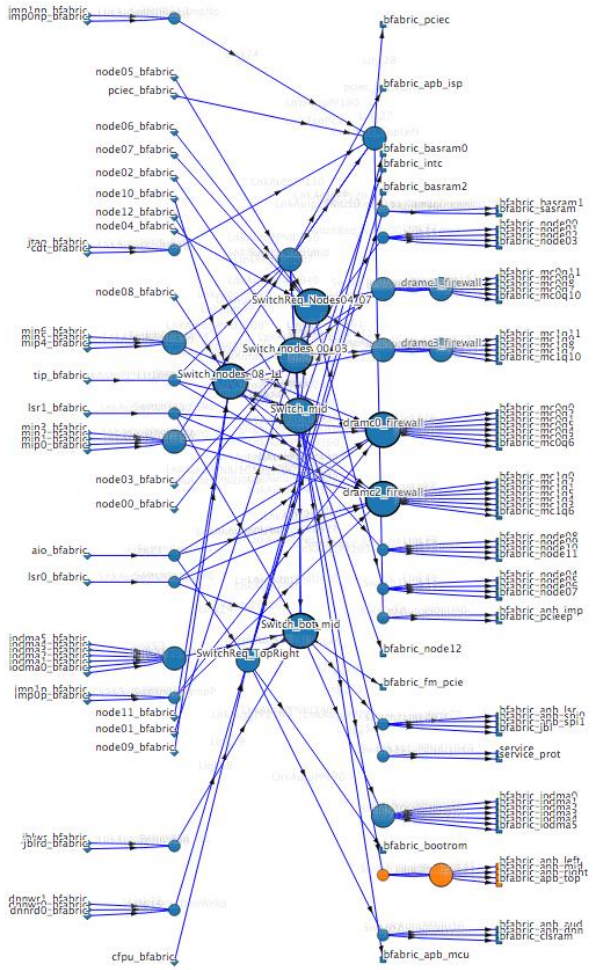
'XMOS' Ring

- **Yes**, we can!
- However:
  - The hardware cannot take advantage of path diversity
    - Load balancing cannot be done using dynamic routing
    - All routes shall be specified manually at design time
  - Avoiding deadlocks is difficult

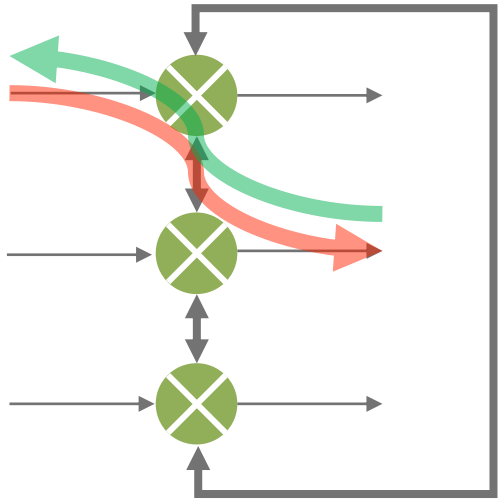
“Real life” topologies

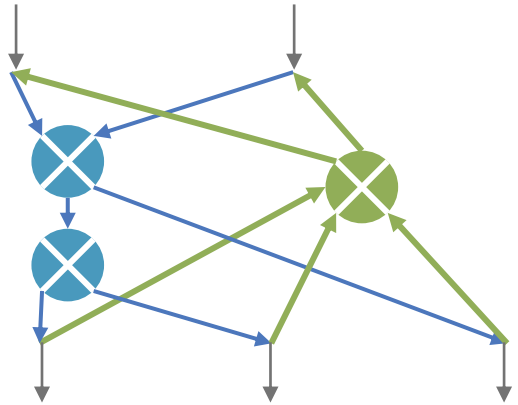
In “real life”, topologies are a mix of crossbars, with various degrees of distributed mergers and splitters





