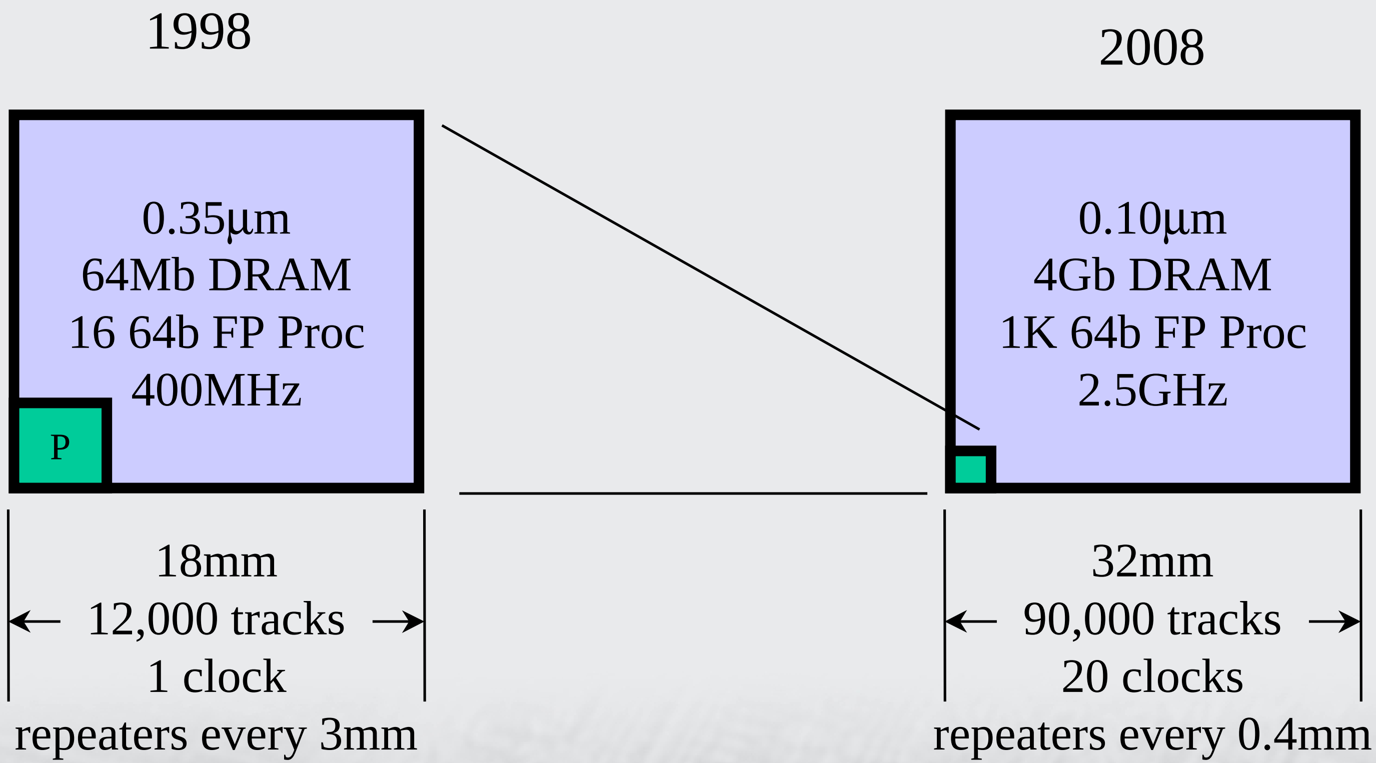
OccN

A Methodology for NoC

AST Grenoble (with collaborators)
Advanced Modeling group

Technology scaling makes communication *the* scarce resource



Source: William J. Dally (Stanford University)

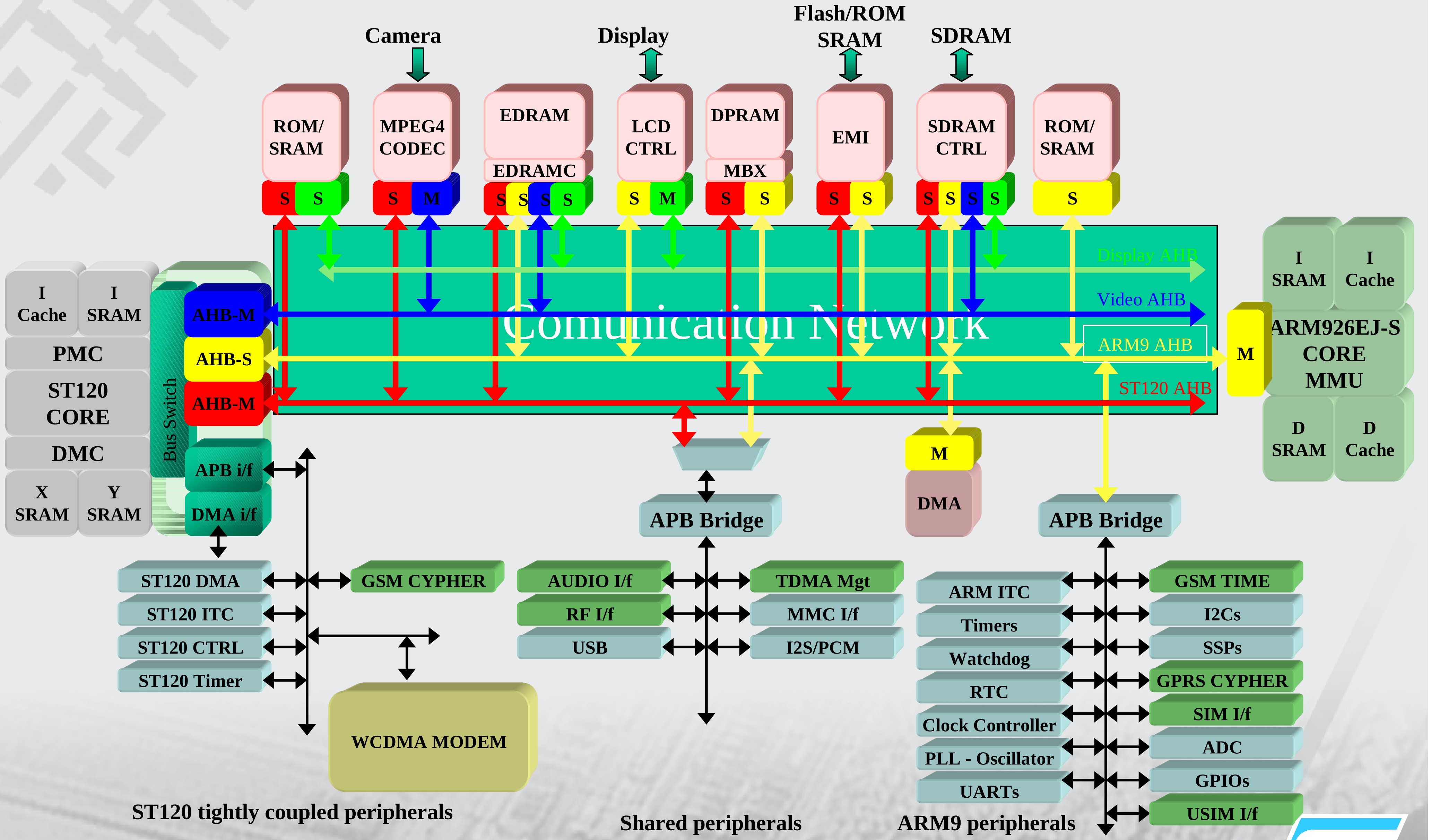
Platforms & Architectures for System Design

Function on chip	Many chips on PCBs External Connections External storage
Algorithm on a chip	Hardwired computation Hardwired interconnectivity Centralized storage
System on a chip	Programmable computation Hardwired interconnectivity Partially distributed storage
Network on a chip	Programmable computation Programmable interconnectivity Fully distributed storage

