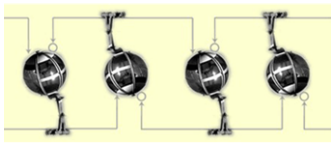
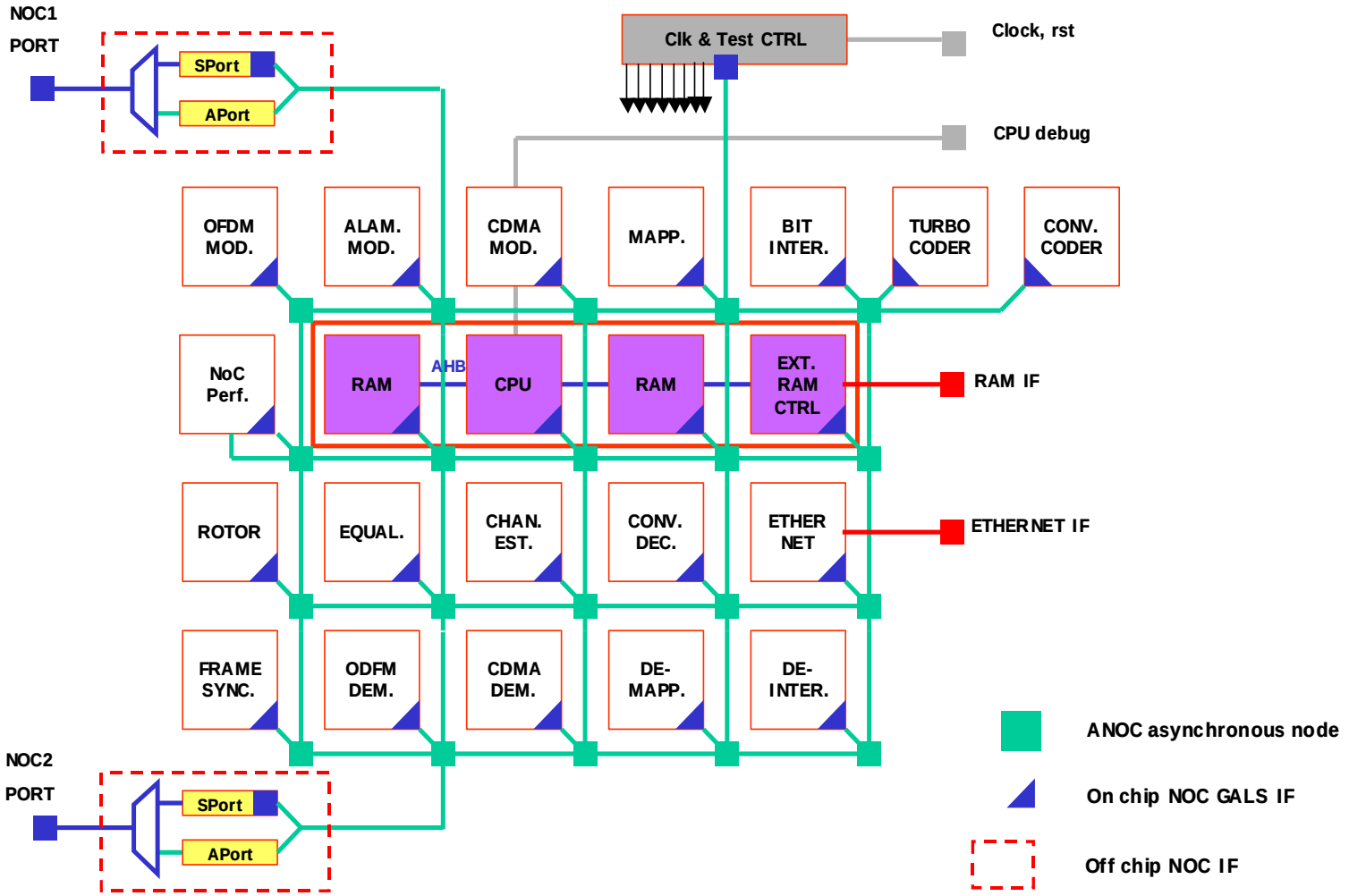
Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture

