ELEC-H410
REAL-TIME SYSTEMS

Chapter 9

Logic analysers

Logic Analysers

CONTENTS

Introduction

why logical analysers ?

- needs
  - debug of the hardware
    - delays, *setup&hold time*
    - race conditions
    - access time of the memories
  - debug of the software
    - measure execution times
    - observation of the real execution cycles
  - tests of integrated circuits (ASIC, FPGA)
  - maintenance
- solution
  - make the signals visible (ex: oscilloscope)

  ! ASIC/µP => a lot of signals to observe (30 - 100+)

Introduction

the oscilloscope

- we want to see the actual shape of analog signals
  - bulky compensated probes
  - signals have highly variable amplitudes =>attenuators followed by amplifiers
  - 1 amplifier per channel
    - short rise time => large bandwith
    - tr [ns] = 350 / BW[MHz]
    - bulky and expensive
  - non repetitive phenomena => digital oscilloscopes
  - not enough channels (2 to 4) to analyse digital systems

Introduction

logic analyzer

- we want to know the binary state of a signal
  - simple logical probes: fast comparators
    - adjustable threshold v_{TH}
      - below V_{TH} => 0
      - above V_{TH} =>1
    - restricted V_{TH} range (1 to 10V)
  - no amplifiers
  - a lot of channels (several 100)

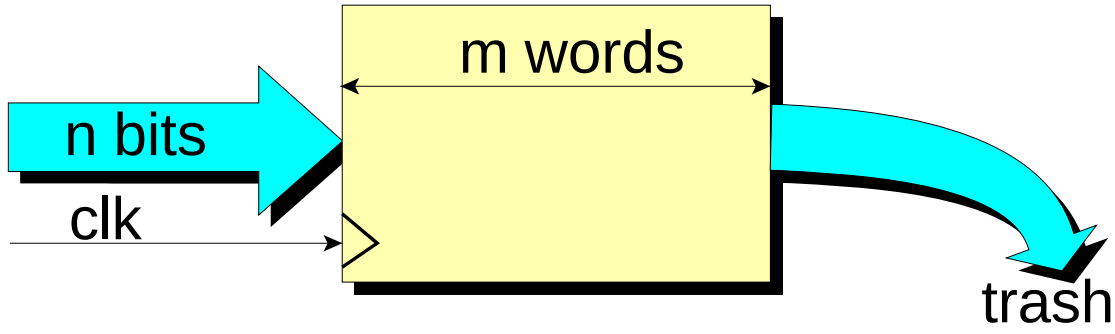
Logic analysers

CONTENTS

Memorisation (classical analyser)

principle : FIFO

- shift register (**n** bits width x **m** words depth)
- sampling rate fixed by the clock
- lifetime of an information: m clock periods



To record data in the logic analyser, N channels are sampled simultaneously (forming a N-bit word). The sampling clock pushes data in the memory structured as a FIFO (First-In-First-Out) shift register.
The register has got a depth of m words and constitutes a **time window of finite size**: after m clock cycles, the information has crossed the whole register and is lost.

Memorisation

figure of merit

- figure of merit = n . m . f
  - n = 1 ... 100+ channels
  - m = 1K ... ??? words (size=price)
  - f = 10MHz ... 5GHz
- obtained by grouping elementary modules
  - in series
  - in parallel
  - interlaced

One can define a **figure of merit** for the analyser as the product

n x m x f (width x depth x sampling frequency)

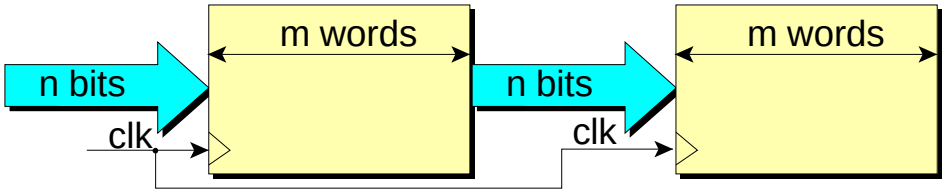
Indeed, it is difficult to increase these 3 values simultaneously.

We will see in the next slide that increasing the performance is obtained by grouping modules within the analyser.

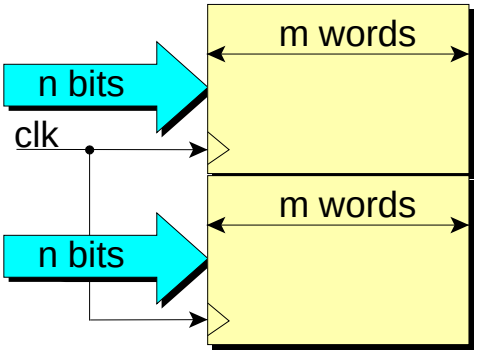
With current maximum frequencies (n GHz) logical analysers are among the fastest measuring instruments ever made.

