UNIVERSITÀ DEGLI STUDI DI PADOVA

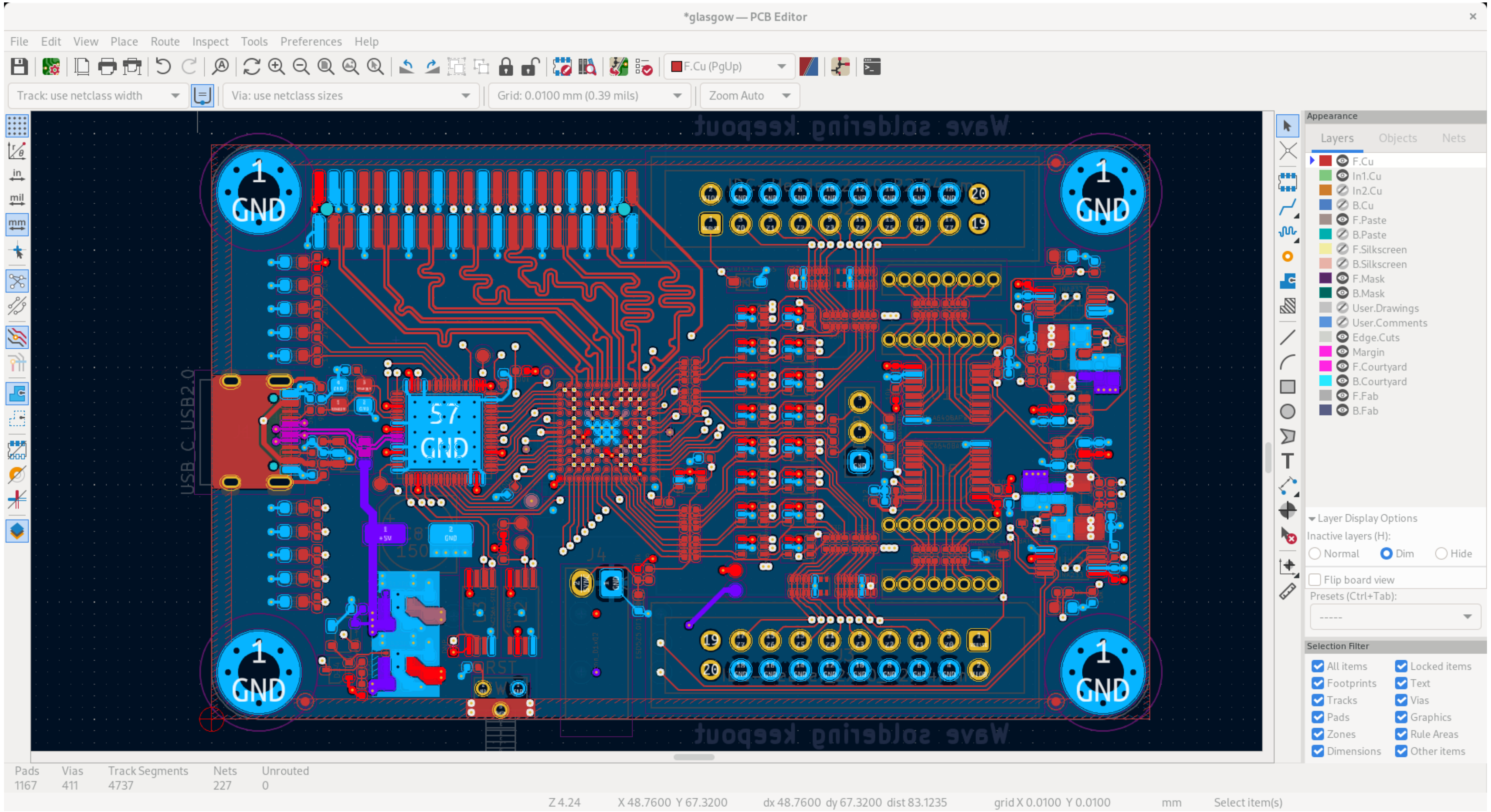
Digital circuit test – An introduction

Lecture #16

Electronic measurements

Claudio Narduzzi, Alessandro Pozzebon

Computer Aided Design (CAD)



Modelling digital devices

- **Models** → Different levels of abstraction
  - Behavioural models
  - Structural models

- **System level models** → Device architecture
- Different subsystems with different tasks

- **Behavioural model** → Interacting processes
- **Structural model** → Integrated components (processors, memory units, the organization of data and control busses, etc…)

Modelling digital devices

- **Algorithms** → Description of the way the system processes data and produces an output

- **Data** → Variables → Registers
- **Algorithm steps** → transfers of processed data among registers

⬇

Register-transfer (RT) model

Modelling digital devices

- **Logic level models** → Basic representation by means of binary variables

- **Behavioural model** → Boolean equations
- **Structural model** → Logic gates and memory elements

