ARTIST Workshop at DATE'06
W4: "Design Issues in Distributed, Communication-Centric Systems"

Information Society Technologies

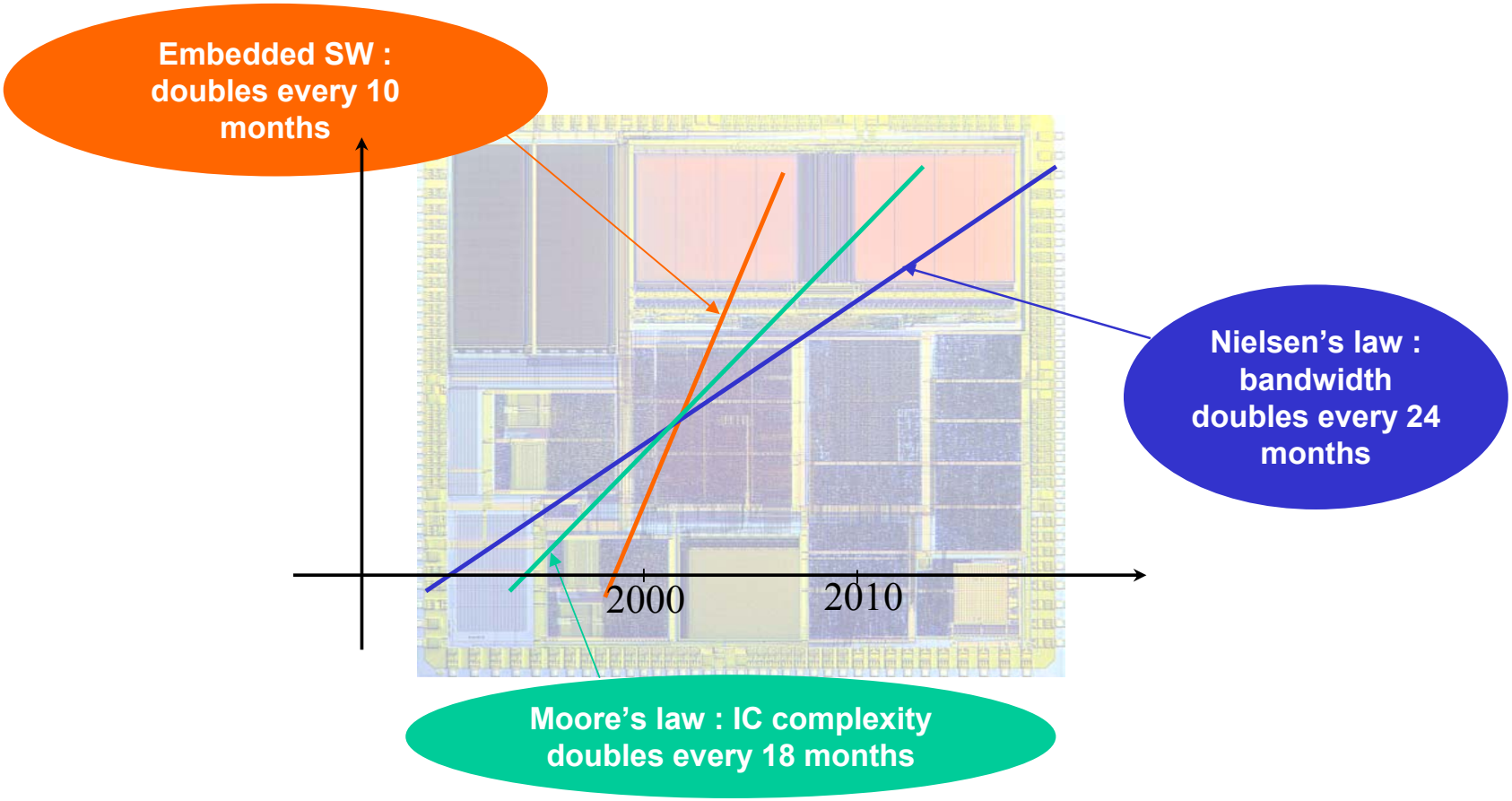
Communication Oriented Design Flow

Marcello Coppola
Head of AST Grenoble Lab
STMicroelectronics

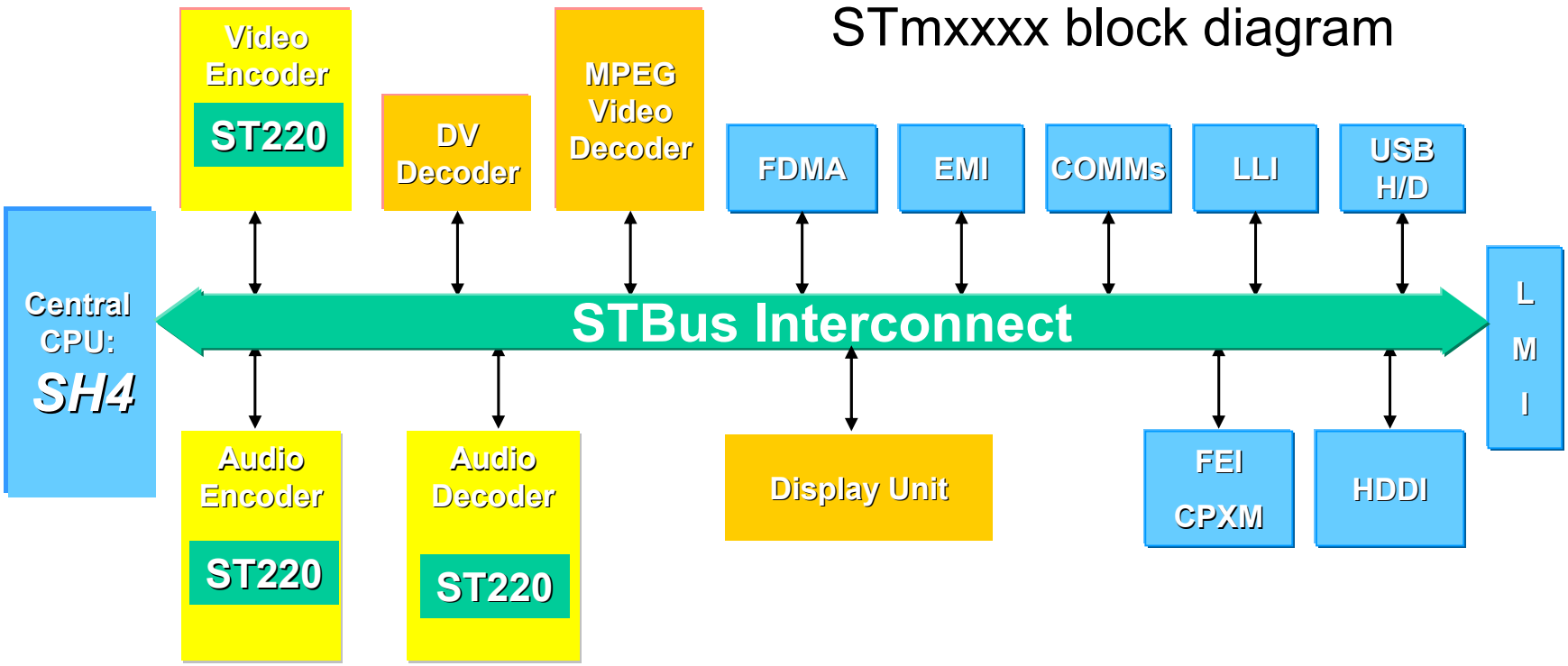
Outline

- MPSOC trends
- NoC
  - Introduction
  - STBus Genkit and STNoC™ extensions
  - TLM Modeling
  - OCCN Modeling
- Conclusion