Memorisation

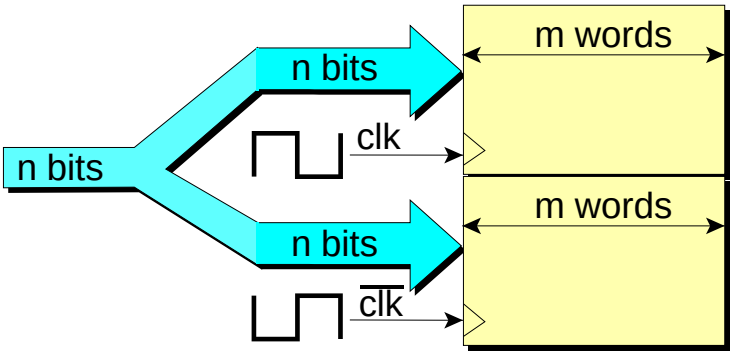
increase the performance by the modularity



series : n.2m.f



parallel : 2n.m.f

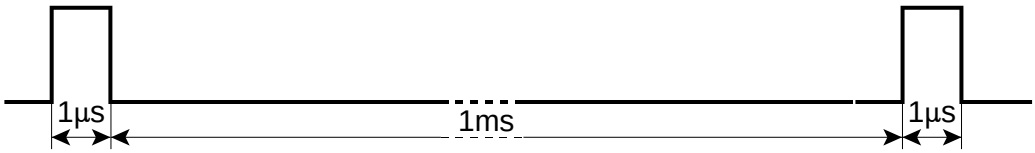


interlaced : n.m.2f

Memorisation

saving memory: transitional analyzer

- problem of very distant short pulses
- Ts < pulse width so as not to miss it
- traditional analyser (shift register) :
  - all the intermediate states are memorized
  - 1100000000000.....0000000001100000
- transitional analyser :
  - the transitions and the state durations are memorized
  - 1(2)0(2000)1(2)
  - large saving of memory



The principle of the shift register is simple, but the ratio between the size of the memory and the number of actual events stored can be very low.

The simplest example is a sequence of short pulses separated by an interval much larger than their width.

The pulse width determines the sampling rate; indeed, if τ is the shortest pulse width to detect, the sampling period has to be less than τ.

In the case of this figure, to capture 1µs pulses; we have to sample at least at 1MHz; let us take 2 MHz. Hence, the shift register will contain 2000 successive "0", to store the interval of 1ms between the two pulses.

In the **"transitional" analysers**, the logical states are stored with their durations (expressed in clock cycles), which leads to a substantial saving in memory.

Logic analysers

CONTENTS

- Introduction
- **Classical logic analyser**
  - principles of memorisation
  - **trigger**
  - clock
  - harnessing and connectors
  - display
  - additional functions
- Specialized analysers
- Conclusions

Trigger

to display, the memory should be frozen

- continous acquisition
- trigger event => memory is frozen
- then data is processed and displayed
- specify the trigger condition is the most important and difficult part of the debugging with a LA

In digital oscilloscopes and logic analysers, once the acquisition has been started, it keeps running continuously. Considering the high rate of acquisition, it is clearly impossible to visualize data in real time.

It is thus necessary to stop acquisition by an event called the *trigger* before looking at the memory.

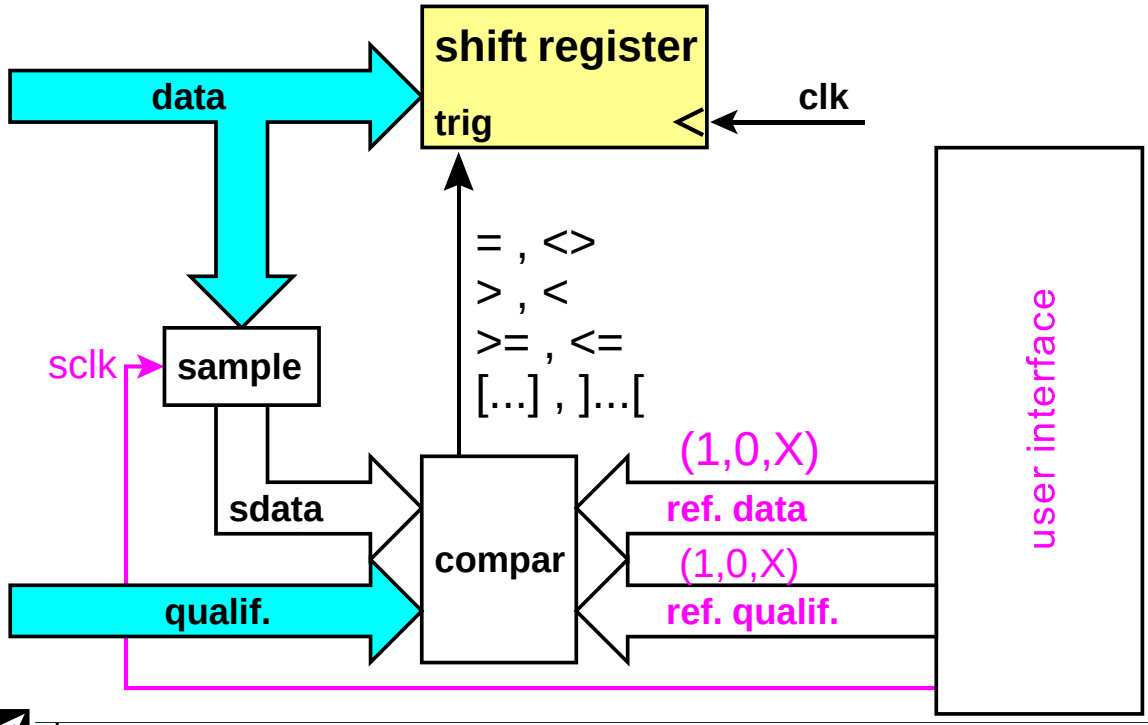
The condition of the trigger will be specified by the operator via the man-machine interface (keyboard, script....); we will see in the following slides that this condition can be very complex.