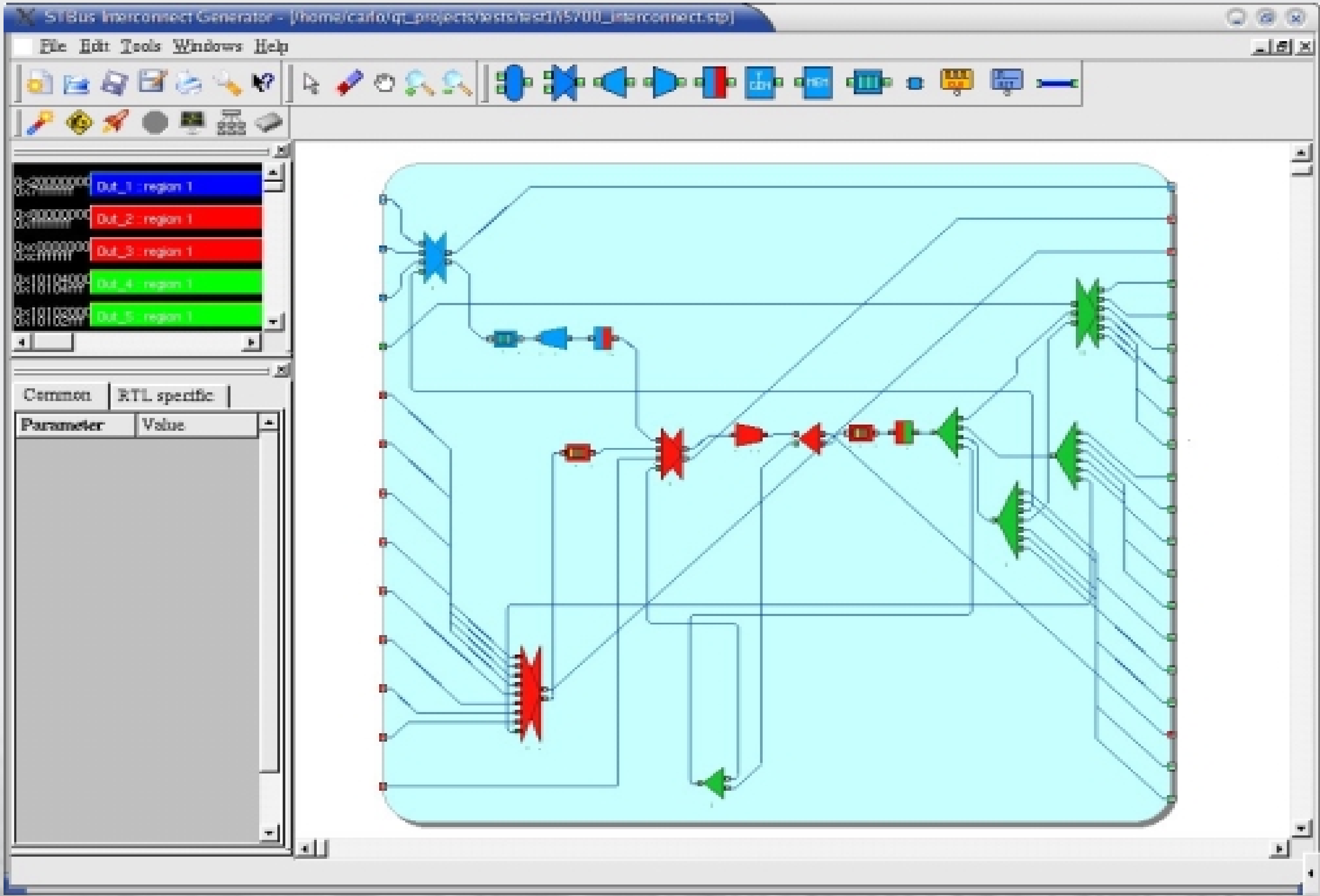
Some definitions

- NoC is a future view as a micro-network of components [Benini and De-Micheli]
- NoC is a parallel computation platform with a task/process level of parallelism; suitable only for high-volume products [J.P Soininen and H Heusala]
- NoC is a set of computation node connected via sophisticated on-chip communication network [A.A Jerraya et alt.]

MP-SoC architecture: AMBA based



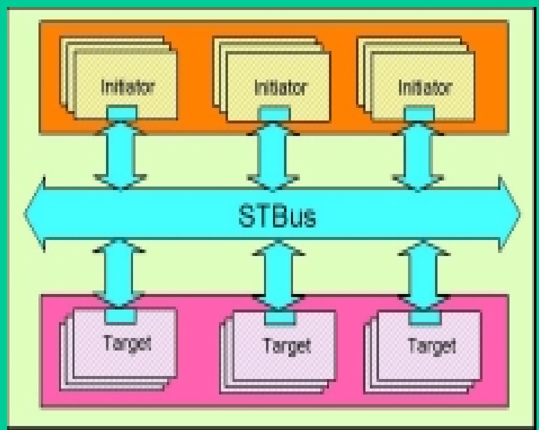
Or STBUS based



From SoC to NoC

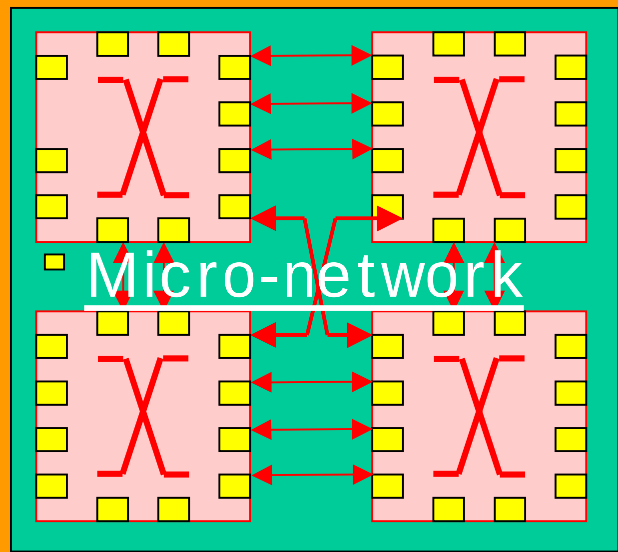
System on a chip

Programmable computation
Hardwired interconnectivity
Partially distributed storage

