Models of Computation for Networks on Chip

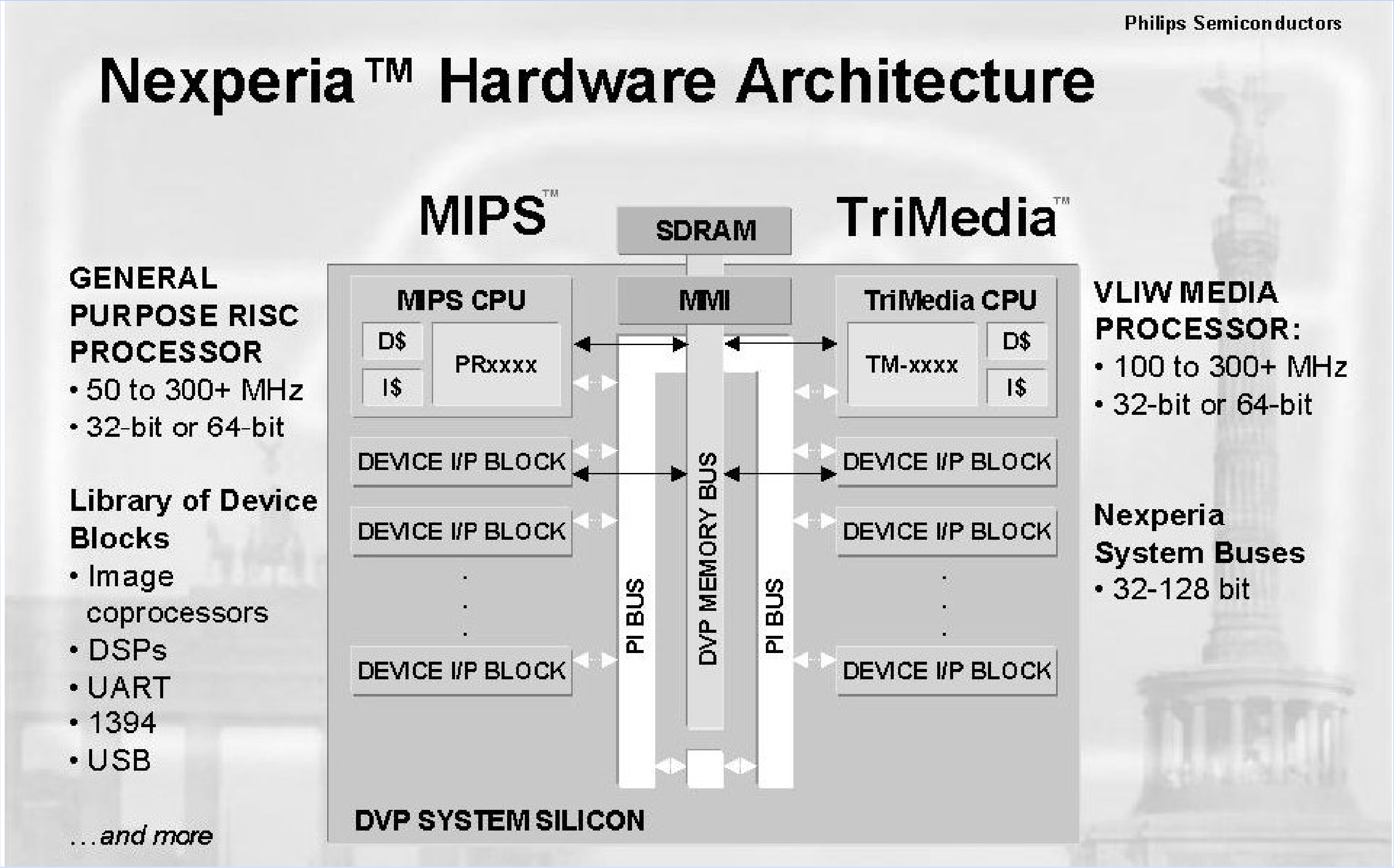
Axel Jantsch
Royal Institute of Technology, Stockholm

ACSD 2006

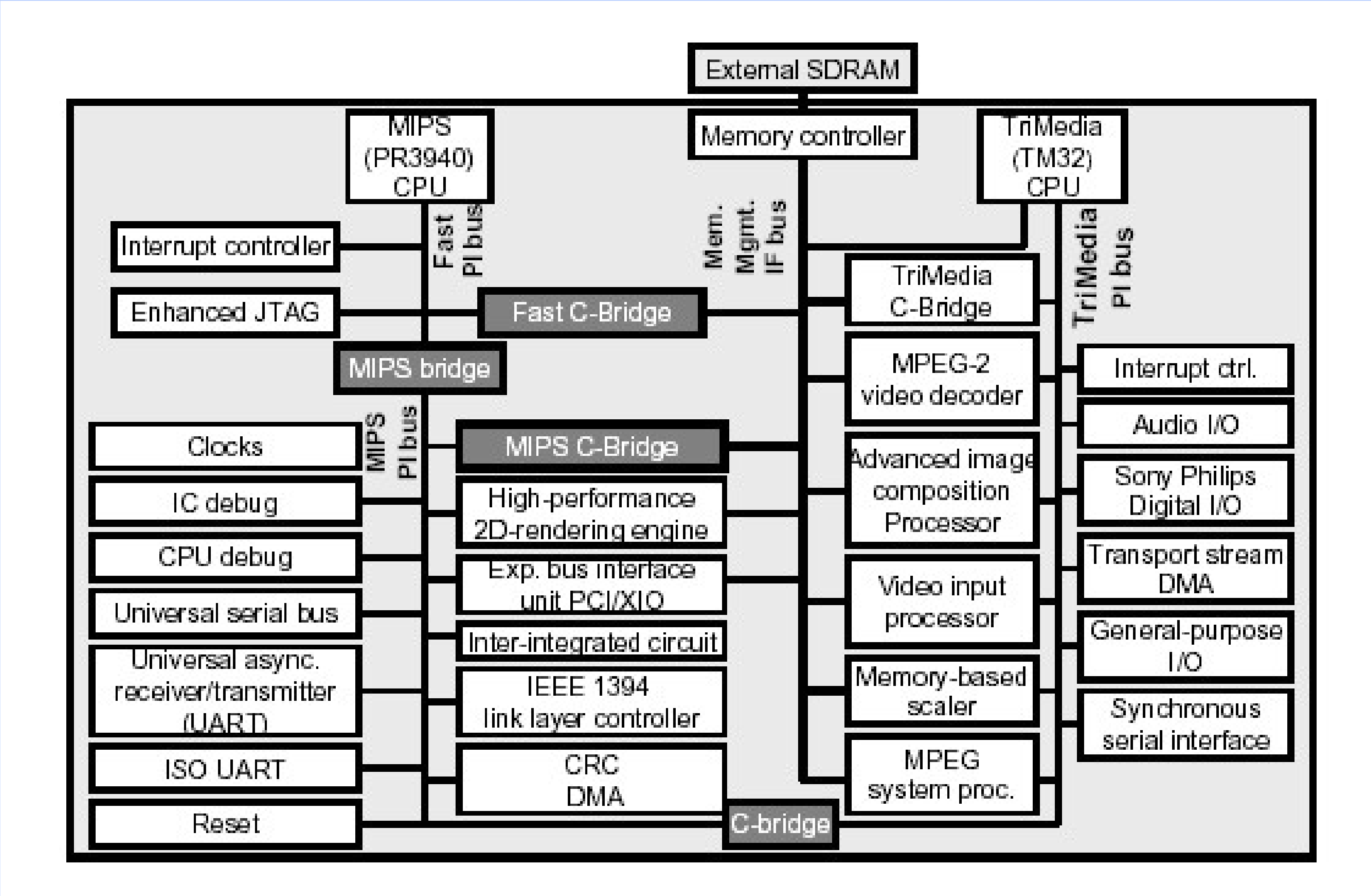
Overview

- System on Chip (SoC) Platforms
- Composability
- The Nostrum NoC
- A Nostrum MoC
  - Composition of Guaranteed Bandwidth traffic
  - Composition of Best Effort traffic
  - MoC Properties
- Summary and Conclusions

