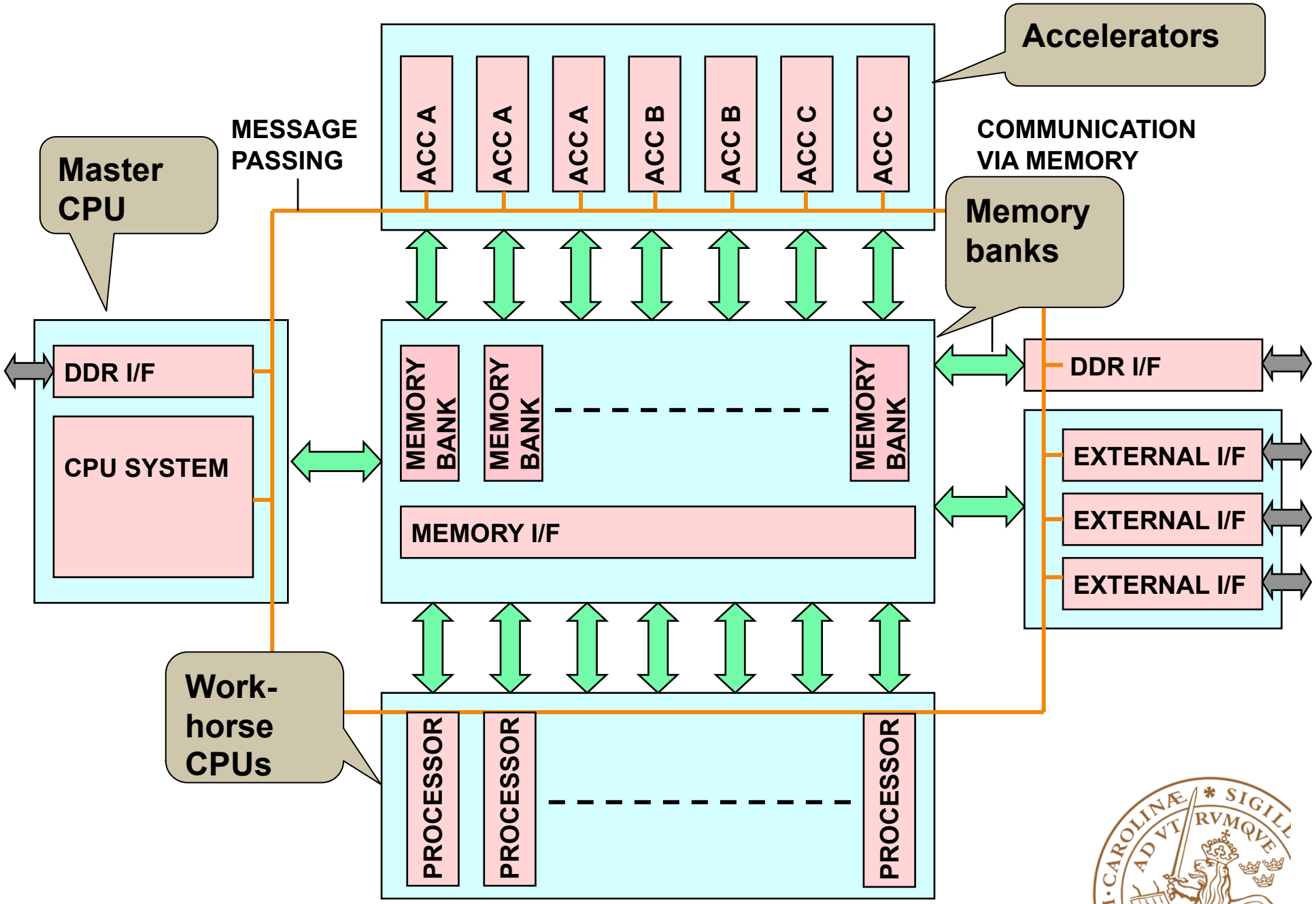
Fault management in an IEEE P1687 (IJTAG) environment

Erik Larsson and Konstantin Shibin
Lund University Testonica Lab

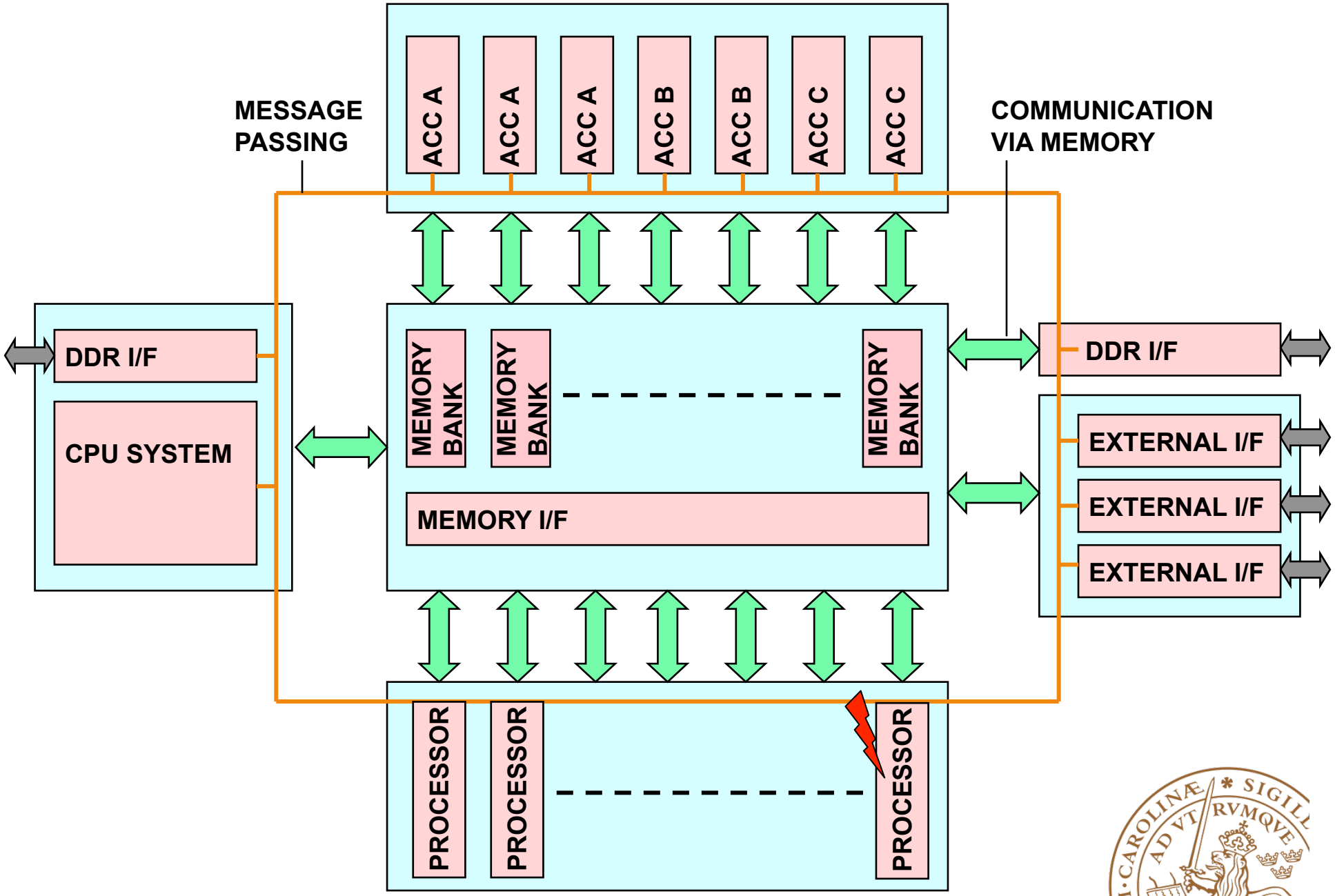
Motivation

- Semiconductor technology development enables design and manufacturing of integrated circuits (ICs) that meet the constant performance demand.
- Yesterday, the performance demand was met by higher clock frequencies.
- Today, the performance demand is met by Multi (Many)-Processor System-on-Chip (MPSoCs) where a number of components such as processors, DSPs, accelerators, memories, etc, are integrated on a single IC.


Accelerators
ACC A
ACC A
ACC A
ACC B
ACC B
ACC C
ACC C
MESSAGE PASSING
COMMUNICATION VIA MEMORY
Master CPU
Memory banks
DDR I/F
CPU SYSTEM
MEMORY BANK
MEMORY BANK
MEMORY BANK
MEMORY I/F
DDR I/F
EXTERNAL I/F
EXTERNAL I/F
EXTERNAL I/F
Work-horse CPUs
PROCESSOR
PROCESSOR
PROCESSOR

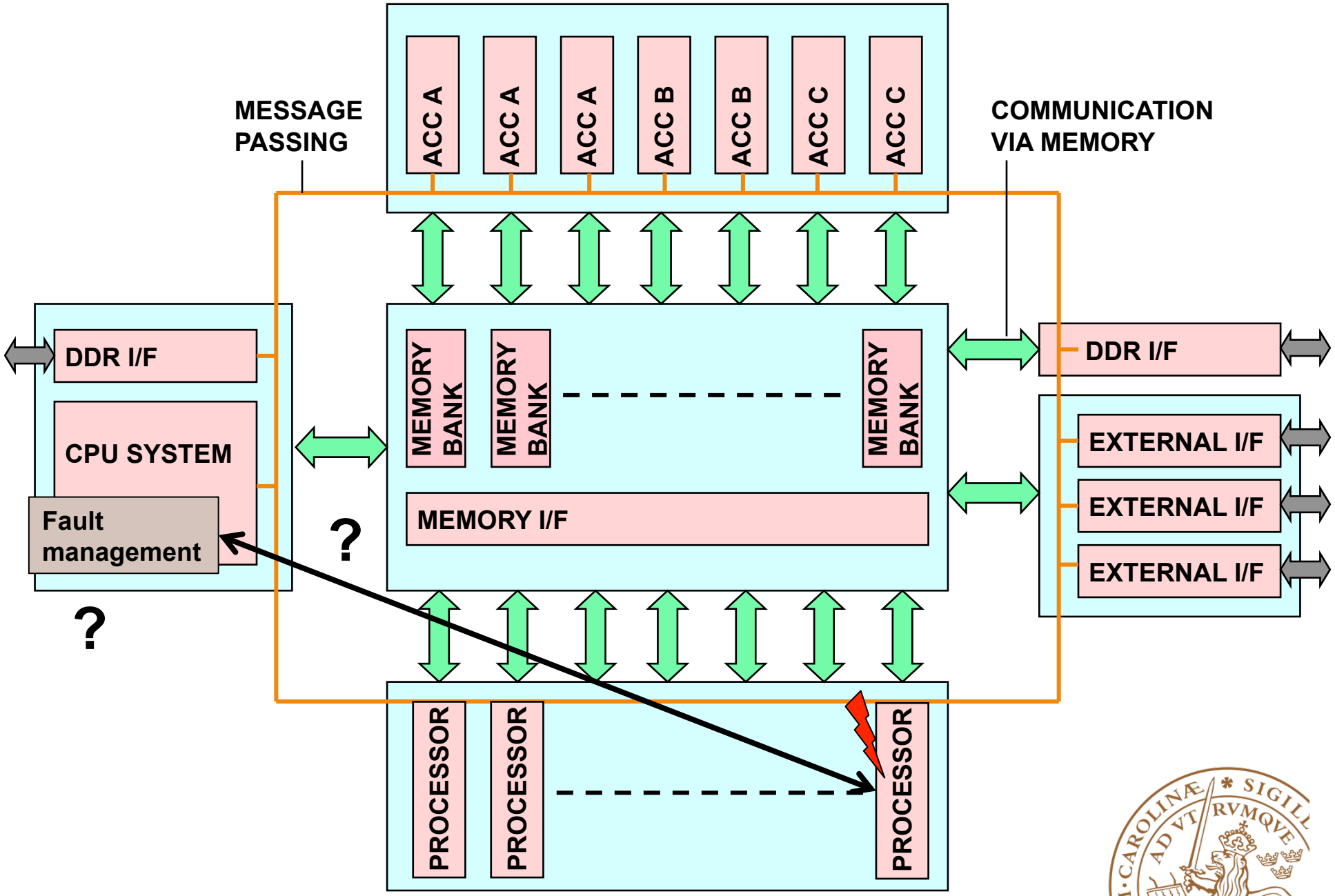
Motivation

- A drawback with ICs developed in later semiconductor technologies is that it is difficult to avoid errors
- Errors can be classified as soft and hard
- Soft errors are transient. These may not show up during manufacturing test or may evolve during operation due to aging
- Hard errors are permanent. Many are detected at manufacturing test but some may escape. Also, hard errors may evolve during operation (for example due to aging)


MESSAGE PASSING
COMMUNICATION VIA MEMORY
ACC A
ACC A
ACC A
ACC B
ACC B
ACC C
ACC C
DDR I/F
CPU SYSTEM
MEMORY BANK
MEMORY BANK
MEMORY BANK
MEMORY I/F
DDR I/F
EXTERNAL I/F
EXTERNAL I/F
EXTERNAL I/F
PROCESSOR
PROCESSOR
PROCESSOR

Motivation

- Detect errors locally to get error indication fast; avoid the effect of the error to spread.
- Adjust at system-level where there is a global view of the system; which units are defective etc.
- For soft errors, re-execute the job
- For hard errors, fault-mark and do not use the component
- To meet this:
  - Architecture to connect component-level with system-level
  - Fault management; what to do when an error occurs.