Edith Beigné – Pascal Vivet
{edith.beigne; pascal.vivet}@cea.fr

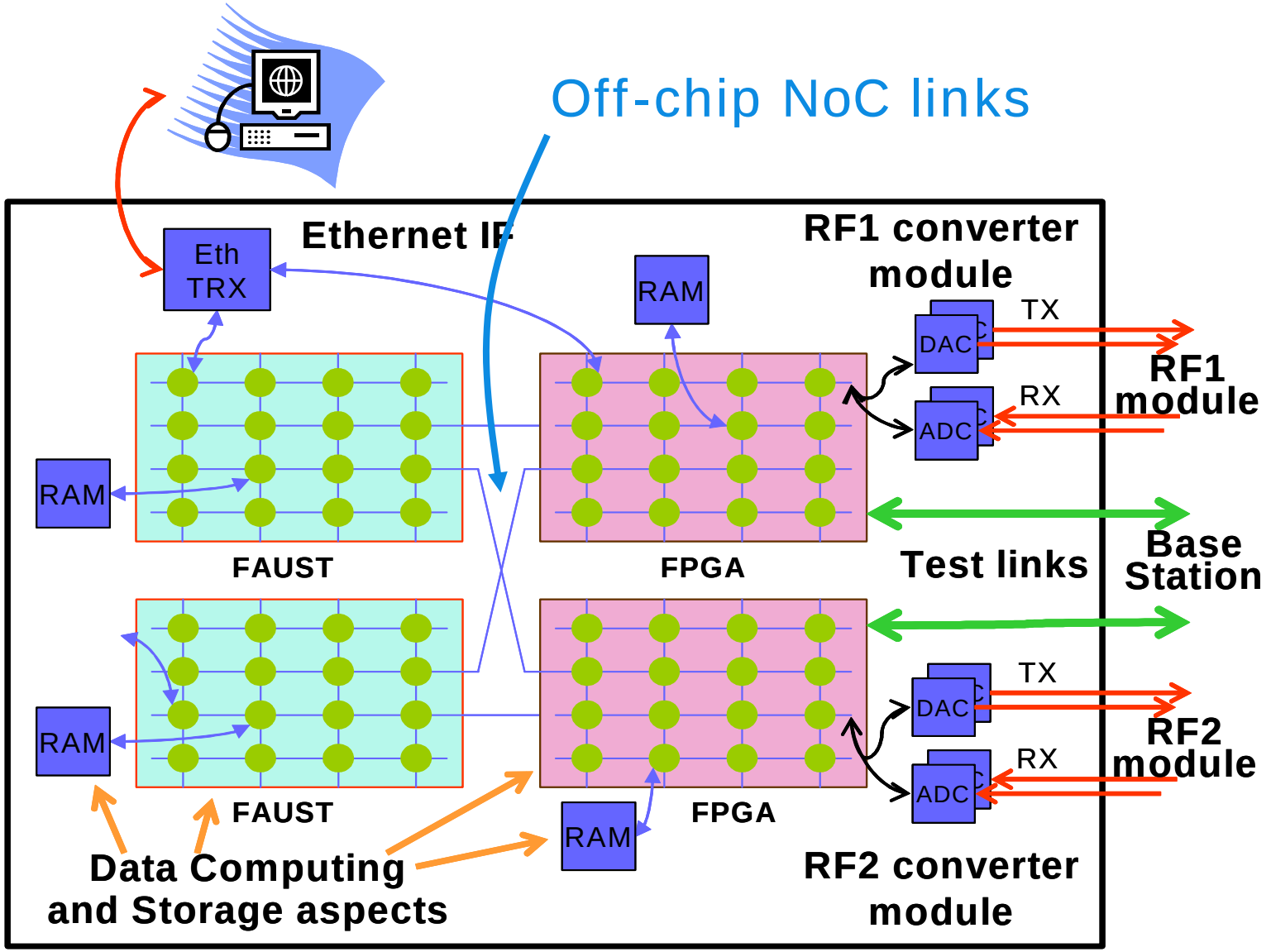


Async'06

FAUST : an Open NoC-based platform for Telecom



FAUST : an Open NoC-based platform for Telecom



Presentation Outline

- ANOC : a NoC GALS architecture
- On-chip interface
- Off-chip interface
- Results
- Conclusions

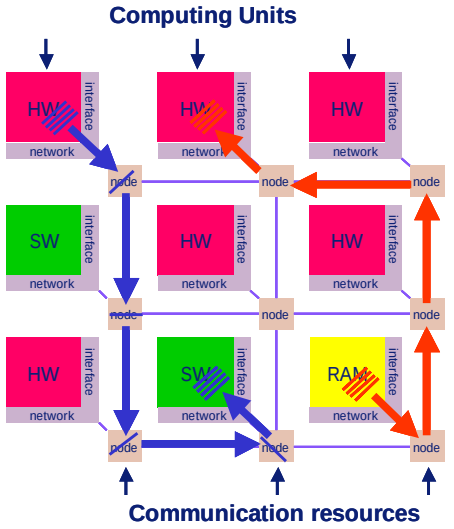
Presentation Outline

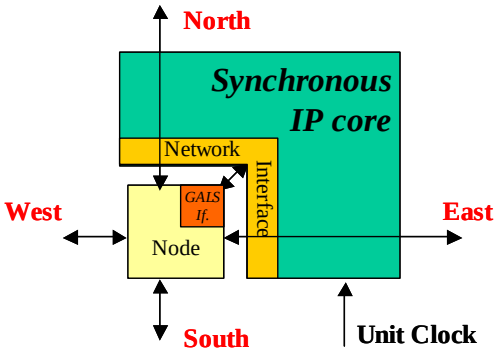
- **ANOC : a NoC GALS architecture**
  - **ANOC overview**
  - **NoC physical layer protocols**
  - **Interface Design Objectives**

- On-chip interface
- Off-chip interface
- Results
- Conclusions

ANOC : Asynchronous NoC characteristics

- ANOC architecture
  - Noc features :
    - 2D mesh, wormhole,
    - dead-lock free source routing
    - QoS with 2 virtual channels
    - Distributed control & synchronization provided by Network Interface
  - GALS features :
    - Low power system
    - ANOC fully implemented in QDI logic (robustness + low power)
    - All NoC units have distinct & independent clock domains





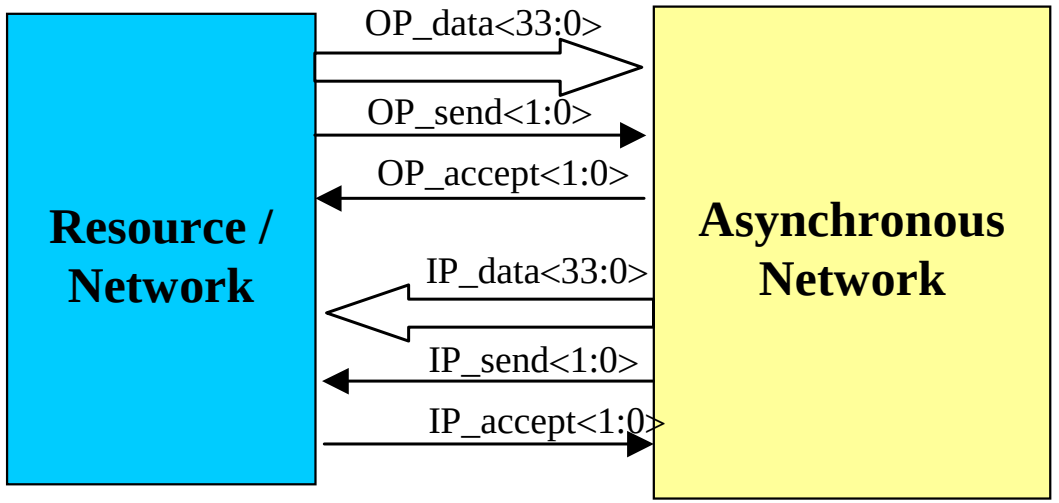
NoC Virtual Channel protocol

- ANOC virtual channels provide Low latency / Quality-of-Service (QoS) :
  - VC0 : *Guaranteed service for high priority packets (Control, Interruption, …)*
  - VC1 : *Best effort traffic for classical data-flow*

- Low latency service is implemented at Flit level :
  - Send/Accept Protocol always guarantees that the physical channel is free :
    - A new data can be sent on Virtual Channel *i* only when Accept*i* is true