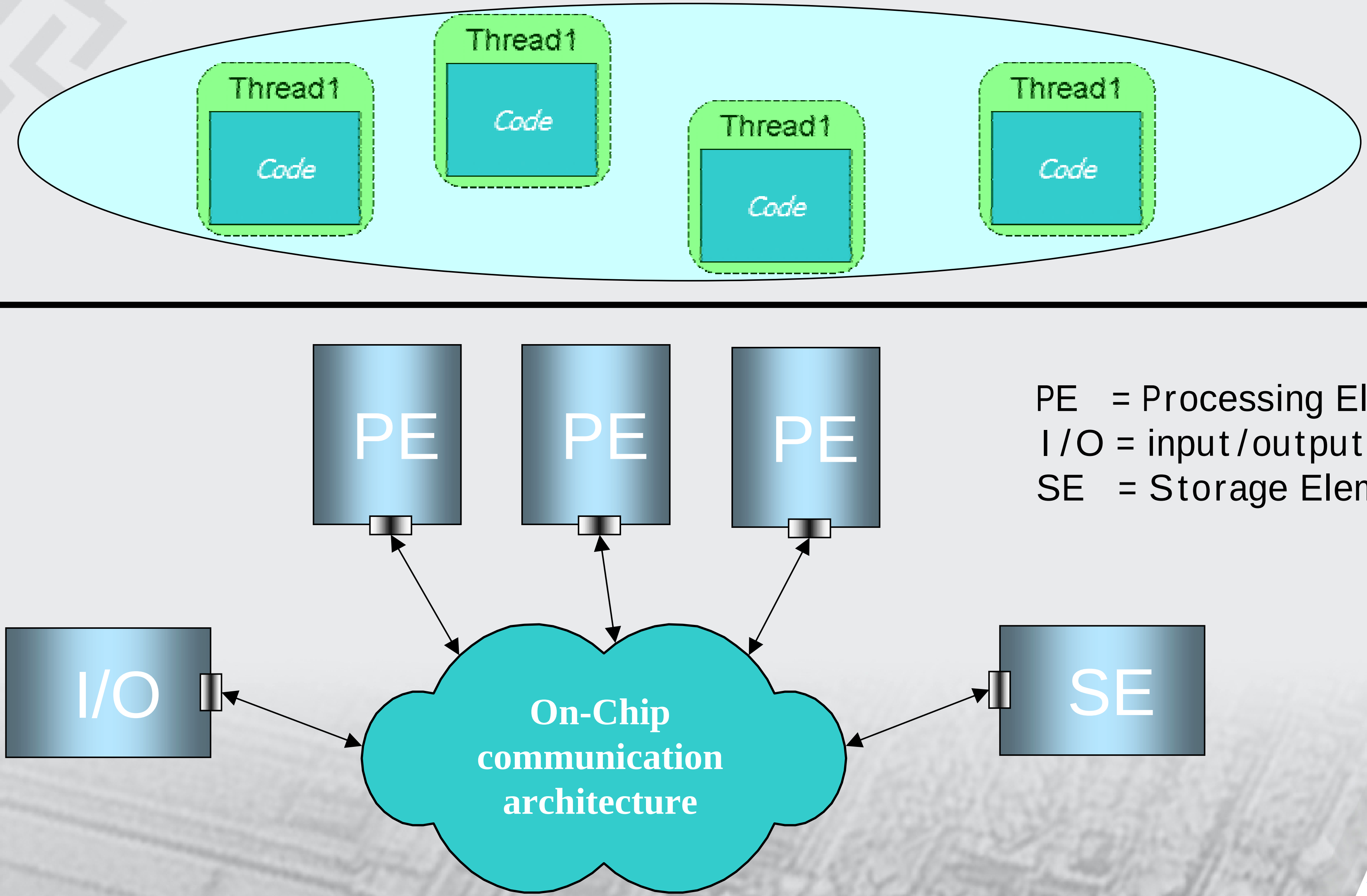


Network on a chip

Programmable computation
Programmable interconnectivity
Fully distributed storage

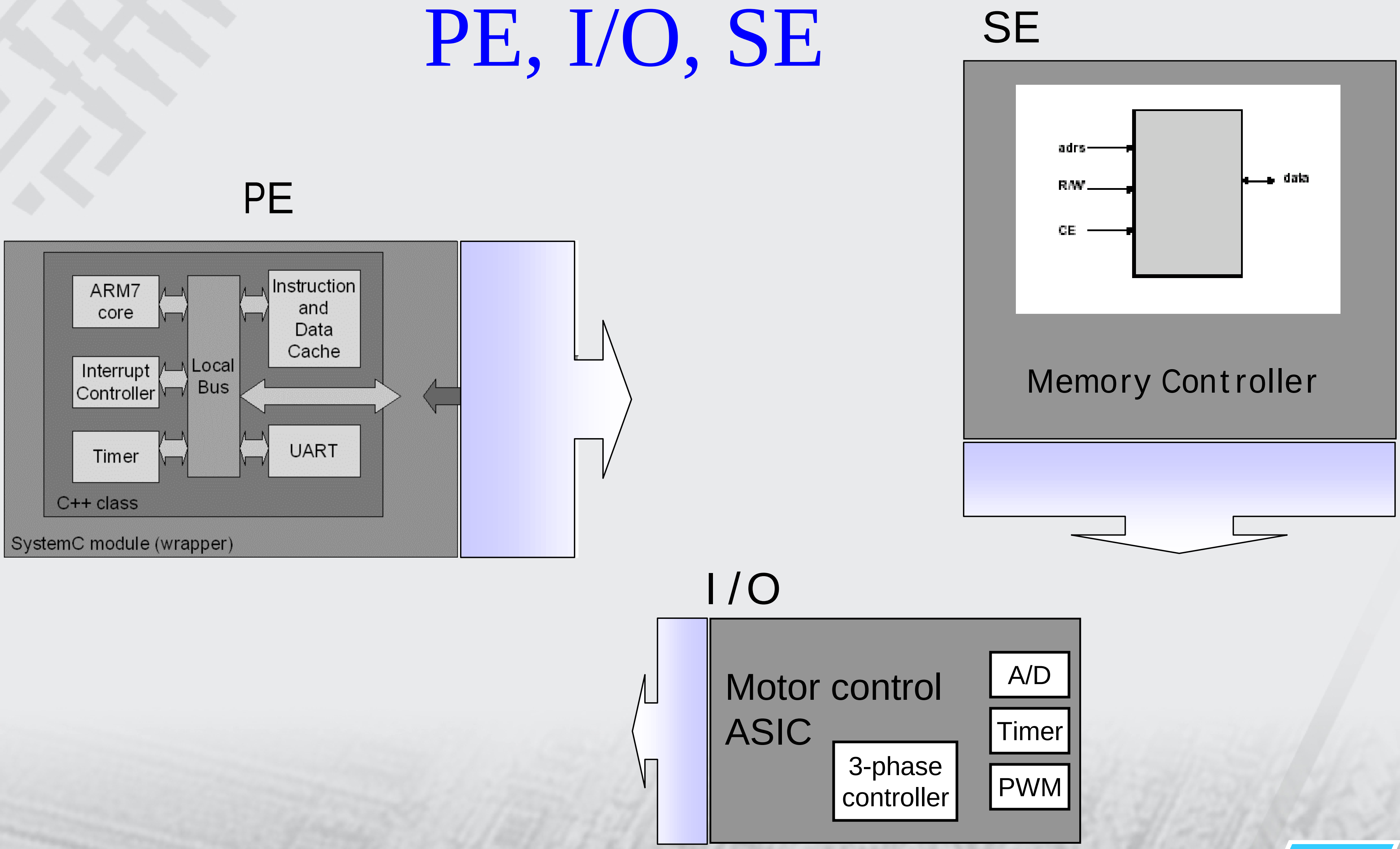


NoC overview



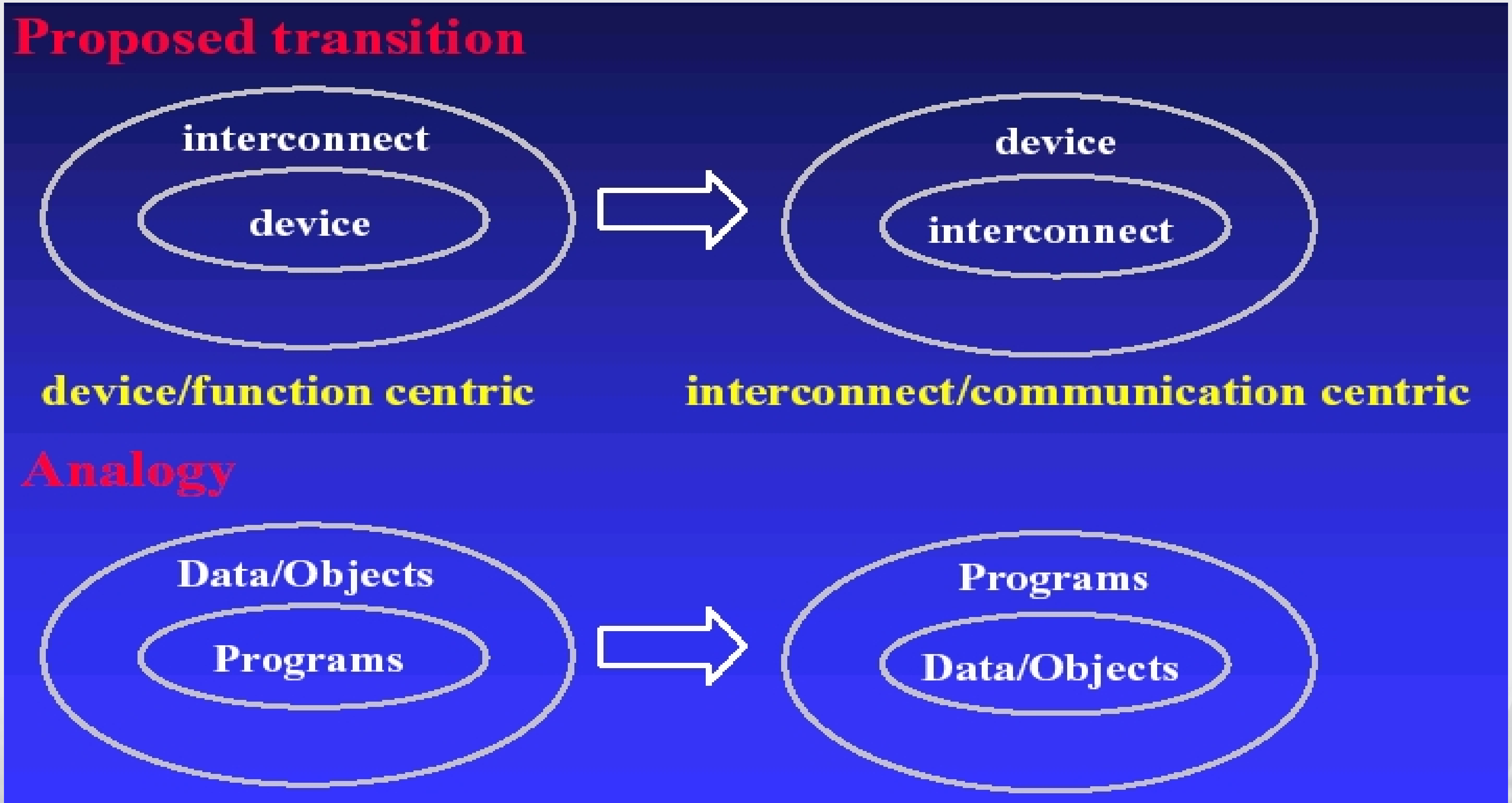
PE = Processing Element
I/O = input/output
SE = Storage Element