Request Networks / Response Networks



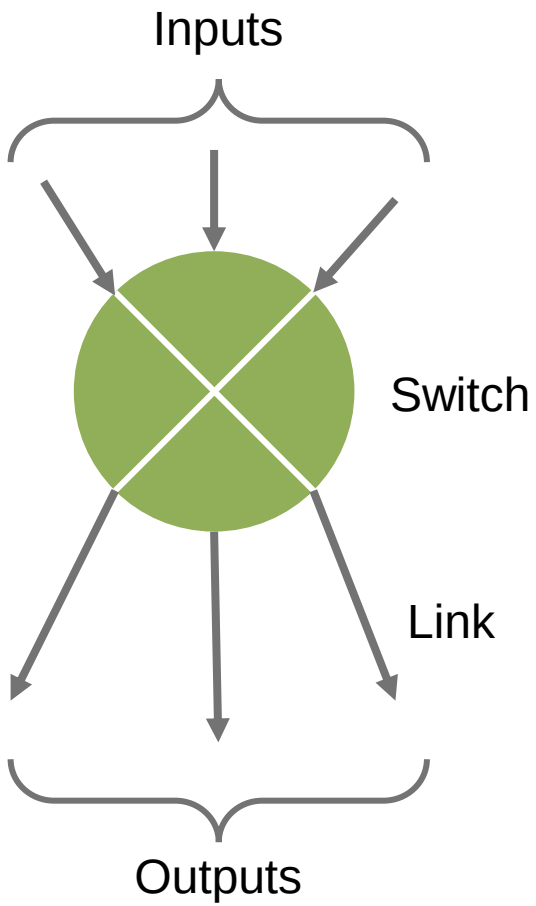


- Some topologies are well suited to unified request+response network
  - E.g.: Ring, mesh. Requests and responses can use the same network.
  - Issues: Deadlocks! If a target needs to move a response before it can accept a new request (most of them do), but the unaccepted request is blocking the response because they share a path: Circular dependency.
- Complete separation of request and response networks are more robust
  - Guaranteed free from deadlocks
  - Allow different optimizations in each directions
  - Better suited to typical SoC networks (irregular)

Switches

NETWORK BUILDING BLOCKS

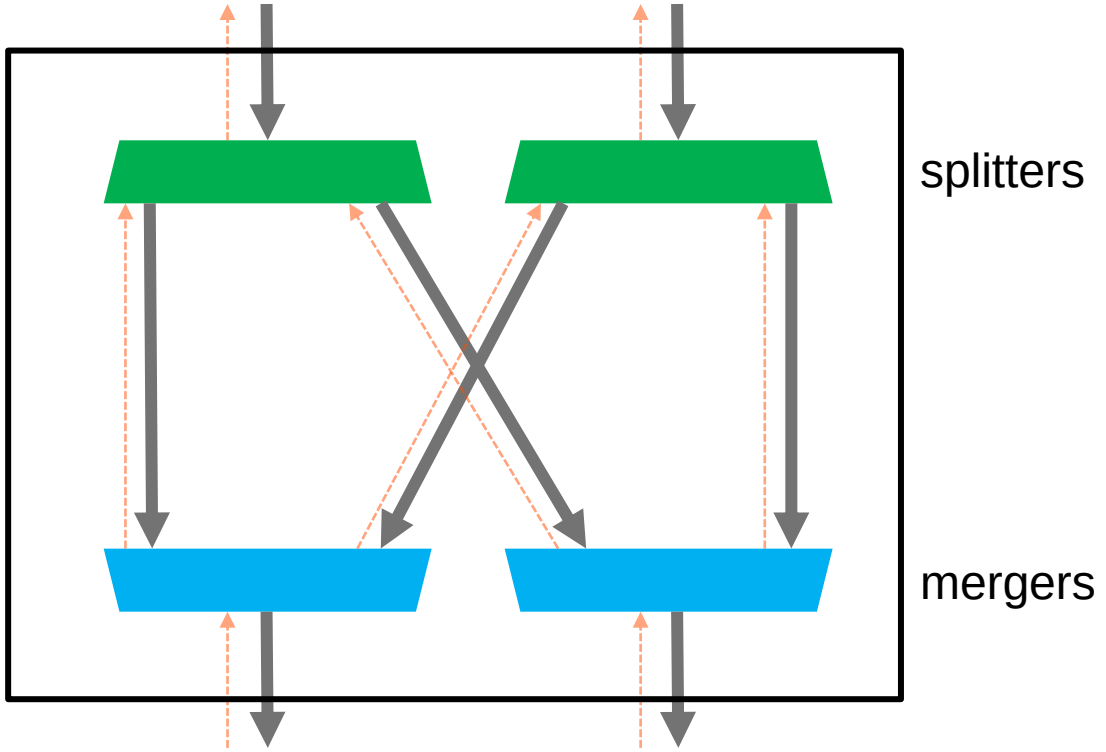
Switch high level view



- A **switch** is a logic function that can route the traffic from I inputs to O outputs
  - It performs **routing**: Mapping a request from its input ports to an output port according to a mapping function
  - It performs **arbitration**: Resolution of conflicts when multiple input ports need to access the same output port
- Switches and links carry **packets**