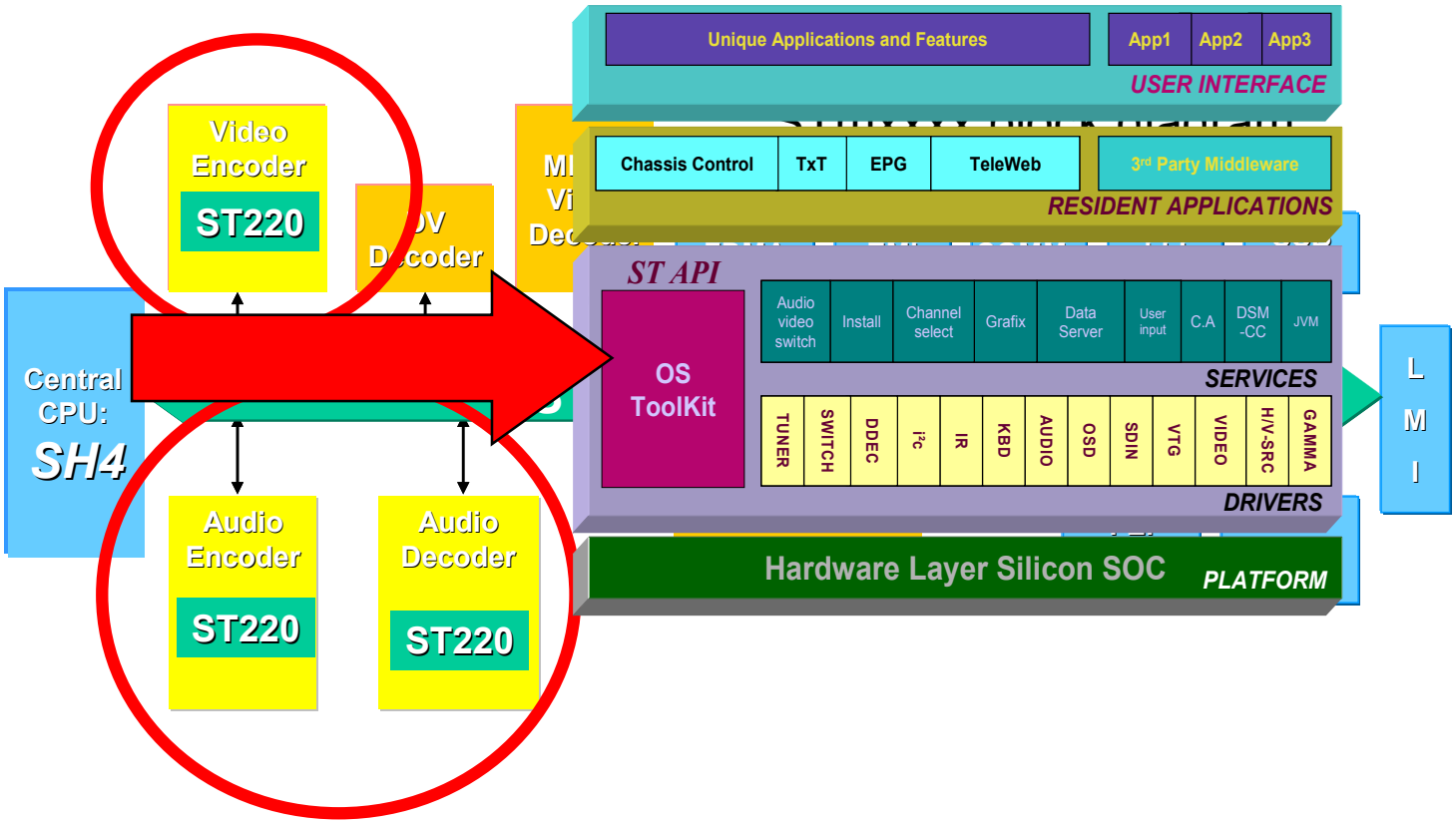
SoC challanges



Multimedia Chip today: HW view



Multimedia Chip today: SW view



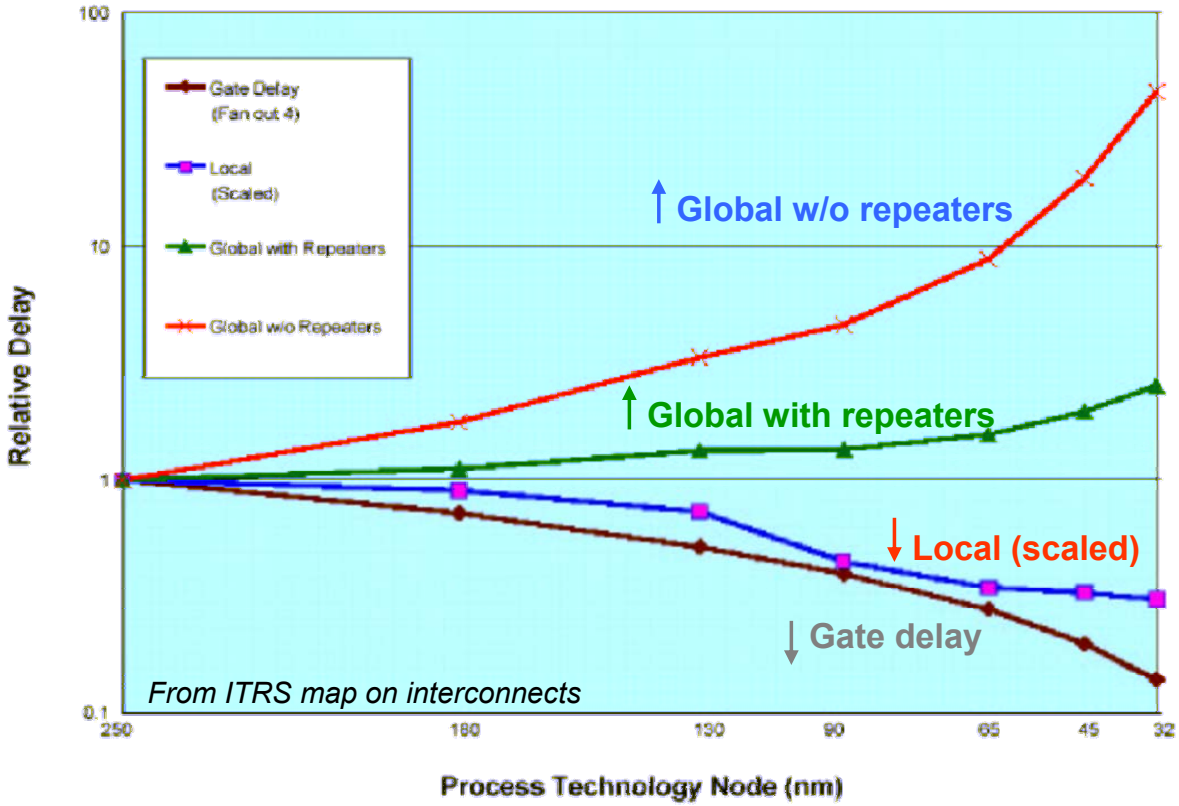
Multi Processor SoC issues

- Scalability is limited by:
  - Complexity
  - Programming models limitations
  - Inadequacy of on-chip communication
  - Power consumption, both dynamic and static
  - Technology
- Costs issues
  - SW complexity and verification
  - HW verification time and costs

SoC Interconnect issues

- Complexity
  - Interconnect must scale to manage increased number of IPs and interfaces in the system
  - Interconnect must support QoS for reduced interference of applications at the interconnect level leading to reduced effort for performance verification
  - Interconnect technology must support ease of generation
- Physical integration
  - Interconnect must deal with increased timing closure issues
  - Interconnect must reduce wire congestion and wire cost
- Bandwidth
  - Interconnect must support operation at high clock frequency
- Low Power
  - Interconnect must support low power operation