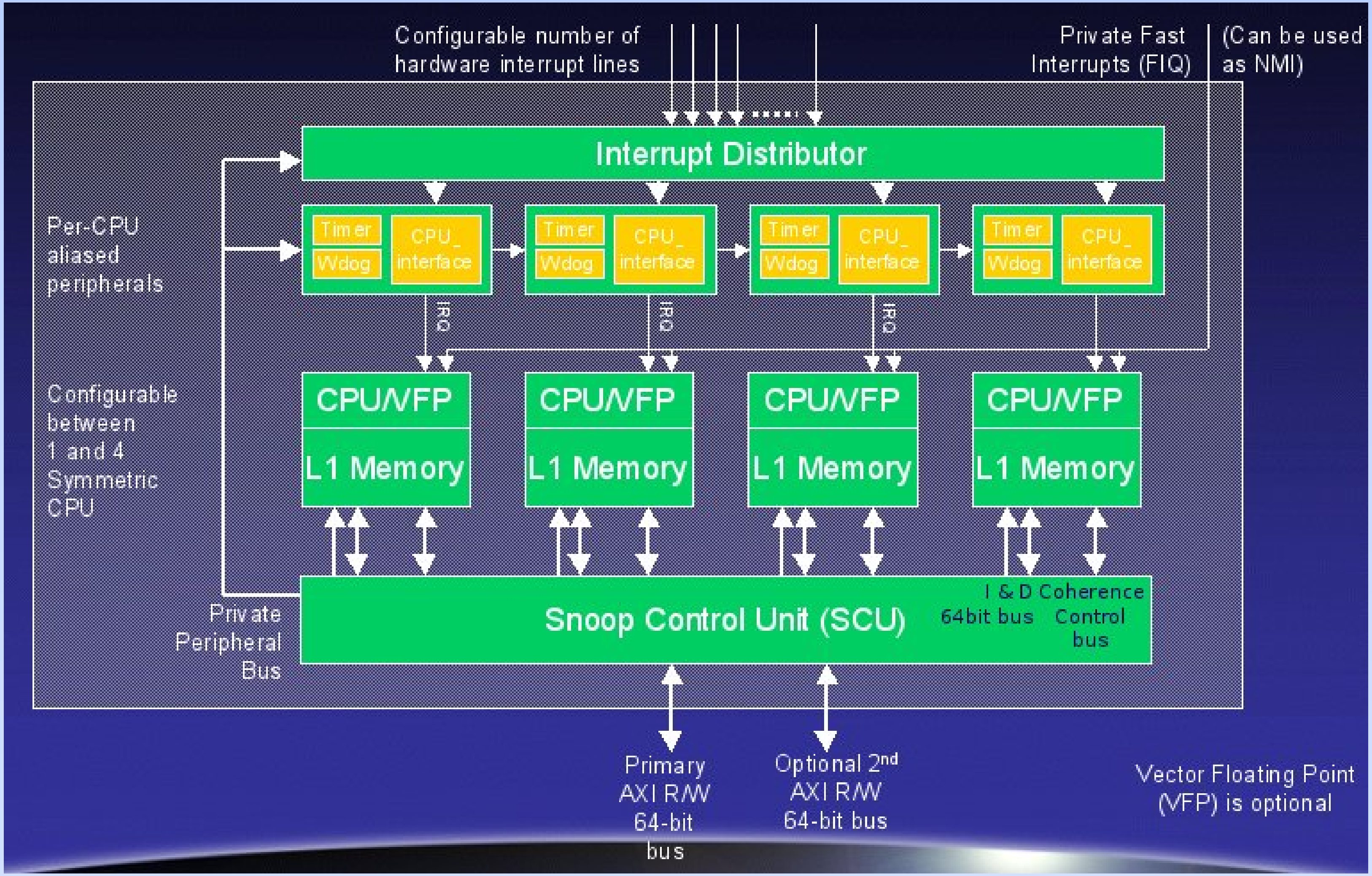
Platform Example: Nexperia



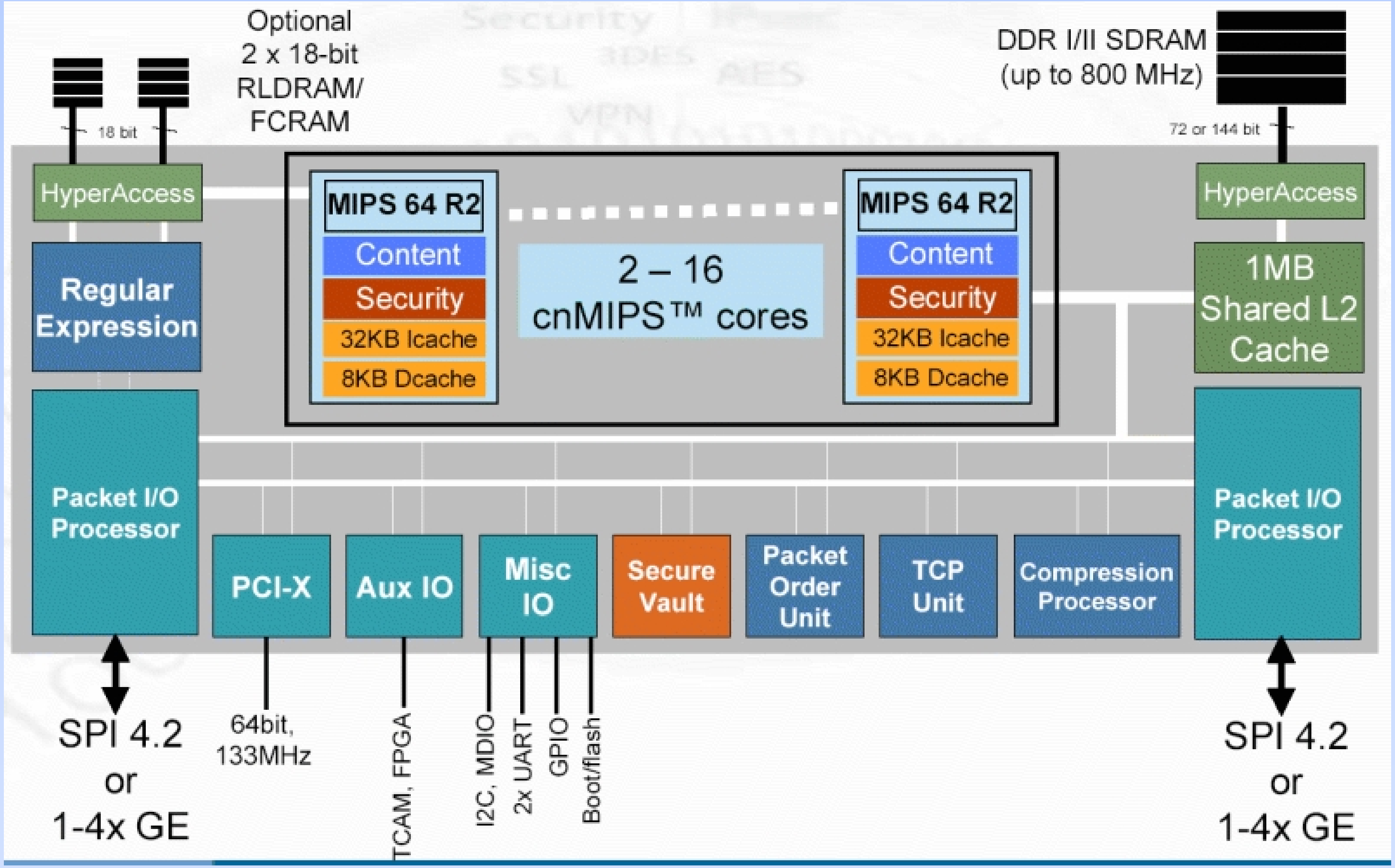
Nexperia Platform Instance: Viper



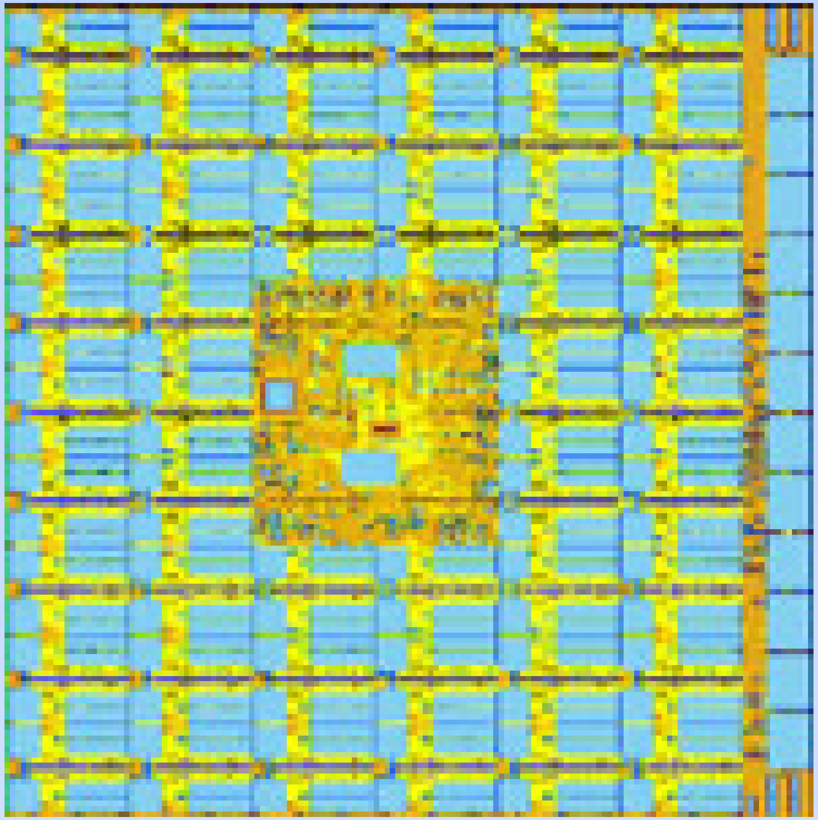
Platform Example: ARM Multiprocessor Core