MESSAGE PASSING
COMMUNICATION VIA MEMORY
ACC A
ACC A
ACC A
ACC B
ACC B
ACC C
ACC C
DDR I/F
CPU SYSTEM
Fault management
?
?
MEMORY BANK
MEMORY BANK
MEMORY BANK
MEMORY I/F
DDR I/F
EXTERNAL I/F
EXTERNAL I/F
EXTERNAL I/F
PROCESSOR
PROCESSOR
PROCESSOR

Outline

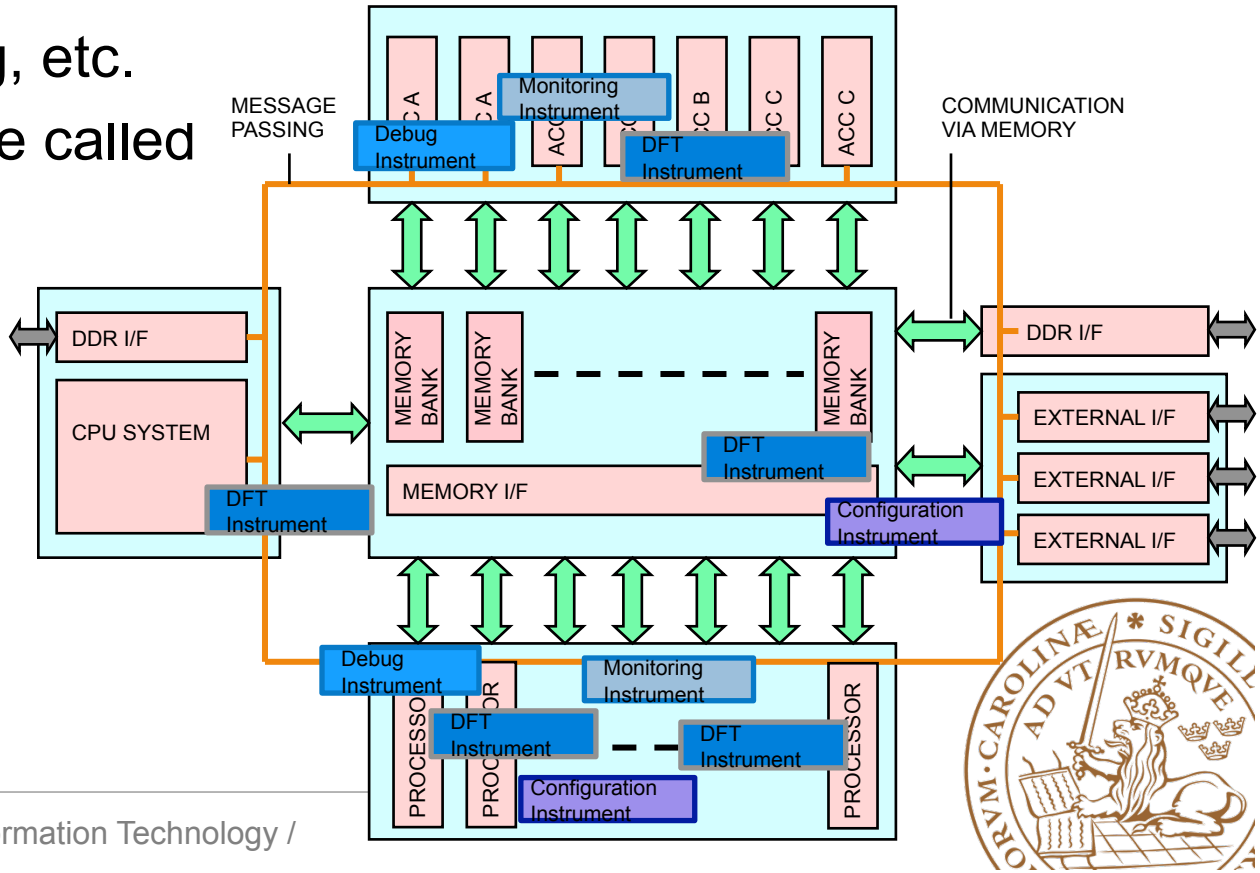
- Architecture to connect component-level with system-level
  - IEEE P1687 (IJTAG)
  - Error propagation
- Fault management
- Demonstrator

IEEE P1687 (IJTAG)

- IJTAG – Internal JTAG
- JTAG – Joint Test Access Group
- JTAG developed the IEEE 1149.1 standard, which often is called JTAG or Boundary Scan
- The objective of JTAG was to find a standardized and low cost solution to test printed circuit boards (PCBs).
- IC vendors included JTAG, which is utilized at board test
- The objective of IJTAG is to find a standardized and low cost solution to access internals of ICs….also after IC manufacturing test

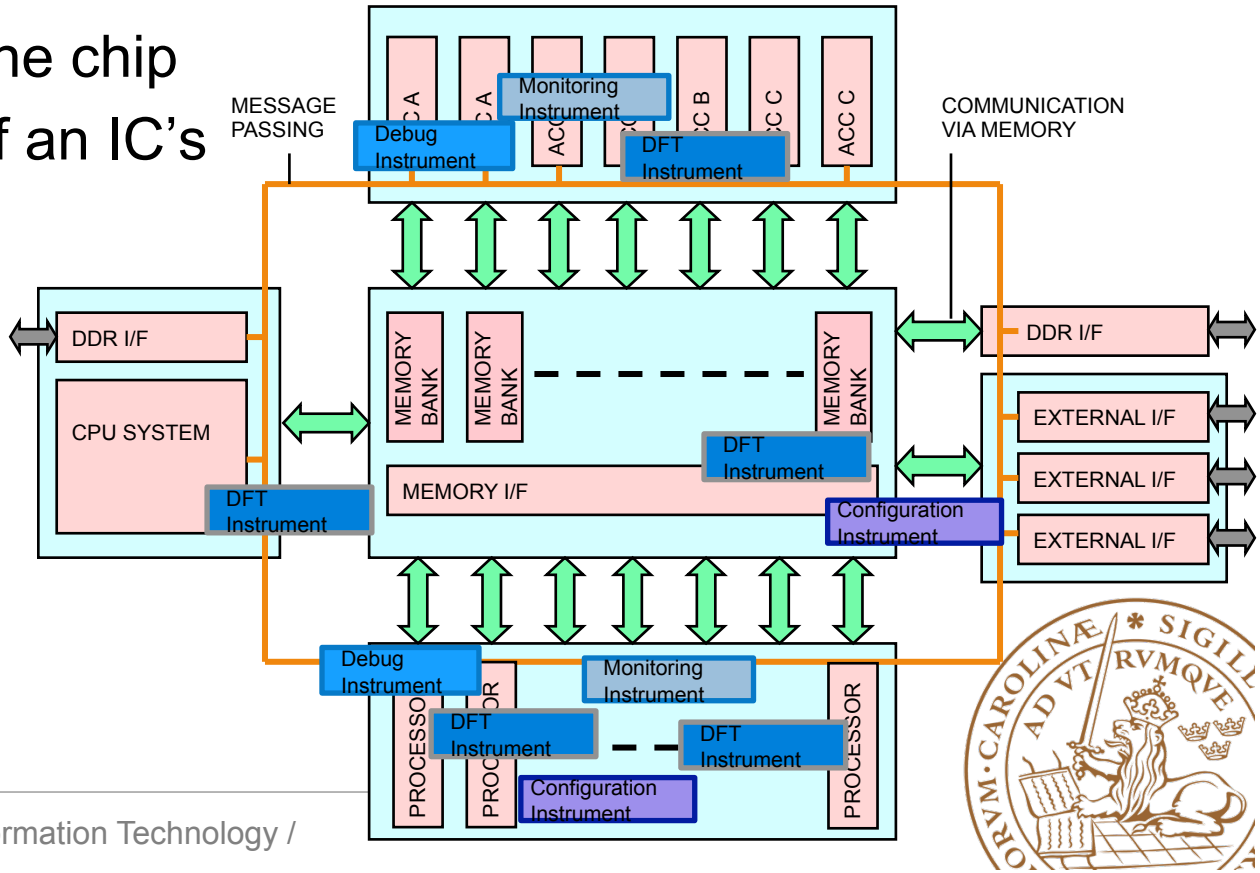
IEEE P1687 (IJTAG)

- Modern ICs contain many embedded features
  - for test, debug, etc.
- These features are called instruments

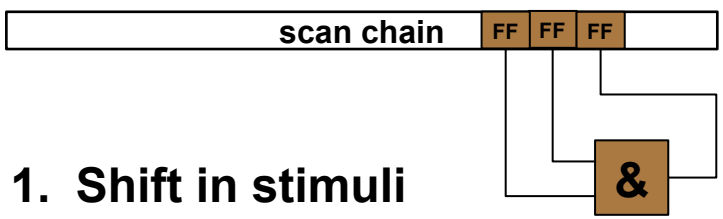


IEEE P1687 (IJTAG)

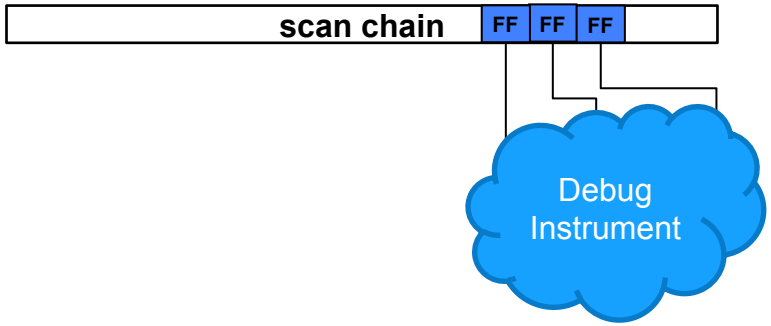
- There is a need to access the instruments
  - from outside the chip
  - in all stages of an IC's lifecycle
- JTAG serves this need

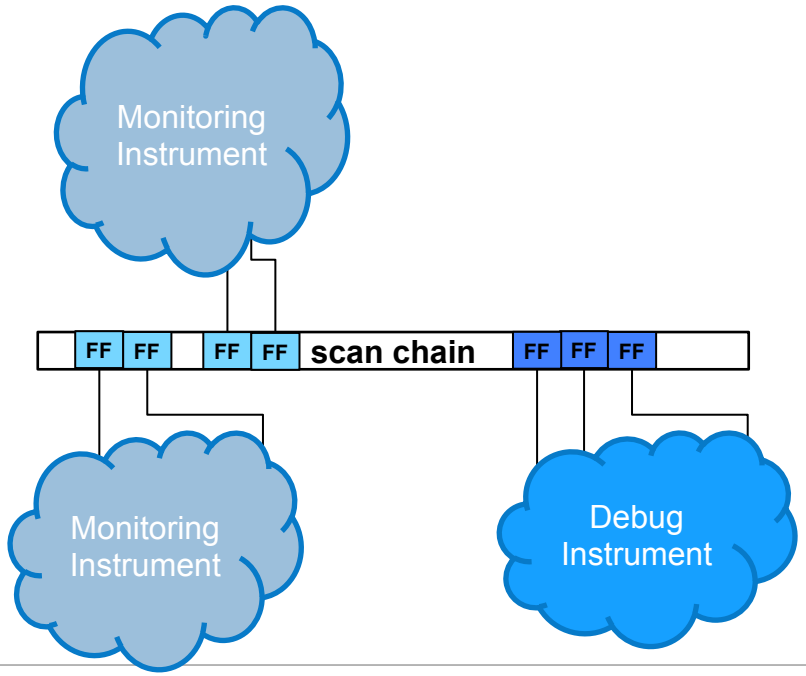


IEEE P1687 (IJTAG)

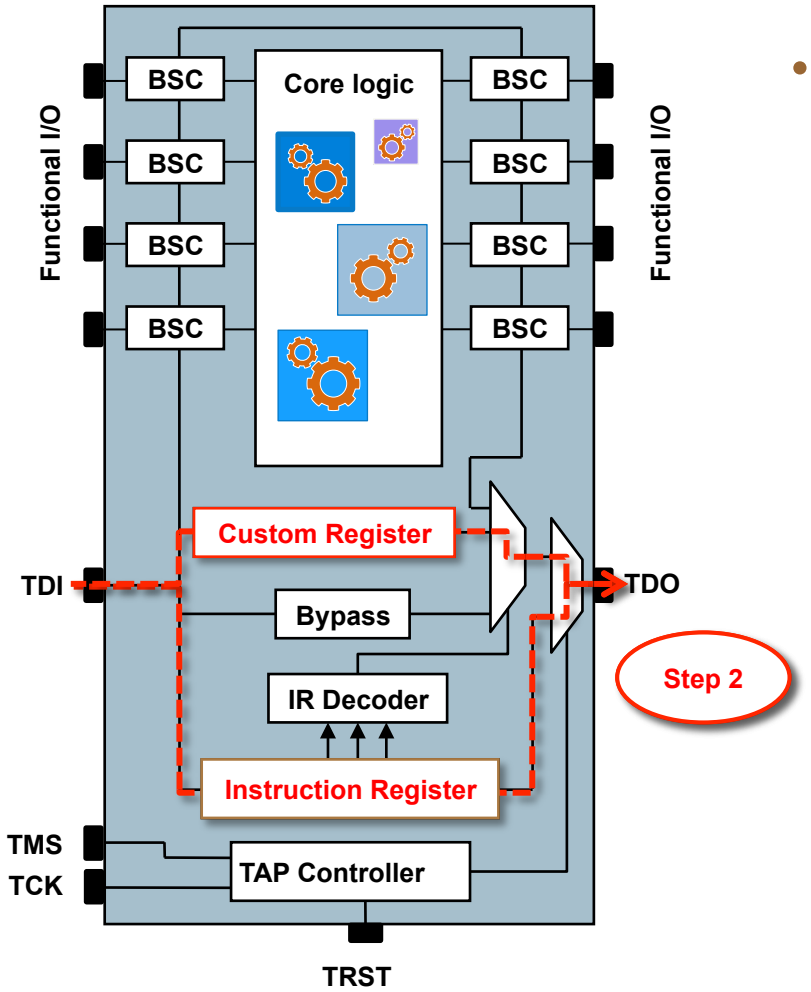


1. Shift in stimuli
2. Capture
3. Shift out responses

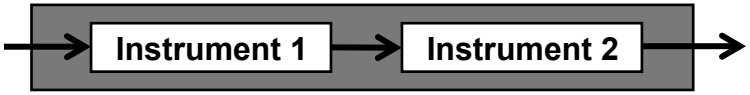




IEEE P1687 (IJTAG)

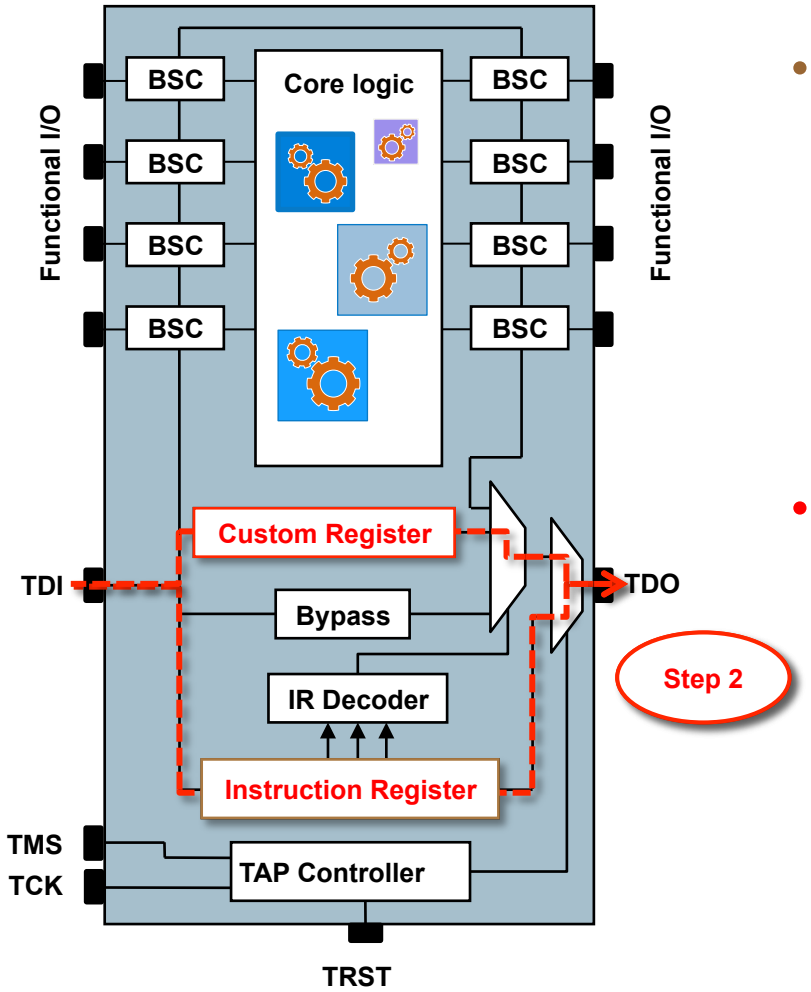


- Assume two instruments
- Alternative1: both instruments in one custom register

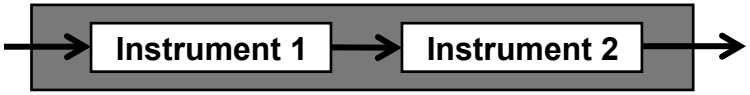


  - Step 1: Set instruction
  - Step 2: Transfer data

IEEE P1687 (IJTAG)