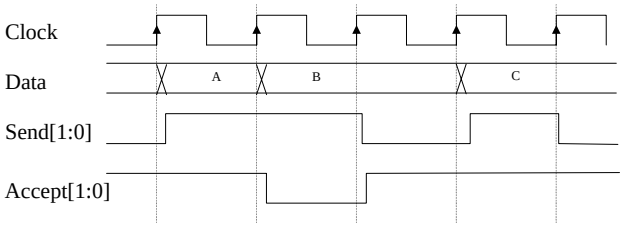
This protocol/service can be implemented using both synchronous & asynchronous circuits

⇒ Well suited for GALS systems



NoC Physical Layer Protocols

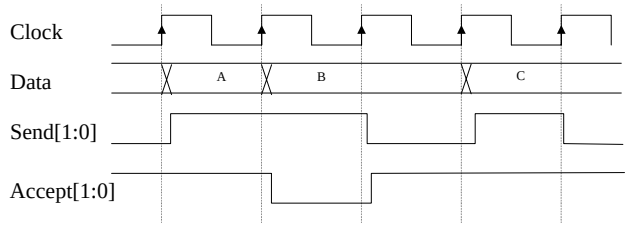
- 3 different physical NoC protocols :
  - Synchronous handshake protocol
    - Send/Accept protocol
    - 76 signals per link
    - Used by **Synchronous** Units

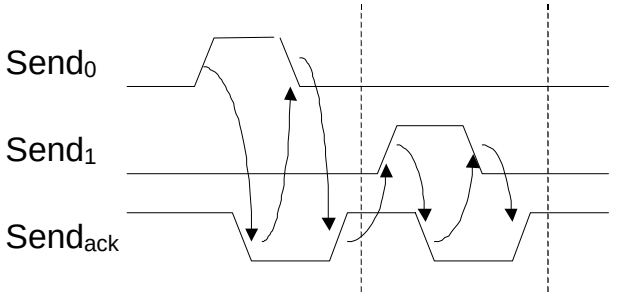


NoC Physical Layer Protocols

- 3 different physical NoC protocols :
  - Synchronous handshake protocol
    - Send/Accept protocol
    - 76 signals per link
    - Used by **Synchronous** Units
  - QDI multi-rail handshake protocol
    - 4-phase, dual-rail/four-rail
    - 180 signals per link
    - Used by **On-chip** NoC links
      - Robustness / low-power

On-chip Interface



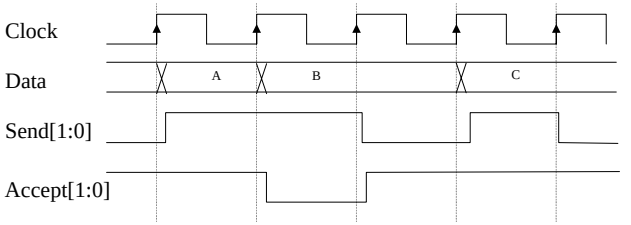


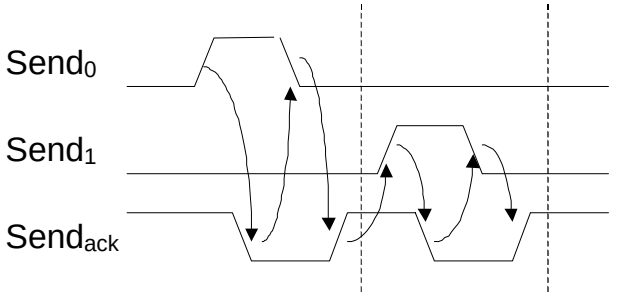
NoC Physical Layer Protocols

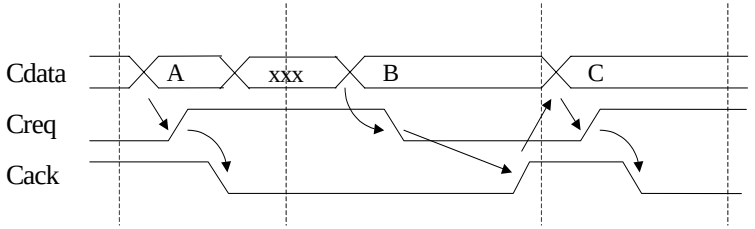
- 3 different physical NoC protocols :
  - Synchronous handshake protocol
    - Send/Accept protocol
    - 76 signals per link
    - Used by **Synchronous** Units
  - QDI multi-rail handshake protocol
    - 4-phase, dual-rail/four-rail
    - 180 signals per link
    - Used by **On-chip** NoC links
      - Robustness / low-power
  - Bundled-data handshake protocol
    - 2-phase, bundled-data
    - 78 signals per link
    - Used by **Off-chip** NoC links
      - Low signal count / better throughput

On-chip Interface

Off-chip Interface







NoC interface design objectives

Design objectives

- On-chip NoC interface :
  - Synchronous/Asynchronous Interface for unit-to-ANOC connection
  - Use **standard-cell & standard tools** at most as possible
  - Low-latency / high throughput

- Off-chip NoC interface :
  - Extend the NoC protocol Off-chip for system level (ASIC/FPGA)
  - Offer **both synchronous & asynchronous** modes
  - Chip pads sharing / acceptable throughput

Presentation Outline

- ANOC : a NoC GALS architecture
- **On-chip interface**
  - **GALS synchronization FIFO's**
  - **Interface design**

- Off-chip interface
- Results
- Conclusions

GALS synchronization

- GALS synchronization issues
  - Synchronize independent clock domains
    - Transfer correctness (MTBF measurements)
  - Simple design with a two (or more) FF's synchronizer easily achievable
    - Throughput & latency concerns

- GALS synchronization state-of-the-art
  - It is a complex design issue : [Seizovic], [Greenstreet], [Ginosar], …
  - Synchronization Fifo's
    - [Chelcea] : A/S, S/A, S/S very efficient fifo's, full-custom design
  - Pausable clock techniques
    - [Yun], [Moore], [Muttersbach] : requires programmable delay lines