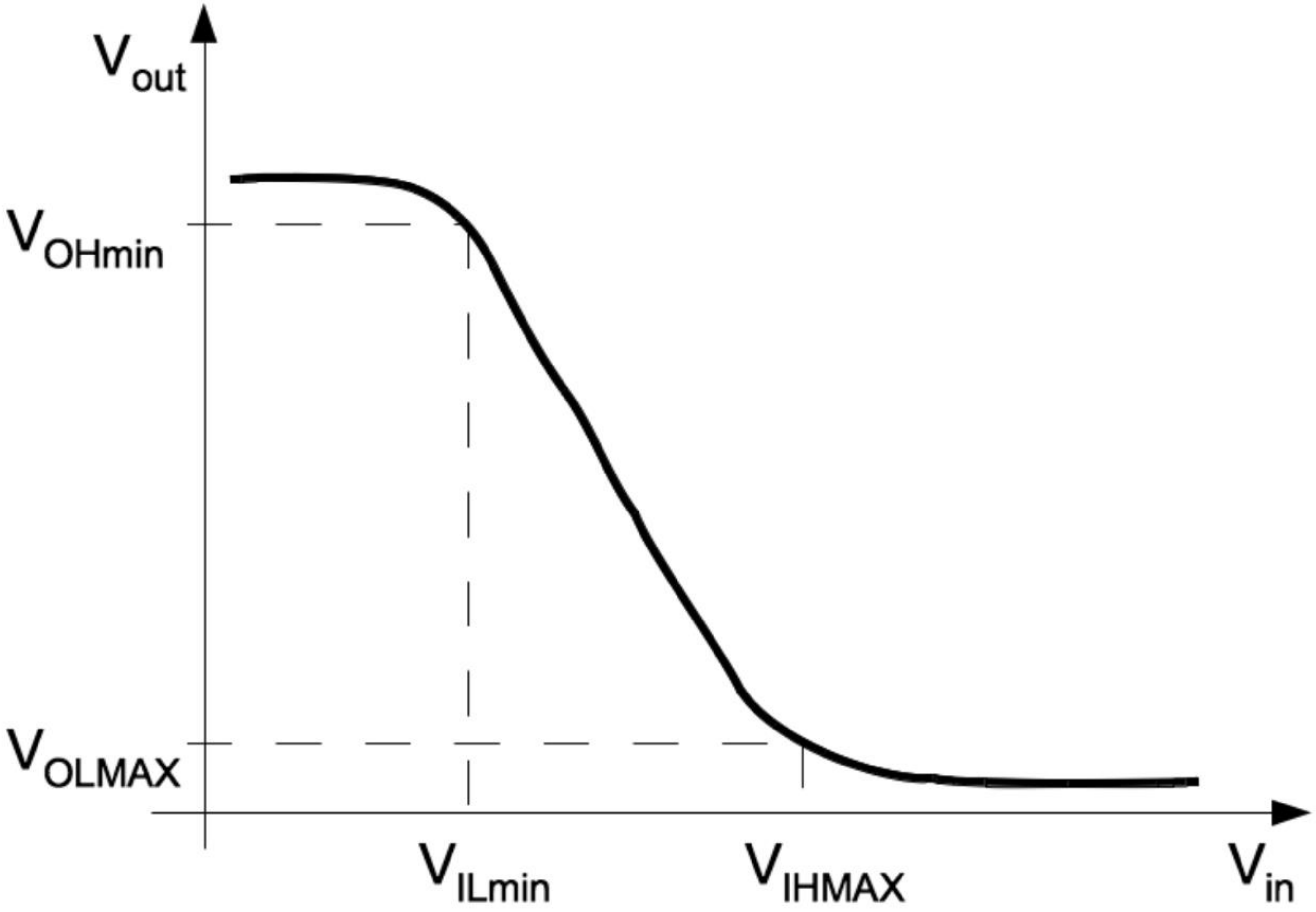
⬇

Primary approach in the characterization of digital electronic devices

- Signal representation:
  - **Analog**: function $x(t)$
  - **Digital**: [0, 1] values

Modelling digital devices

Transfer characteristic of a logic inverter



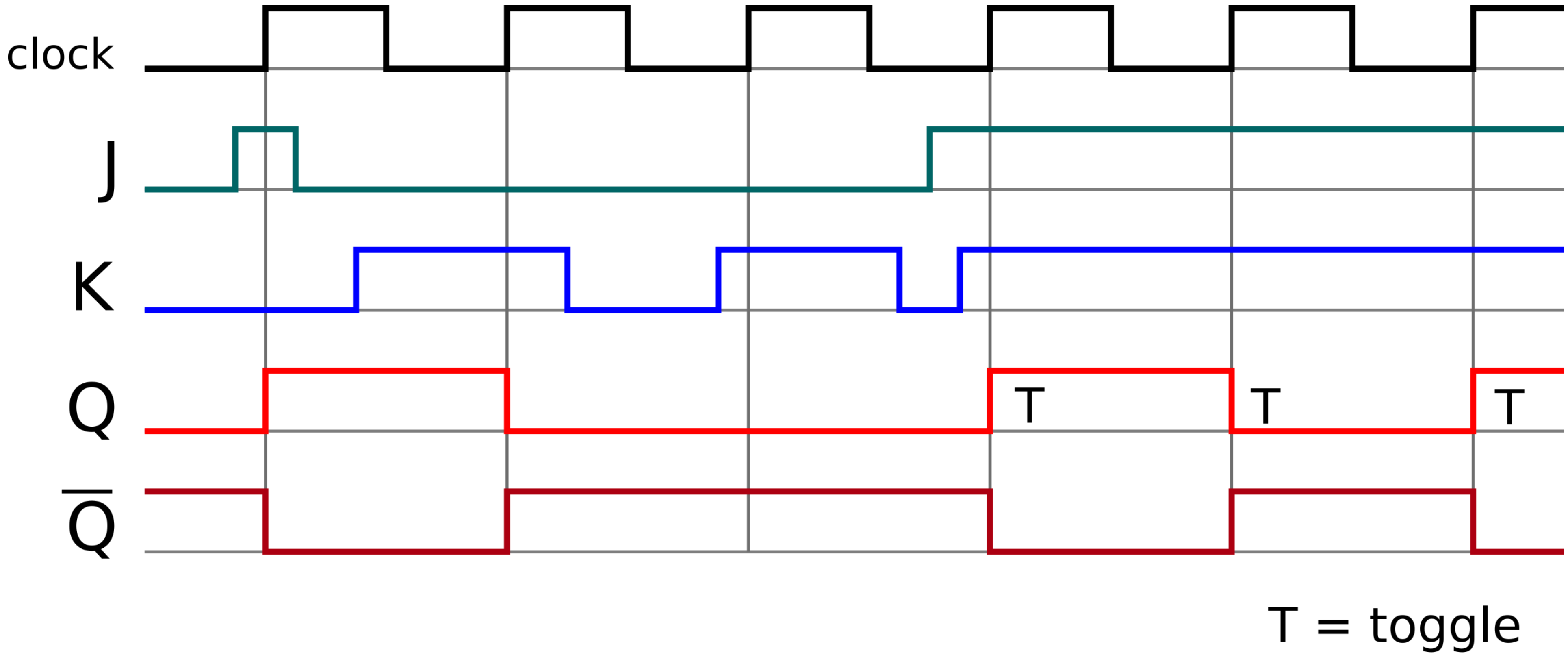
Modelling digital devices

- **Internal circuits timing** → Clock → Square wave
- **Rising or falling edges** determine the instants when digital memory elements are allowed to update their values → Active edges

- **Combinational logic networks** → Truth tables
- **Synchronous sequential state machines** → State diagrams

- Verification of **correct logic behaviour** → measurement instrument uses the circuit clock
- Analysis of **response times** → Measurement instrument uses an internal high-speed time reference → Timing analysis (**Timing diagrams**)

Timing diagram



Fault models

- **Physical-level defects → Logic faults**
  - Digital components like EEPROM, PLD or FPGA, are **programmed only when employed in a specific device**. Component-level verification cannot ensure a proper system functionality check
  - An electronic product goes through **several assembly stages** (component positioning on PCB, soldering, etc...) that may introduce defects
  - Electronic components may be subjected to **thermal, mechanical, chemical and electrical stress**

⬇

Digital testing → identification of the logic faults induced by defects

Fault models

- **Fault model**: translates the possible effects of a physical-level defect into the description of a **faulty logic-level behaviour**

- **Logic fault** → Device logic function is altered
  - Combinational logic network → Modification in the **input-output logic behaviour**
  - Sequential machine → Modification in the **state diagram**

- **Timing faults** → critical variations in the device **timing diagrams**

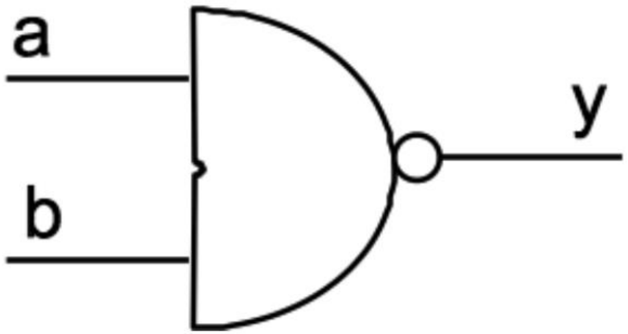
Stuck-at faults

- In some circuit node, switching between the two logic levels **becomes impossible for some reason**
- The logic level of one specific line is thus forced to just one value → **Stuck-at**