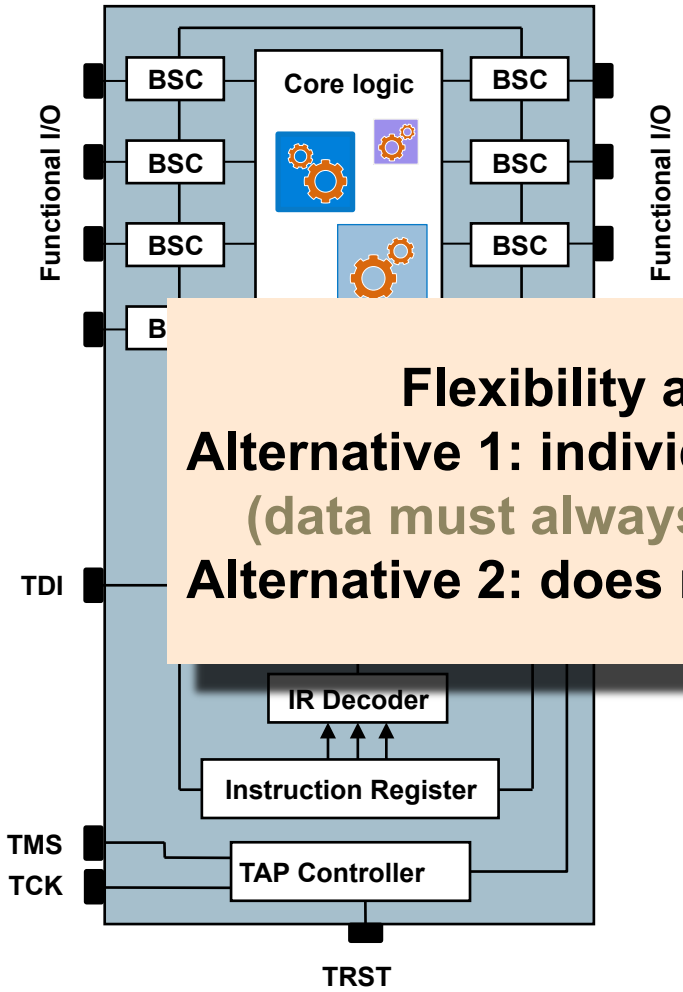
- Assume two instruments
- Alternative1: both instruments in one custom register



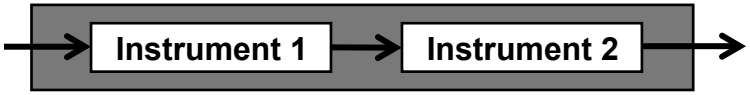
  - Step 1: Set instruction
  - Step 2: Transfer data
- Alternative 2: per instrument one custom register

  - For each instrument:
    - Step 1: Set instruction
    - Step 2: Transfer data

IEEE P1687 (IJTAG)



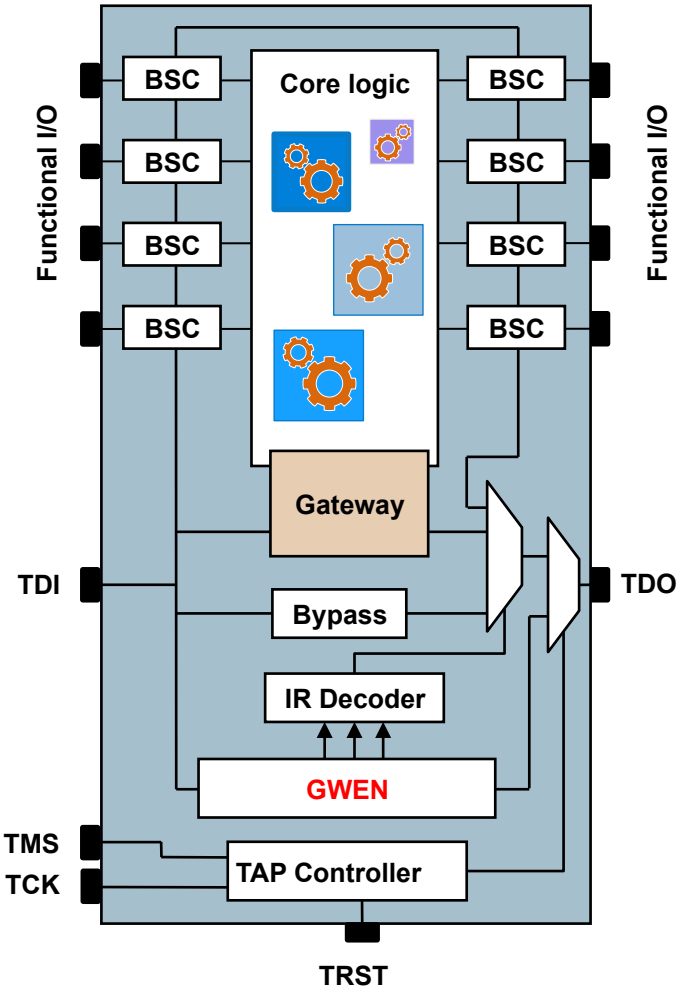
- Assume two instruments
- Alternative1: both instruments in one custom register

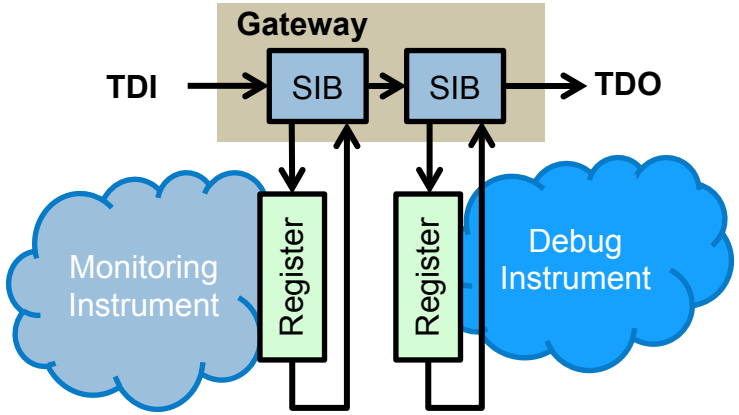


Flexibility and scalability are the problems!
Alternative 1: individual access results in high overhead
(data must always be shifted through both instruments)
Alternative 2: does not support concurrent access

- For each instrument:
  - Step 1: Set instruction
  - Step 2: Transfer data

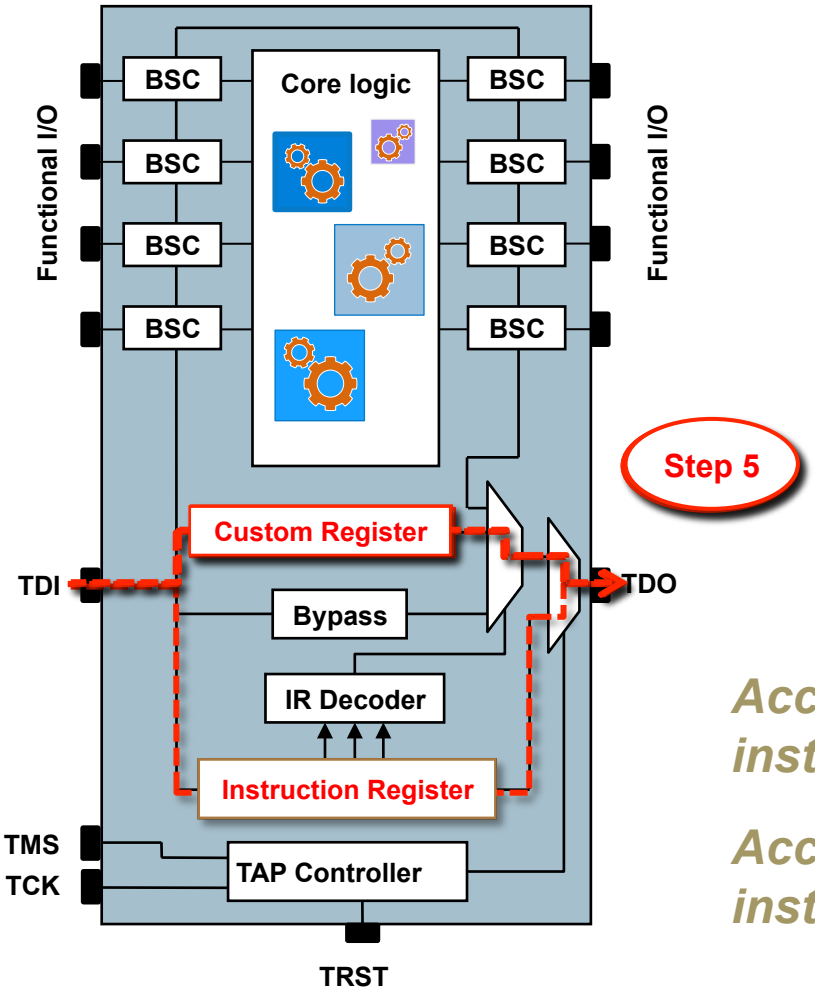
IEEE P1687 (IJTAG)



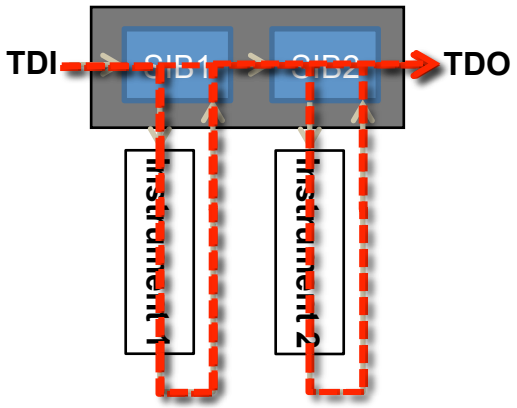


- **P1687 introduces**
  - **A single test data register (Gateway)**
  - **A single JTAG instruction (GWEN)**
  - **Segment Insertion Bit (SIB)**
- **Solves the flexibility and scalability problems**

IEEE P1687 (IJTAG)



- **Assume the same instruments**



Access both instruments

Access one instrument

- **Step 1: Set instruction**
- **Step 2: Program SIBs**
- **Step 3: Transfer data**
- **Step 4: Program SIBs**
- **Step 5: Transfer data**

IEEE P1687 (IJTAG)

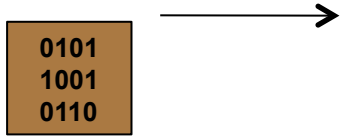
- JTAG:
  - Boundary Scan Definition Language (BSDL)
  - Serial Vector Format (SVF)
- BSDL
  - describes JTAG circuitry
- SVF
  - de facto language for operating the JTAG TAP
  - describes the test vectors
- There is no language connection between BSDL and SVF

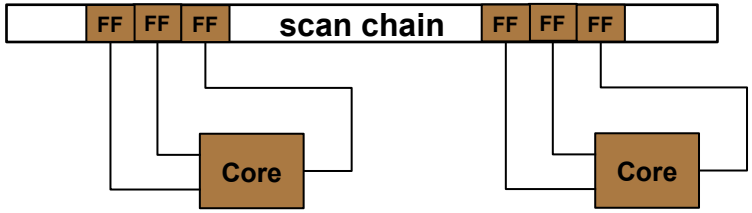
IEEE P1687 (IJTAG)

- P1687 introduces two languages:
  - Instrument Connectivity Language (ICL)
  - Pattern Description Language (PDL)
- ICL used to describe
  - instrument port functions
  - on-chip instrument access network
- PDL
  - used to describe how to operate an instrument
  - supports loops, parameters, enums, aliases, …
- ICL/PDL relation can be used for retargeting of instrument access procedures to higher levels in the design hierarchy (to the chip pins)

IEEE P1687 (IJTAG)

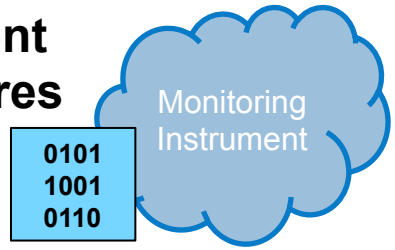
Core vendor
1. **Core**
2. **Test data**