Platform Example: Octeon Network Processor



Platform Example: Emulator Chip with 768 Processing Units



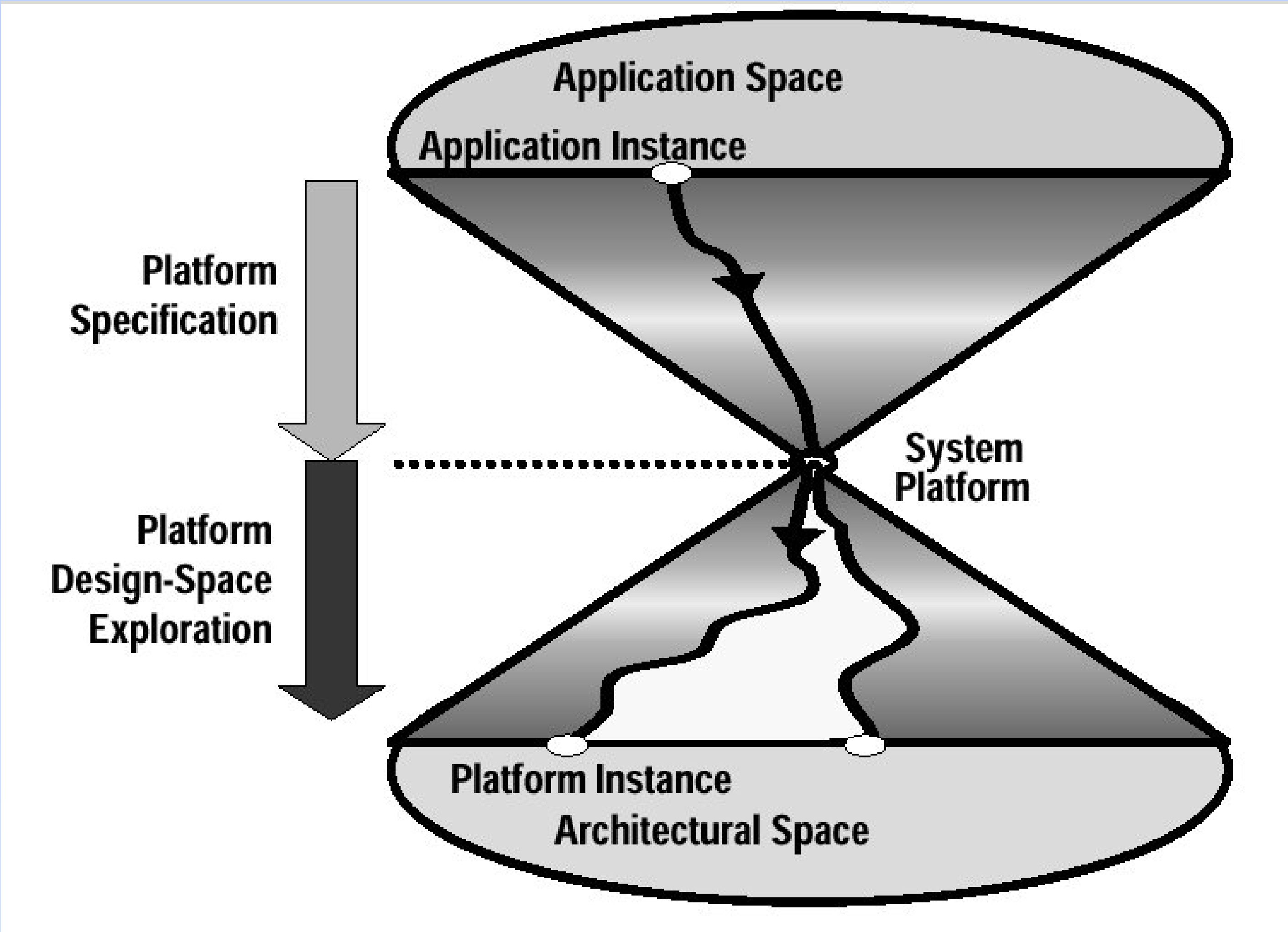
What is a SoC Platform

1. Library of HW and SW IP blocks
2. Communication infrastructure
3. Resource management services
4. Design methodology and tools

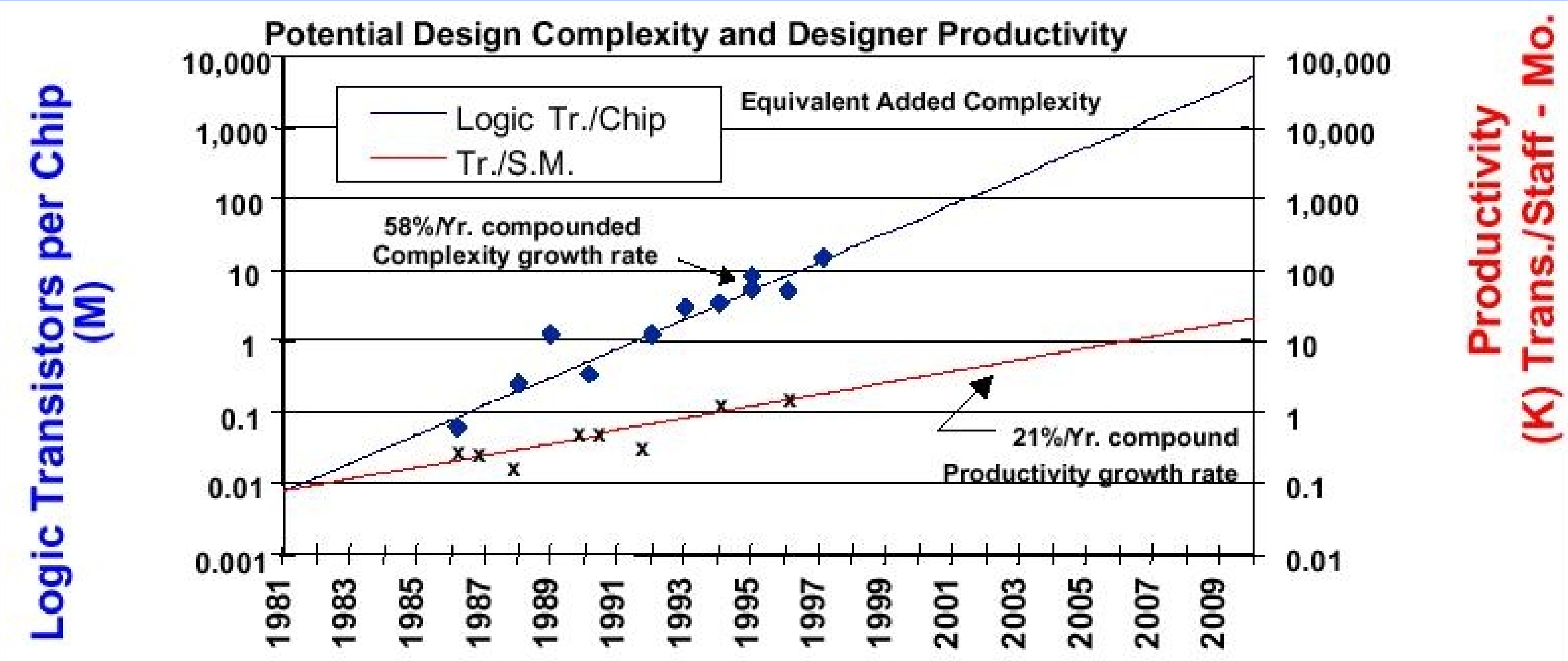
Platform Characteristics

- Tradeoff between efficiency and cost
- Application area specific
- Predictable performance characteristics
  (Guarantees if possible)
- Scalability (Size, Performance, Functionality)
  - ⋆ Performance - Cost
  - ⋆ Reliability
  - ⋆ Design methodology

Platform Based Design



Design Productivity Gap



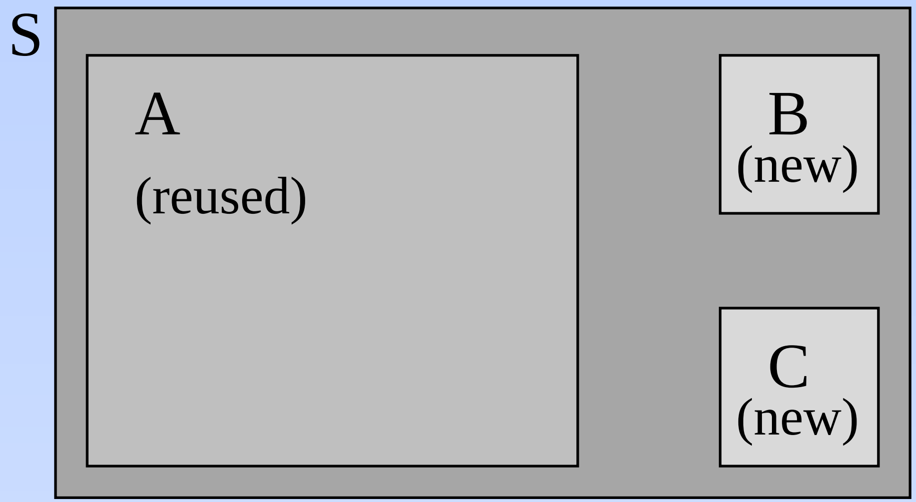
Source: International Technology Roadmap for Semiconductors 1999

Arbitrary Composability

Given a set of components C and combinators O.
Let A_1 be a component assemblage.
(C, O) is arbitrary composable if

$$A_1 + B \Rightarrow A_2$$

can be done for any $B \in C, + \in O$ without changing the relevant behaviour of A_1.