- Functional verification: check that **all inputs and outputs can freely change** their logic state in response to suitable test stimuli

Stuck-at faults

- **Example: NAND gate**

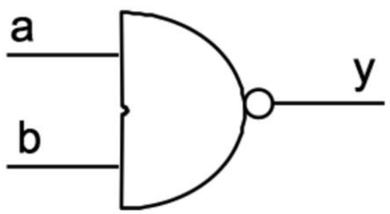


a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

Stuck-at faults

- **Gate output behaviour**
  - $a = 1, b = 1$
  - One of the other three combinations

- **Gate input behaviour**
  - $a = 1, b = 0$
  - $a = 0, b = 1$

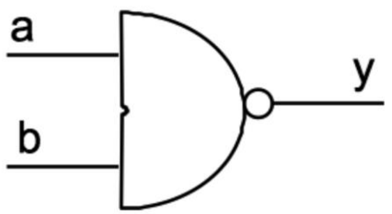


a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

Test vectors

Test vectors

- **Test Vector:** any combination of input logic values employed in the test of a logic network, associated with the indication of the expected correct response



- **NAND gate**
  - 6 stuck-at single faults (stuck-at 0 and stuck-at 1)
  - Minimum number of test vectors required for a complete test = 3

a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

test vector no.	a	b	y	$\bar{y}$	detectable faults		
1	0	1	1	0	a stuck-at-1	y stuck-at-0	
2	1	0	1	0	b stuck-at-1	y stuck-at-0	
3	1	1	0	1	a stuck-at-0	b stuck-at-0	y stuck-at-1

Test vectors

test vector no.	a	b	y	$\bar{y}$	detectable faults		
1	0	1	1	0	a stuck-at-1	y stuck-at-0	
2	1	0	1	0	b stuck-at-1	y stuck-at-0	
3	1	1	0	1	a stuck-at-0	b stuck-at-0	y stuck-at-1

- $a = 1, b = 1, y = 1 \rightarrow$ a fault condition is detected (**Fault detection**), but it is not possible to establish where the fault occurred (**Fault location**)

- Minimum number of test vectors N_{tv}:

$$n + 1 \leq N_{tv} \leq 2^n$$

with n inputs

UNIVERSITÀ DEGLI STUDI DI PADOVA

Test vectors

$$n + 1 \leq N_{tv} \leq 2^n$$

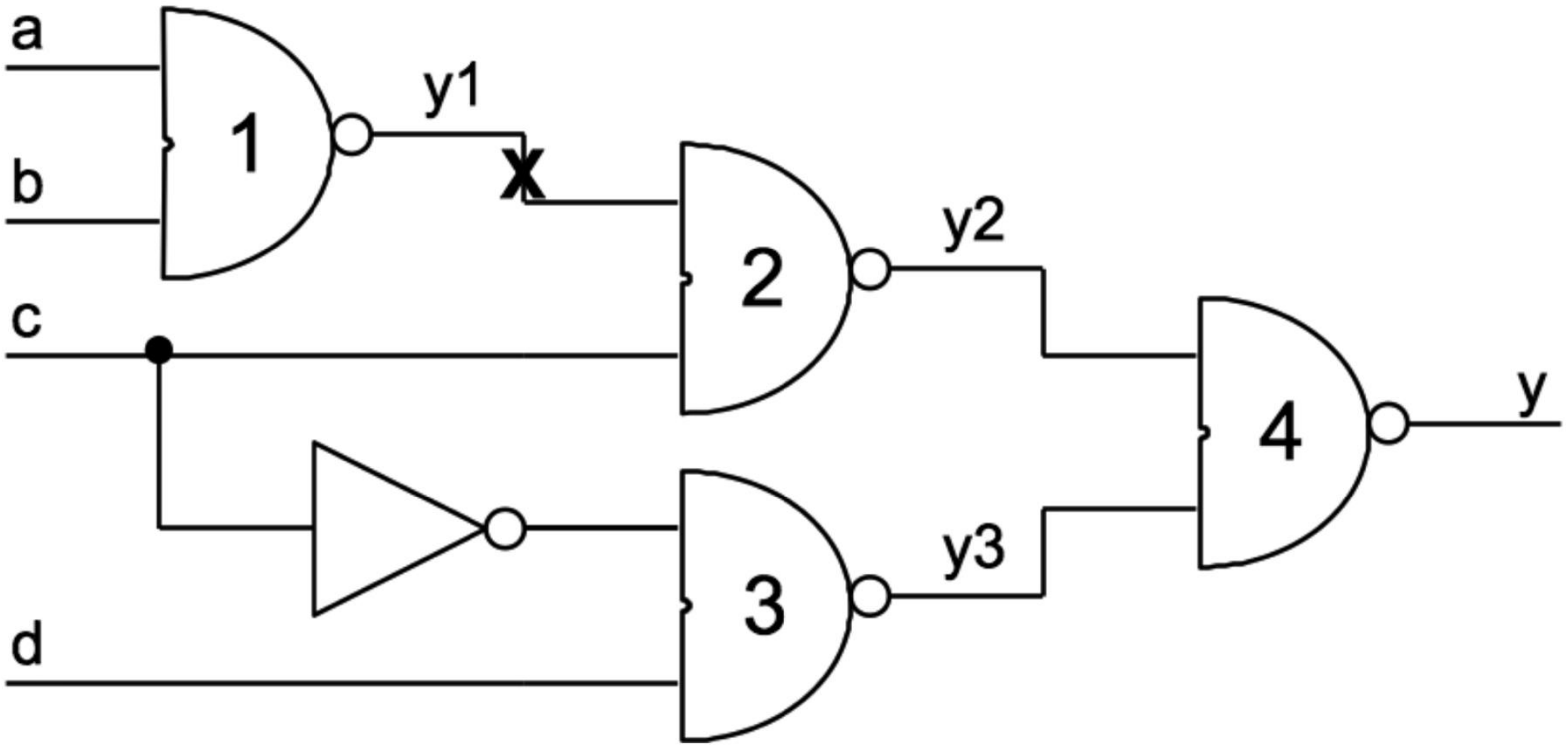
- The number of test vectors **increases with the number of inputs**
- The actual complexity **depends on the specific Boolean function**, as well as on how that function is implemented by a logic gate network
- **Different combinations of logic gates** can be used to implement a given Boolean function