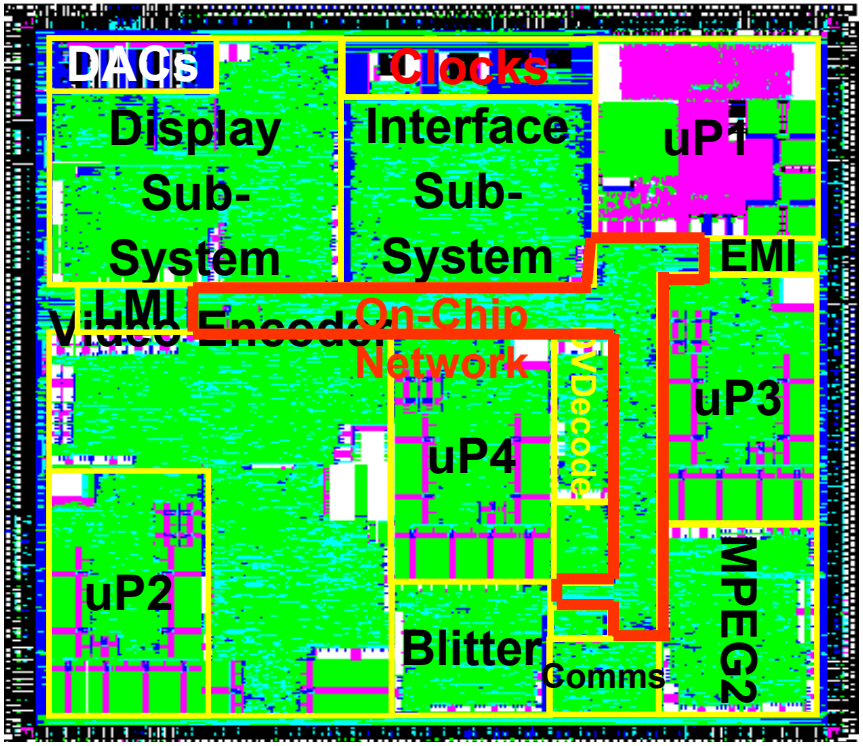
Interconnect: Global wire delay issues



- Global wires delay increase !
  (Logic and local wires delay downscale)

Interconnect: area issues

- Stbus based,more than 40 bus targets/initiators
- Large interconnect area
- Cross-talks problems
- Clock skew



Network on Chip

- NoC is a consolidated **research topic**

  (first explicit literature contributions in year 2000)

- Convergence of **multiple disciplines**: NoC is a synthesis

  (On-chip communication, Multiprocessing, Networking, Parallel computing)

NoC is a layered approach for on-chip communication design proposed in literature to cope with issues of current SoC architectures

NoC Paradigm

- The NoC paradigm is one enabling the integration of a large number of computational logic, and storage blocks in a SoC
- The adoption and deployment of NoCs face important issues relating to design and test methodologies and automation tools.
- Effective on-chip implementation of NoC paradigms requires developing and deploying a whole new set of infrastructure IPs and supporting tools and methodologies.

NoC is a communication-centric approach

STNoC™ : flexible packet based

- **Performance:** emulate crossbar but at lower cost and higher frequency
- **Technology issue:** reduce wires, split long connections
- **Productivity:** QoS/methodology, real Fast prototyping, easy backend
- **Reuse:** modularity for silicon open platform
- **Configurability/flexibility/scalability**