PE, I/O, SE



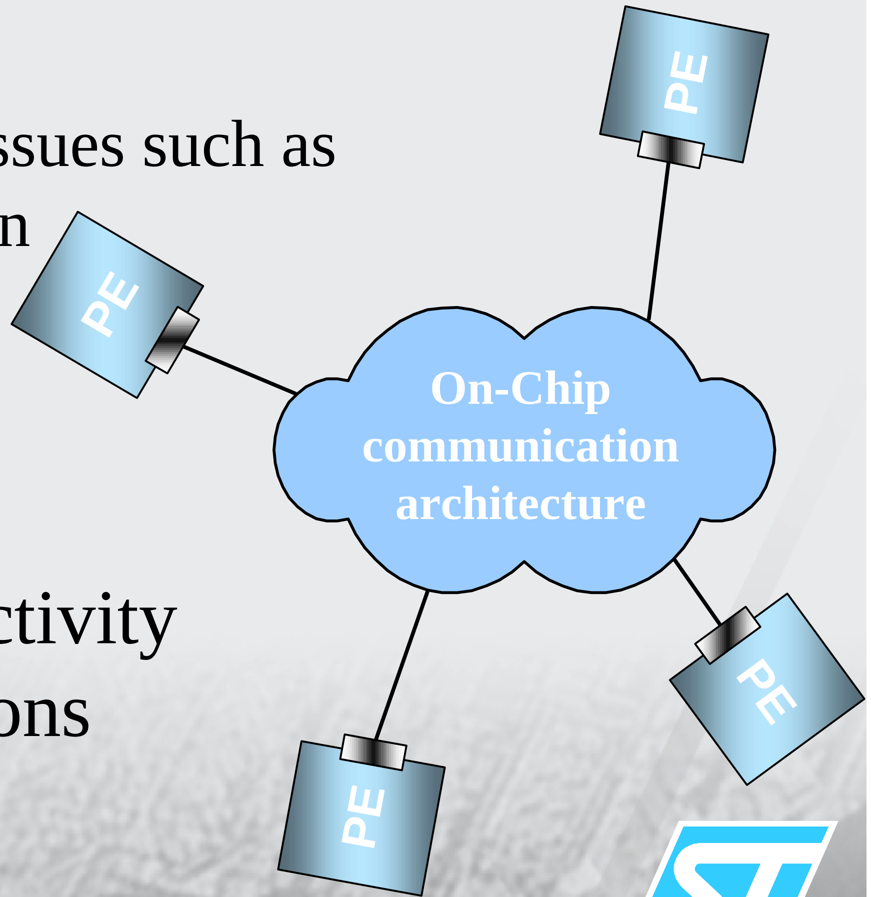
SE
adrs
R/W
CE
data
Memory Controller
PE
ARM7 core
Instruction and Data Cache
Interrupt Controller
Local Bus
Timer
UART
C++ class
SystemC module (wrapper)
I/O
Motor control ASIC
A/D
Timer
PWM
3-phase controller

Communication Centric Methodology

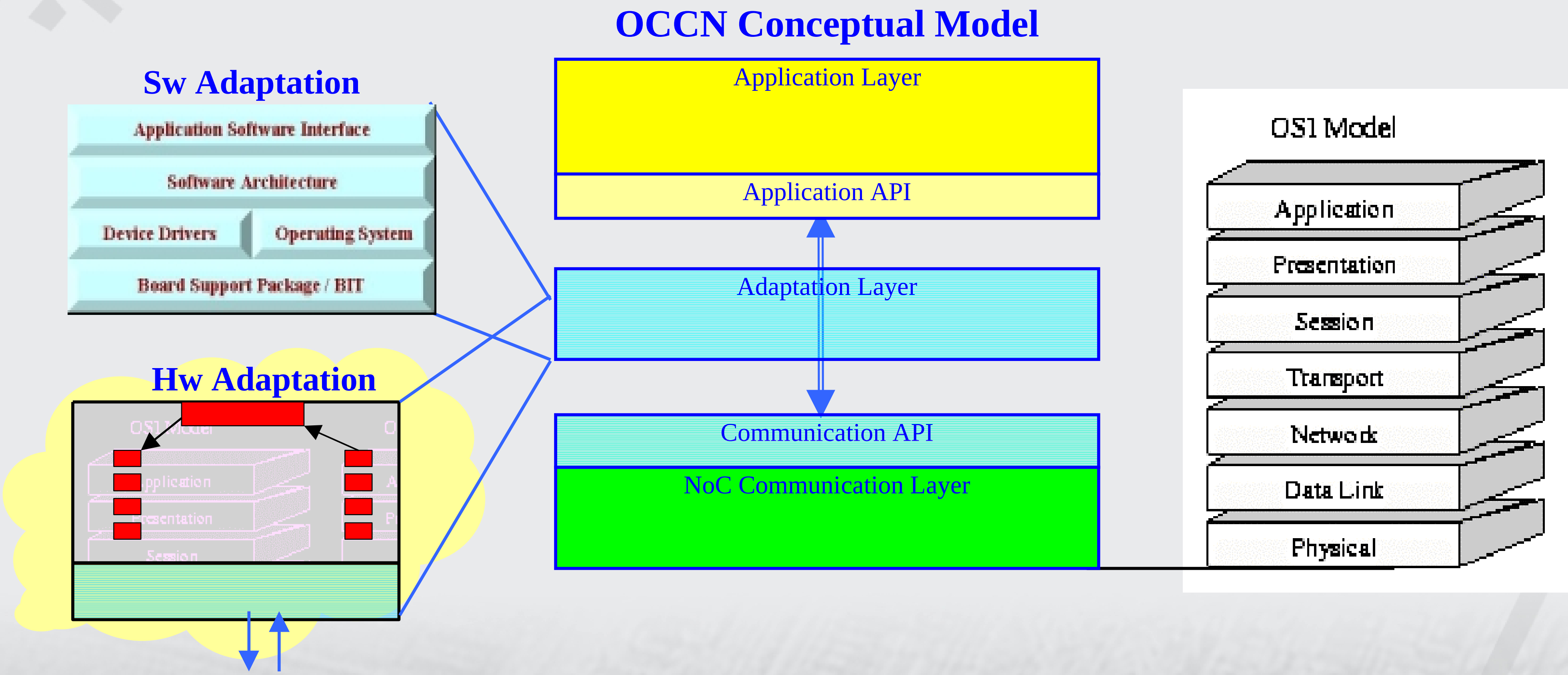


OCCN : methodology for communication modeling

- Generic communication-centric design methodology based on C++ and SystemC
- OCCN addresses
  - high level performance modeling issues such as speed, latency and power estimation
  - modeling productivity
  - model portability
  - simulation speed-up
- OCCN is an on-going research activity between several R&D organizations