$\Downarrow$

Whenever possible, a designer should favour easily testable circuit structures

Test vectors

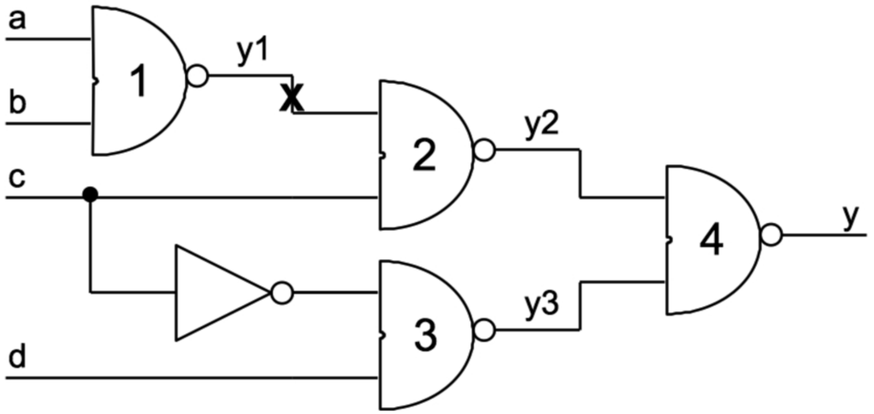
- Verification of lines that are **not directly accessible** from the network input/output terminals



a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Test vectors

- **Input c → Enable input** that allows to select as the network output y, the logic NAND of the two inputs a and b, or the value of logic variable d
- Test vectors for the NAND gate allow the verification of gate no. 1 (a and b), but **the output is not accessible** unless input $c = 1$
- Setting $c = 1$ allows the gate no. 1 output value to propagate through NAND gate no. 2
- Care needs to be taken to define some of the test inputs so that **the response of logic gates at intermediate levels can be observed**



Test vectors

test vector no.	a	b	c	d	y	$\bar{y}$	detectable faults
1	X	$\bar{\text{X}}$	0	0	0	1	d s-a-1 $y1$ s-a-0 $y2$ s-a-0 $y3$ s-a-0 y s-a-1
2	X	$\bar{\text{X}}$	0	1	1	0	c s-a-1 d s-a-0 $y3$ s-a-1 y s-a-0
3	0	1	1	1	1	0	a s-a-1 c s-a-0 $y1$ s-a-0 $y2$ s-a-1 y s-a-0
4	1	0	1	1	1	0	b s-a-1 c s-a-0 $y1$ s-a-0 $y2$ s-a-1 y s-a-0
5	1	1	1	1	0	1	a s-a-0 b s-a-0 c s-a-0 $y1$ s-a-1 $y2$ s-a-0 $y3$ s-a-0 y s-a-1

Test vectors

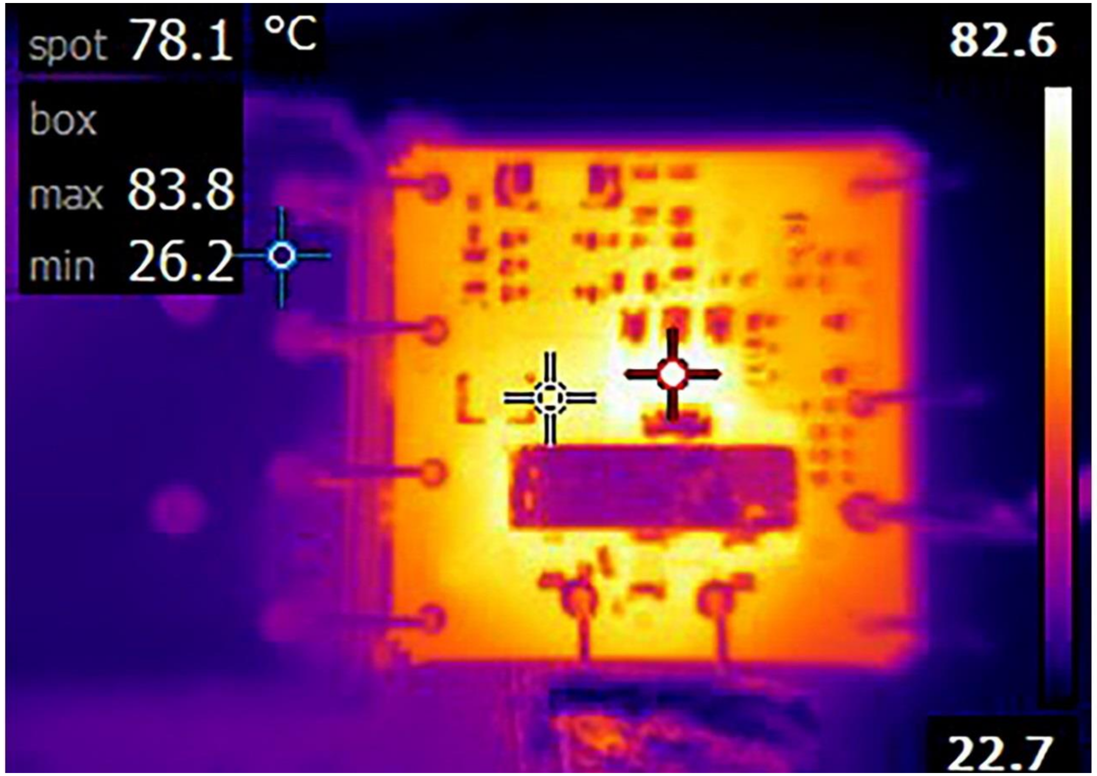
- Test vectors are defined under the assumption that a **single** fault may occur
  - The full path bringing the input stimulus to the logic gate concerned is enabled (**Path sensitization**)
  - The full path that propagates the test response to an observable output is enabled (**Fault propagation**)
- A single vector can detect **more than one single fault condition**

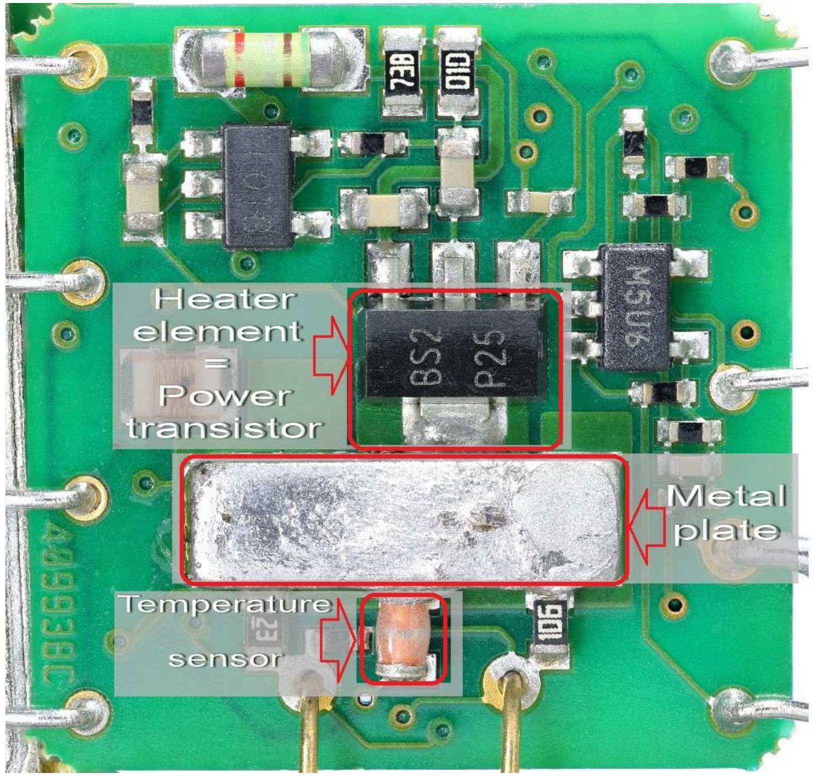
- Generation of test vectors is usually the task of **automatic test pattern generation (ATPG)** tools
- Often the test vector set produced by an ATPG is not designed to cover all possible faults, but to to provide a **sufficiently high level of fault coverage** (for instance, 90%)