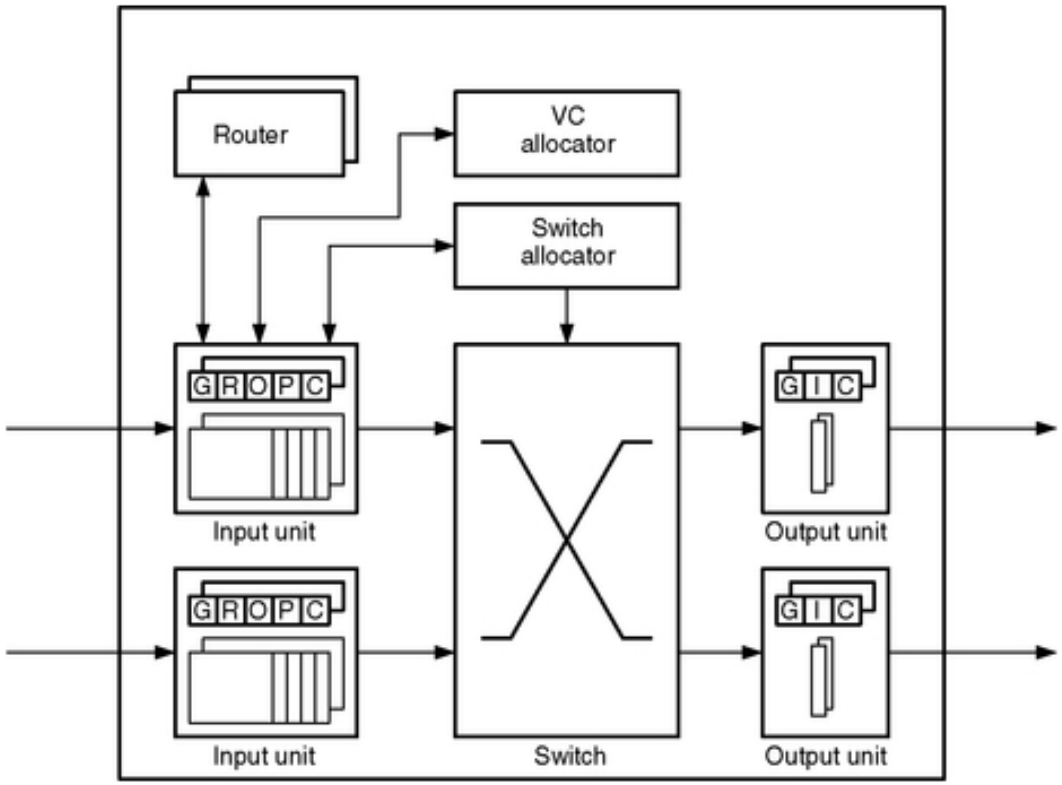
Simplified switch architecture

Valid/Ready flow control



- **Splitters** perform:
  - **Decoding** of the output port from header information (**routing**)
  - Tracking of the packet FLITS and holding of the decoded value until the end of the packet (state)
  - Demux of the forward packet signals to the output port merger
  - Mux of the flow control signal(s) from the output port merger
- **Mergers** perform:
  - **Arbitration** of the incoming forward headers and choice of a winner
  - Tracking of the packet FLITS and holding the winner status until the end of the packet
  - Mux of the forward packet signals of the winner
  - Demux of the flow control signal(s)

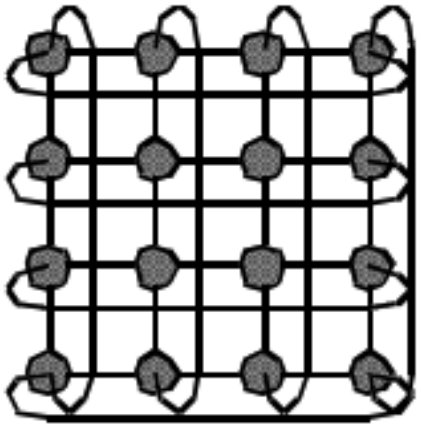
Complex switch architecture – with VC support