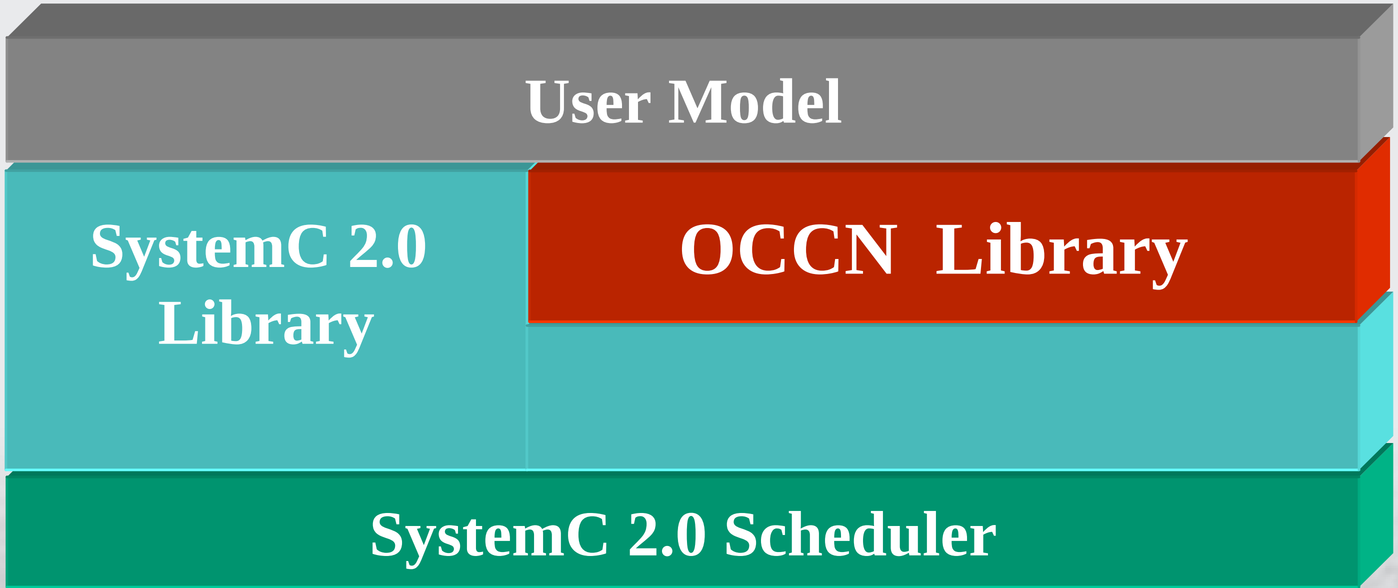


OCCN Conceptual Model



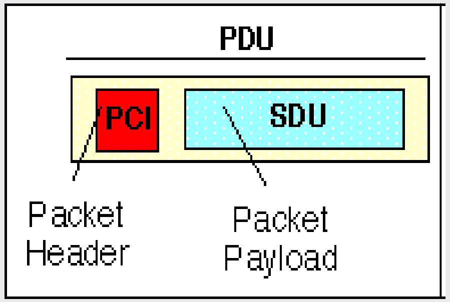
What is OCCN ?

OCCN aims at IC modeling, providing a real **object-oriented methodology** based on a **C++ library** and a fully documented design flow **based on SystemC 2.0**

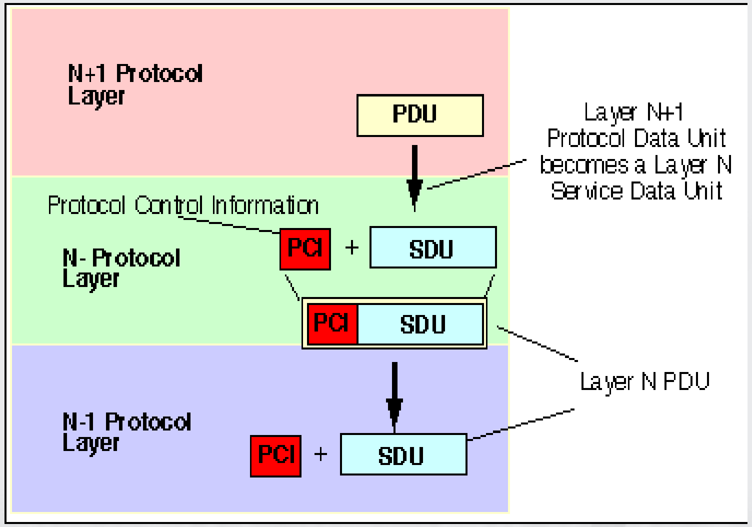


OCCN core: the PDU

- Protocol = syntax + semantics
  - syntax = PDU
  - semantics = how the PDU are exchanged



- The PDUs exchanged have two parts:
  - a header also known as the Protocol Control Information (PCI)
  - a payload also known as a Service Data Unit (SDU)



- Several operators are defined for handling protocol operations (segmentation/reassembling)

- Syntax example

```
Struct MyHeader { int P; char T};
Pdu<MyHeader,char,4> my_pdu;
```

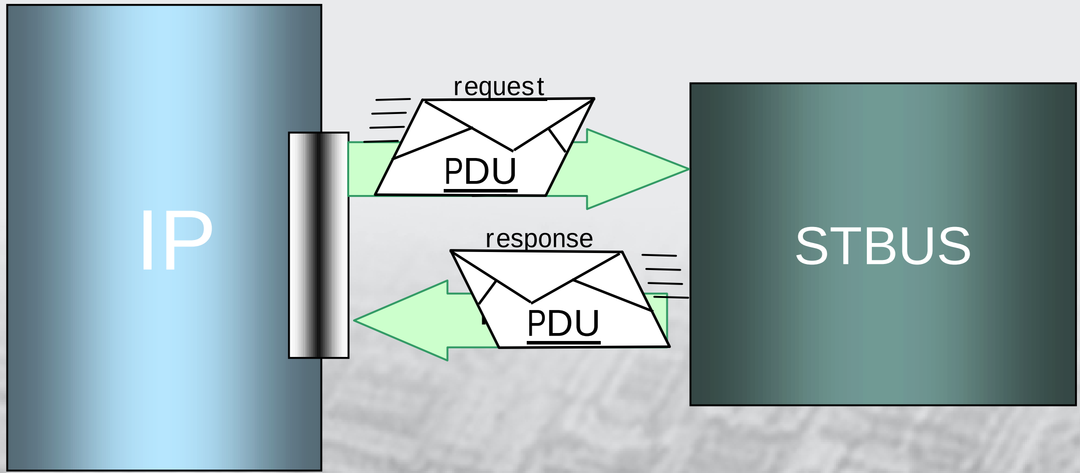
Generic representation of a connection

- Any connection of a module to the communication node (network) is based on 2 sets of PDU
  - Pdu< PCIRequest,uint32>
  - Pdu<PCIResponse,uint32>

- The PCI and SDU sets are defined according to the bus specification and thus are specific to a model. For instance it will be different for an AHB model and an STBUS model

```
Struct
{
  bool Request;
  unsigned int address;
  unsigned char Opcode;
  bool Lock;
  unsigned char SrcId;
   …
}
PCIRequest
```

```
Struct
{
  bool ReturnRequest;
  unsigned char ReturnOpcode;
  unsigned char SrcId;
  …
}
PCIResponse
```