Università degli Studi di Padova

Test vectors

- Better test efficiency is obtained by a **combination of test techniques**
- Observation of the **thermal image** of an assembled printed circuit board





Sequential state machines

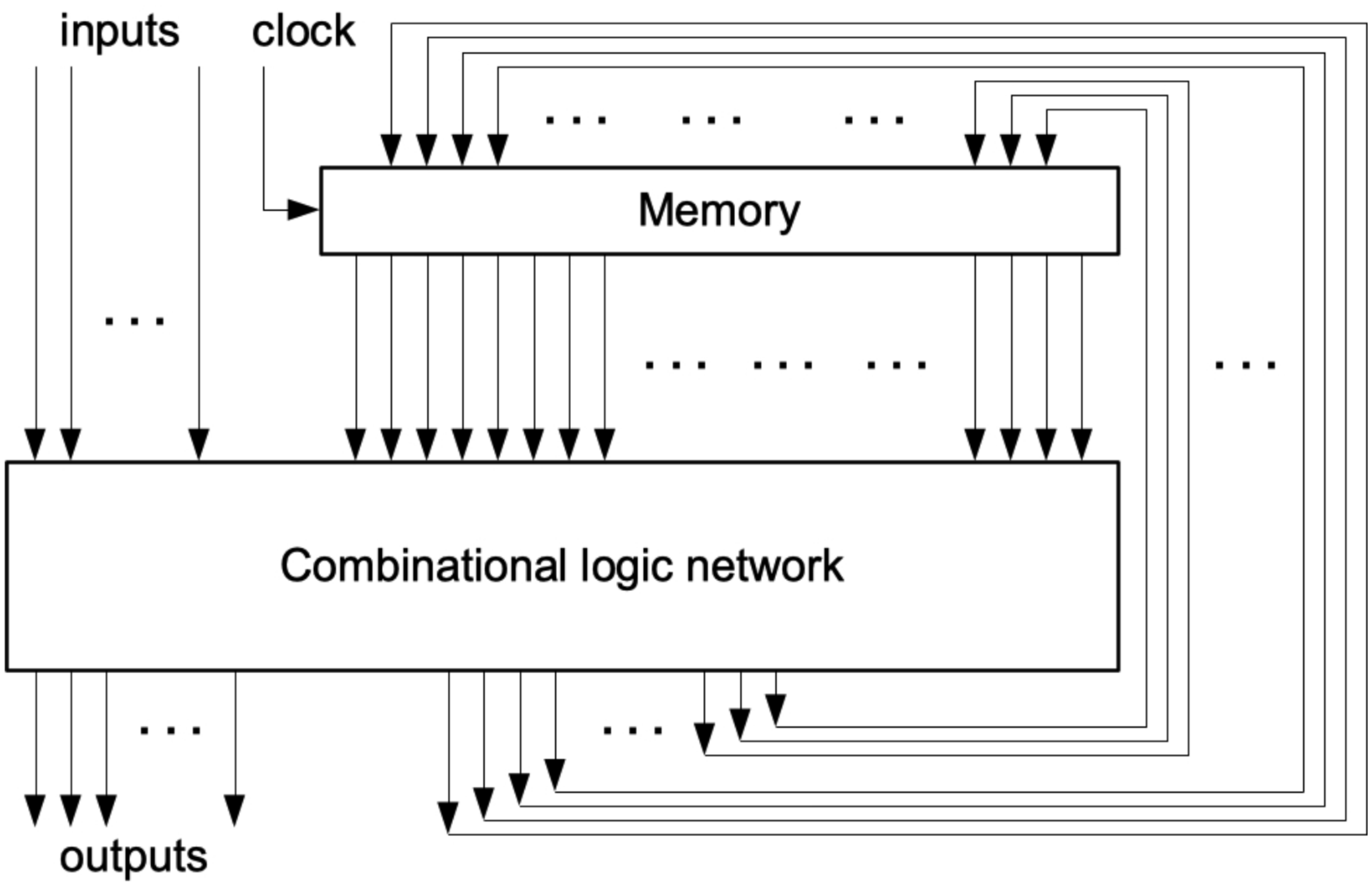
- **Sequential State Machines**:
  - A number of **states**
  - A set of state **transition rules**

- **Clock** circuit for timing
- **Machine states** → Contents of memory elements
- **State transitions** → Triggered by clock active edges → Outputs of combinational logic functions

⬇

State transition functions define the values that will be stored in the memory elements at the next clock edge

Sequential state machines



Sequential state machines

- Verification of a sequential machine presents far **more complex problems** than a purely combinational network
- **Input and output lines** are considered the only test points

⬇

Tests should also take into account all possible states of the machine

Internal variables

- ***m*** memory cells
- 2^m different states

⬇

The correctness has to be verified for each state

Sequential state machines

- ***n*** inputs
- **2^n** input combinations
- **$2^m x 2^n$** test vectors

- **Difficulties**:
  - An extremely high number of combinations
  - The difficulty to preset the sequential machine to a known state before a test starts, since reaching a specific state usually implies the application of a suitable sequence of inputs