STNoC™ : architecture

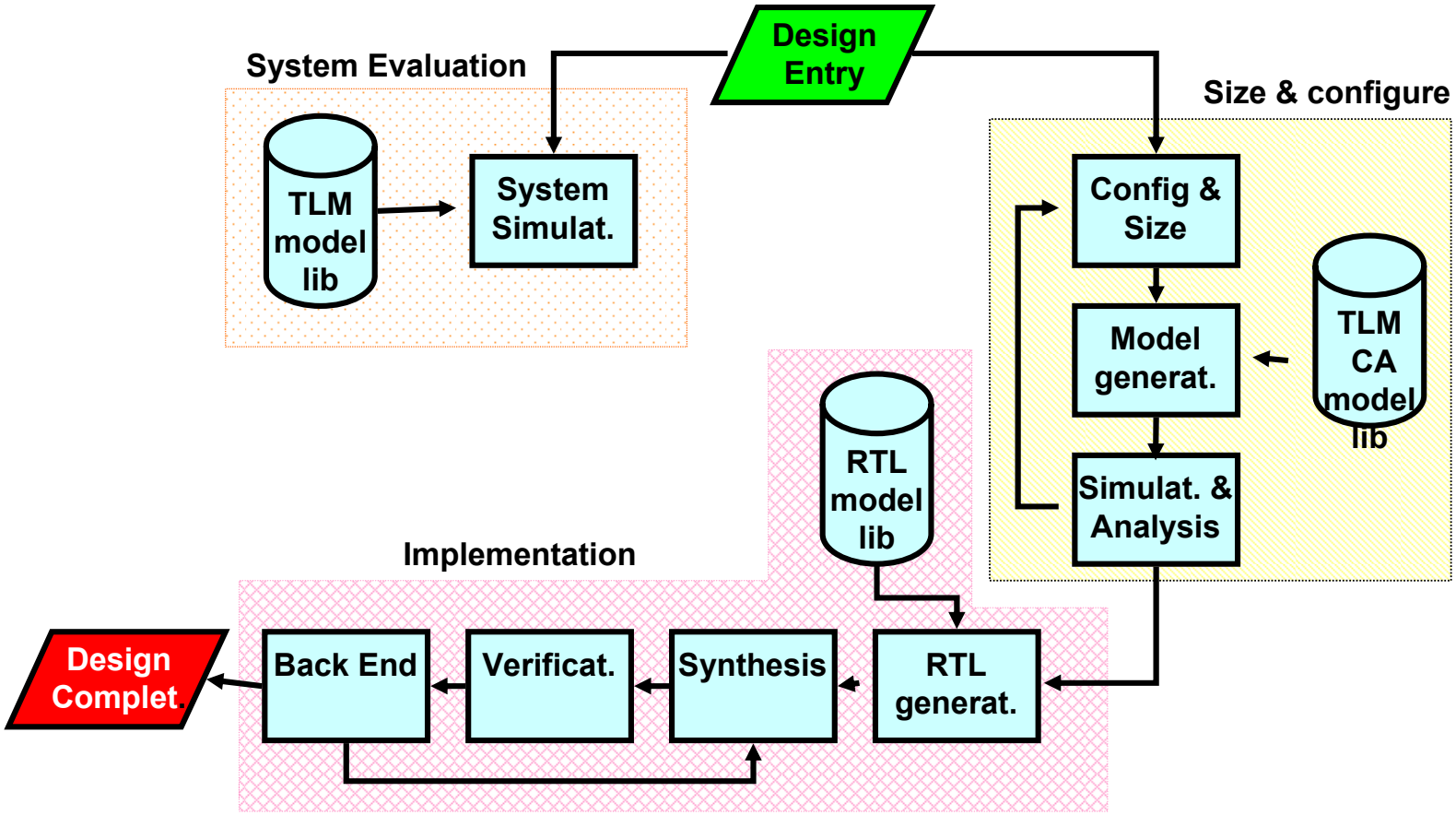
- Octagon and STBus as a starting point
- Fixed and regular topology
- Granularity of 2 nodes
- Routing algorithm:
  - ✓ easy to implement
  - ✓ deterministic
  - ✓ based on packet source and destination address (no stored table)
  - ✓ diameter = ceil(N/4)
- Wormhole switching technique
- Output buffers
- Use of Virtual Channels and smart selection to avoid deadlock

STNoC™ target: to provide on-chip services for a wealth of systems

- **Computer peripherals**
  - Data storage
  - Webcam
  - Printers and imaging
  - Monitor
- **Digital consumer**
  - DVD