- Complex switches supporting virtual channels need input buffers
  - More on virtual channels later – in the flow control section
- The amount of state in a VC-capable switch is an order of magnitude higher compared to the simple switch

Routing

- Routing = Find a route between a source and a destination
- Most NoC routing functions are
  - Static: Defined at design time
  - Oblivious: Do not depend on the state of the system
  - Static routing cost is low, routing function computation is fast
    - Issue: Does not balance load on different links when there is path diversity
  - FlexNoC routing is static, explicit: All routes are defined by the designer
- However, routing can be dynamic, adaptive
  - When path diversity exists: Choice between multiple path can be made dynamically
  - If routing is a function of the system state: Adaptive
  - Example: In a 2D torus, a packet can go in either of the 4 directions at each node
    - Can be made a function of link occupancy, congestion
    - Beware: Deadlocks!

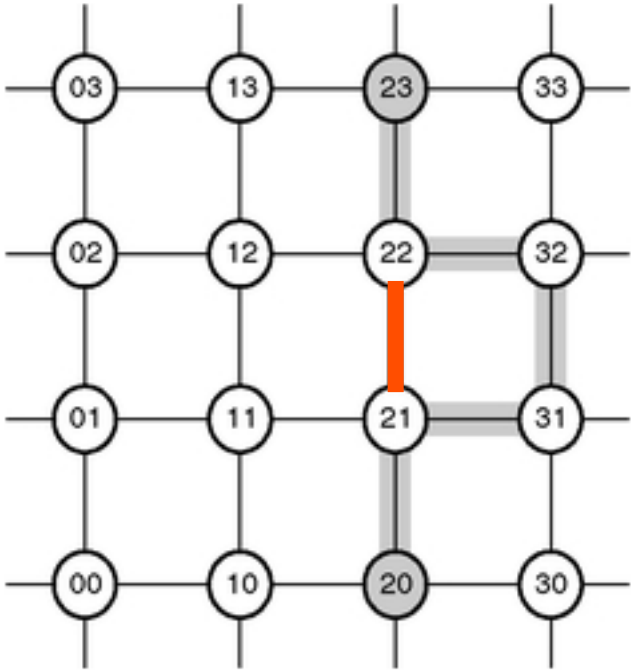


2D Torus

Benefit of static routing – lower cost!

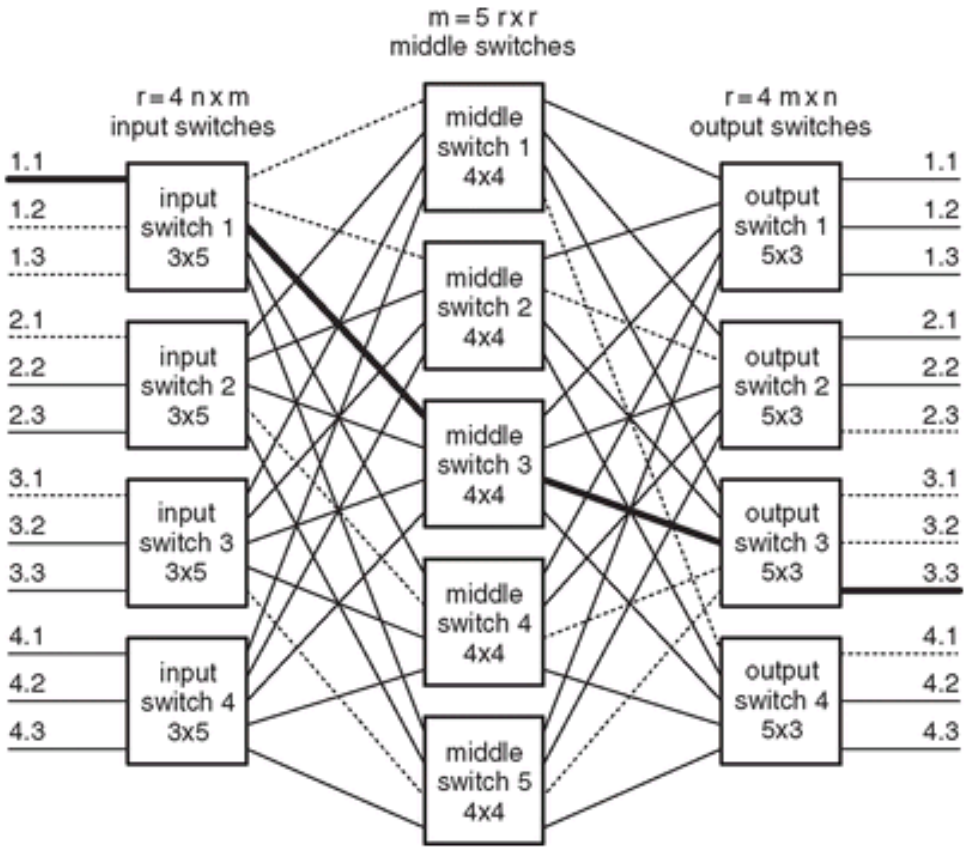
- At a splitter, choosing the destination output port of the switch is in the critical path
- Routing function at each switch is implemented using a table
- If the routes are static – defined at design time – synthesis will simplify the table as a set of logic equations
- If the table is dynamic – can be changed at runtime – this requires a RAM lookup
- Hybrid schemes exist:
  - A static table is defined at design time, but encodes multiple possible routes
  - A dynamic selection amongst the possibility is made by a function
    - Based on network workload, random (for load balancing), etc.