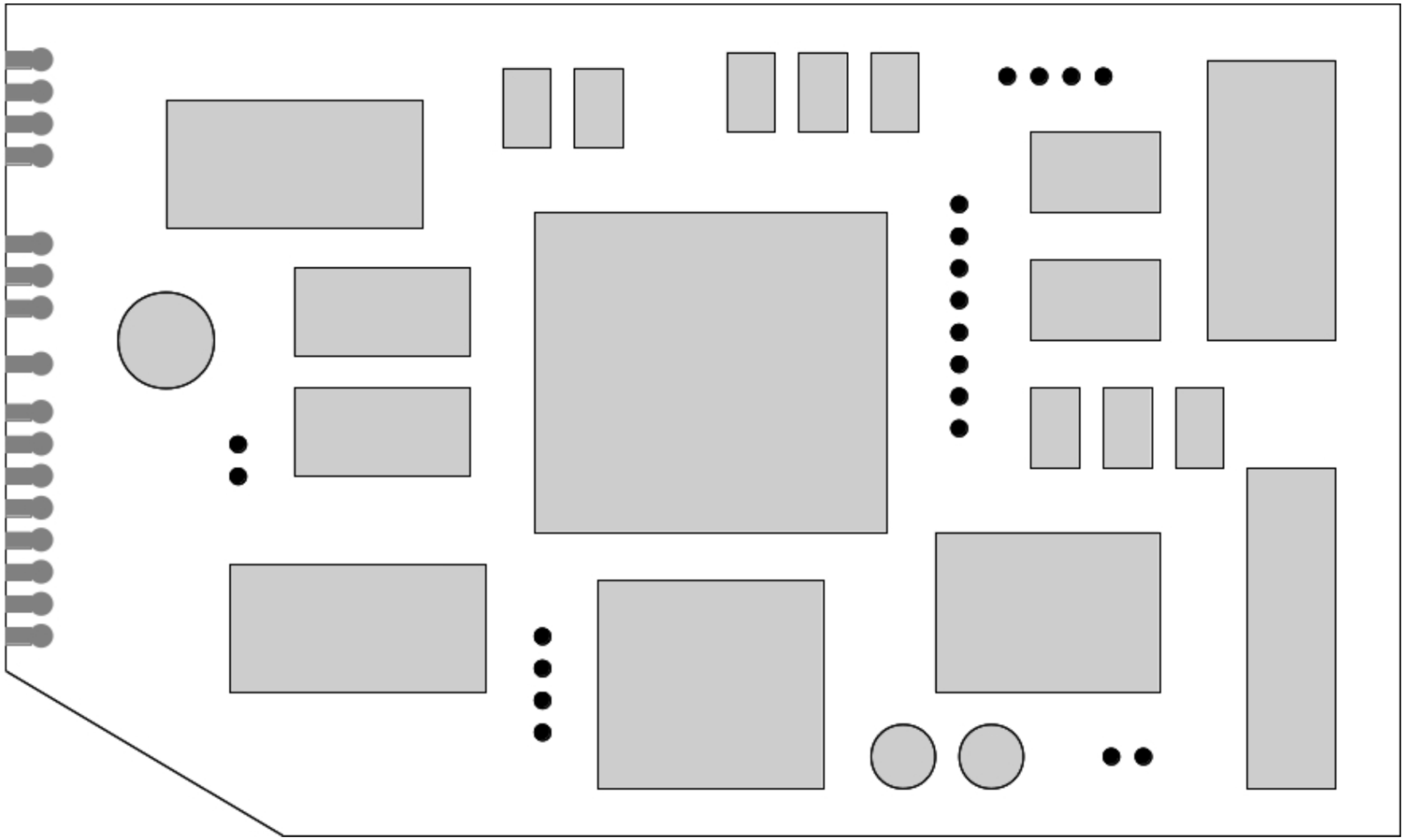
Printed Circuit Boards

- **Testing of PCBs**
  - Test stimuli and the corresponding system responses can only be observed at the board input/output connectors

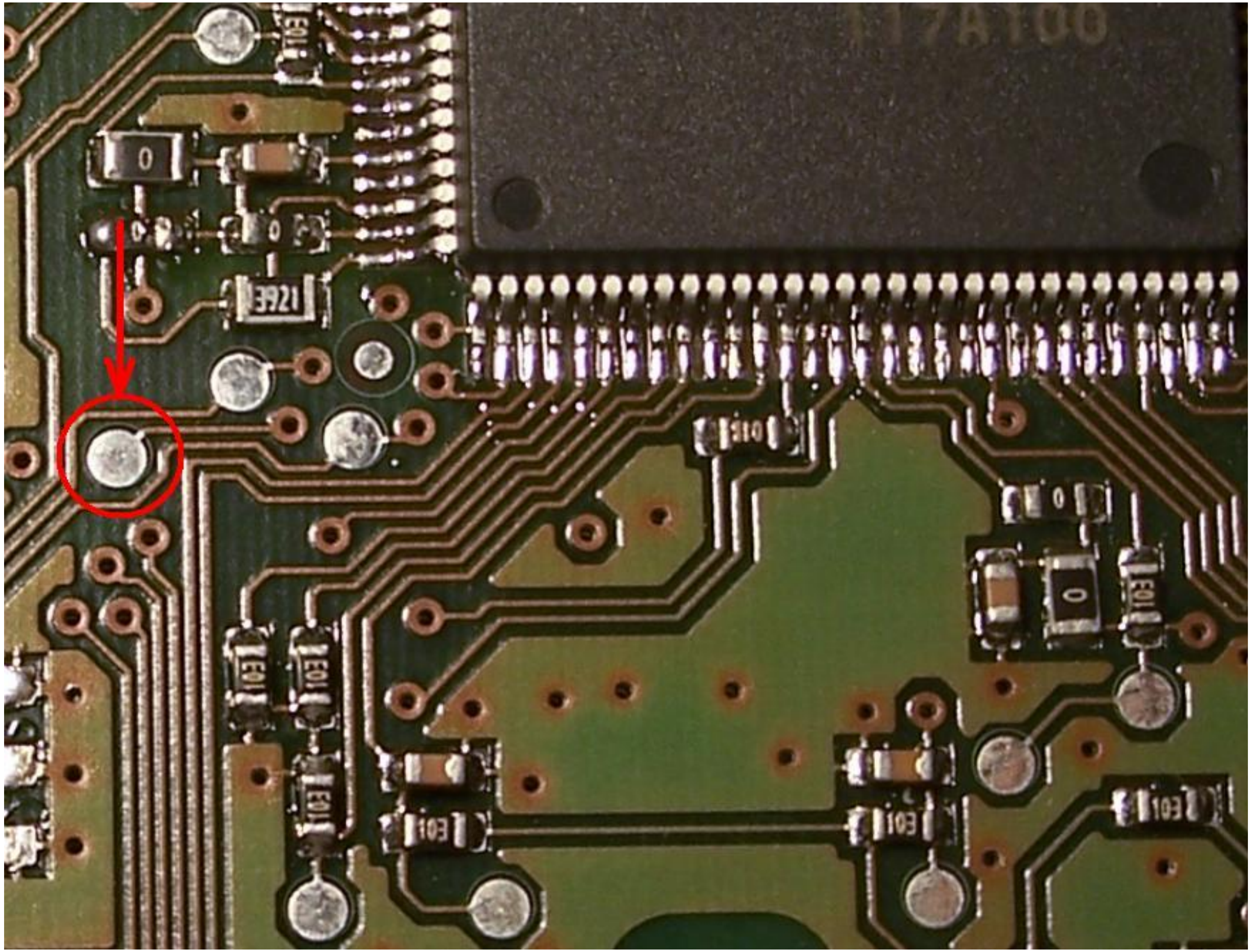
The **number of test points** remains extremely limited compared to system complexity

- Design of **additional test points** to provide access to selected signals, internal to the board
- **Partitioning of the system functions** into more easily testable subsystems

Sequential state machines



Sequential state machines



Printed Circuit Boards

- **Dedicated test probes**
- **In-circuit testing → Automatic test systems**

- **Test fixture:** interface between the test machine and the system under test
  - **Power supply** for the system under test
  - **Access to the test points** represented by system connectors
  - **Connections** to the internal test points → **Bed of nails**

- **Nails:** telescopic, spring-loaded pins whose position in the test fixture matches exactly that of test points in the system under test

- Each test fixture is **specific to a certain system under test**

Boundary scan

- **Boundary scan or Scan Path:** standardized testing technique
- **JTAG**: Joint Test Action Group → IEEE 1149.1 Standard

- Testing of complex and VLSI (Very large scale integration) Ics
- Progressive decrease in the accessibility

- **Microcontroller**: several thousand logic gates and just a few tens input/output lines