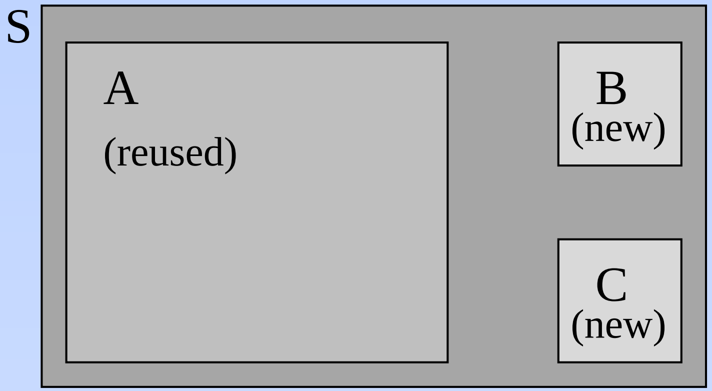


Platform and Composability

- A good platform has the arbitrary composability property.
- There are building blocks that can be added without changing the rest of the system.
- The building blocks can be:
  - ⋆ Computation resources
  - ⋆ Communication resources
  - ⋆ Storage resources
  - ⋆ I/O resources
  - ⋆ Resource manager modules (Scheduler, OS, ...)
  - ⋆ Features: Resources + System functionality
- The "relevant behaviour" includes functionality, performance, cost, reliability, power consumption.
- $\Longrightarrow$ **We can make guarantees.**

Linear Effort Property

Given a set of components C and combinators O.
Let $A_1, \ldots, A_n$ be component assemblages.
A design process using C and O to build a system has the linear effort property if $A_1, \ldots, A_n$ can be integrated into a system S with an effort dependent on n but not on the size of the assemblages: $\texttt{Ieffort}(n)$.
Total design effort for S is



$$\texttt{Deffort}(S) = \texttt{Deffort}(A_1) + \cdots + \texttt{Deffort}(A_n) + \texttt{Ieffort}(n)$$

Methodology and Linear Effort

- A good platform comes with a methodology that has the linear effort property.
- The platform is then scalable with respect to capacity increase by reusing ever larger components.
- This implies an invariance with respect to hierarchy: Composition works as well for primitive components as for arbitrary assemblages.

Platform Summary

- A good Platform greatly restricts the design space.
- It trades in optimality for design efficiency and predictability.
- The arbitrary composability and the linear effort properties provide a scalable platform.
- The reuse of ever bigger assemblages and components is platform inherent.
- Predictability of functionality, performance, cost, power consumption and reliability is a prerequisite as well as a consequence for the arbitrary composability property.

Model of Computation

A MoC is an abstraction of a computation device that

- Exposes relevant properties;
- Eliminates irrelevant details;
- Allows for efficient analysis, design, simulation, verification, synthesis, ...;

Model of Computation

A MoC is an abstraction of a computation device that

- Exposes relevant properties;
- Eliminates irrelevant details;
- Allows for efficient analysis, design, simulation, verification, synthesis, ...;

Examples:

- Turing Machine
- Lambda calculus
- Algorithm
- Random Access Machine (RAM)
- Parallel Random Access Machine (PRAM)

Model of Computation

A MoC is an abstraction of a computation device that

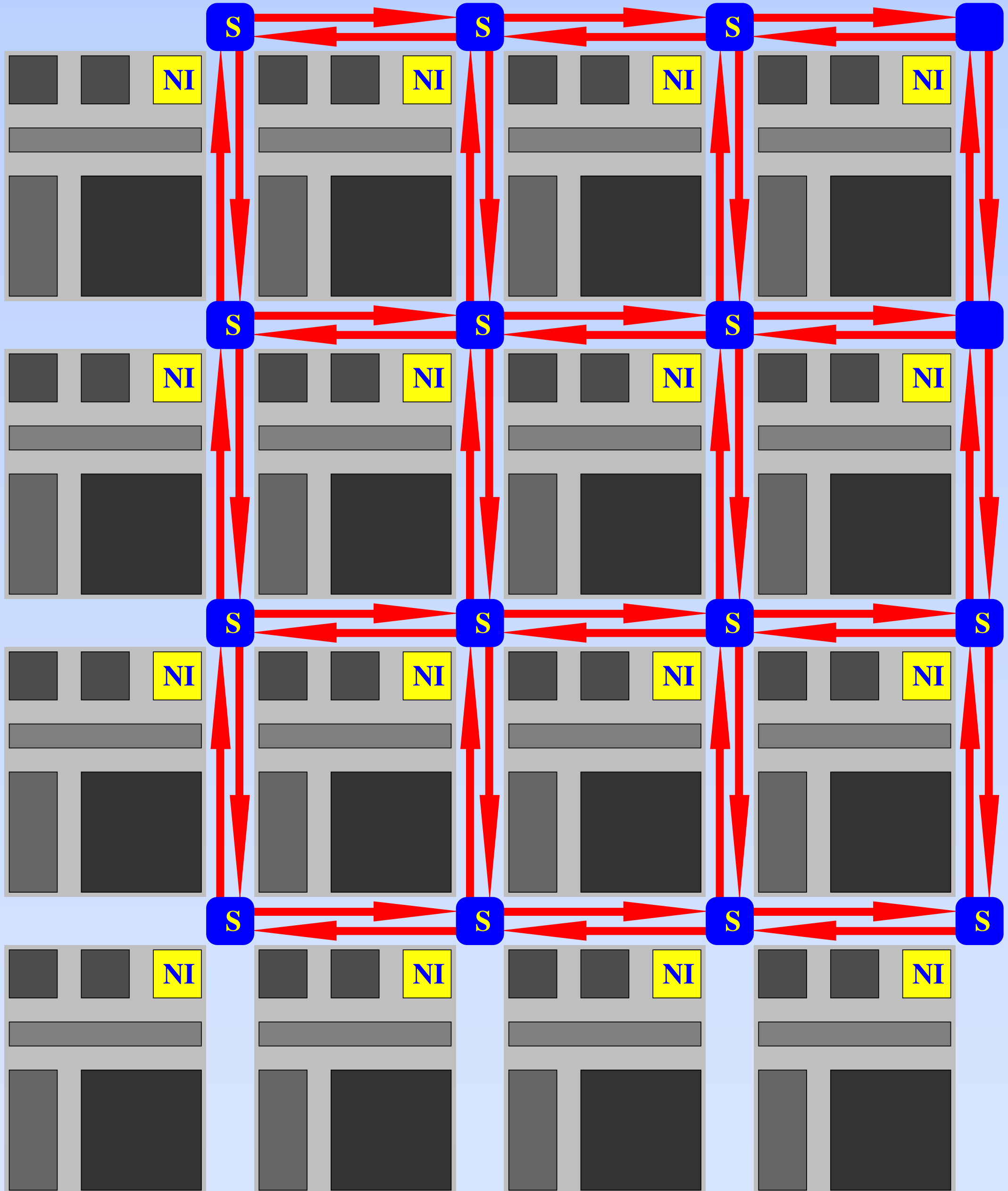
- Exposes relevant properties;
- Eliminates irrelevant details;
- Allows for efficient analysis, design, simulation, verification, synthesis, ...;

Examples:

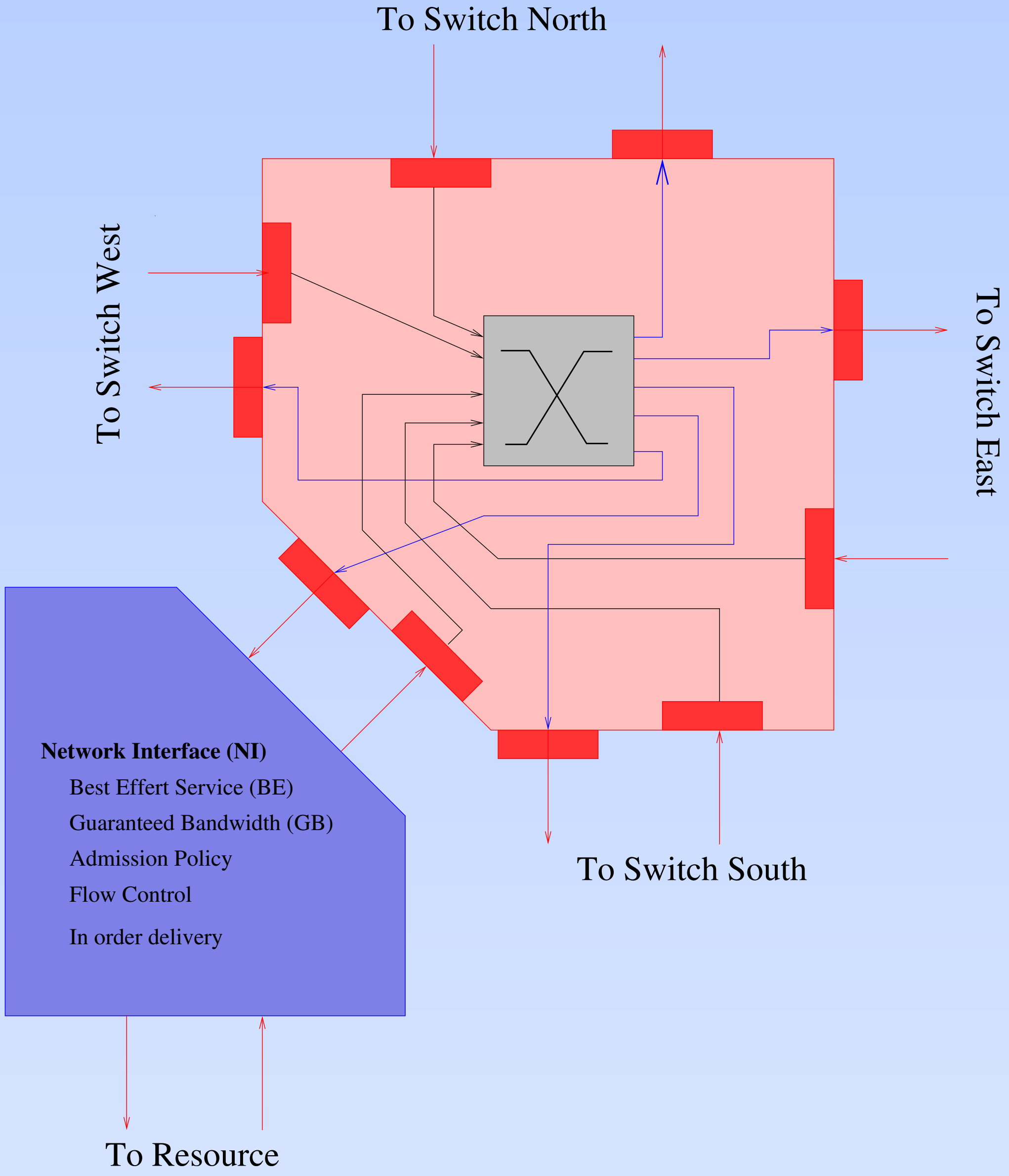
- Turing Machine
- Lambda calculus
- Algorithm
- Random Access Machine (RAM)
- Parallel Random Access Machine (PRAM)
- Petri net
- Kahn Process Network
- Synchronous Data Flow
- Boolean Logic
- Clocked synchronous model