Benefit of dynamic routing – congestion avoidance



- If routing decisions can adapt to the network load, the network can be used more efficiently
- Beware livelocks: Packets that moves without ever reaching their destination.

Benefit of dynamic routing – non-blocking property

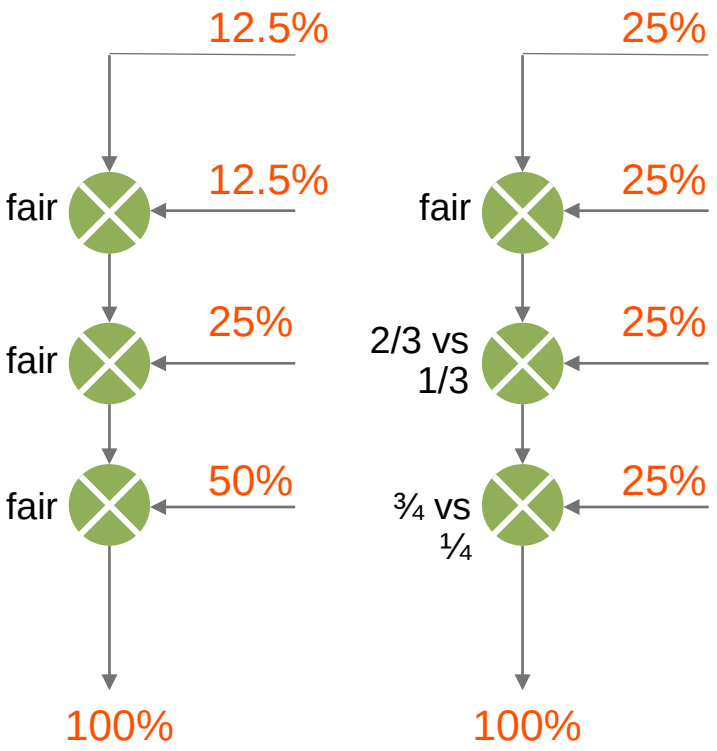


- If routes can be adapted to a particular situation, a complex network (more complex than a single switch!) can be made **non-blocking:**
  - There is a solution for every routes as long as there are no destination conflicts (each source goes to a different destination)
- Example: Clos network
  - Each routing “session” needs to find a set of routes from source X.Y to destination I.J
  - If the network topology satisfies a criteria for its path diversity, Clos networks are guaranteed to be non-blocking

Arbitration

- An arbiter's role is to select a winner amongst a set of contenders for a resource
  - At the heart of a merger, in its critical path
- An important property of an arbiter is **fairness**: It provides an equal opportunity to all requests to be served.
  - Strong fairness: Requesters will be served equally often
  - Weak fairness: Requesters will be served "eventually" (no starvation)
  - FIFO fairness: Requesters are served in the order of arrival (and strong fairness amongst simultaneously arrived requests)
  - Weighted fairness:
    - Two requests might represent completely different amounts of data transferred
    - Example: A byte read and a cache line read
    - Weighted fairness applies weights to requesters and guarantee contenders will be served with respect to their weights – as long as there is continuous contention
    - Or, one might need to mitigate the effects of topology on arbitration