OCCN core : API syntax

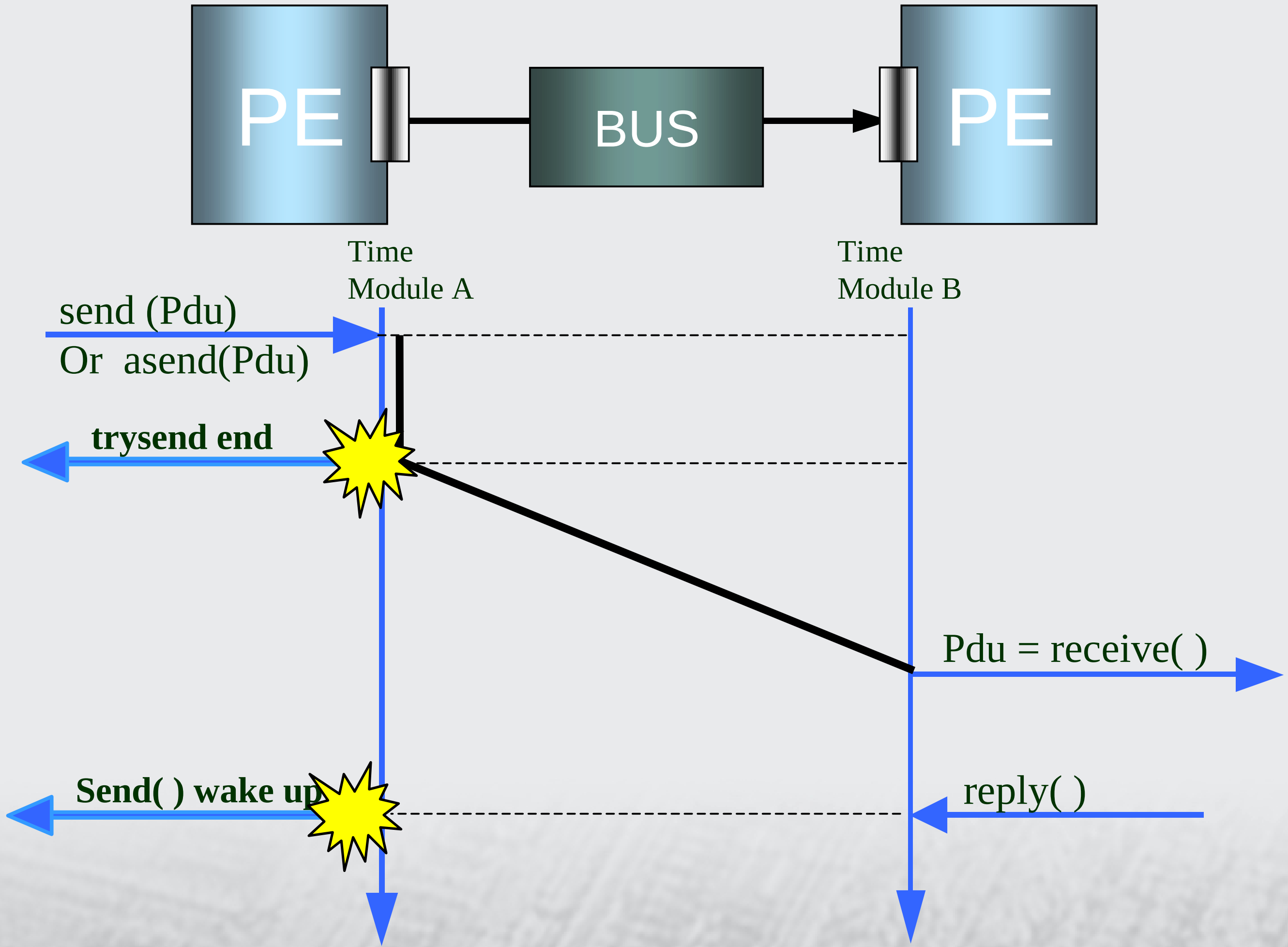
simple message passing API


```
void asend(Pdu<>* p)
void send(Pdu<>* p)
void send(Pdu<>* p, sc_time& time_out, bool& sent)
void asend(Pdu<>* p, sc_time& time_out, bool& sent)
```

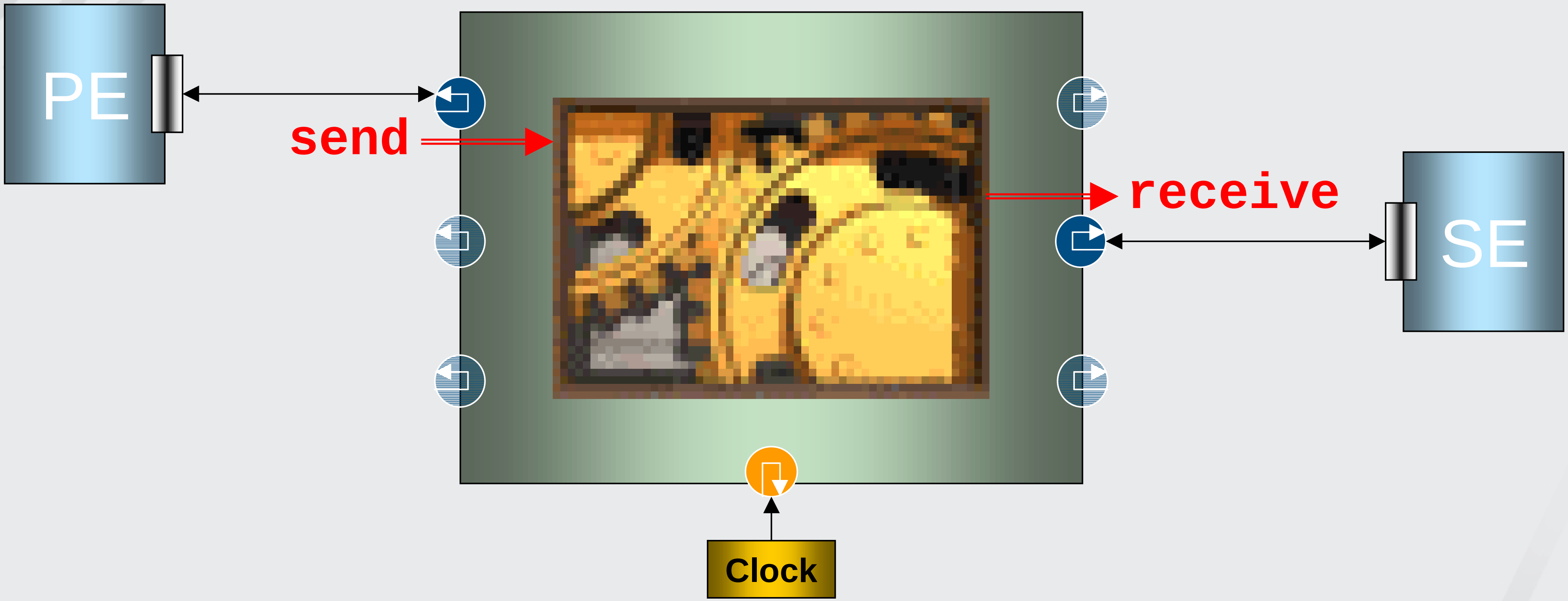
```
Pdu<>* receive()
void reply()
void reply(uint nb_cycles)
void reply(sc_time& delay)
Pdu<>* receive(sc_time& time_out, bool& received)
```