The Nostrum Network on Chip Platform

The Nostrum Network on Chip Platform

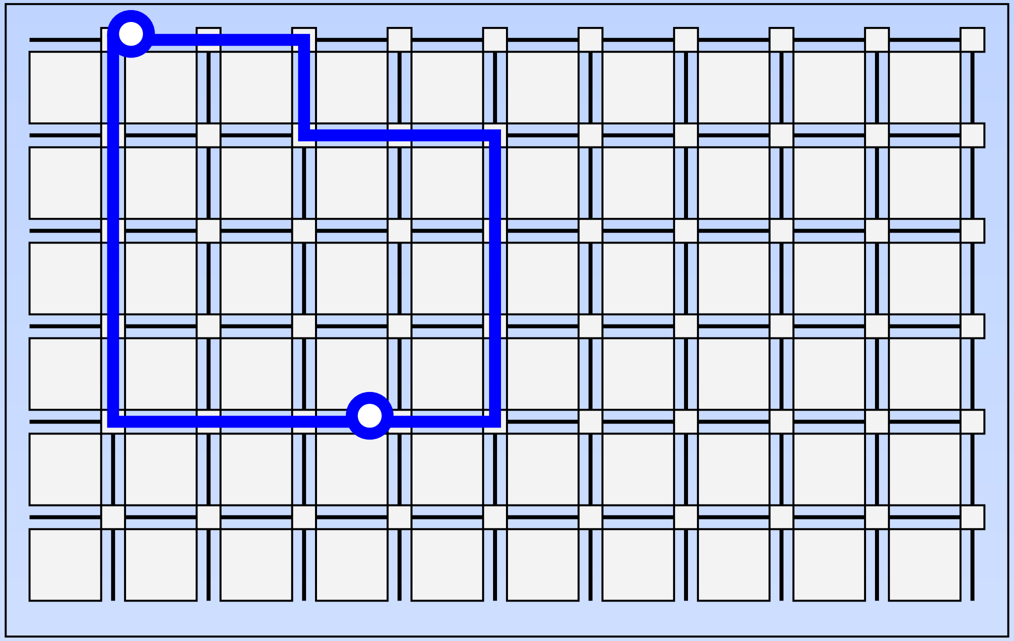


Nostrum Characteristics



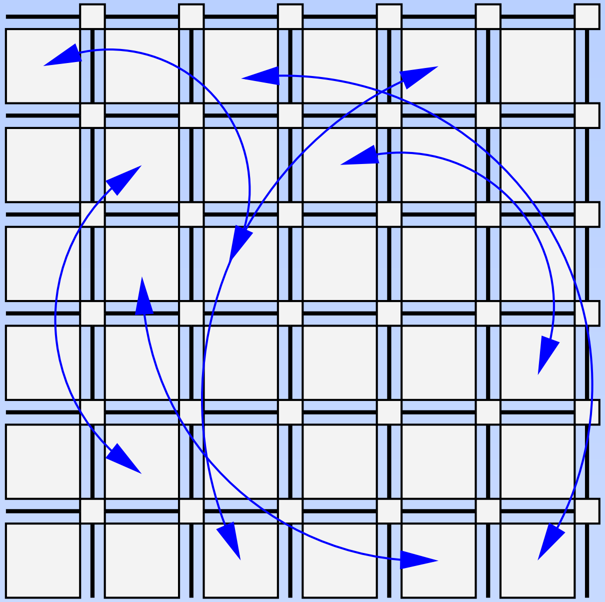
- Adaptive, hot potato routing
- No buffering in switches
- Access policy and buffering in the network interface
- Wide links
- Pseudo-synchronous network operation
- Best Effort service
- Guaranteed Bandwidth service based on virtual circuits

Nostrum Communication Services

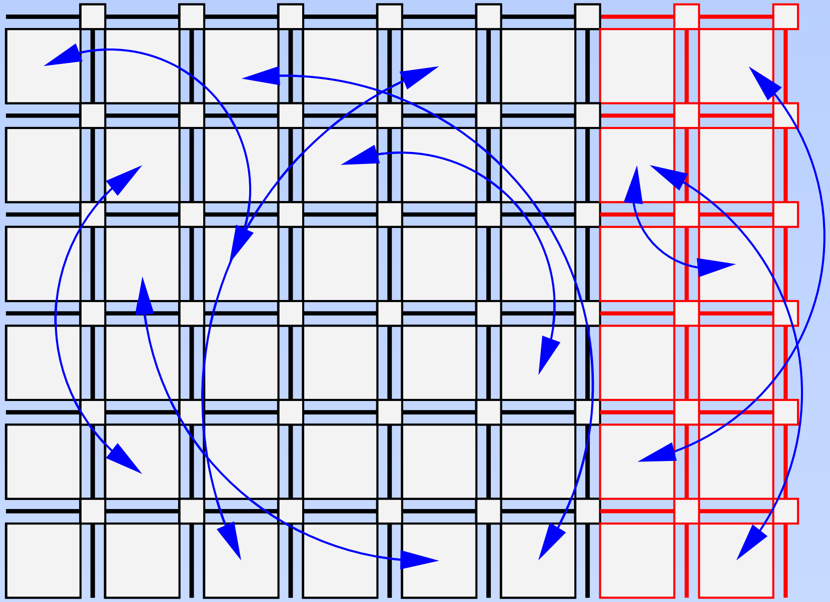


- Best Effort:
  - On congestion packets are deflected
  - Higher Priority:
    - Older Packets
    - Shorter distance to destination
- Guaranteed Bandwidth
  - Virtual Circuits (VC)
  - Looping containers reserve resources