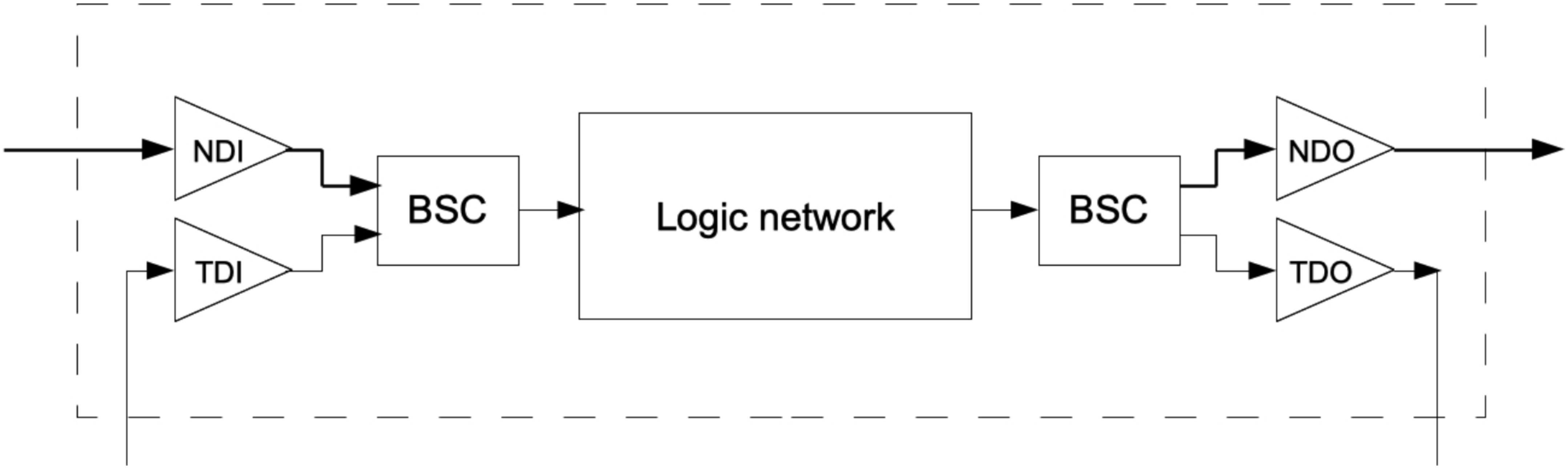
Boundary scan

- **Dedicated additional circuits** to facilitate the observation of internal states in a device and its initialization to a desired state → Activated exclusively during testing

- JTAG standard supports the realization of those test circuits and defines the system architecture for their use

- **Boundary Scan Cell (BSC):** memory cell → elementary JTAG circuit
- Associated to each input and output line in a logic network

Boundary scan



Boundary scan

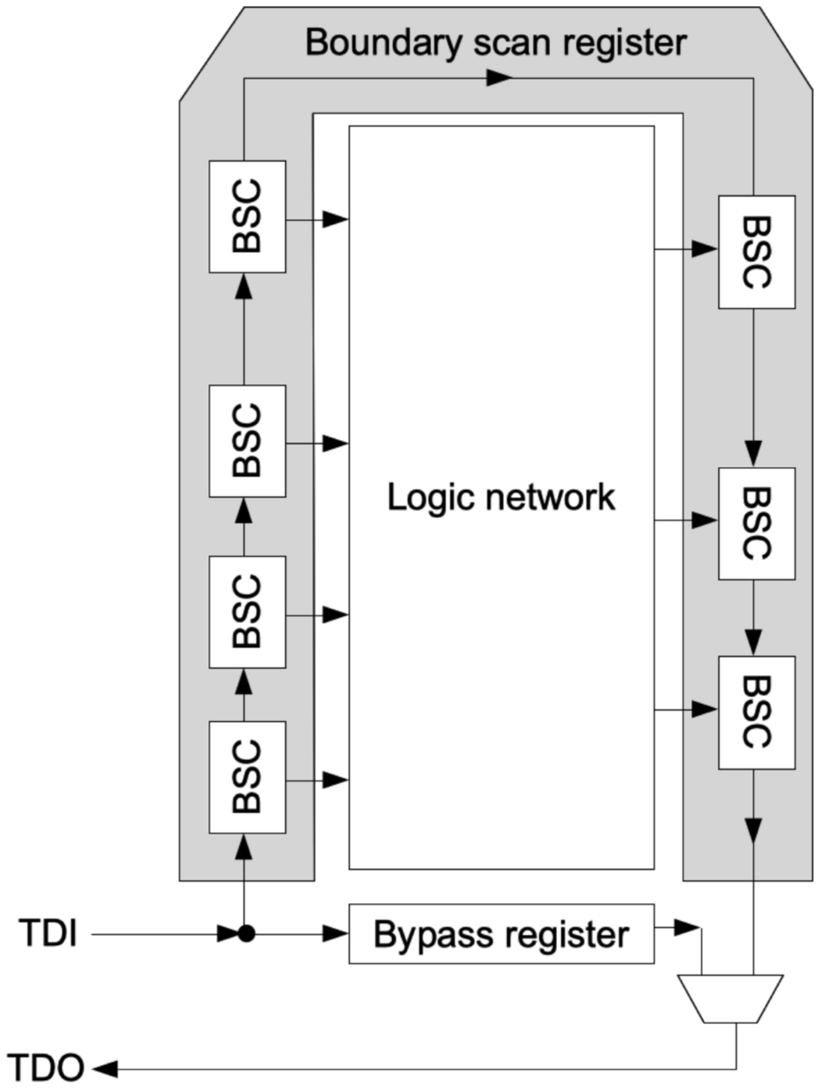
- Device in **normal operation** → BSC is "transparent" to the exchange of data and does not affect the IC in any way
  - **Normal Data Input (NDI)**
  - **Normal Data Output (NDO)**

- **Test mode** activation → BSC disconnects the internal logic network from NDI and NDO lines, replacing them with test lines
  - **Test Data Input (TDI)**
  - **Test Data Output (TDO)**