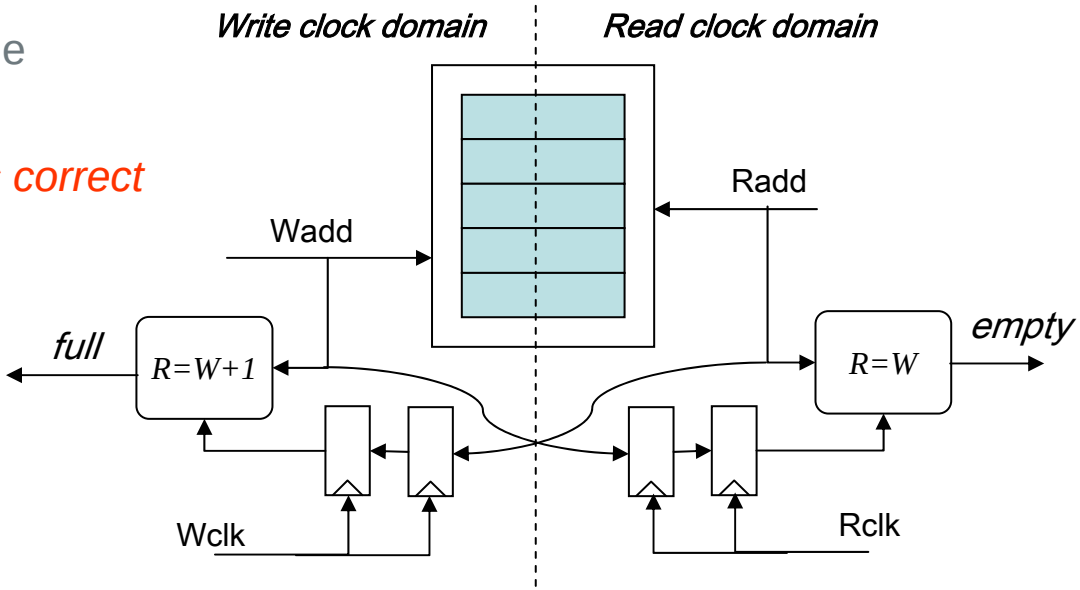
- **Idea ?**
  - *Reuse and adapt a standard dual-clock gray fifo* [Cummings]

GALS synchronization fifo's

- Dual-clock **Gray** fifo principle :
  - Fifo is controlled by Write & Read pointers
  - Synchronize the opposite pointer with two (or more) FF's
  - Pointers are encoded using **Gray code**

- Only one bit is modified between any successive values
- If FF's fail to get the new value, result is "wrong" of at most one value (not 2^N)

=> *Full/empty comparison is always correct*



Performance aspects :

- Two clock cycle latency
- High throughput (concurrent read & write accesses as long as fifo is not empty nor full)

A-to-S synchronization FIFO

- A-to-S gray fifo
  - Based on a 8word * 34-bit gray fifo
  - Rclk : ***equal*** to the synchronous unit clock
  - Wclk : ***generated*** by the asynchronous side of the GALS interface

- Required modifications
  - Since Wclk is off after the last write, the full flag is cleared differently
  - To facilitate send/accept NoC handling, use a "*almost-full*" information

Use asymetric C-elements

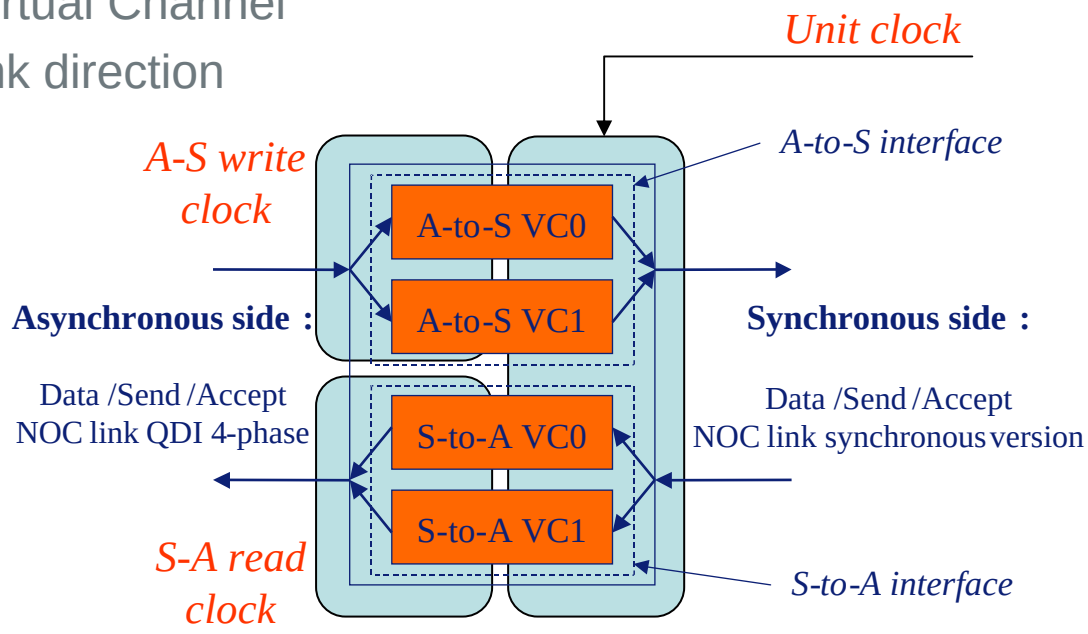
Fifo is *empty* when :
read_add[2:0]=write_add[2:0]
Fifo is *almost-full* when :
read_add[2:1]=(write_add[2:1]) +1

$$
\begin{aligned}
&Full \wedge CKR_b \wedge (Radd \neq Wadd+1) \Rightarrow Reset\_Full+\\
&\neg Full \Rightarrow Reset\_Full-\\
&Full_b \wedge CKW_b \wedge (Radd = Wadd+1) \Rightarrow Set\_Full+\\
&\neg Full_b \Rightarrow Set\_Full-\\
&\neg Set\_Full \wedge \neg Reset\_Full_b \Rightarrow Full-\\
&Set\_Full \Rightarrow Full+
\end{aligned}
$$

On-chip GALS NoC Interface

- GALS NoC Interface Architecture :
  - Interfaces synchronous and QDI 4-rail send/accept protocol
    - Performs data conversion from 4-rail to binary
    - Manages priority between VC0 & VC1 channels
      - a VC0 flit can take over suspended VC1 flits
  - Composed of 4 synchronization gray fifos
    - One fifo per NoC Virtual Channel
    - One fifo per NoC link direction

Interface is built of 3 clocks :
- Unit clock
- A-S write clock
- S-A read clock



A-to-S NoC Interface

4-rail to binary data conversion :