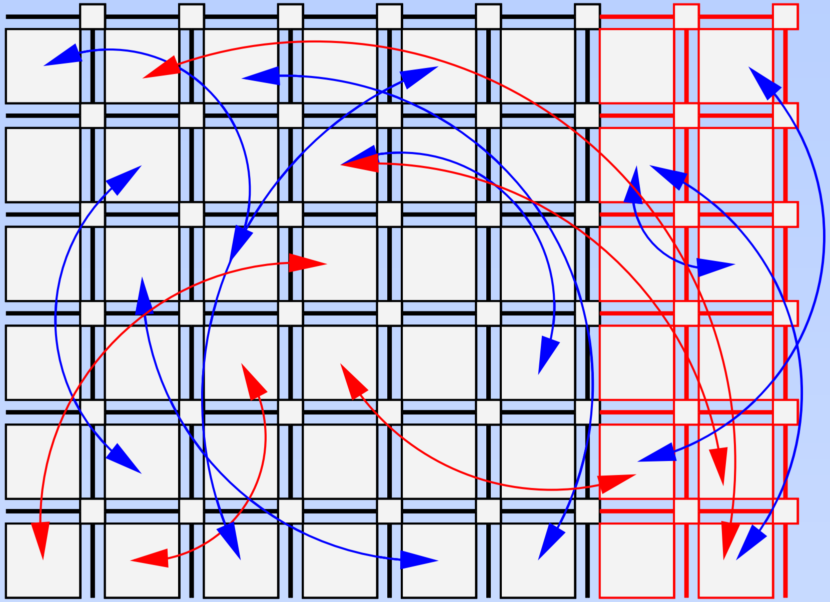
A Nostrum Model of Computation



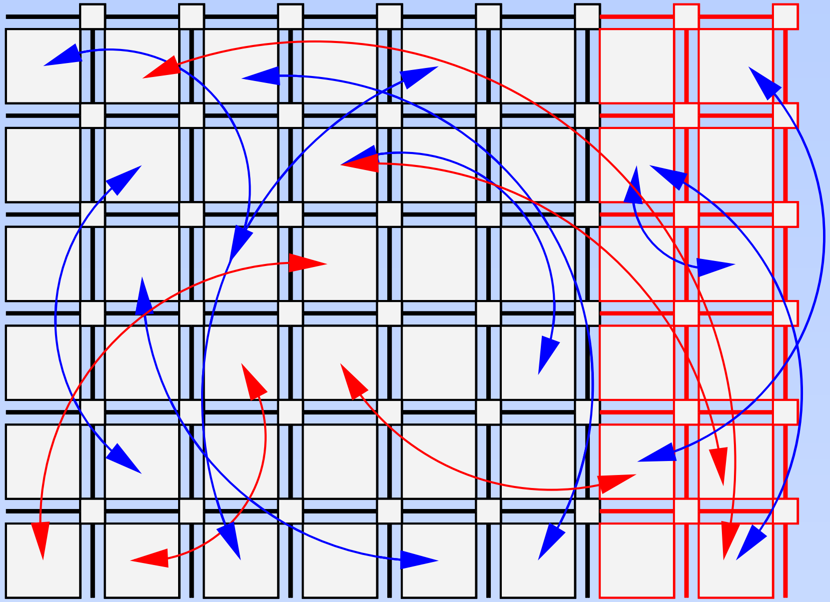
A Nostrum Model of Computation



A Nostrum Model of Computation

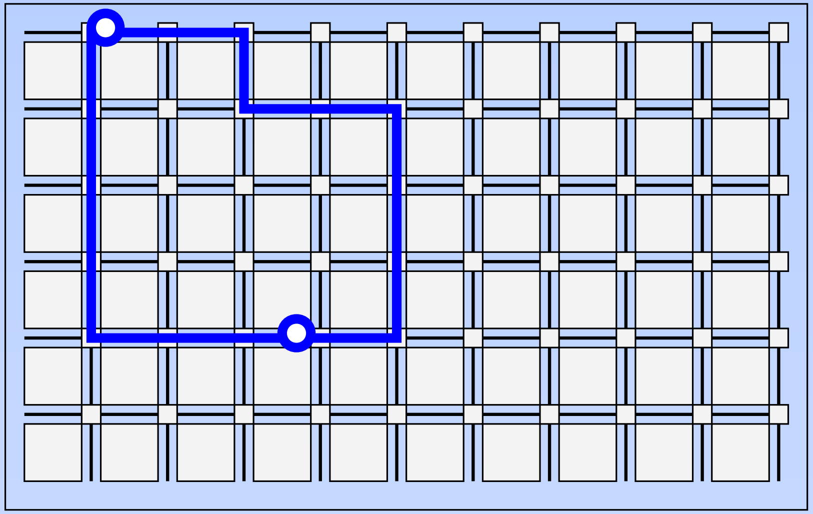


A Nostrum Model of Computation

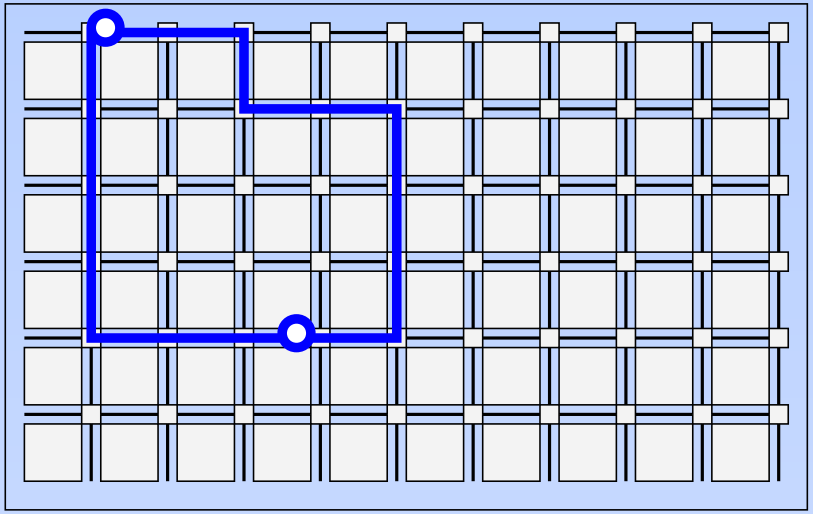


- Composition of Functionality with predictable performance
- Composition of Functions in network nodes
- Composition of Traffic
  - ⋆ GB traffic composition
  - ⋆ BE traffic composition

GB Traffic Composition

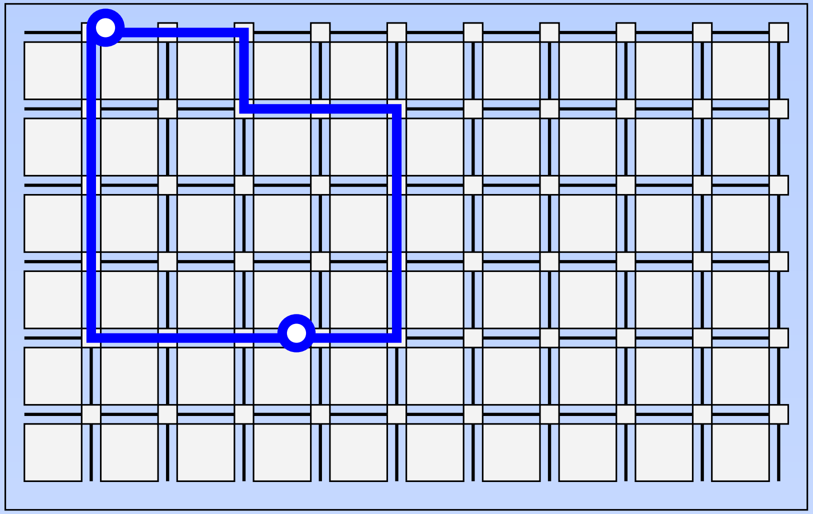


GB Traffic Composition



Load and performance is considered within a time window W cycles.

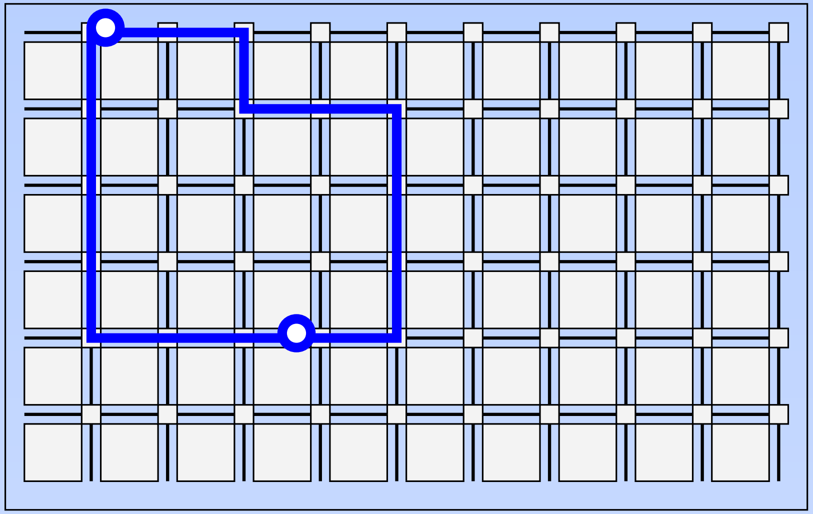
GB Traffic Composition



Load and performance is considered within a time window W cycles.

$v_{i,k}$: the load on link i by VC k in window W;

GB Traffic Composition

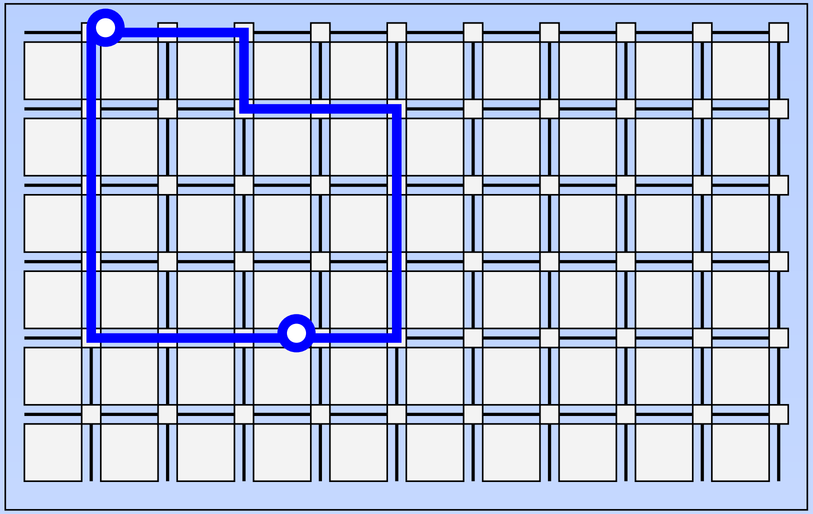


Load and performance is considered within a time window W cycles.