  - Digital TVs
  - Digital cameras
  - MP3 players
  - Set-top boxes
- **Automotive**
  - Car radio
  - Engine/body/safety
  - Car multimedia

- Communications
  - Wireless handset
  - Wireless infrastructure
  - Access
  - Networking
- Smartcards/industrial
  - Telephone
  - Banking
  - User ID
  - Security

STM Communication centric Flow Diagram in today SoC



STM GenKit(*) : Why?

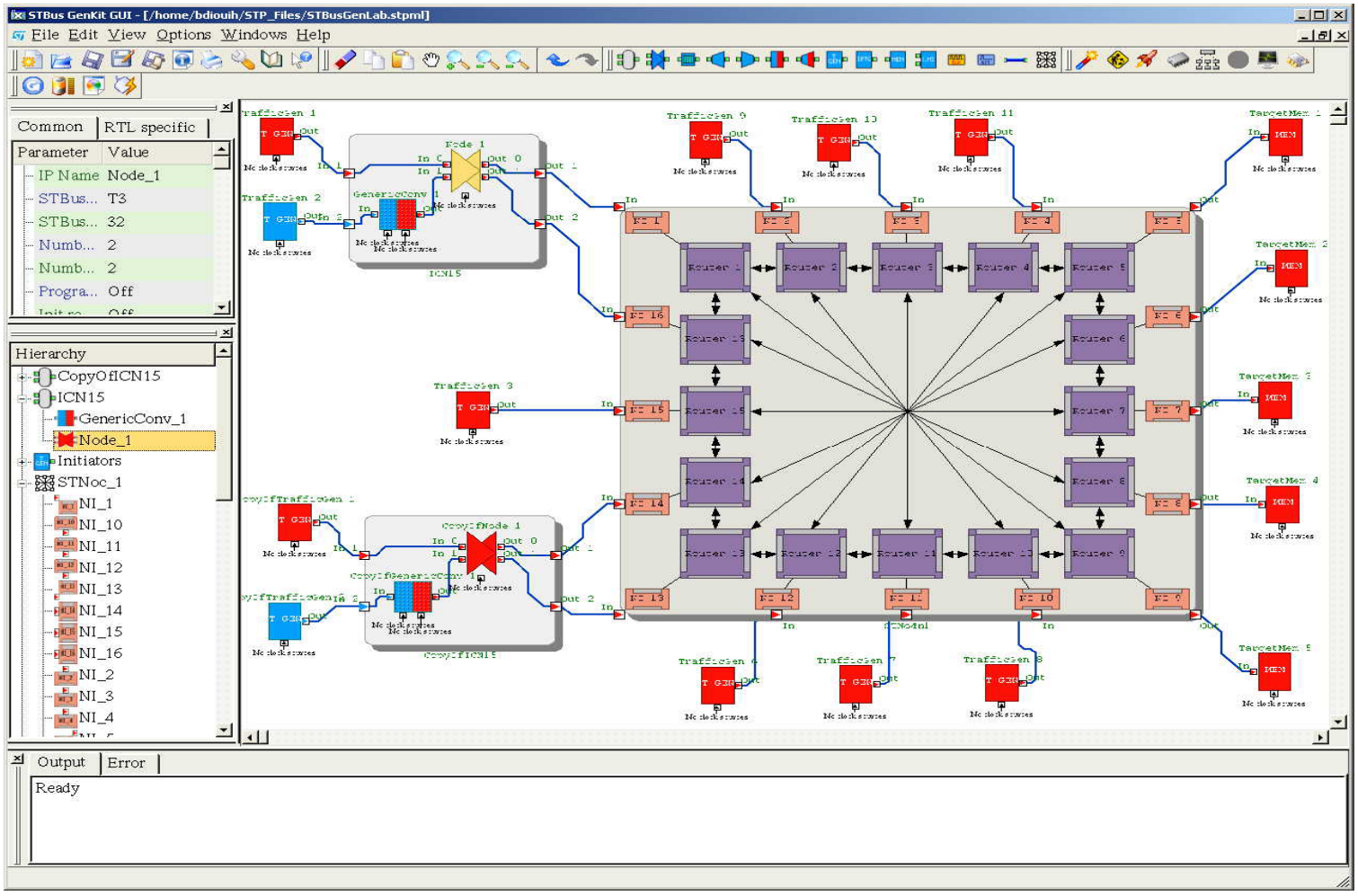
- To have an STBus & STNoC™ development platform able to exploit and maximize the main STBus & STNoC™ features:
  - **Flexibility & Configurability**
  - **Extensibility**
  - **Fast Integration**