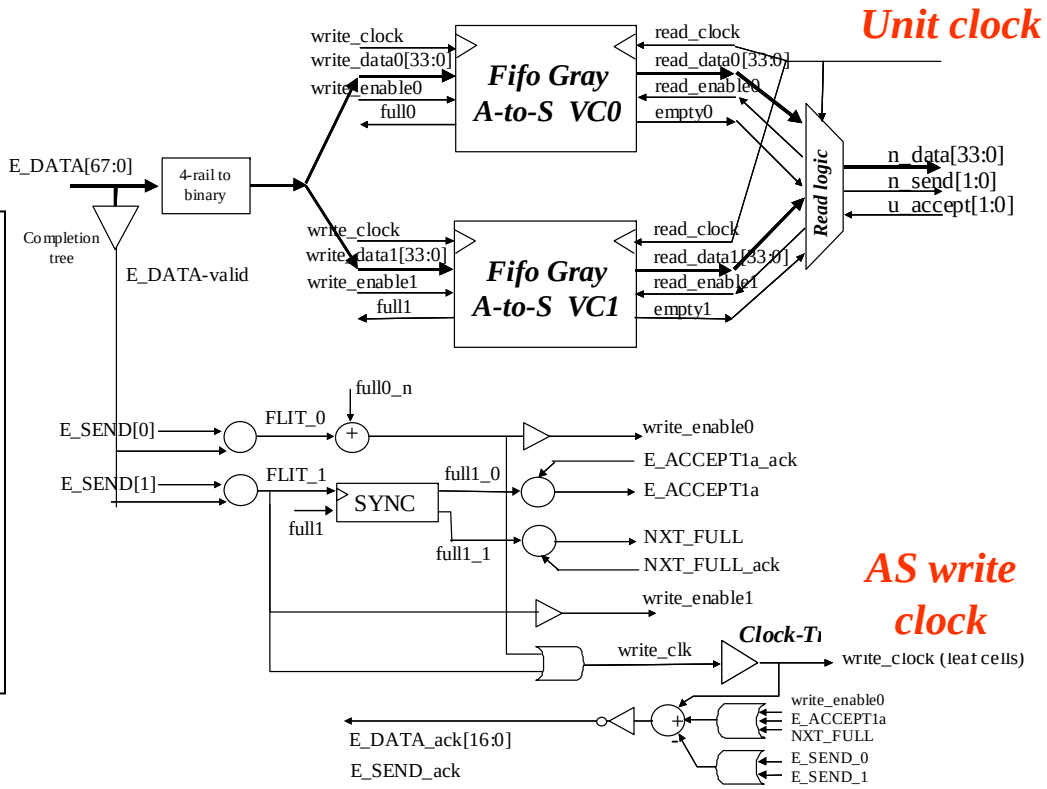
```
data[2*i+1]=OR(E_DATA[4*i+3], E_DATA[4*i+2])
data[2*i]  =OR(E_DATA[4*i+3], E_DATA[4*i+1])
```

- A-to-S NoC interface :
  - Composed of 2 A-to-S Gray fifos (VC0 & VC1)
  - Read-clock is equivalent to synchronous side
  - Write-clock is locally generated
    - *asymetric clock-pulse*

A-to-S CHP model :

```
(1)*[ #E_SEND=0 and #E_DATA =>
     [ #full0=0 =>
        Wclk!, write0!, E_SEND?, E_DATA?data0
     ]
    @ #E_SEND=1 and #E_DATA =>
        Wclk!, write1!, E_SEND?, E_DATA?data1,
        [  #full1=0 => E_ACCEPT1a!
        @@ #full1=1 => NXT_FULL!
    ]   ]

(2) *[ [ #CUR_FULL and #full1=0 =>
              CUR_FULL?, E_ACCEPT1b! ] ]
```

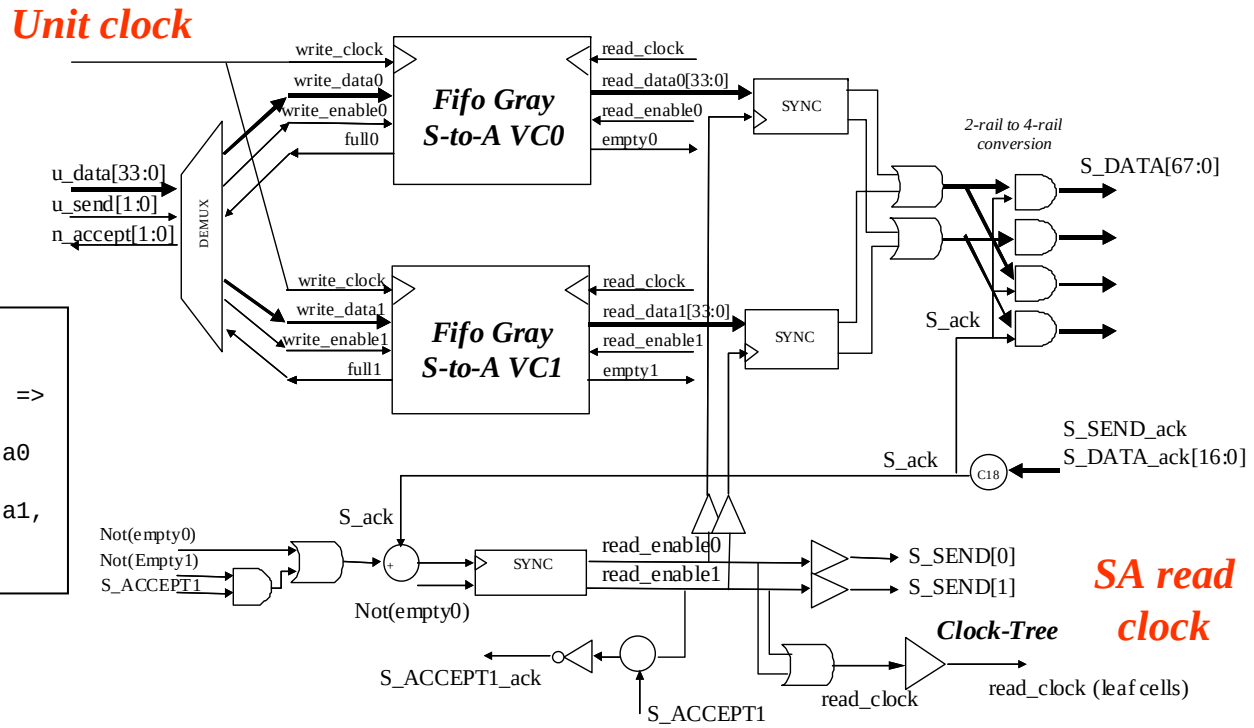


S-to-A NoC Interface

binary to 4-rail data conversion :

```
data[4*i+3]=AND(rail1[2*i+1], rail1[2*i])
data[4*i+2]=AND(rail1[2*i+1], rail0[2*i])
data[4*i+1]=AND(rail0[2*i+1], rail1[2*i])
data[4*i]  =AND(rail0[2*i+1], rail0[2*i])
```