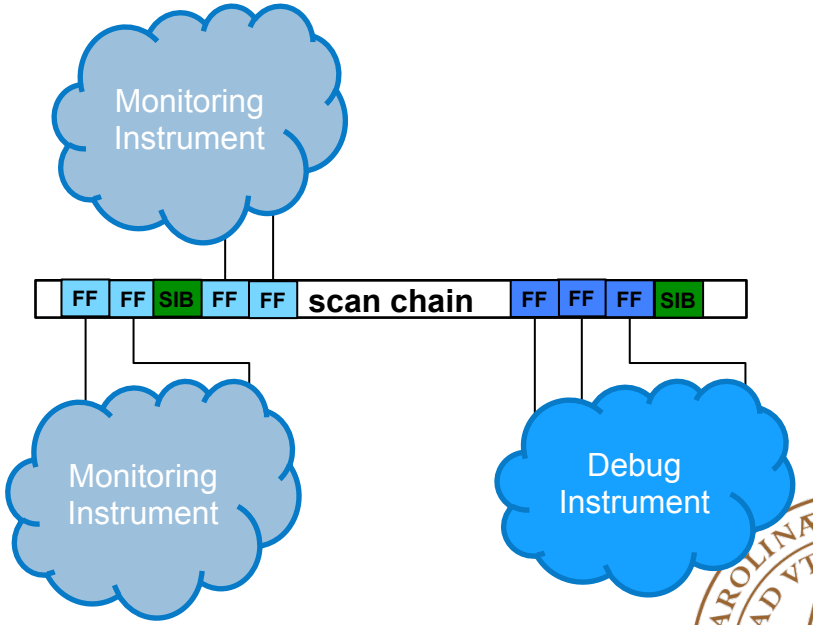




Put test data in tester corresponding to where core is in the chain

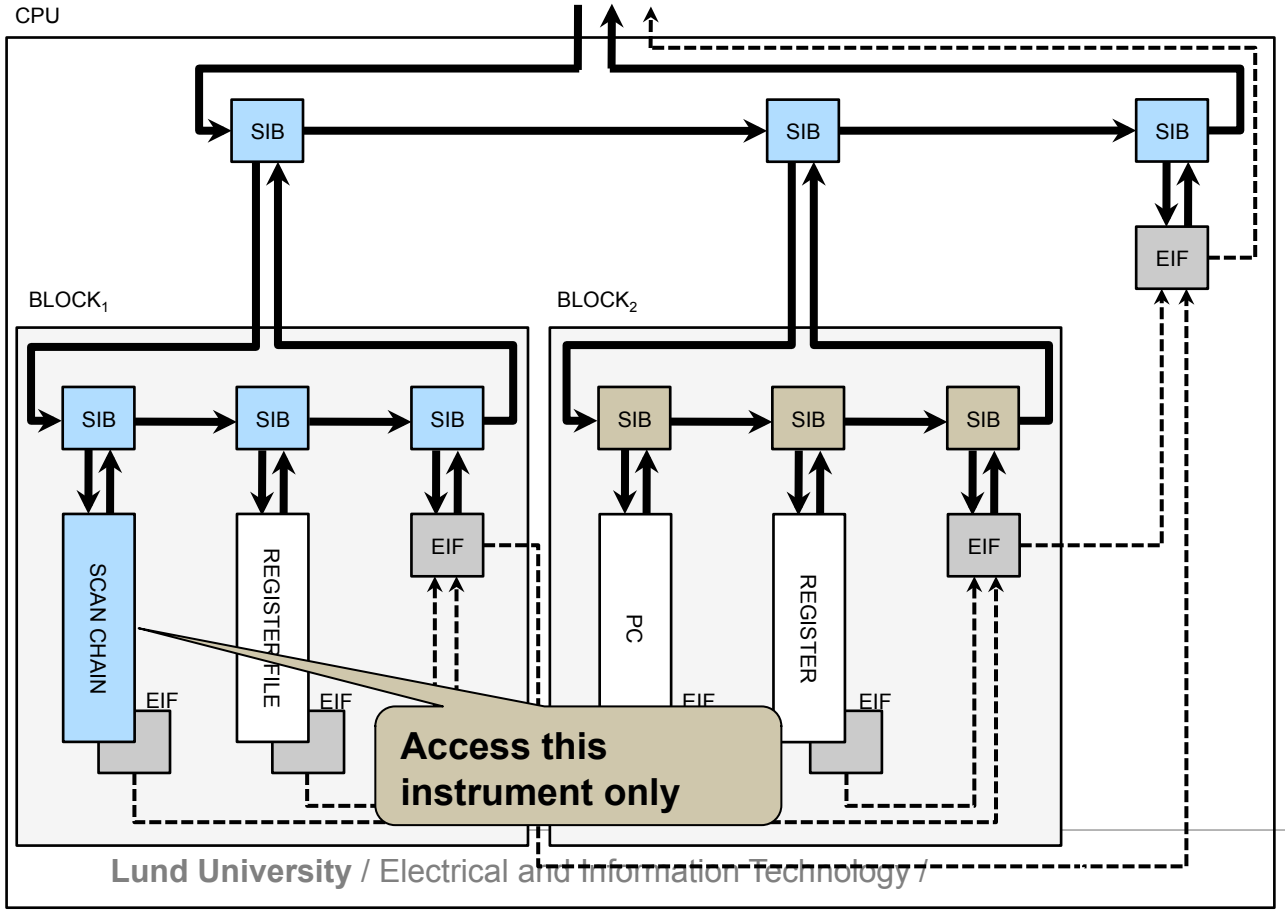
Instrument vendor:
1. **Instrument**
2. **Procedures**

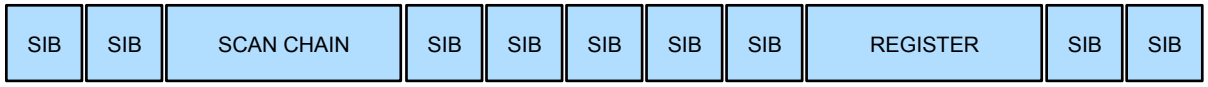


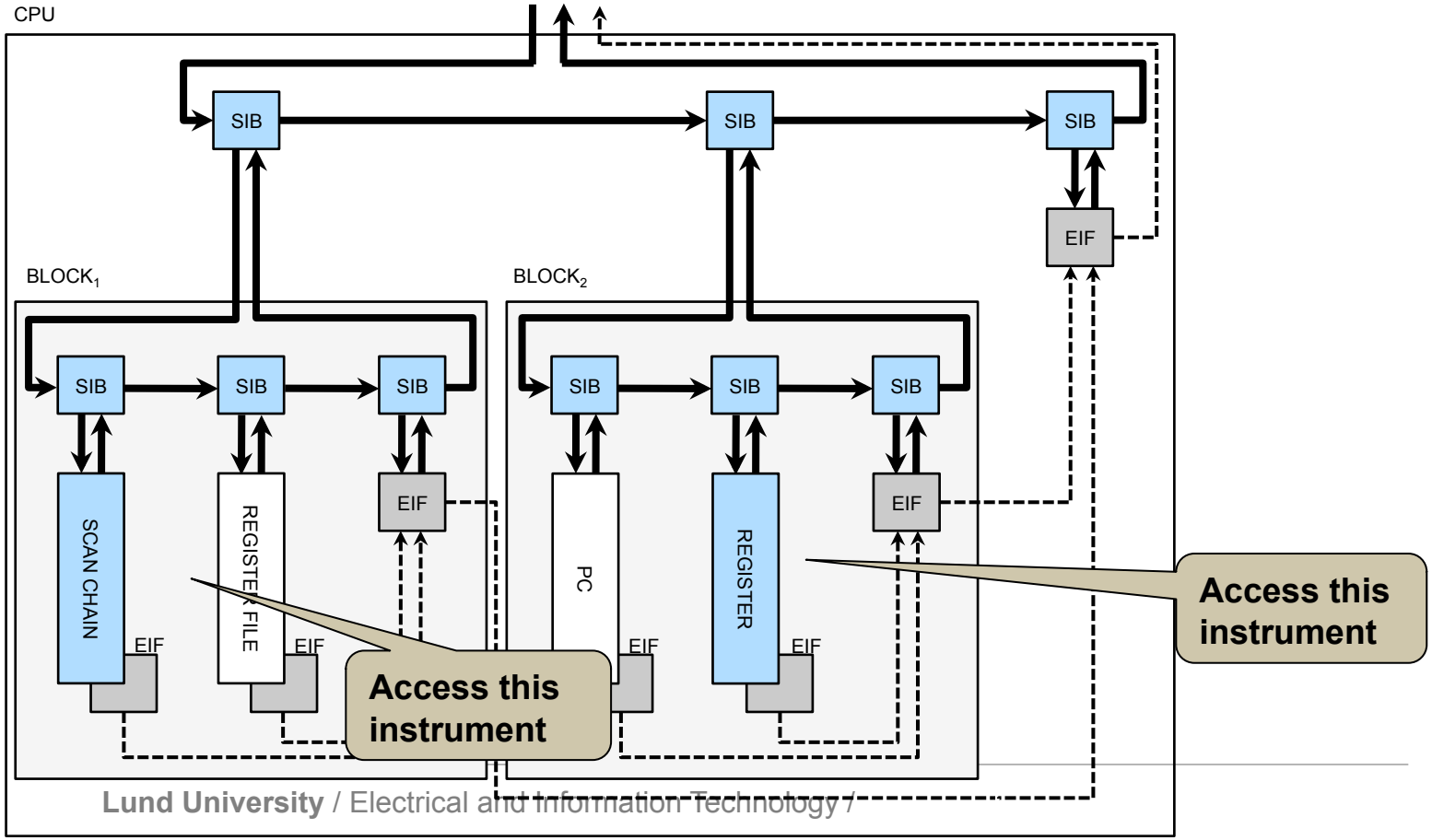


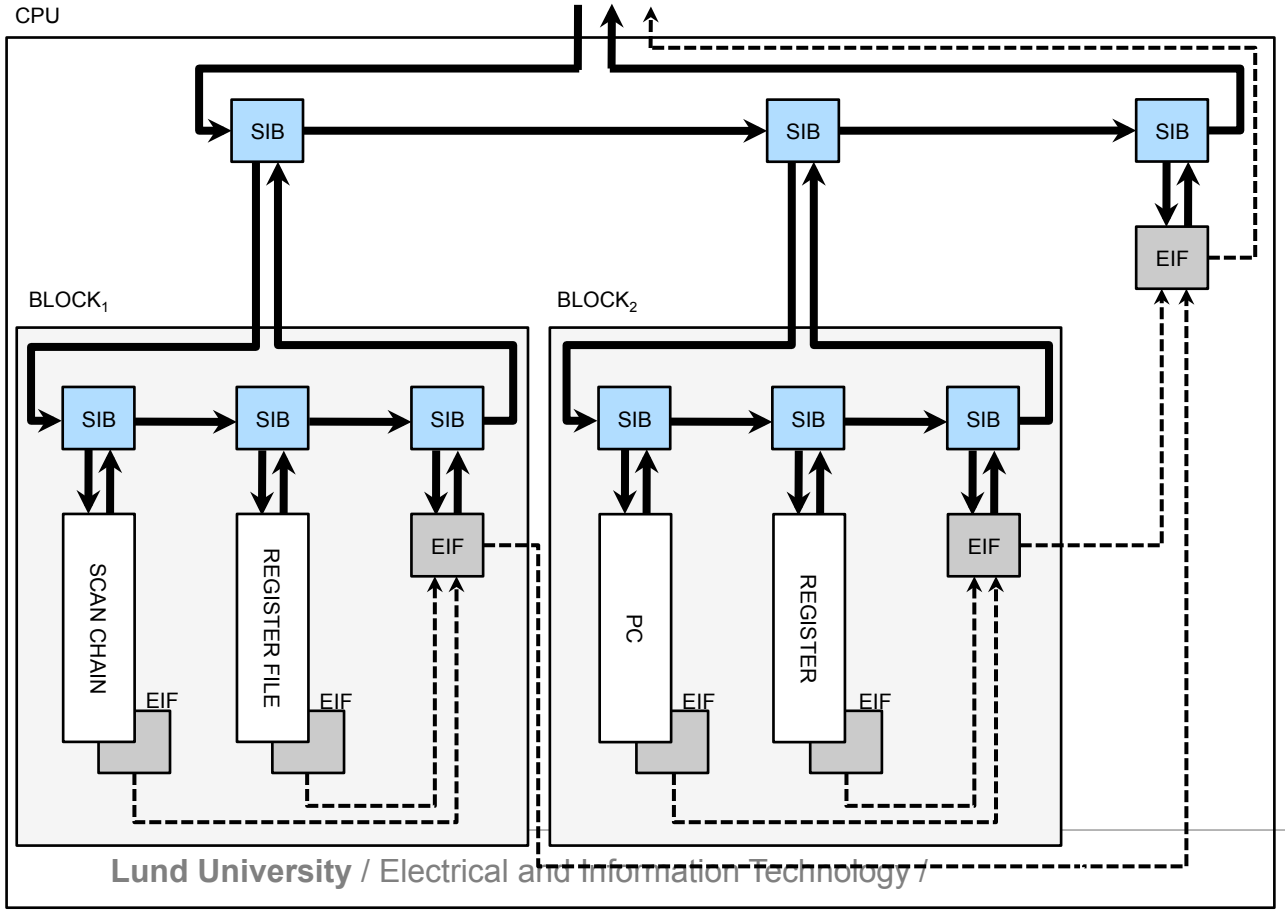
Chain changes based on SIBs depending on which instruments to access