(*) For more info contact Carlo Pistritto (carlo.pistritto@st.com)

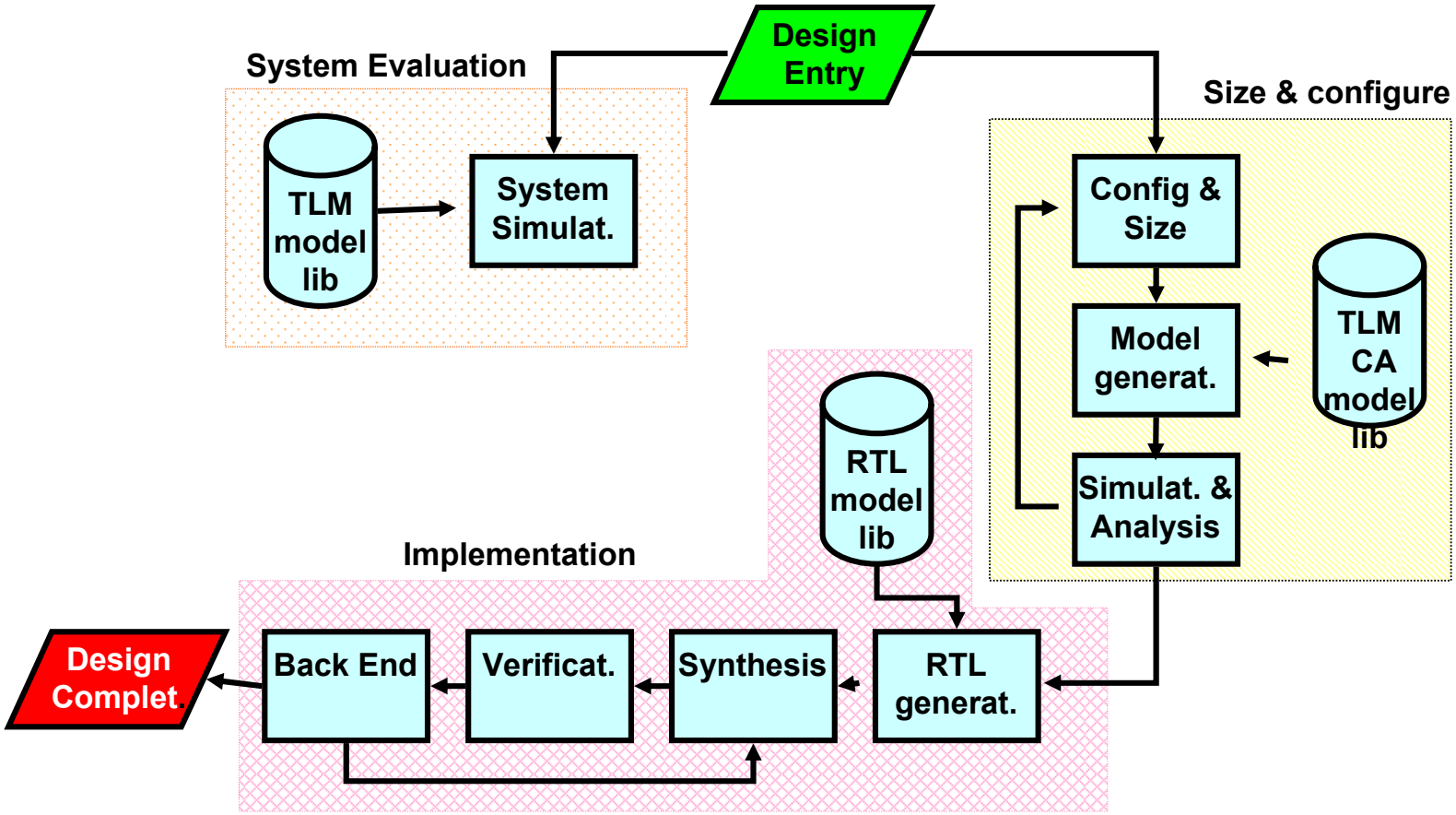
STM GenKit GUI: Introduction

- The "GenKit" GUI is the entry point for STBus & STNoC™ interconnects generation. It allows:
  - Creating projects
  - Instantiating STBus & STNoC™ interconnects, components, clocks…and connecting them