$v_{i,k}$: the load on link i by VC k in window W;

If VC k uses a single container, $v_{i,k} = 1$ on all links of the VC path;

GB Traffic Composition



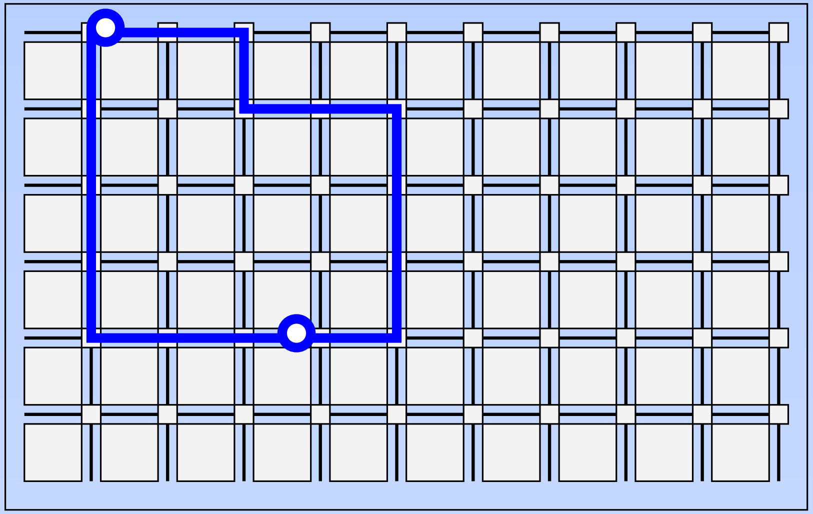
Load and performance is considered within a time window W cycles.

$v_{i,k}$: the load on link i by VC k in window W;

If VC k uses a single container, $v_{i,k} = 1$ on all links of the VC path;

$v_{i,k} \leq W$ for all links i and all VCs k.

GB Traffic Composition



Load and performance is considered within a time window W cycles.

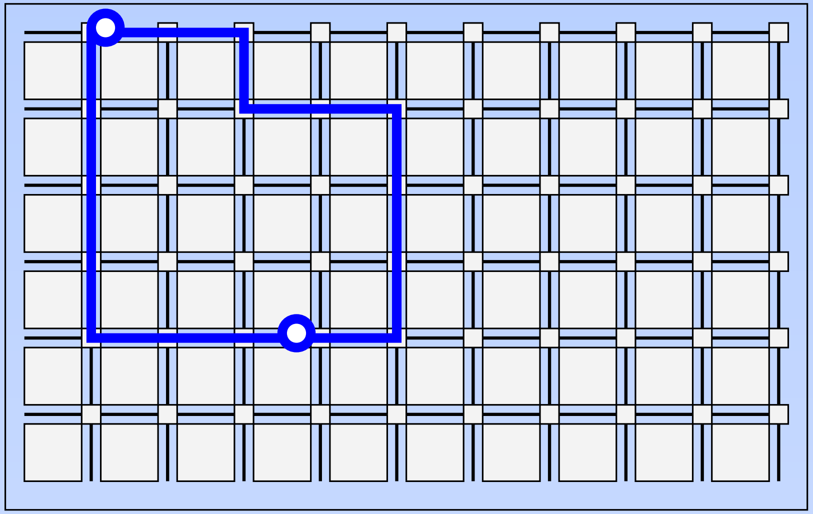
$v_{i,k}$: the load on link i by VC k in window W;

If VC k uses a single container, $v_{i,k} = 1$ on all links of the VC path;

$v_{i,k} \leq W$ for all links i and all VCs k.

$V_k = \sum_i v_{i,k}$ is the load of VC k on the network.

GB Traffic Composition



Load and performance is considered within a time window W cycles.

$v_{i,k}$: the load on link i by VC k in window W;

If VC k uses a single container, $v_{i,k} = 1$ on all links of the VC path;

$v_{i,k} \leq W$ for all links i and all VCs k.

$V_k = \sum_i v_{i,k}$ is the load of VC k on the network.

MoC Constraints:

$$\sum_k V_k \leq CG_{\mathrm{VC}} \leq WL$$

$$\sum_k v_{i,k} \leq CL_{\mathrm{VC}} \leq W \quad \text{for all links } i$$

Properties of GB Traffic

c_k: number of containers in the VC k;
len_k: the length of the VC in cycles.
$\mathrm{maxInit}_k$: maximum time between two containers.

Properties of GB Traffic

Bandwidth:

$$BW_k = \frac{c_k}{\mathrm{len}_k} \quad \frac{\text{packets}}{\text{cycle}}$$

c_k: number of containers in the VC k;
len_k: the length of the VC in cycles.
$\mathrm{maxInit}_k$: maximum time between two containers.

Properties of GB Traffic

Bandwidth:

$$BW_k = \frac{c_k}{\text{len}_k} \quad \frac{\text{packets}}{\text{cycle}}$$

Maximum Latency:

$$\text{maxLat}_k = \text{maxInit}_k + \text{len}_k$$

c_k: number of containers in the VC k;
len_k: the length of the VC in cycles.
maxInit_k: maximum time between two containers.

Properties of GB Traffic

Bandwidth:

$$BW_k = \frac{c_k}{\mathrm{len}_k} \quad \frac{\text{packets}}{\text{cycle}}$$

Maximum Latency:

$$\mathrm{maxLat}_k = \mathrm{maxInit}_k + \mathrm{len}_k$$

Average latency:

$$\mathrm{avgLat}_k = \frac{\mathrm{len}_k}{2c_k} + \mathrm{len}_k$$

c_k: number of containers in the VC k;
len_k: the length of the VC in cycles.
$\mathrm{maxInit}_k$: maximum time between two containers.

BE Traffic Composition - Network Load

BE traffic between source **A** and **B** is **channel based**.

BE Traffic Composition - Network Load

BE traffic between source **A** and **B** is **channel based**.
Channel h loads the network with

$$E_h = n_h \, d_h \, \delta$$

n_h: the number of packets **A** injects on channel h during the window W
d_h: the shortest distance between **A** and **B**
δ: the **average deflection factor**

BE Traffic Composition - Network Load

BE traffic between source **A** and **B** is **channel based**.
Channel h loads the network with

$$E_h = n_h \, d_h \, \delta$$

n_h: the number of packets **A** injects on channel h during the window W
d_h: the shortest distance between **A** and **B**
δ: the **average deflection factor**

$$\delta = \frac{\text{sum of traveling time of all packets}}{\text{sum of shortest path of all packets}}$$

BE Traffic Composition - Link Load

Channel h loads individual links with

$$e_{h,i} = n_h \, p_{h,i} \, \delta$$

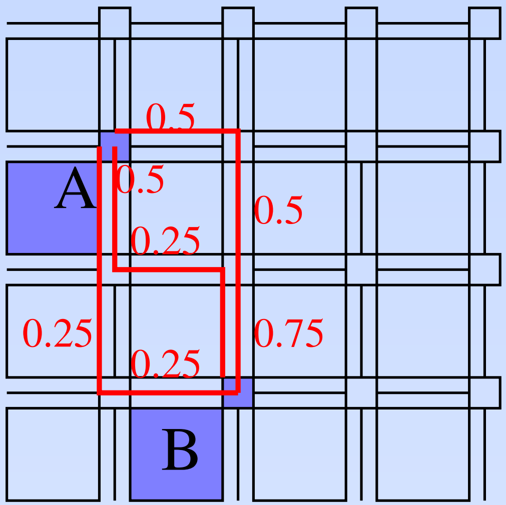
$p_{h,i}$ is the probability that link i is used by a packet of channel h on the shortest path between **A** and **B**.

BE Traffic Composition - Link Load

Channel h loads individual links with

$$e_{h,i} = n_h \, p_{h,i} \, \delta$$

$p_{h,i}$ is the probability that link i is used by a packet of channel h on the shortest path between **A** and **B**.



MoC Constraints for BE Traffic - Channels

$$\sum_h E_h \leq CG_{\mathrm{BE}} \leq LW - CG_{\mathrm{VC}}$$

$$\sum_h e_{i,h} \leq CL_{\mathrm{BE}} \leq W - CL_{\mathrm{VC}} \quad \text{for all links } i$$

MoC Constraints for BE Traffic - Resources

$$\sum_{h \in H_r^o} E_h \leq B_r^o$$

$$\sum_{h \in H_r^i} E_h \leq B_r^i$$

$$\sum_r B_r^o = \sum_r B_r^i \leq CG_{\mathrm{BE}}$$

B_r^o: Outgoing traffic budget for resource r
B_r^i: Incoming traffic budget for resource r

Traffic Ceiling

$$CG_{\mathrm{BE}} = \kappa(LW - CG_{\mathrm{VC}}) \quad \text{with } 0 \leq \kappa \leq 1$$

κ is the margin that allows for accommodation of temporal and spatial burstiness of traffic

Properties of BE Traffic

Under incoming and outgoing resource budget constraints;
n_h: number of emitted packets in each window on channel h;
d_h: shortest distance on channel h;
D: diameter of the network;
N: number of nodes in the network;

Properties of BE Traffic

Bandwidth:

$$BW_r = \sum_{h \in H_r^o} \frac{n_h}{W}$$

Under incoming and outgoing resource budget constraints;
n_h: number of emitted packets in each window on channel h;
d_h: shortest distance on channel h;
D: diameter of the network;
N: number of nodes in the network;

Properties of BE Traffic

Bandwidth:

$$BW_r = \sum_{h \in H_r^o} \frac{n_h}{W}$$

Maximum Latency:

$$\mathrm{maxLat}_k = 5DN$$

Under incoming and outgoing resource budget constraints;
n_h: number of emitted packets in each window on channel h;
d_h: shortest distance on channel h;
D: diameter of the network;
N: number of nodes in the network;