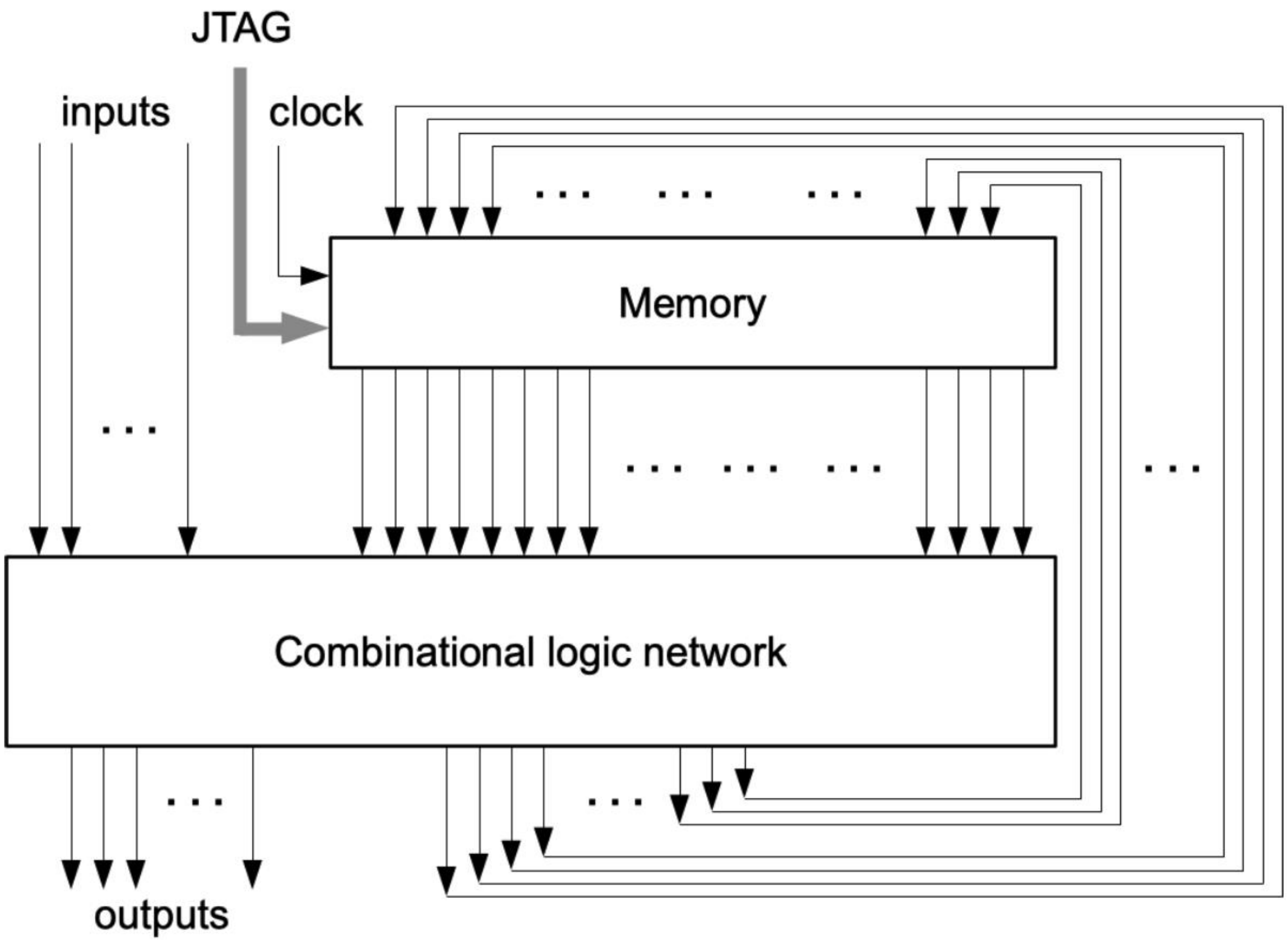
- A test input can thus be sent to the logic network through TDI, and the response is observable on line TDO

Boundary scan register

- BSC cells within a device are interconnected to form a register, called boundary scan register (BSR)
- During the actual test, all the component input and output lines are operated in parallel
- Four test lines are defined by the JTAG standard and form the **test access port (TAP):**
  - **TDI**
  - **TDO**
  - **TCK: test clock**
  - **TMS: test mode select**

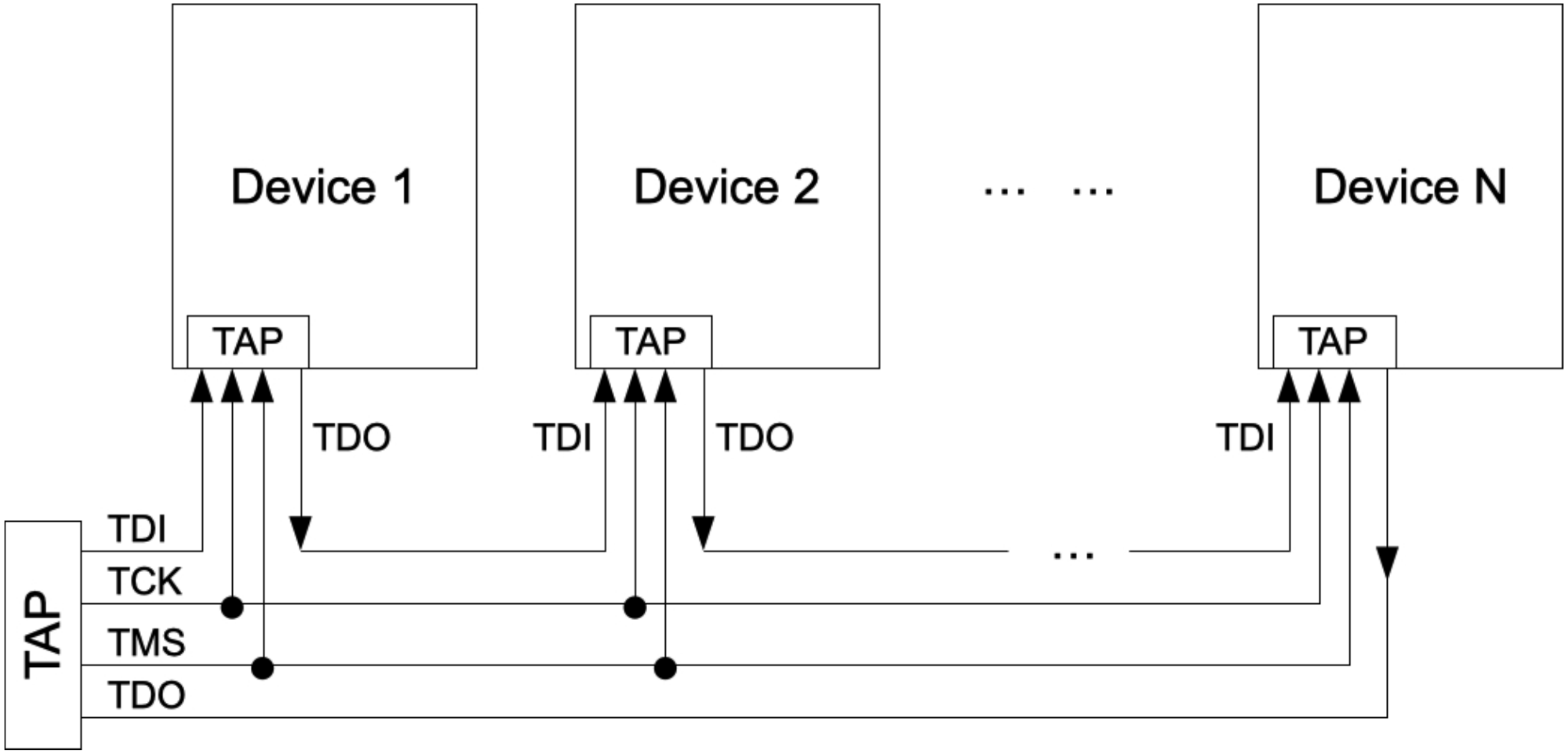


Boundary scan register



Boundary scan register

- In a system, all JTAG-compatible components are serially linked to form a daisy chain



Università
degli Studi
di Padova