  - SystemC TLM interconnects generation & simulation
  - SystemC BCA interconnects generation, simulation & analysis
  - Power consumption analysis
  - Area estimation analysis
  - RTL scripts for coreAssembler generation

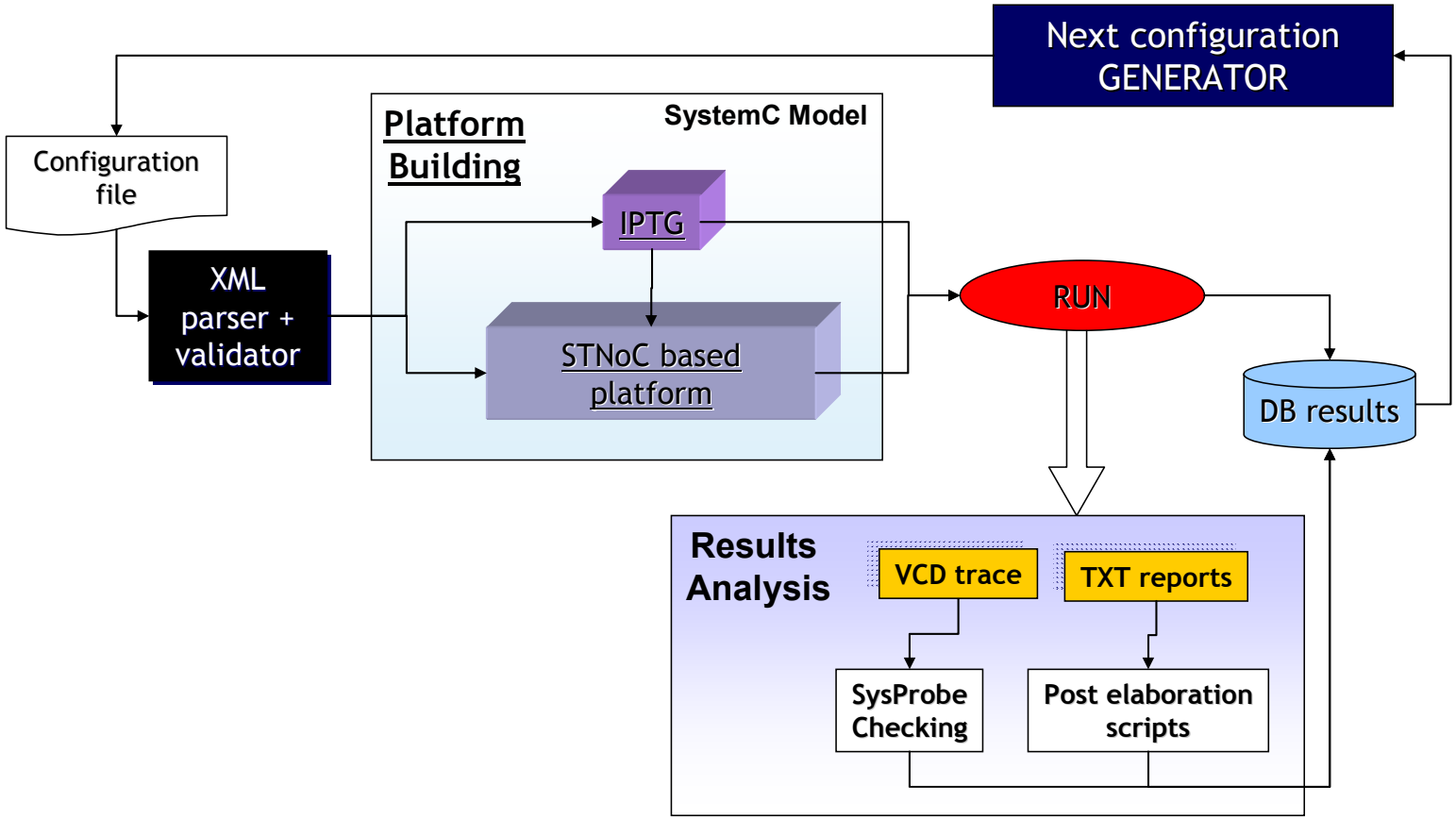
GenKit Snapshot for STNoC™



STNoC™ Platform Builder within the Genkit



STNoC™ Platform Building in Detail (1/2)