Properties of BE Traffic

Bandwidth:

$$BW_r = \sum_{h \in H_r^o} \frac{n_h}{W}$$

Maximum Latency:

$$\mathrm{maxLat}_k = 5DN$$

Average latency:

$$\mathrm{avgLat}_k = d_h\, \delta$$

Under incoming and outgoing resource budget constraints;
n_h: number of emitted packets in each window on channel h;
d_h: shortest distance on channel h;
D: diameter of the network;
N: number of nodes in the network;

Nostrum MoC Summary

MoC Constraints:

$$\begin{aligned}
\sum_k V_k &\leq CG_{\mathrm{VC}} \leq WL \\
\sum_k v_{i,k} &\leq CL_{\mathrm{VC}} \leq W \\
\sum_{h \in H_r^o} E_h &\leq B_r^o \\
\sum_{h \in H_r^i} E_h &\leq B_r^i \\
\sum_r B_r^o = \sum_r B_r^i &\leq CG_{\mathrm{BE}} = \kappa(LW - CG_{\mathrm{VC}})
\end{aligned}$$

MoC Properties:

$$\begin{aligned}
BW_k &= \frac{c_k}{\mathrm{len}_k} \quad \frac{\mathrm{packets}}{\mathrm{cycle}} \\
BW_r &= \sum_{h \in H_r^o} \frac{n_h}{W} \\
\mathrm{maxLat}_k &= \mathrm{maxInit}_k + \mathrm{len}_k \\
\mathrm{maxLat}_k &= 5DN \\
\mathrm{avgLat}_k &= \frac{\mathrm{len}_k}{2c_k} + \mathrm{len}_k \\
\mathrm{avgLat}_k &= d_h \, \delta
\end{aligned}$$

MoC Design Methodology

MoC Design Methodology

1. Divide the communication capacity between GB and BE;
   Set CG_{VC} and CL_{VC}.

MoC Design Methodology

1. Divide the communication capacity between GB and BE;
   Set CG_{VC} and CL_{VC}.
2. Set the BE traffic ceiling κ;

MoC Design Methodology

1. Divide the communication capacity between GB and BE; Set CG_{VC} and CL_{VC}.
2. Set the BE traffic ceiling κ;
3. Distribute the available BE traffic capacity among the resources; for a uniform allocation we have

$$B_r^o = \frac{\kappa LW}{N}$$

MoC Design Methodology

1. Divide the communication capacity between GB and BE; Set CG_{VC} and CL_{VC}.
2. Set the BE traffic ceiling κ;
3. Distribute the available BE traffic capacity among the resources; for a uniform allocation we have

$$B_r^o = \frac{\kappa LW}{N}$$

4. Determine δ empirically; Use D_1 as an upper bound for δ.

$$1 - 10^{-i} \text{ of all packets } p\text{: } \frac{\mathrm{delay}(p)}{\mathrm{mindelay}(p)} \leq D_i \qquad (1)$$

90% of all packets have a delay less or equal D_1.

Deflection Factor δ Measured for Various Traffic Budgets

B_r^o/W	16	20	30	40	50	60	70	80	90	100
0.05	1.02	1.02	1.02	1.02	1.02	1.02	1.02	1.02	1.02	1.02
0.10	1.04	1.05	1.04	1.06	1.06	1.05	1.05	1.06	1.05	1.05
0.15	1.07	1.08	1.08	1.11	1.10	1.09	1.09	1.10	1.09	1.09
0.20	1.10	1.12	1.11	1.17	1.15	1.15	1.14	1.16	1.14	1.14
0.25	1.14	1.17	1.15	1.26	1.24	1.23	1.22	1.24	1.22	*sat.*
0.30	1.18	1.22	1.20	1.46	1.41	1.36	1.33	1.33	1.29	*sat.*
0.35	1.23	1.28	1.27	1.78	1.65	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*
0.40	1.28	1.36	1.40	1.98	1.84	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*
0.45	1.35	1.54	1.75	1.99	1.85	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*
0.50	1.57	1.98	1.82	1.99	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*

δ depending on traffic budget and network size;

D_1 Measured for Various Traffic Budgets

B_r^o/W	16	20	30	40	50	60	70	80	90	100
0.05	1.12	1.12	1.12	1.15	1.15	1.15	1.16	1.16	1.11	1.11
0.10	1.12	1.12	1.15	1.25	1.23	1.23	1.23	1.27	1.23	1.23
0.15	1.12	1.28	1.30	1.41	1.41	1.36	1.35	1.41	1.35	1.35
0.20	1.36	1.44	1.40	1.46	1.46	1.46	1.46	1.46	1.47	1.55
0.25	1.44	1.44	1.45	1.65	1.64	1.71	1.80	2.35	3.46	*sat.*
0.30	1.44	1.60	1.61	4.65	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*
0.35	1.60	1.61	1.72	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*
0.40	1.60	1.81	6.10	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*
0.45	1.80	3.44	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*
0.50	6.17	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*	*sat.*

δ depending on traffic budget and network size;

Deflection Factor δ Measured for Various Traffic Patterns

B_r^o/W	Uniform	BitComp.	BitRev.	BitRot.	BitShuf.	BitTrans.	BUni.
0.05	1.02	1.01	1.01	1.00	1.01	1.00	1.00
0.10	1.04	1.02	1.02	1.00	1.02	1.01	1.02
0.15	1.07	1.05	1.03	1.00	1.03	1.02	1.03
0.20	1.10	1.07	1.04	1.01	1.05	1.02	1.05
0.25	1.14	1.11	1.04	1.01	1.06	1.02	1.05
0.30	1.18	1.17	1.06	1.02	1.09	1.02	1.07
0.35	1.23	1.24	1.07	1.02	1.11	1.03	1.08
0.40	1.28	1.34	1.09	1.03	1.15	1.03	1.09
0.45	1.35	1.62	1.11	1.04	1.18	1.04	1.11
0.50	1.57	1.62	1.14	1.04	1.21	1.05	1.12

δ depending on traffic budget and traffic pattern for a 4×4 net;

How to Use the Nostrum MoC

How to Use the Nostrum MoC

For a new platform instance:

How to Use the Nostrum MoC

For a new platform instance:

1. Model the application as a process graph
2. Configure the NoC platform
3. Mapping
   - Map the processes to resource nodes
   - Map the process communication to network services
4. Determine κ, assign the traffic budgets to channels and resource nodes
5. Derive the deflection factor empirically
6. Back-annotate the process graph with performance figures

How to Use the Nostrum MoC

For a new platform instance:

1. Model the application as a process graph
2. Configure the NoC platform
3. Mapping
   - Map the processes to resource nodes
   - Map the process communication to network services
4. Determine κ, assign the traffic budgets to channels and resource nodes
5. Derive the deflection factor empirically
6. Back-annotate the process graph with performance figures

For an existing platform instance:

1. Model the application as a process graph
2. Mapping
   - Map the processes to resource nodes
   - Map the process communication to network services
3. assign the traffic budgets to channels and resource nodes
4. Back-annotate the process graph with performance figures

Summary

Summary

- An MoC provides communication performance characteristics for a NoC platform

Summary

- An MoC provides communication performance characteristics for a NoC platform
- Allows for composition of traffic with predictable performance

Summary

- An MoC provides communication performance characteristics for a NoC platform
- Allows for composition of traffic with predictable performance
- Based on contracts between service users (nodes, applications) and service providers (communication network)

Summary - MoC Design Options

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)
- Network characteristics

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)
- Network characteristics
  - ⋆ Deterministic vs adaptive routing

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)
- Network characteristics
  - ⋆ Deterministic vs adaptive routing
  - ⋆ Cost of guarantees

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)
- Network characteristics
  - Deterministic vs adaptive routing
  - Cost of guarantees
  - Delay characteristics under load

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)
- Network characteristics
  - ⋆ Deterministic vs adaptive routing
  - ⋆ Cost of guarantees
  - ⋆ Delay characteristics under load
- Application characteristics

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)
- Network characteristics
  - Deterministic vs adaptive routing
  - Cost of guarantees
  - Delay characteristics under load
- Application characteristics
  - Traffic scenarios

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)
- Network characteristics
  - ⋆ Deterministic vs adaptive routing
  - ⋆ Cost of guarantees
  - ⋆ Delay characteristics under load
- Application characteristics
  - ⋆ Traffic scenarios
  - ⋆ Well known or unknown

Summary - MoC Design Options

- Link based (GB) vs. node based traffic allocation (BE)
- Central planning (GB) vs. distribution of budgets (BE)
- Design time allocation (GB) vs. run time allocation (BE)
- Accurate prediction (GB) vs. estimated bounds (BE)
- Network characteristics
  - ⋆ Deterministic vs adaptive routing
  - ⋆ Cost of guarantees
  - ⋆ Delay characteristics under load
- Application characteristics
  - ⋆ Traffic scenarios
  - ⋆ Well known or unknown
  - ⋆ Real-time requirements

Conclusion

Conclusion

- Designing a MoC for NoC platforms is **necessary**

Conclusion

- Designing a MoC for NoC platforms is **necessary**
- Designing a MoC for NoC platforms is **feasible**

Conclusion

- Designing a MoC for NoC platforms is **necessary**
- Designing a MoC for NoC platforms is **feasible**
- It depends on the network characteristics

Conclusion

- Designing a MoC for NoC platforms is **necessary**
- Designing a MoC for NoC platforms is **feasible**
- It depends on the network characteristics
- It depends on the application characteristics