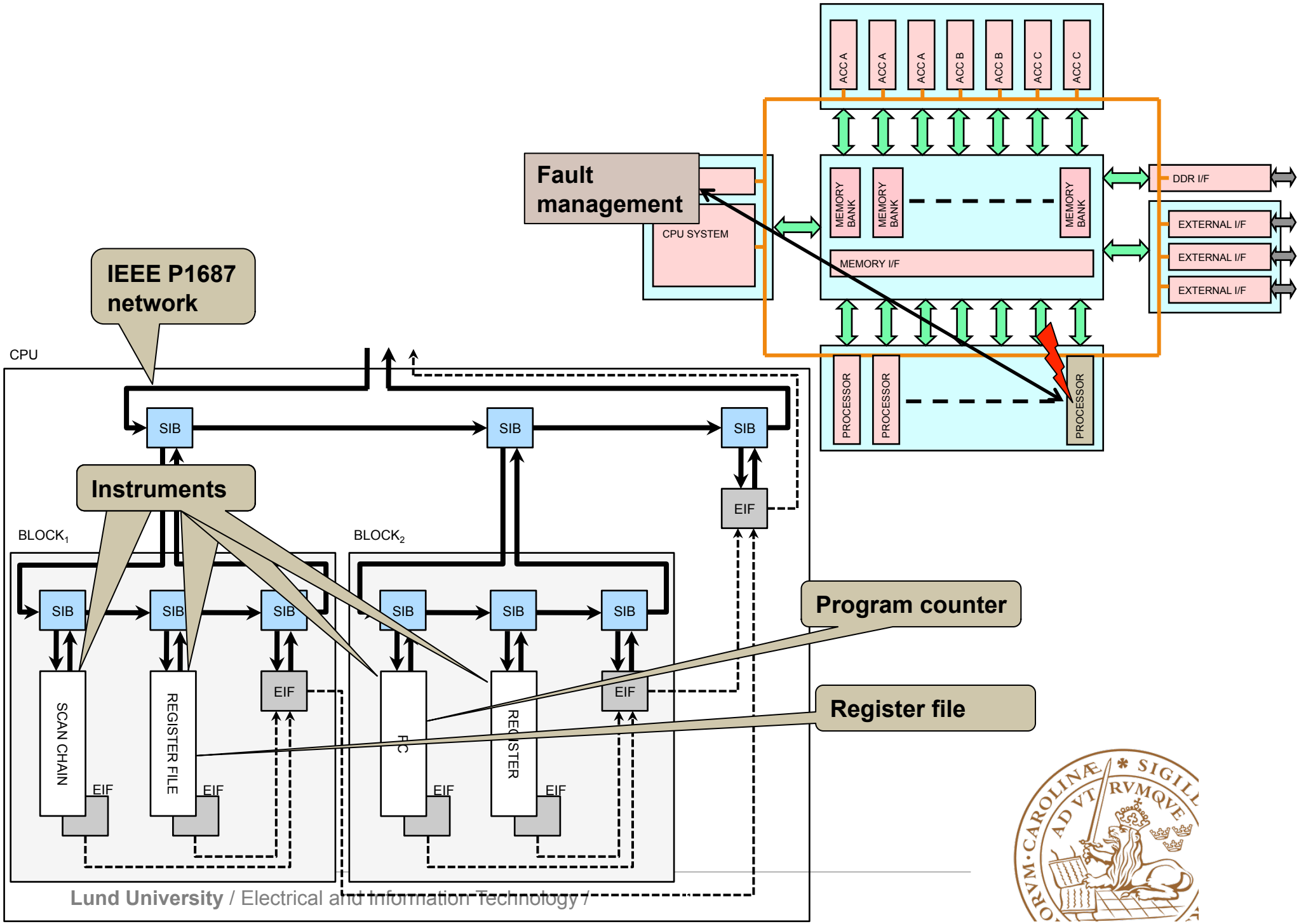

SIB
SIB
SCAN CHAIN
SIB
SIB
SIB
SIB

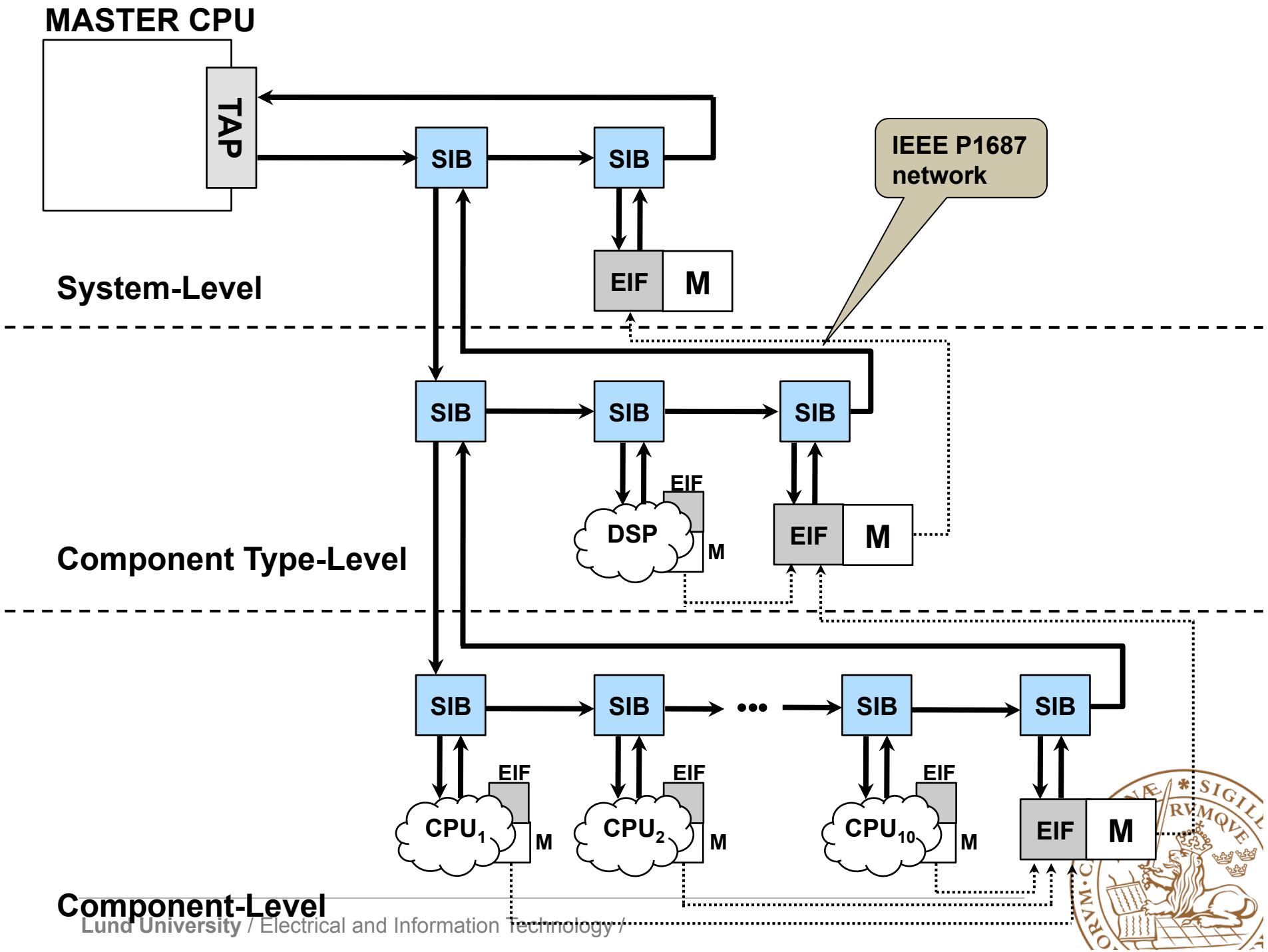

CPU
SIB
SIB
SIB
EIF
BLOCK1
BLOCK2
SIB
SIB
SIB
SIB
SIB
SIB
SCAN CHAIN
REGISTER FILE
EIF
PC
REGISTER
EIF
EIF
EIF
EIF
EIF
Access this instrument only


SIB
SIB
SCAN CHAIN
SIB
SIB
SIB
SIB
SIB
REGISTER
SIB
SIB


CPU
SIB
SIB
SIB
EIF
BLOCK1
BLOCK2
SIB
SIB
SIB
SIB
SIB
SIB
SCAN CHAIN
REGISTER FILE
EIF
PC
REGISTER
EIF
EIF
EIF
EIF
Access this instrument
Access this instrument


CPU
SIB
SIB
SIB
EIF
BLOCK1
BLOCK2
SIB
SIB
SIB
SIB
SIB
SIB
SCAN CHAIN
REGISTER FILE
EIF
PC
REGISTER
EIF
EIF
EIF
EIF
EIF


Fault management
IEEE P1687 network
Instruments
Program counter
Register file
CPU
CPU SYSTEM
ACC A
ACC A
ACC A
ACC B
ACC B
ACC C
ACC C
MEMORY BANK
MEMORY BANK
MEMORY BANK
MEMORY I/F
DDR I/F
EXTERNAL I/F
EXTERNAL I/F
EXTERNAL I/F
PROCESSOR
PROCESSOR
PROCESSOR
SIB
SIB
SIB
EIF
BLOCK1
BLOCK2
SIB
SIB
SIB
SIB
SIB
SIB
SCAN CHAIN
REGISTER FILE
FIC
REGISTER
EIF
EIF
EIF
EIF
EIF
EIF


MASTER CPU
TAP
SIB
SIB
IEEE P1687
network
EIF
M
System-Level
SIB
SIB
SIB
EIF
DSP
M
EIF
M
Component Type-Level
SIB
SIB
•••
SIB
SIB
EIF
EIF
EIF
CPU1
M
CPU2
M
CPU10
M
EIF
M
Component-Level

IEEE P1687 (IJTAG)

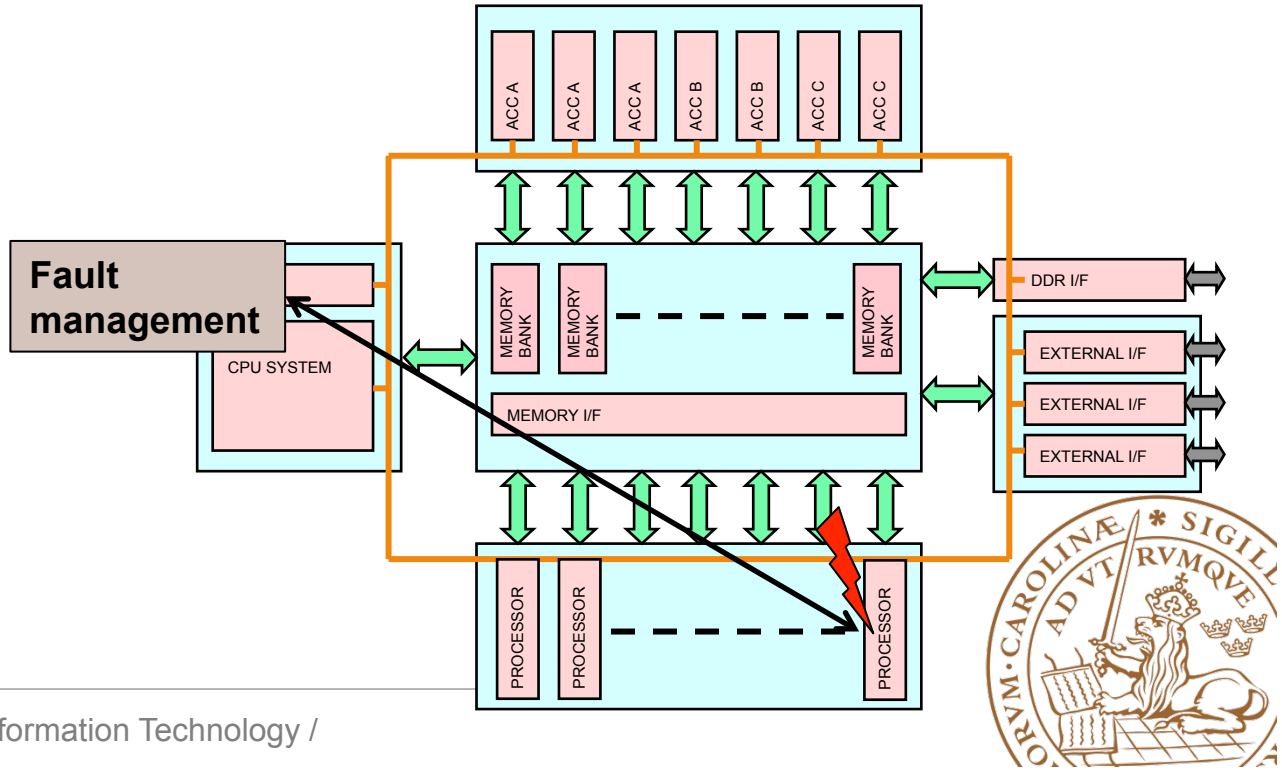
- IEEE P1687 network designed in a hierarchical way to ease access to each component
- However, polling for errors in the network is time consuming.
- If there are 100 components, polling will take at least 100 clock cycles.

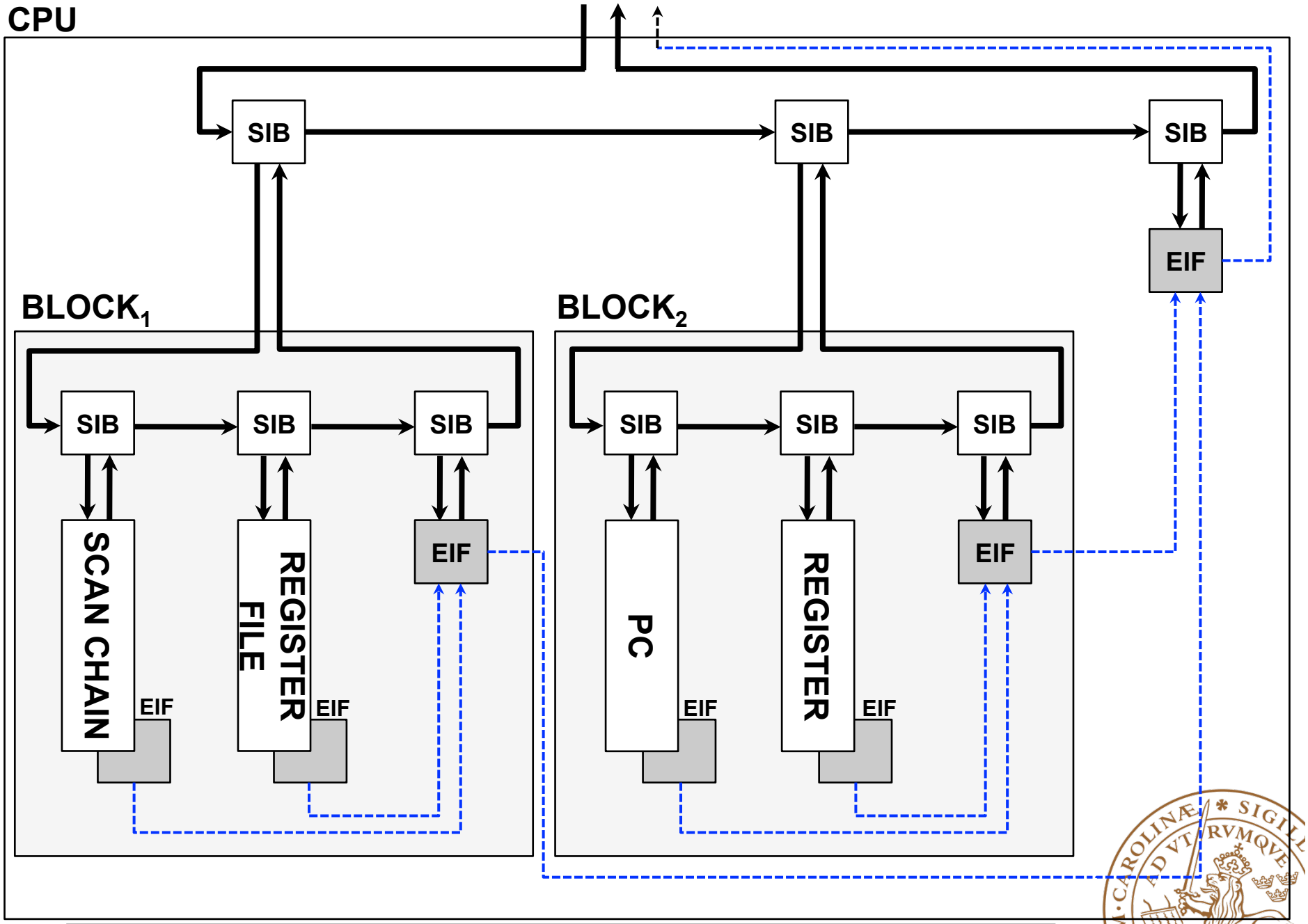
Outline

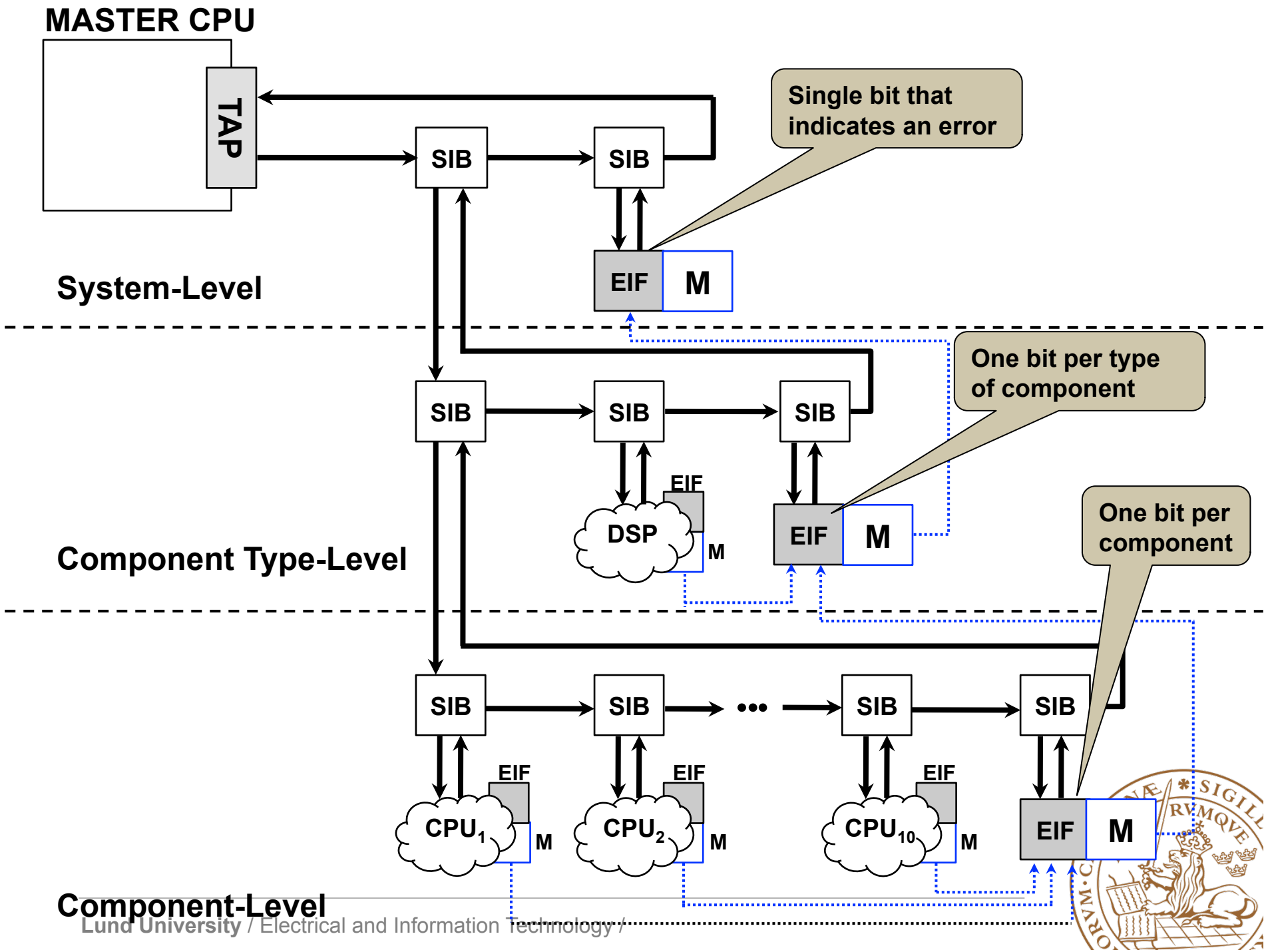
- Architecture to connect component-level with system-level
  - IEEE P1687 (IJTAG)
  - Error propagation
- Fault management
- Demonstrator