OCCN core : API semantic

with or without acknowledge



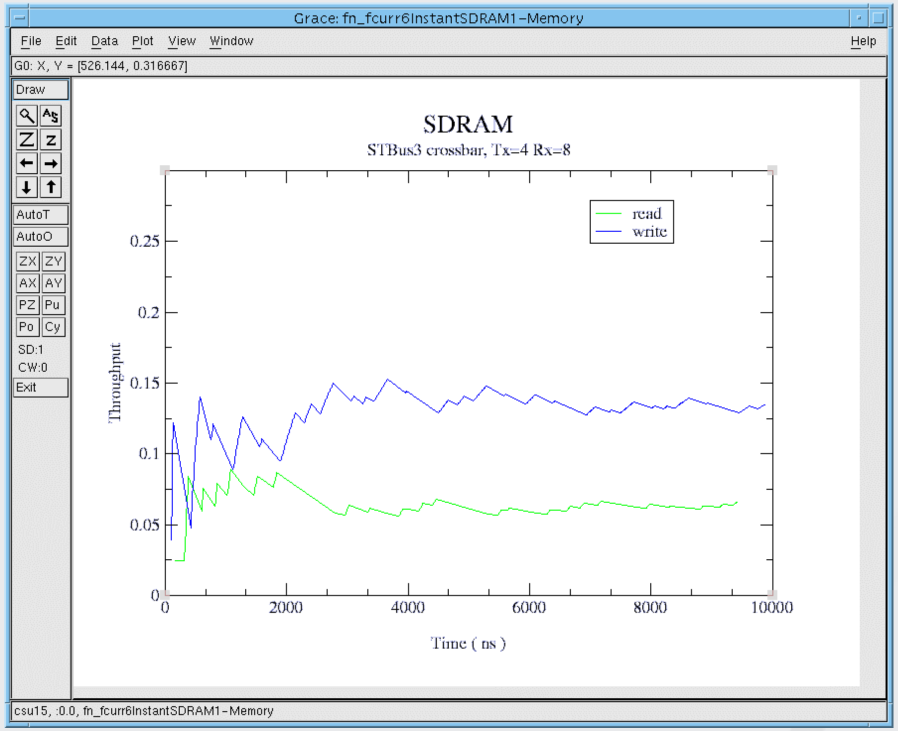
OCCN core : protocol state machine centralized



- allows synchronous and asynchronous communication modeling
- For synchronous com, PEs don’t need to be connected to the clock signal

Performance measurement with Grace

- XY graph, XY charts, pie charts, polar, and fixed graphs.
- User-defined scaling, ticks, labels, symbols, line styles, fonts, colors.
- Merging, validation, cumulative average, curve fitting, regression, filtering, DFT/FFT, cross/auto-correlation, sorting, interpolation, integration, differentiation...
- Internal language, and dynamic module loading (C, Fortran, etc).
- Hardcopy support with PS, PDF, GIF and PNM formats.



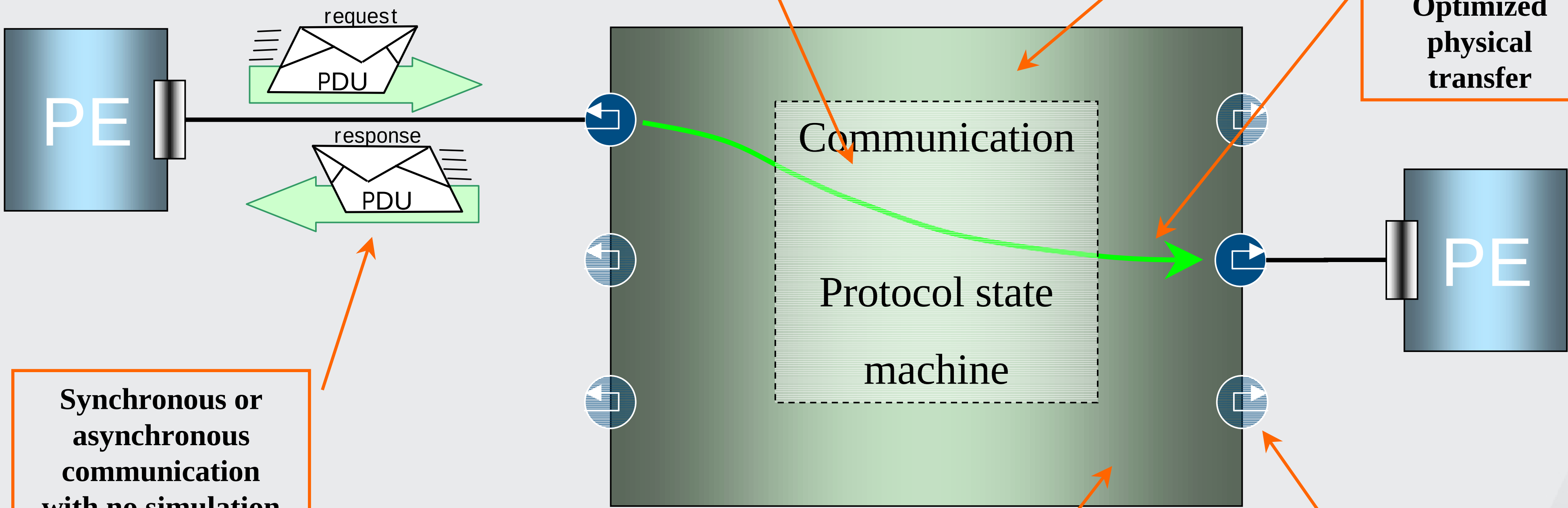
OCCN framework keypoints

Message passing oriented communication

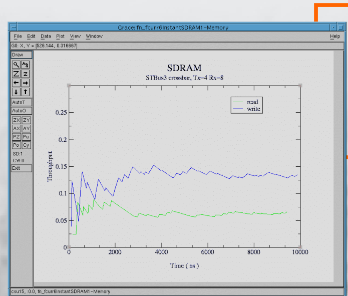
Protocol state machine centralized

Generic structure for communication node

Optimized physical transfer



Synchronous or asynchronous communication with no simulation overhead

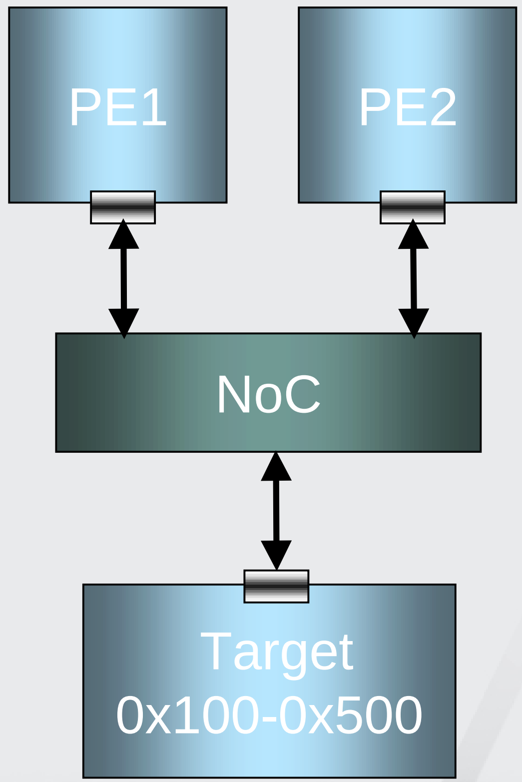


System performance metrics

Dynamic number of bound PE

MP SoC architecture

```
main()
{
        sc_clock my_clock(10, SC_NS);
        PE pe1, pe2;
        SE se1;
        NoC occa();

        occa.clk(my_clock);
        pe1(occa);
        pe2(occa);
        se(occa);

        occa.set_address_range(&se1.port,0x100,0x500);
        occa.set_priority(&pe1.port, 2);
        occa.set_priority(&pe2.port, 5);
        sc_start(-1);
}
```