Weighted fairness – adjusting arbitration to topology

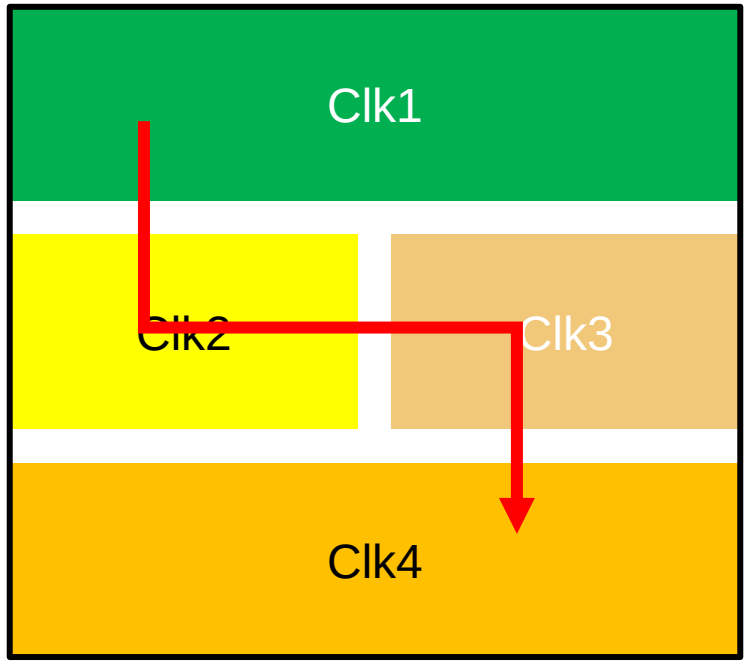


- Cascaded arbiters can lead to great differences in how requesters can access a resource
- Special QoS mechanisms can mitigate this problem
- Weighted fairness can also mitigate this effect
  - Can become tricky if state must be maintained over time so that the arbiter “remembers” the past to allow the weighting to apply even if there is no continuous contention.

Domain crossings

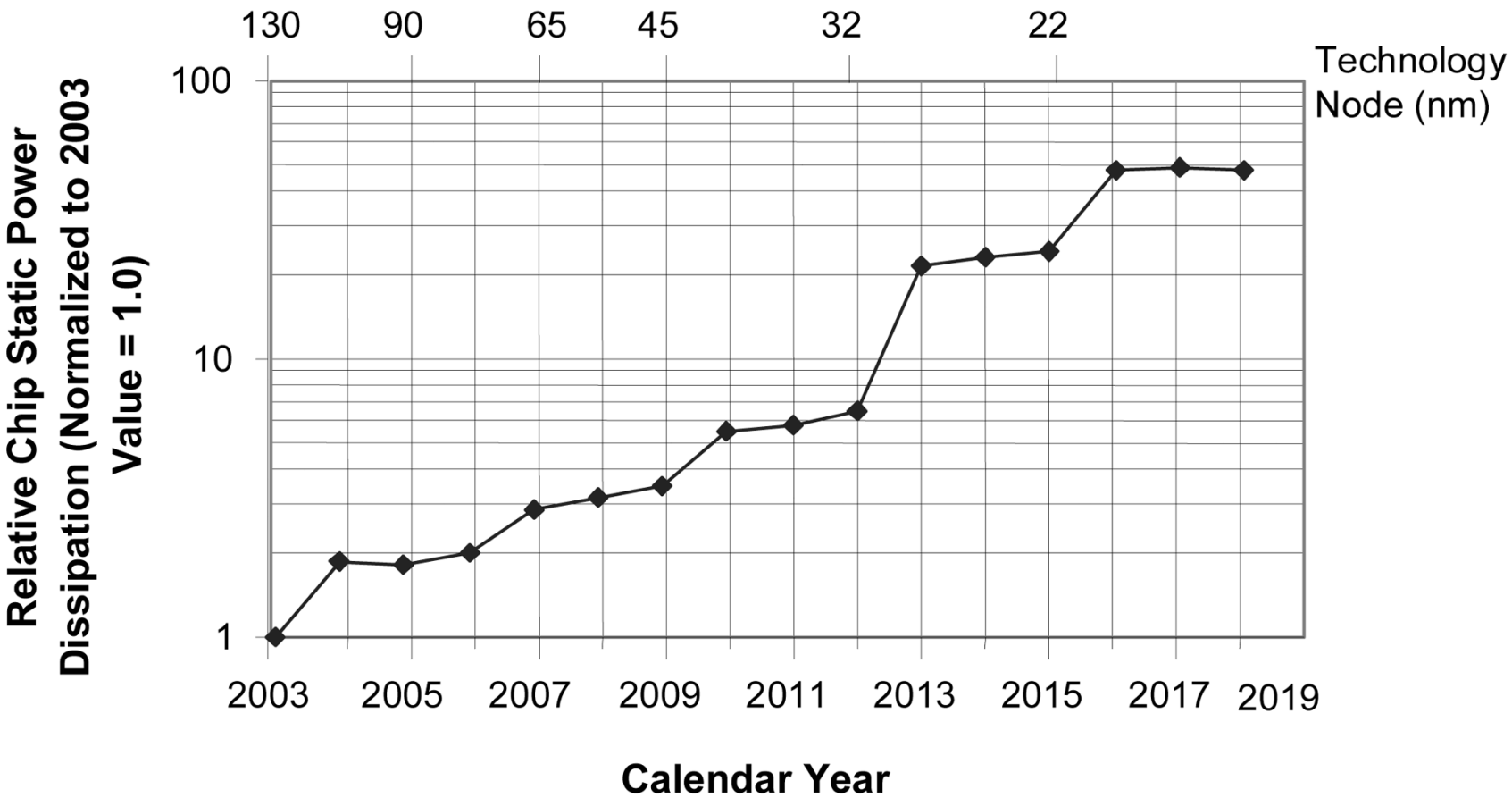
WHAT TIME IS IT ON THE OTHER SIDE?