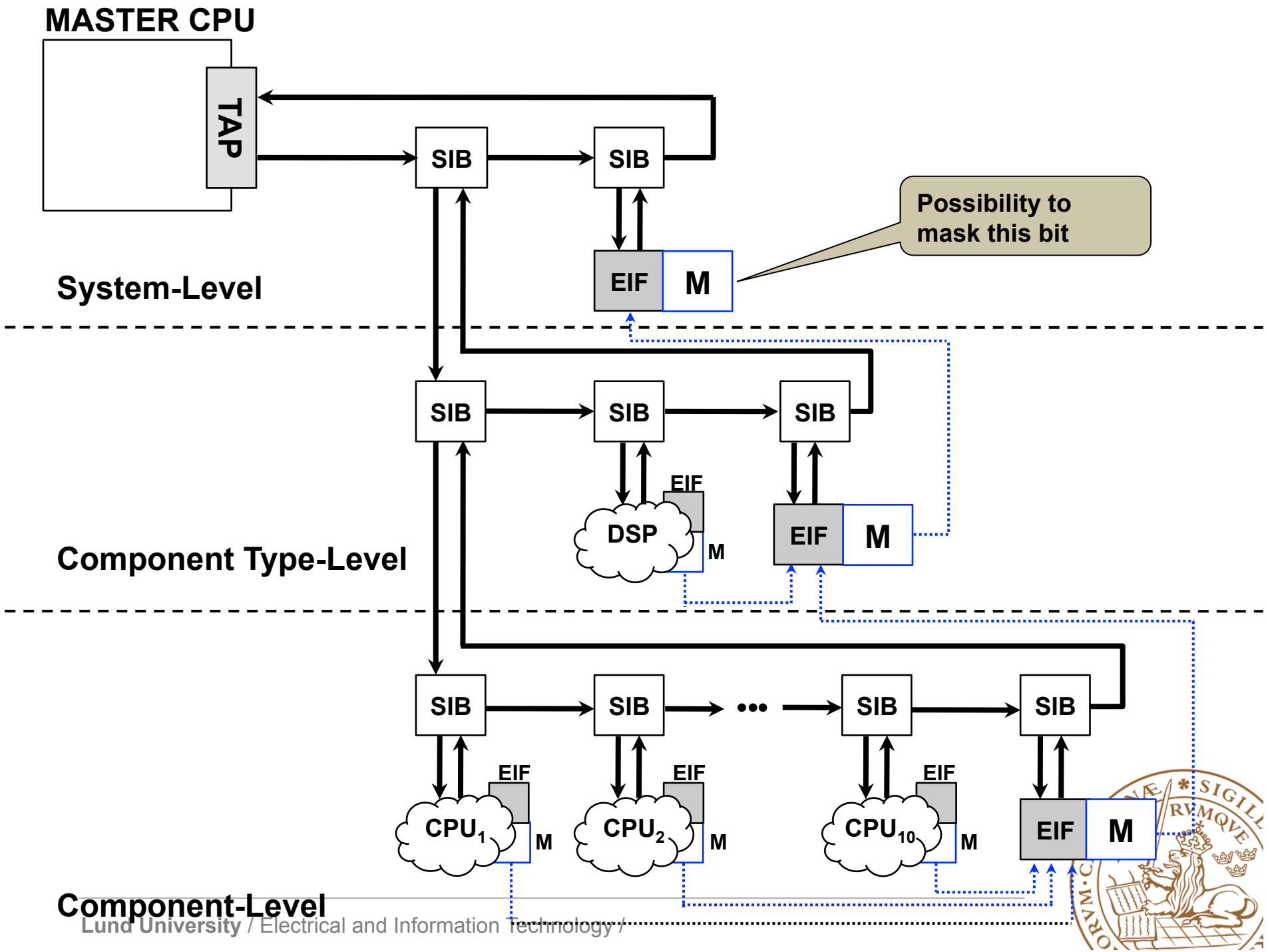
Error propagation

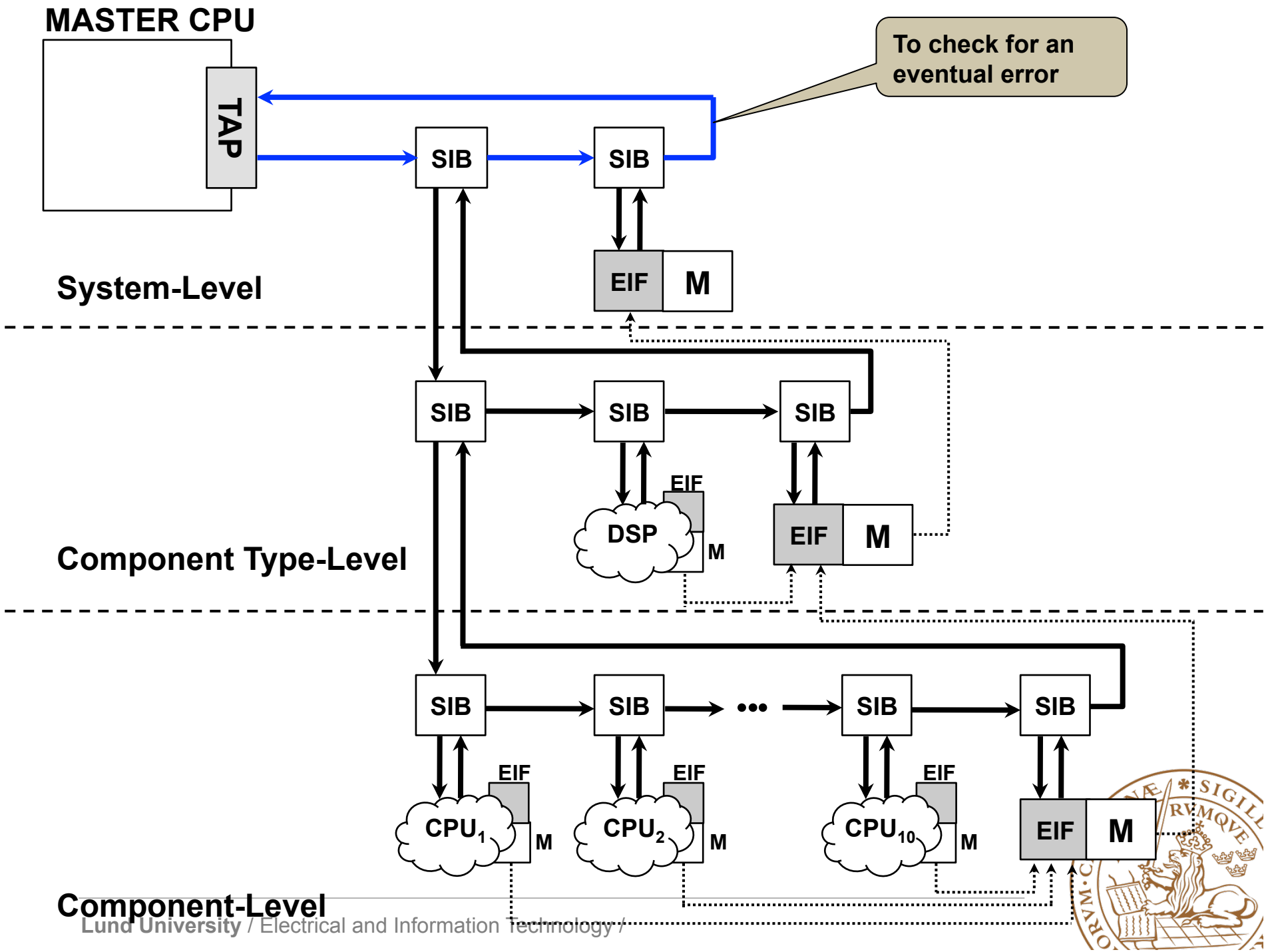
- To reduce the polling time, we have designed an error propagation network