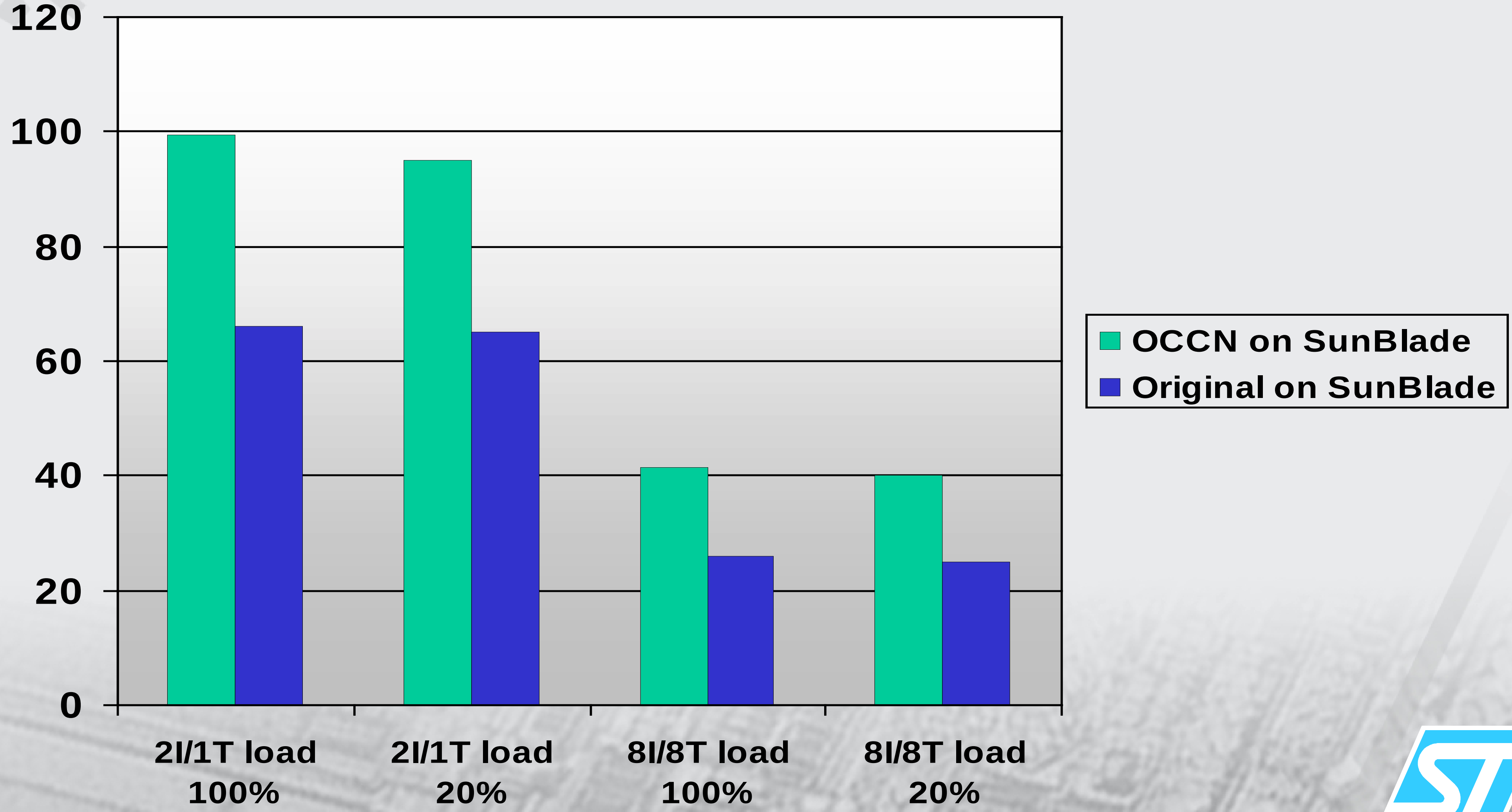


OCCN: PE code example

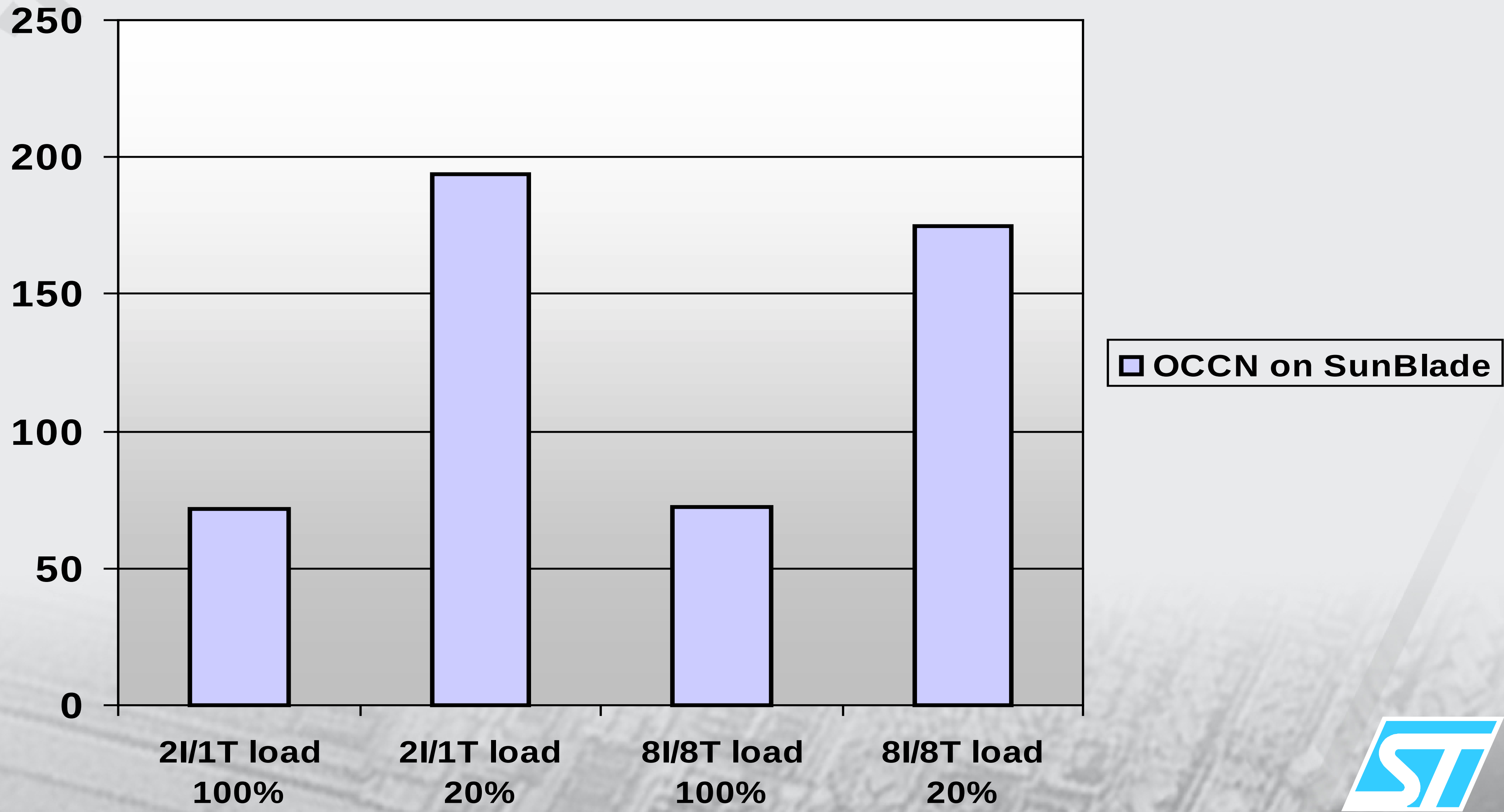
```
#include “producer.h”
producer::producer(sc_module_name name) : sc_module(name)
{SC_THREAD(read);}


void producer::read() {
  char c;
  Pdu<char>* msg;
  while (cin.get(c)) {
       msg = new Pdu<char>;
       // producer sends c
      *msg = c;
       out.send(msg);
  } // after the send the msg is not usable
}
```

Protocol inlining:
protocol is automatic generated

Speed-up for STBUS T3 model
(benefit of message passing)



Speed-up for AHB 2.0 model (benefits of message passing & centralization of the protocol state machine)



Conclusion

- OCCN
  - based on SystemC methodology
  - open & flexible API
  - simulation speed-up
  - reusability
  - productivity
  - communication architecture exploration
- Public part -> http://occn.sourceforge.net
- Related work: Gigascale Silicon Research Center (GSRC) effort Princeton University: MESCAL Project

  Modern Embedded Systems Compilers Architectures and Languages

  Princeton and UC Berkeley

Collaborations

Bologna University

Pisa University

Ancona University

- Internal Cooperation: AST R&I(R. Zafalon), CMG