Clock domains



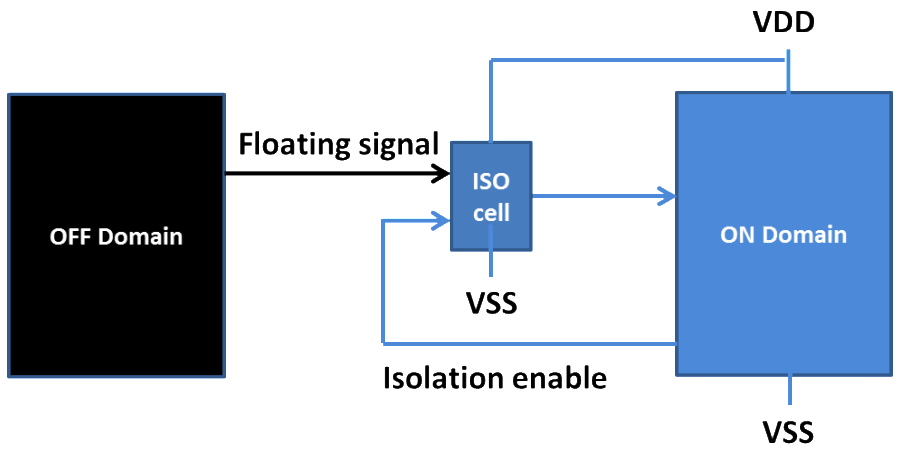
- GALS concept:
  - "Globally Asynchronous, Locally Synchronous"
  - SoC divided into multiple synchronous islands
  - Islands asynchrnous w.r.t. each other
  - Benefits:
    - No clock balancing across long distances == easier timing closure
    - Lower stress on power supply (no synchronized transitions == spikes on the whole chip)
- The NoC must ensure communication between islands!
- Solution 1: run the NoC synchronous, adapt between NoC and IP
  - Bad for timing closure (NoC spans long distances)
  - Bad for gate count (wide asynch FIFO)
- Solution 2: implement Asynch FIFO in the transport
  - Best for timing closure, lower gate count (packets narrower then transactions)

Dynamic Power management



- Dynamic power management required by increase in leakage currents in CMOS planar
  - Leakage = current flowing in absence of activity

Crossing Power domains – Isolation cells



- To avoid floating signals if they are no longer driven because the driver is OFF
  - Need special isolation cells (clamp)
  - Need to clamp at the value of the input signal at reset
    - To avoid glitches at power up

Thank you