- S-to-A NoC interface :
  - Composed of 2 S-to-A Gray fifos (VC0 & VC1)
  - Write-clock is equivalent to synchronous side
  - Read-clock is locally generated
    - *Use SYNC cells to make the 4-rail output value stable*



S-to-A CHP model :

```
*[[#empty0=0 or (#empty1=0 and #S_ACCEPT1) =>
    [  #empty0=0 =>
       Rclk!, read0!, S_SEND!0, S_DATA!data0
    @@ #empty0=1 =>
       Rclk!, read1!, S_SEND!1, S_DATA!data1,
       S_ACCEPT1?
 ]]
```

Presentation Outline

- ANOC : a NoC GALS architecture
- On-chip interface

- **Off-chip interface**
  - **Asynchronous protocol converters**
  - **Interface design**

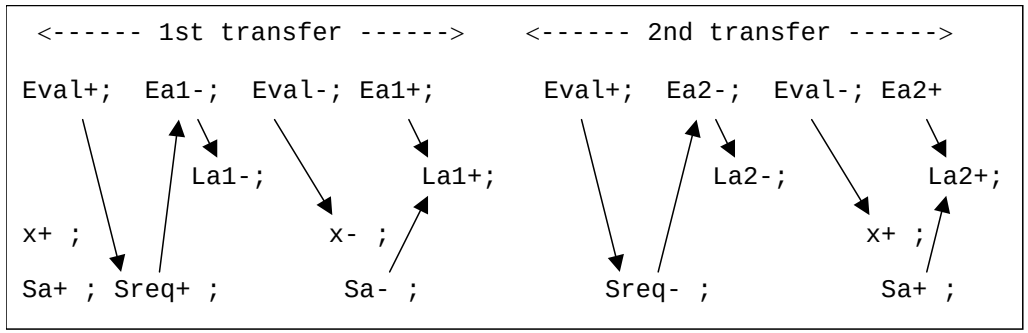
- Results
- Conclusions

Off-chip NoC Interface

- For synchronous mode :
  - Use previous A-from/to-S interfaces
- For asynchronous mode :
  - Internal QDI protocol does not fit off-chip communication
    - 4-rail is too costly in number of pads
    - 4-phase is too slow (pad latency occurs 4 times per data-transfer)
  - Idea ?
    - Keep a 32-bit parallel NoC link, but convert the QDI protocol into a bundled-data protocol

QDI-to-BD interface

- QDI-to-BD interface
  - Convert 4-rail/4-phase to bundled-data/2-phase
  - State variable required to identify odd/even transfers
  - Output data is latched



QDI-to-BD Handshake Expansion

$Eval \wedge Sa \wedge x \Rightarrow Sreq+$
$\neg Eval_b \wedge \neg Sa \wedge \neg x \Rightarrow Sreq-$

$Eval \wedge Sreq \wedge x \Rightarrow Ea1_b+$
$\neg Eval \Rightarrow Ea1_b-$

$Eval \wedge Sreq_b \wedge x_b \Rightarrow Ea2_b+$
$\neg Eval \Rightarrow Ea2_b-$

$\neg Ea1 \Rightarrow La1-$
$Ea1 \wedge Sa_b \Rightarrow La1+$

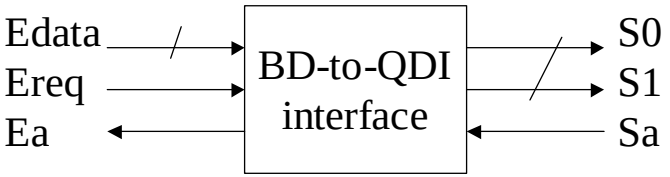
$\neg Ea2 \Rightarrow La2-$
$Ea2 \wedge Sa \Rightarrow La2+$

$Sreq \wedge Eval_b \Rightarrow x-$
$\neg Sreq \wedge \neg Eval \Rightarrow x+$

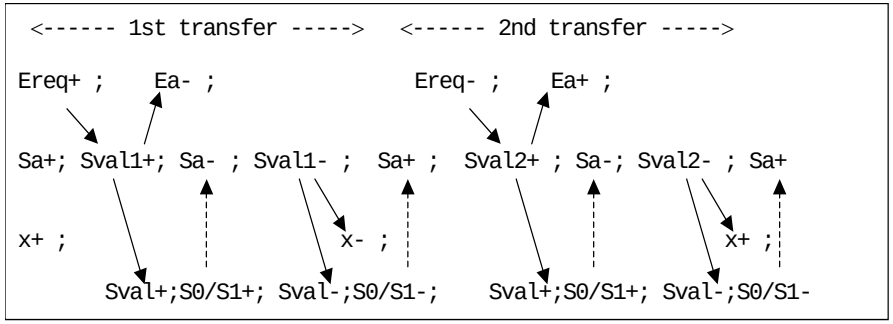
QDI-to-BD Production Rules

(can be mapped onto asymetric C-elements)

BD-to-QDI interface



- BD-to-QDI interface
  - Convert bundled-data/2-phase to 4-rail/4-phase
  - State variable required to identify odd/even transfers
  - Output data built from MUTEX cells



BD-to-QDI Handshake Expansion

$$
\begin{aligned}
&Sa \wedge Ereq \wedge x \Rightarrow Sval1+\\
&\neg Sa \Rightarrow Sval1-\\
&Sa \wedge Ereq_b \wedge x_b \Rightarrow Sval2+\\
&\neg Sa \Rightarrow Sval2-\\
&x \wedge Sval1 \Rightarrow Ea-\\
&\neg x \wedge \neg Sval2_b \Rightarrow Ea+\\
&\neg Ea \wedge \neg Sval1 \Rightarrow x-\\
&Ea \wedge Sval2_b \Rightarrow x+\\
&Sval \wedge Edata \wedge S0_b \Rightarrow S1+\\
&Sval \wedge Edata_b \wedge S1_b \Rightarrow S0+\\
&\neg Sval \Rightarrow S0-, S1-
\end{aligned}
$$