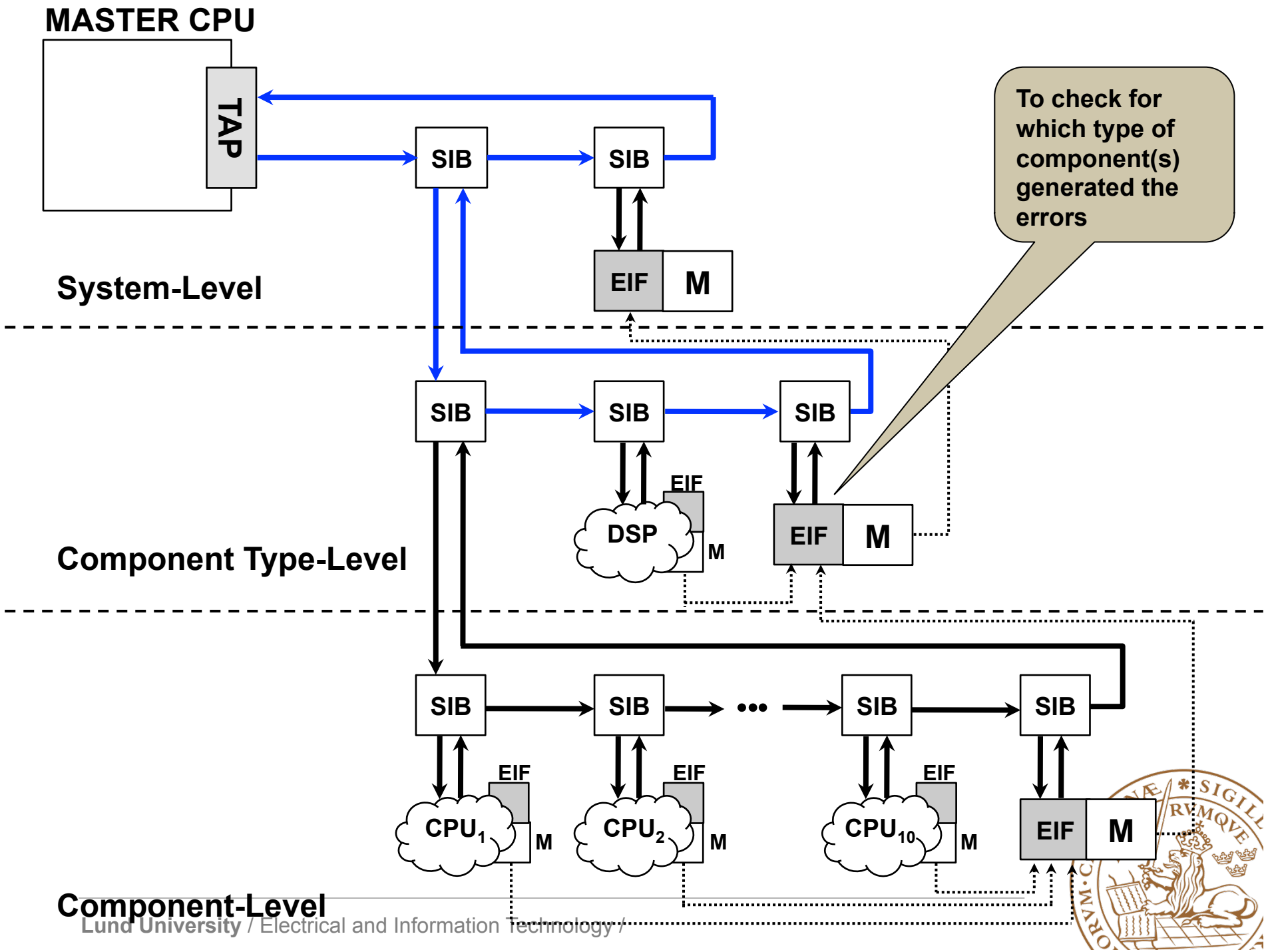


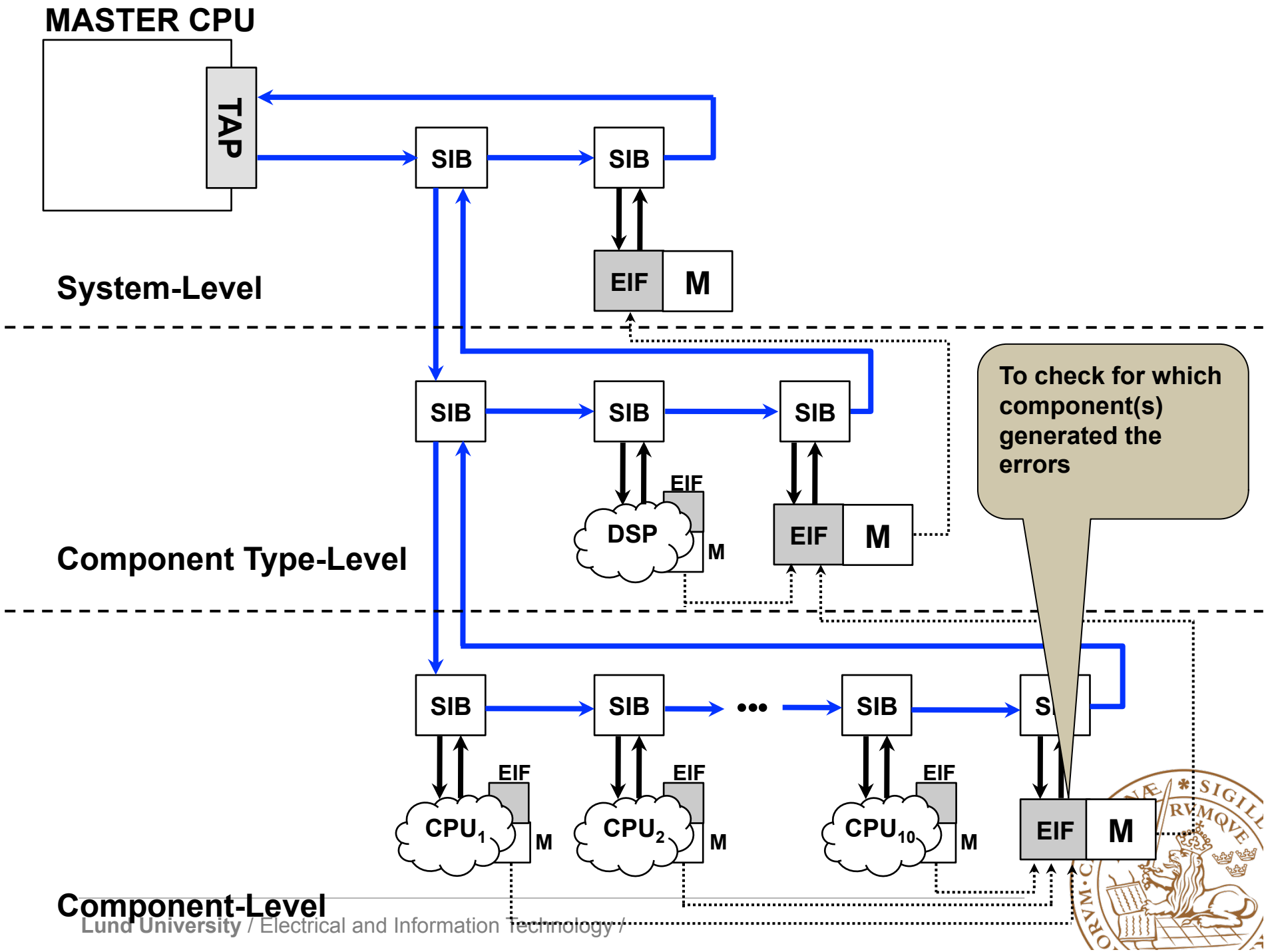

CPU
SIB
SIB
SIB
EIF
BLOCK1
BLOCK2
SIB
SIB
SIB
SIB
SIB
SIB
SCAN CHAIN
REGISTER FILE
EIF
PC
REGISTER
EIF
EIF
EIF
EIF
EIF

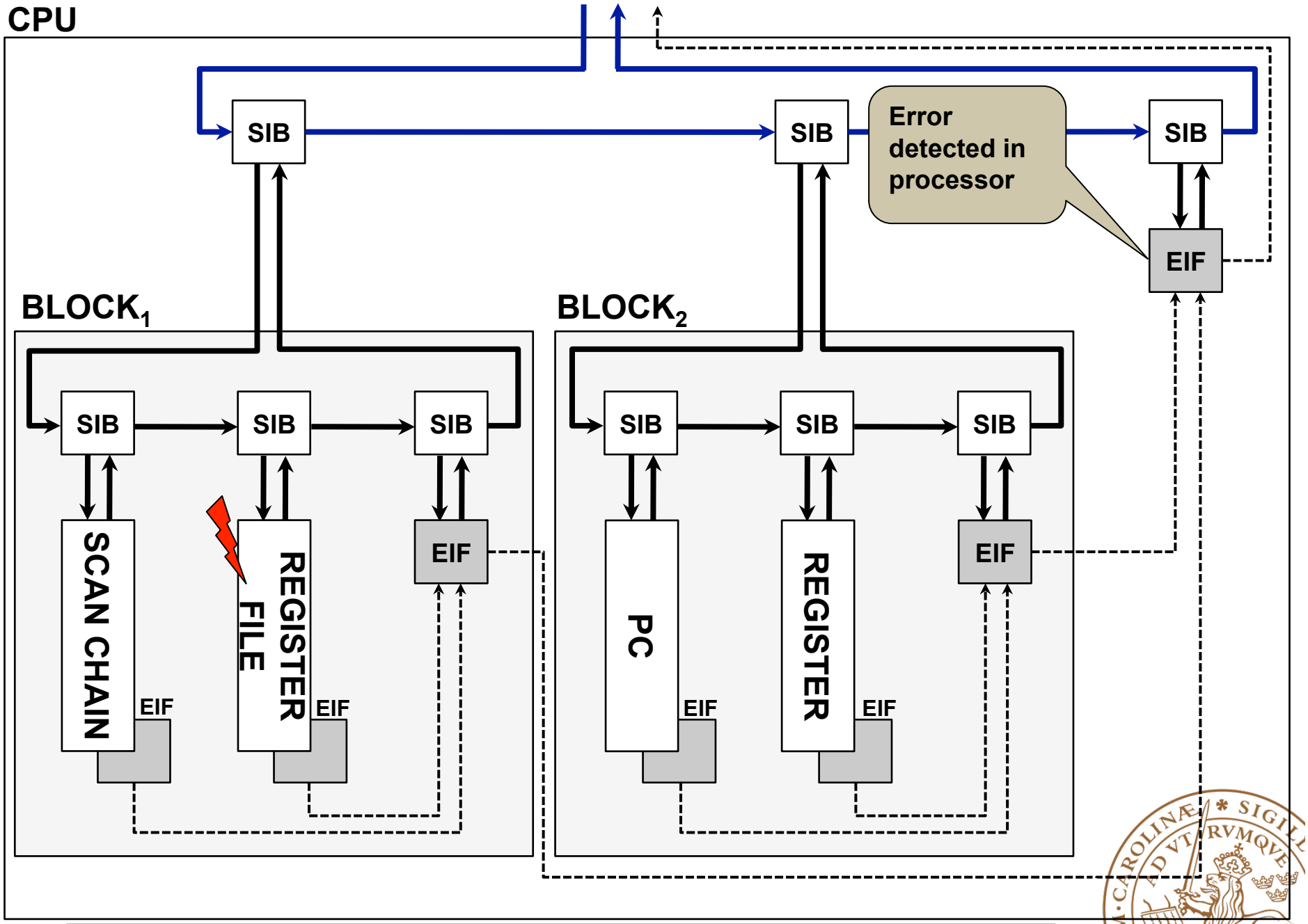

MASTER CPU
TAP
SIB
SIB
Single bit that indicates an error
EIF
M
System-Level
SIB
SIB
SIB
One bit per type of component
EIF
DSP
M
EIF
M
Component Type-Level
One bit per component
SIB
SIB
SIB
SIB
EIF
CPU1
M
EIF
CPU2
M
EIF
CPU10
M
EIF
M
Component-Level

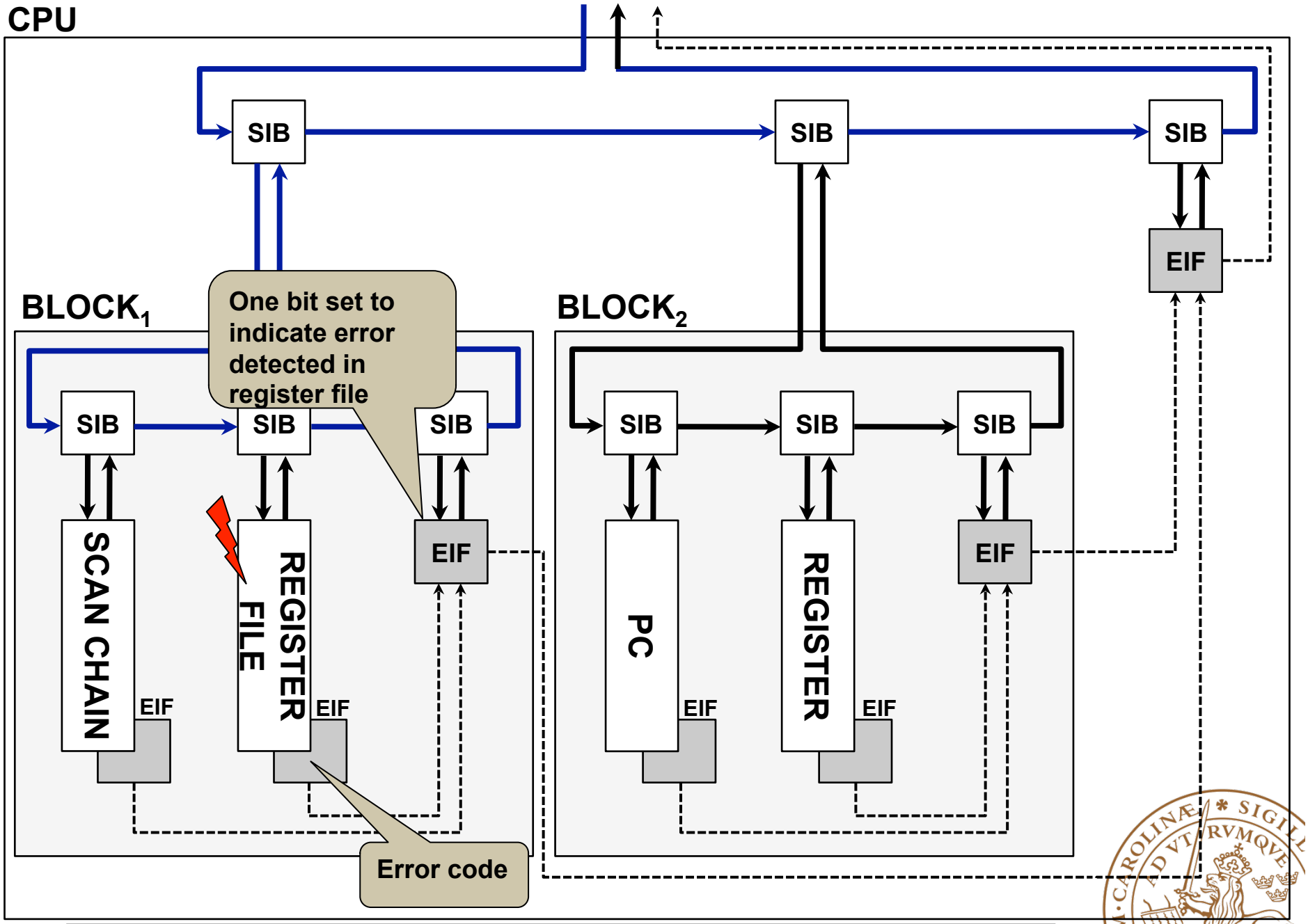

MASTER CPU
TAP
SIB
SIB
EIF
M
Possibility to mask this bit
System-Level
SIB
SIB
SIB
EIF
DSP
M
EIF
M
Component Type-Level
SIB
SIB
SIB
SIB
EIF
CPU1
M
EIF
CPU2
M
EIF
CPU10
M
EIF
M
Component-Level

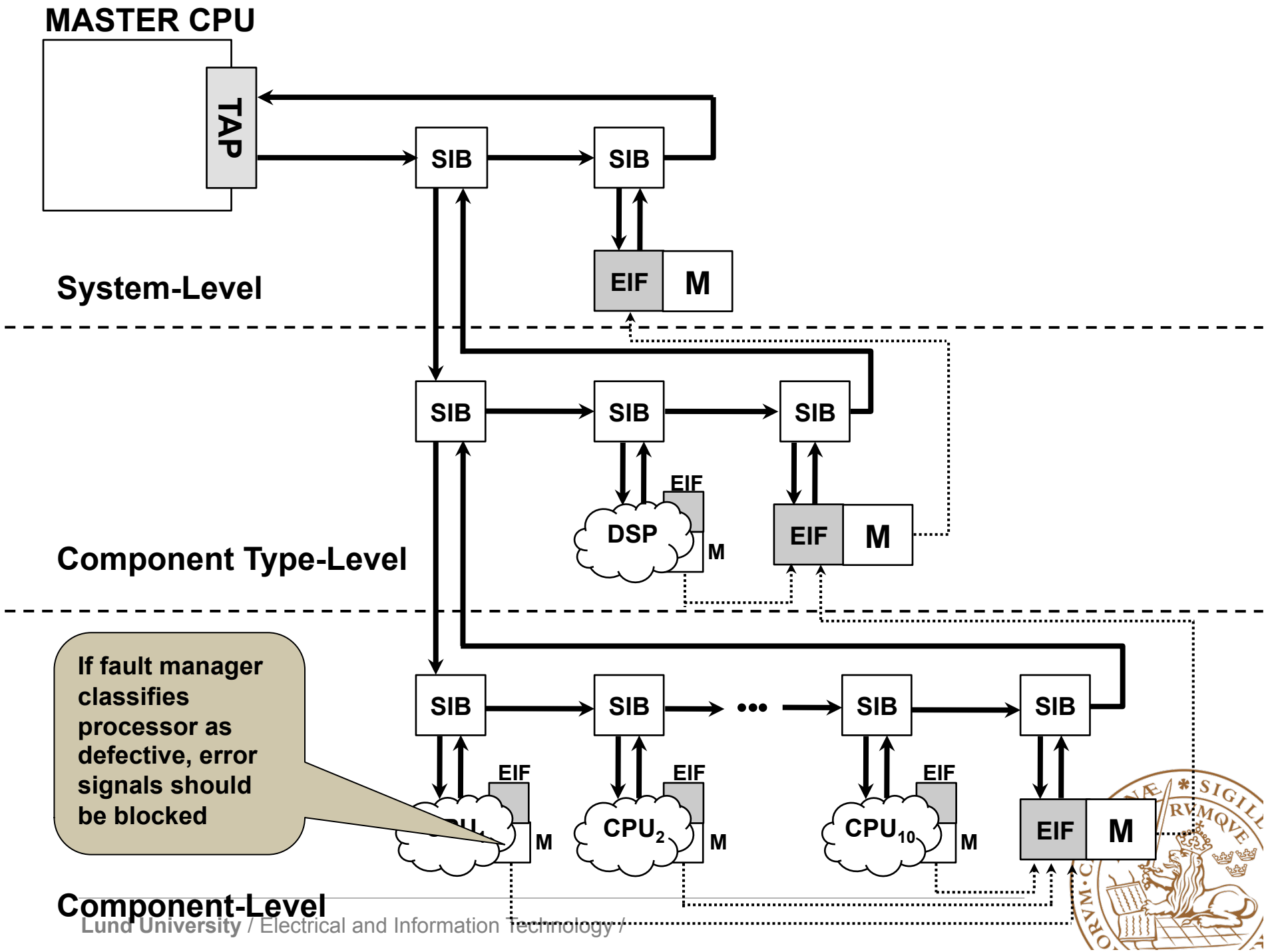

MASTER CPU
TAP
To check for an eventual error
SIB
SIB
EIF
M
System-Level
SIB
SIB
SIB
EIF
DSP
M
EIF
M
Component Type-Level
SIB
SIB
SIB
SIB
EIF
CPU1
M
EIF
CPU2
M
EIF
CPU10
M
EIF
M
Component-Level


MASTER CPU
TAP
SIB
SIB
EIF
M
To check for which type of component(s) generated the errors
System-Level
SIB
SIB
SIB
EIF
DSP
M
EIF
M
Component Type-Level
SIB
SIB
SIB
SIB
EIF
CPU1
M
EIF
CPU2
M
EIF
CPU10
M
EIF
M
Component-Level


MASTER CPU
TAP
SIB
SIB
EIF
M
System-Level
SIB
SIB
SIB
EIF
DSP
M
EIF
M
To check for which component(s) generated the errors
Component Type-Level
SIB
SIB
SIB
SIB
EIF
CPU1
M
EIF
CPU2
M
EIF
CPU10
M
EIF
M
Component-Level


CPU
SIB
SIB
Error detected in processor
SIB
EIF
BLOCK1
BLOCK2
SIB
SIB
SIB
SIB
SIB
SIB
SCAN CHAIN
REGISTER FILE
EIF
PC
REGISTER
EIF
EIF
EIF
EIF
EIF


CPU
SIB
SIB
SIB
EIF
BLOCK1
One bit set to indicate error detected in register file
BLOCK2
SIB
SIB
SIB
SIB
SIB
SIB
SCAN CHAIN
REGISTER FILE
EIF
EIF
EIF
PC
REGISTER
EIF
EIF
EIF
Error code