Platform building in detail

- Standard Component Library composed by TLMStBus, IPTG, STBusGenKit, TLMInfra
  - using GNU Autotools to be faster and more portable,
  - removing where possible the abuse of generic programming (e.g. using template to propagate constants etc.) to be more flexible
- STNoC SystemC/OCCN model library
- Standard configuration interface through XML descriptions using Libxml2
- Automatic system binding XML driven

STNoC™ Platform Building in Detail (2/2)

- The platform is generate at run-time according to the XML description. It's composed by:
  - Traffic Generators (IPTG)
  - STBus Genkit components:
    Ideal Targets
    STBus Nodes
    STBus Buffers
  - STNoC™ components:
    Network interfaces (STBus, AXI)
    Concentrators
    Routers
    TLM/BCA wrappers
- Each of these components can be configured at run-time
- All the components are shared libraries linked at dynamically, so it's possible to change one of these without recompiling the entire system

IP Traffic Generator (IPTG) (1/2)

- IPTG is a block developed in SystemC that read a traffic characterization file as input and generate the corresponding traffic as output on a communication structure.
- IPTG is instantiated for a specific IP to generate the IP traffic on a platform that may contain components of three different abstraction levels:
  - TLM Transaction Level Modelling
  - BCA Bus Cycle Accurate
  - RTL Register Transfer Level

IP Traffic Generator (IPTG) (2/2)

```
IPTG_NAME                       IPcore1
$IPTG_COM                        TLM_STBUS
$IPTG_MODEL                      STBUS_PLUG
$IPTG_CHK_SIZE                   64
$IPTG_MAX_OPC                    ST32
$IPTG_MIN_OPC                    ST32 #
$IPTG_FIFO_SIZE                  256
$IPTG_BEHAVIOUR_SEQ              {1,9,(2,8*32,6)*1}
#------------------- 1 -------------------------------------
#Description: do nothing while the system being initialized
#-----------------------------------------------------------
$BEHAVIOUR_NAME                  WAIT_RESET
$BEHAVIOUR_SEQ                   NOP 7000
#------------------- 8 -------------------------------------
#Description:
#-----------------------------------------------------------
$BEHAVIOUR_NAME                  WRITE_LINE_BLIT1
$BEHAVIOUR_FREQ                  200
$BEHAVIOUR_DATA_SIZE             2
$BEHAVIOUR_DATA_LENGTH           1286 # 1920
$BEHAVIOUR_ADDR                  {(0x10630000~0x106300ff)}
$BEHAVIOUR_ADDR_POLICY           INCREMENTAL
```

Example of a traffic characterization file

Result Analysis – VCD Trace

```
=====================================
      IP:Initiator_at_1_
=====================================
STbus type : 3
Data size: 64 bits
Simulation Start Time: 0 ps
Simulation End Time: 500998000 ps
Simulation duration: 500998000 ps
Clock period: 4000 ps
STbus Frequency: 250.00 Mhz
STbus Clock cycles 125248
Window Time Frame        10
LATENCY :
```

Name	Min	Max	Ave
Req2Gnt	0	46	0.92
Req2R_Req	25	116	38.62
Req2R_Eop	25	116	38.62
R_Req2R_Gnt	0	0	0.00

- VCD trace is parsed by SysProbe in batch modality to obtain:
  - Protocol checking
  - Cycle accurate throughput and latency values

Example of SysProbe output

Results Analysis – TXT Report

- Textual reports of all components are elaborated by several Perl scripts in order to obtain:
  - Bandwith statistics
  - Arbiters statistics
  - Performance results
  - Total bytes transferred
  - Power estimation
  - other

```
--- < Average BW in [ 0 - 0.000501 ] > ---
Source 1: 174.946 MB/s
Source 2: 31.681 MB/s
Source 5: 155.146 MB/s
Source 6: 26.954 MB/s
Source 17: 153.293 MB/s
Source 18: 29.637 MB/s
Source 21: 156.104 MB/s
Source 22: 30.723 MB/s
Source 33: 147.481 MB/s
Source 34: 27.337 MB/s
Source 37: 154.443 MB/s
Source 38: 30.403 MB/s
  TOTAL Bandwith is 1118.14770459082
```

Example of a script's output

Example of XML configuration file

```
<platform>
<interconnect>
   <topology>spidergon</topology>
   <node id="0">
     <type> Master </type>
     <initiator>
        <config_r>CUSTOM_R</config_r>
        <config_w>CUSTOM_W</config_w>
     </initiator>
   </node>
   <node id="1">
     <type> Slave </type>
     <target>
        <start_address>0x100</start_address>
        <end_address>0x1FF</end_address>
     </target>
   </node>
   <routing>
        <req_path>AF_0</req_path>
        <res_path>AL_0</res_path>
   </routing>
</interconnect>
<simulation>
   <vcd_trace>enabled</vcd_trace>
</simulation>
</platform>
```

Conclusion

- System level analysis and design flow provide tremendous advantages but we need tools for
  - ❑ **System level estimation (bandwidth, latency, power),**
  - ❑ **Requirements analysis,**
  - ❑ **Application mapping**
- ST endeavours to develop tools providing a complete and fast path to silicon
  - ❑ **To meet design constraints**
  - ❑ **With short time-to-market**
  - ❑ **Keeping innovation pace**

Marcello.Coppola@st.com

.....Thank you for listening!!!