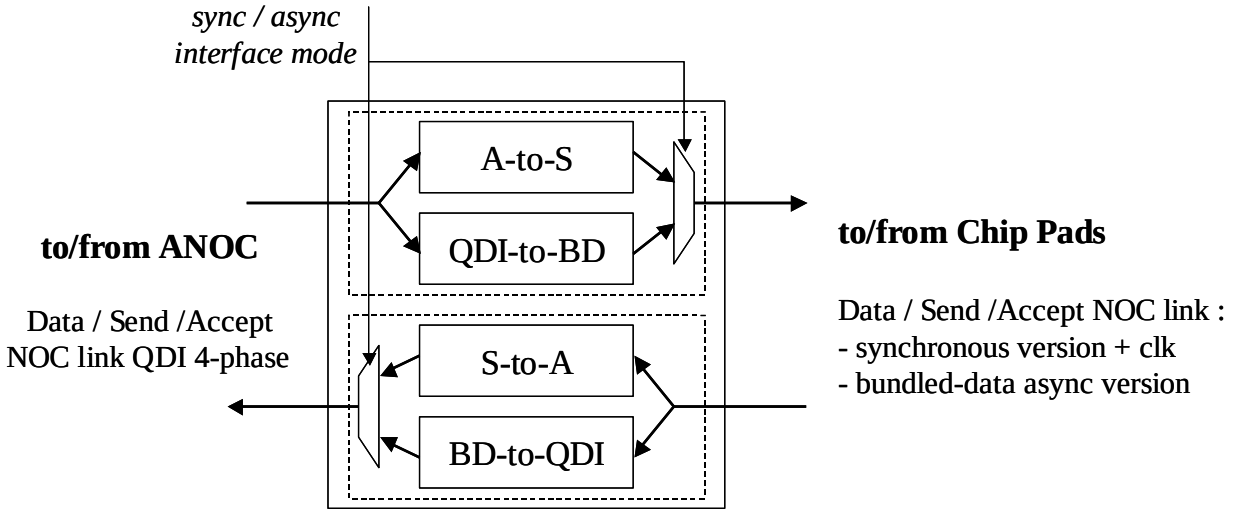
BD-to-QDI Production Rules

(can be mapped onto asymetric C-elements)

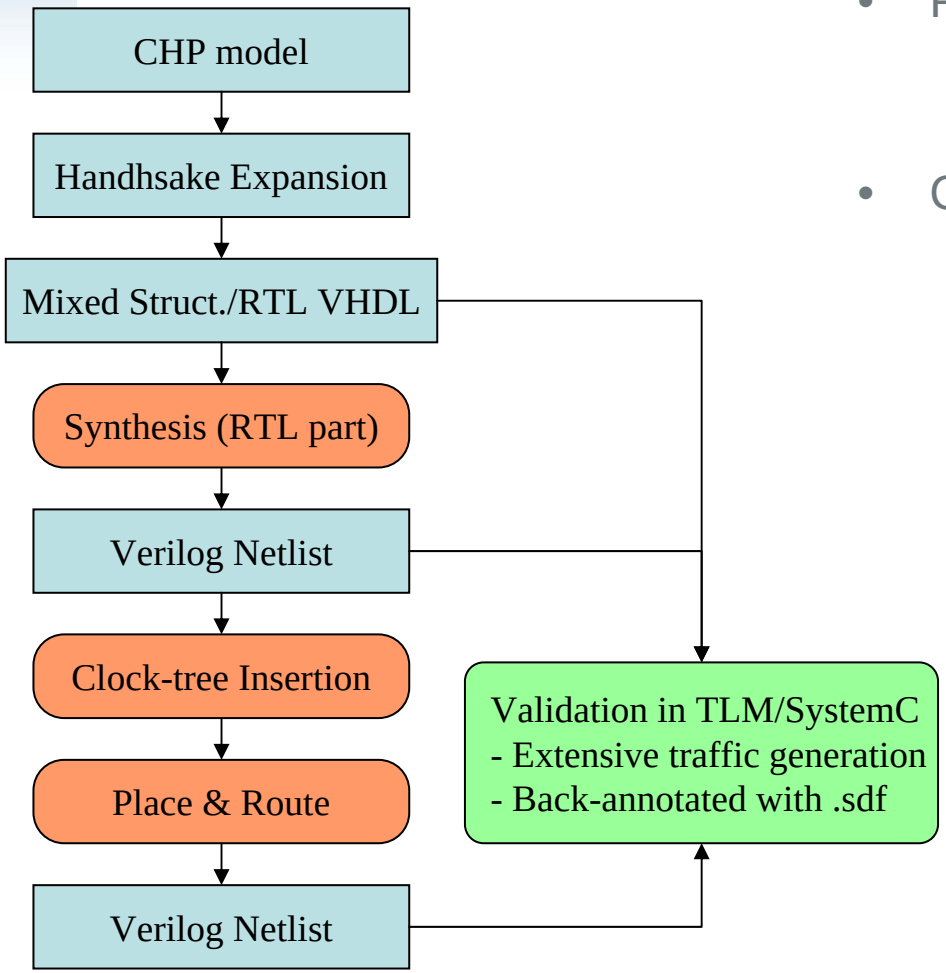
Off-chip NoC Interface

- Off-chip interface design :
  - synchronous & asynchronous off-chip NoC connection
  - Composed of A-S & BD-QDI protocol converters
    - Synchronous version is pipelined (gray fifo's)
    - Asynchronous version is ***not*** pipelined
  - chip PADs are multiplexed :
    - 78 signals per NoC link
    - For the synchronous mode, a NoC port ***clk*** signal is provided

Off-chip NoC connexion for ASIC/FPGA boards

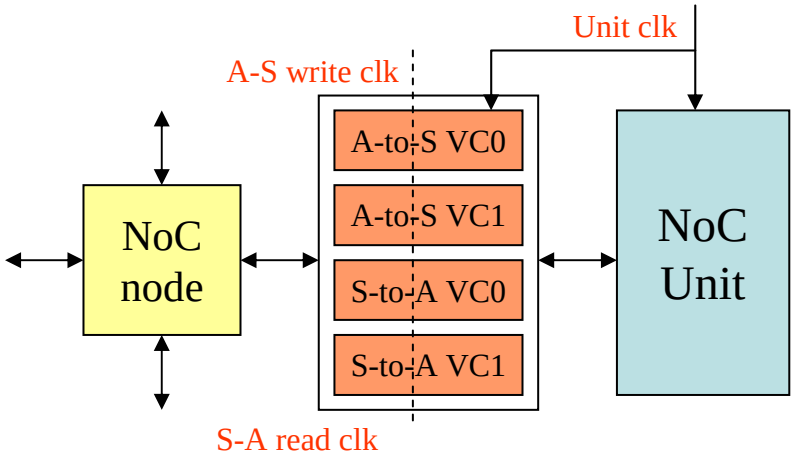


NoC Interfaces Design (1)