MASTER CPU
TAP
SIB
SIB
EIF
M
System-Level
SIB
SIB
SIB
EIF
DSP
M
EIF
M
Component Type-Level
If fault manager classifies processor as defective, error signals should be blocked
SIB
SIB
SIB
SIB
EIF
EIF
EIF
EIF
M
CPU2
M
CPU10
M
M
Component-Level

Outline

- Architecture to connect component-level with system-level
  - IEEE P1687 (IJTAG)
  - Error propagation
- Fault management
- Demonstrator

Fault management

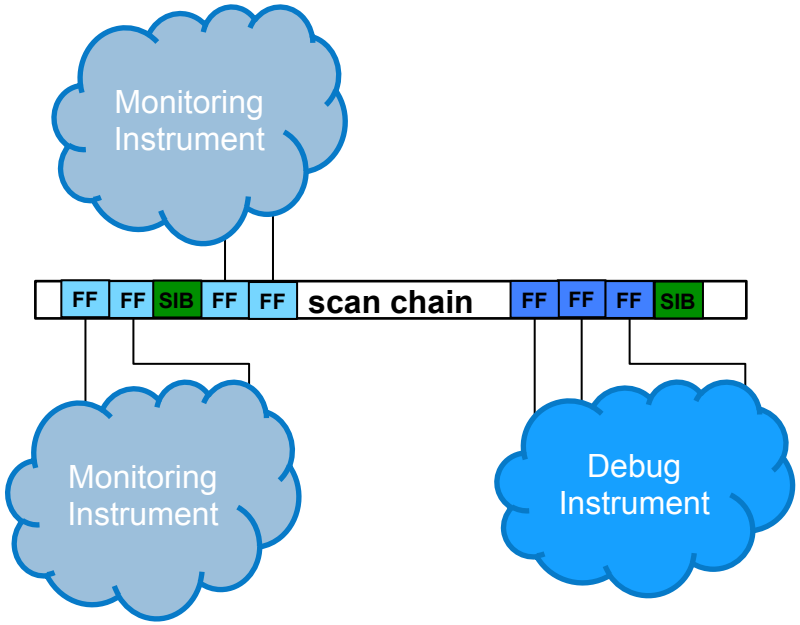
- Resource manager
- Instrument manager

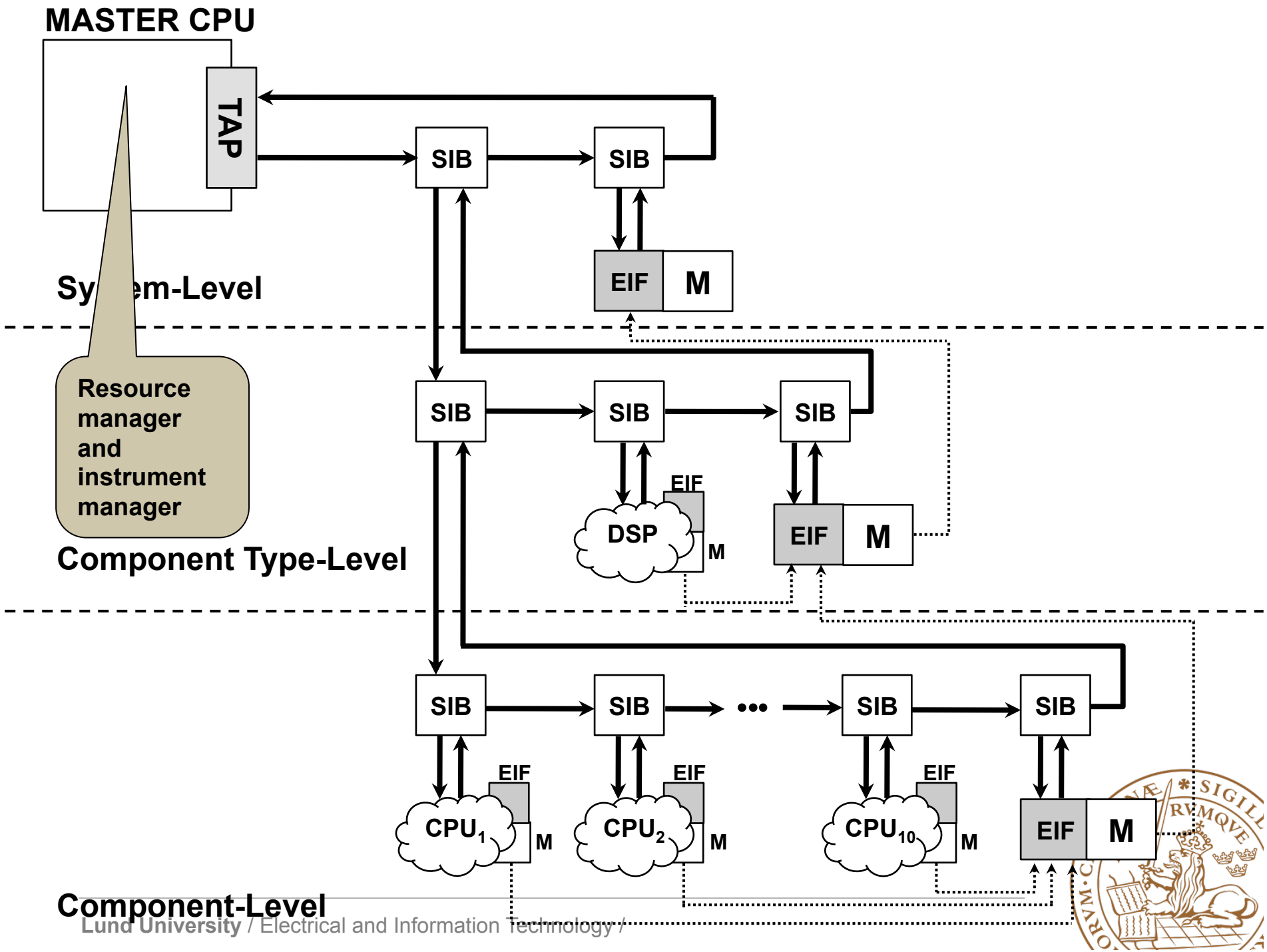
Fault management

- Resource manager
  - Receives error message
  - Identifies defective component
  - Check history (fault map) to see how defective a component is
  - If component is too defective (above threshold), classify component as defective
  - Otherwise, re-execute job

Fault management

- Instrument manager
  - Translates the commands from the resource manager to bit strings (JTAG/IJTAG)



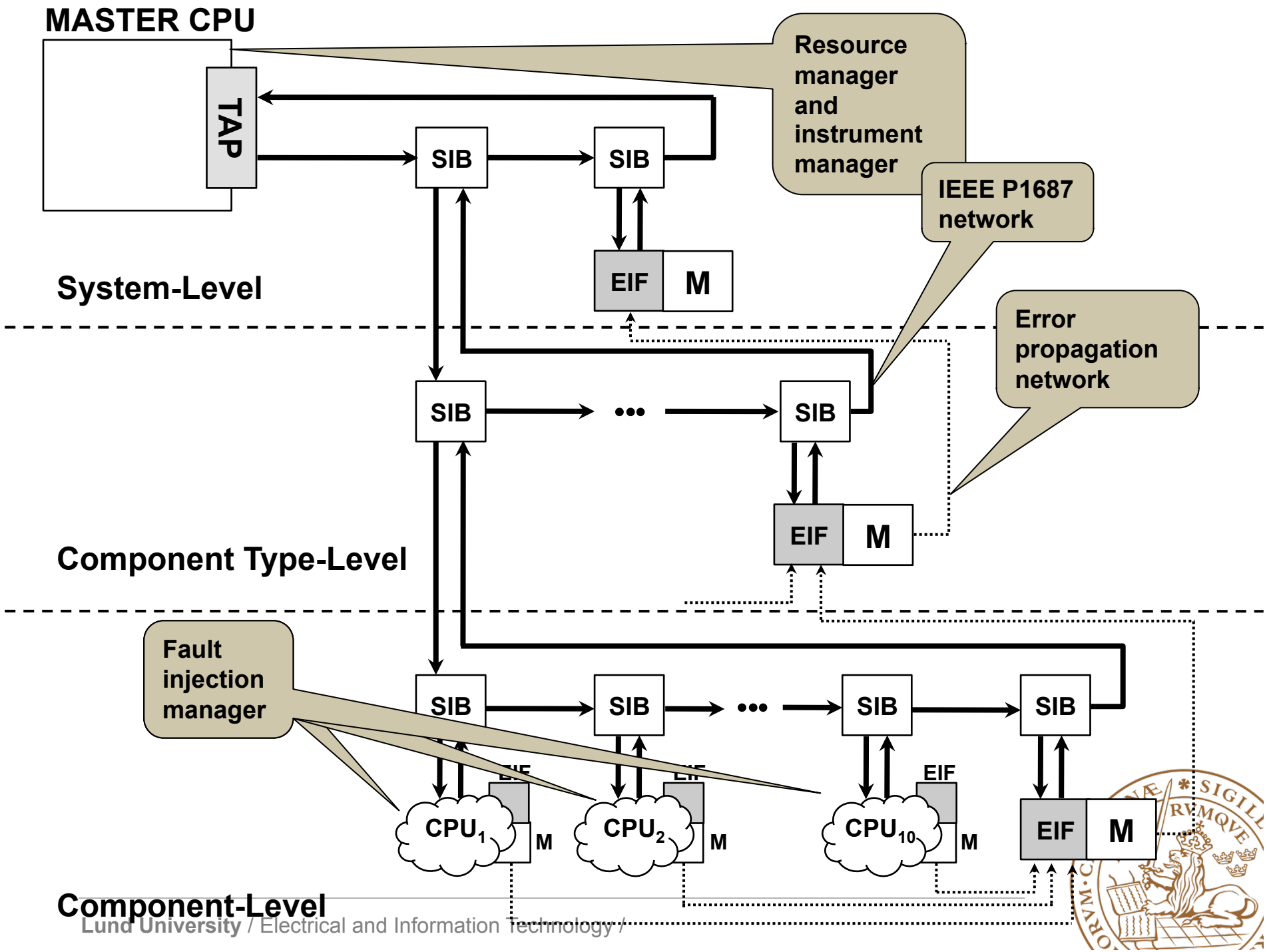

MASTER CPU
TAP
SIB
SIB
EIF
M
System-Level
Resource manager and instrument manager
SIB
SIB
SIB
EIF
DSP
M
EIF
M
Component Type-Level
SIB
SIB
SIB
SIB
EIF
CPU1
M
EIF
CPU2
M
EIF
CPU10
M
EIF
M
Component-Level

Outline

- Architecture to connect component-level with system-level
  - IEEE P1687 (IJTAG)
  - Error propagation
- Fault management
- Demonstrator

Demonstration

- Resource manager
  (implemented in C)
- Instrument manager
  (implemented in C)
- Fault injection manager
  (implemented in C)
  - Prior to execution, define the error profile – when and what types of errors to occur.
- System (VHDL)
  - 10 processors


MASTER CPU
TAP
SIB
SIB
EIF
M
Resource manager and instrument manager
IEEE P1687 network
System-Level
Error propagation network
SIB
SIB
EIF
M
Component Type-Level
Fault injection manager
SIB
SIB
SIB
SIB
EIF
EIF
CPU1
M
CPU2
M
CPU10
M
EIF
M
Component-Level

Demonstration

- Case one: show how resource manager is notified when one processor is affected by a soft error.
- Given: job list, fault profile
- Action:
    1. Error indication indicates that an error has occurred
    2. The faulty processor is identified
    3. The job is re-executed
    4. The error should be reported

Demonstration

- Case two: a system with 10 processors where one processor is affected by a hard error and others with soft errors
- Given: job list, fault profile, threshold on classification
- Action:
    1. Error indication indicates that an error has occurred
    2. The faulty processor is identified
    3. The job is re-executed
    4. The error should be reported
    5. Repeated reporting indicates a hard error

References

- Dimitar Nikolov, Mikael Väyrynen, Urban Ingelsson, Erik Larsson, and Virendra Singh, Optimizing Fault Tolerance for Multi-Processor System-on-Chip, Design and Test Technology for Dependable Systems-on-Chip, Raimund Ubar, Jaan Raik, Heinrich Theodor Vierhaus (Eds.), 2010, Hardcover, ISBN: 978-1-6096-0212-3.
- Farrokh Ghani Zadegan, Urban Ingelsson, Gunnar Carlsson and Erik Larsson, Reusing and Retargeting On-Chip Instrument Access Procedures in IEEE P1687, IEEE Transactions on Computers, EDA Industry Standards issue of IEEE Design & Test Magazine, ISSN: 0740-7475, Digital Object Identifier: 10.1109/MDT.2012.2182984, Mar/Apr 2012
- Kim Petersen, Dimitar Nikolov, Urban Ingelsson, Gunnar Carlsson, Erik Larsson, An MPSoCs demonstrator for fault injection and fault handling in an IEEE P1687 environment, IEEE European Test Symposium (ETS), Annecy, France, May 2012.
- Farrokh Ghani Zadegan, Urban Ingelsson, Gunnar Carlsson and Erik Larsson, Access Time Analysis for IEEE P1687, IEEE Transactions on Computers, ISSN: 0018-9340 Digital Object Identifier: 10.1109/TC.2011.155
- Farrokh Ghani Zadegan, Urban Ingelsson, Golnaz Asani, Gunnar Carlsson and Erik Larsson, Test Scheduling in an IEEE P1687 Environment with Resource and Power Constraints, Asian Test Symposium (ATS 2011) , Delhi, India, November 2011.
- Dimitar Nikolov, Urban Ingelsson, Virendra Singh and Erik Larsson, Level of Confidence Evaluation and Its Usage for Roll-back Recovery with Checkpointing Optimization, 5th Workshop on Dependable and Secure Nanocomputing (WSDN'11) Hong Kong, June, 2011.
- Farrokh Ghani Zadegan, Urban Ingelsson, Gunnar Carlsson and Erik Larsson, Design Automation for IEEE P1687, Design Automation and Test in Europe (DATE), Grenoble, France, March 2011.
- Farrokh Ghani Zadegan, Urban Ingelsson, Gunnar Carlsson and Erik Larsson Test Time Analysis for IEEE P1687, IEEE 19th Asian Test Symposium(ATS2010), Shanghai, China, December 2010
- Mudassar Majeed, Daniel Ahlström, Urban Ingelsson, Gunnar Carlsson and Erik Larsson, Efficient embedding of deterministic test data, (Power Point presentation), IEEE 19th Asian Test Symposium(ATS2010) Shanghai, China, December 2010
- Mikael Väyrynen, Virendra Singh, and Erik Larsson, Fault-Tolerant Average Execution Time Optimization for General-Purpose Multi-Processor System-on-Chips, Design Automation and Test in Europe (DATE 2009), pages 484-489, Nice, France, April, 2009

Conclusions

- Handling of errors – soft and hard – is a necessity for high quality computer systems
- Detect errors locally and handle eventual errors globally
- Access becomes crucial:
  - IJTAG and complemented with error propagation becomes efficient
- Shown a demonstrator with a 10 processor system, an instrument manager for handling of embedded instrument, a resource manager to plan actions

Fault Management in an IEEE P1687 (IJTAG) environment

Erik Larsson and Konstantin Shibin
Lund University Testonica Lab