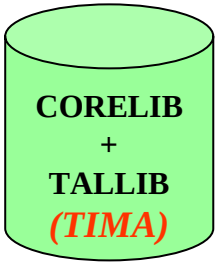
- Physical Implementation
  - NoC node is bult as a "hard-macro"
  - GALS interface is built as a "soft-macro"
- Clock management

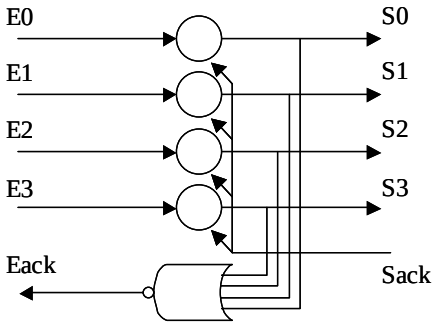
  3 clocks per GALS NoC interface :
  - 'as' & 'sa' clocks are locally generated
    - clock latency is matched with local delays
  - Interface clock is equal to (and skew balanced with) the NoC unit clock



NoC Interfaces Design (2)

- Targeted technology
  - STMicroelectronics 130nm CMOS technology
  - Standard-cell design, mapped onto :
    - CORELIB standard library
    - TALH9 library ***(Collaboration with TIMA)***
      - contains C-elements, Mutex

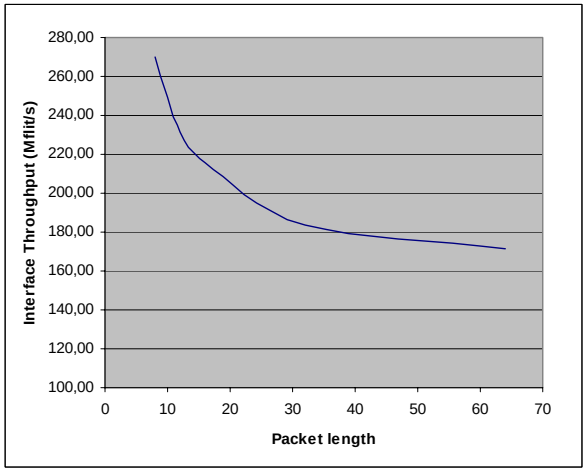


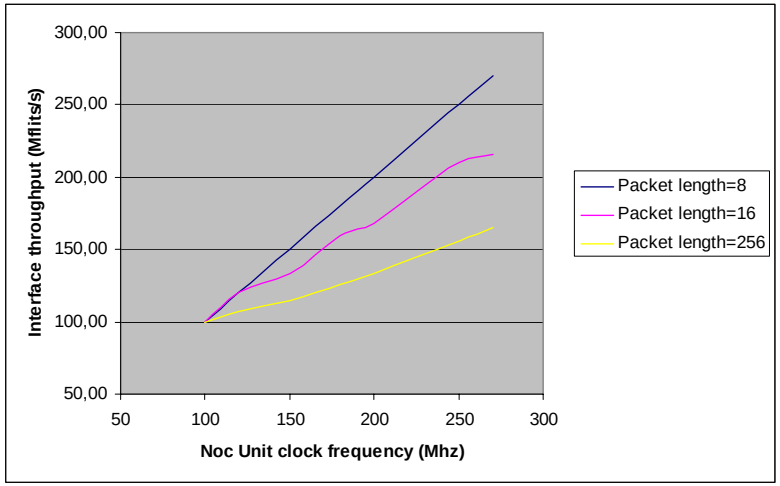


- Testability :
  - NoC nodes + NoC interfaces
    - functional test patterns
  - All NoC units are full-scan + bist
    - ***Scan patterns are transported by the NoC***

NoC Interfaces Results

- On-chip NoC Interfaces (***Measured Results***)
  - 180 Mflit/s for packet-length=32
  - 270 Mflit/s for packet-length=8
    - As fast as NoC nodes



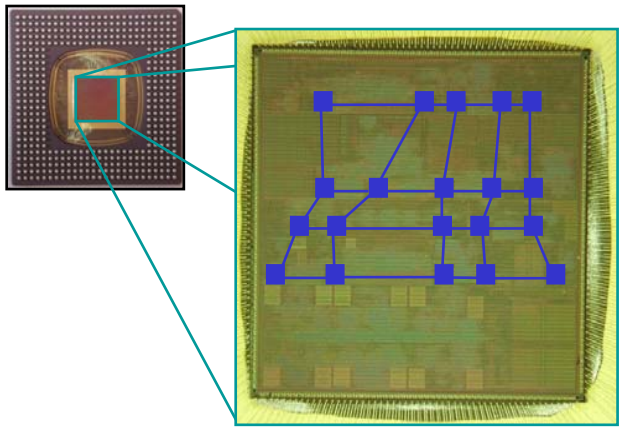


- Off-chip NoC Interfaces (***Simulation Results*** *WC, 105C, 1.02V*)
  - Synchronous mode : 120 Mflit/s
  - Asynchronous mode : 100 Mflit/s

Conclusions & Perspectives

- Designed On-chip & Off-chip NoC Interfaces
  - Adapted to a GALS NoC architecture
  - On-chip GALS interfaces are based on a standard dual-clock gray fifo
  - Off-chip interfaces to extend NoC protocol at system level

- Complex design achieved
  - 8 Million gates prototype chip for 4-G Telecom applications
  - Silicon is currently under test
    - **On-chip & Off-chip interfaces validated**
    - **More tests are currently on-going for perf. analysis**



- Future works
  - Optimizations of A-to-S & S-to-A interfaces
    - Mainly ‘as’ & ‘sa’ clock tree latency issues
  - Optimizations of Off-chip interfaces
    - Add pipeline for the asynchronous mode

The end

- Thank you