Logical analysis relies on two principles:

- connect the good signals to the inputs (i.e. those which enable to highlight the problem)
- write a **suitable trigger condition** which "traps" in the memory the phenomenon to be observed; this is not a trivial task

Trigger

basic block diagram



This slide presents the block diagram of the trigger system whose heart is a **logical comparator,** which compares the input data with **reference data** specified by the user.

Noticed that the signals are first sampled (block "sample"); this block can contain a counter (not represented) to filter short parasitic pulses whose width is less than N periods of the sampling clock ***sclk***, generally faster than the clock of the shift register.

The operator specifies the condition of trigger in the form " **1/0/X**", where X marks a bit whose state is "don't care".

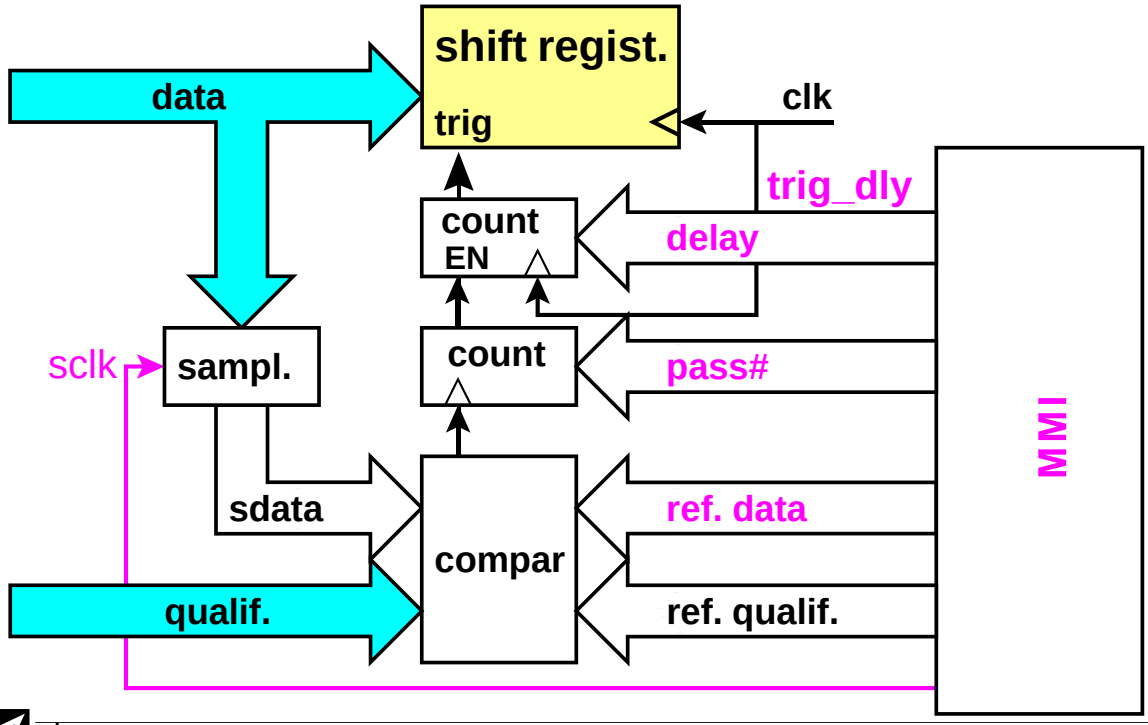
The comparator is very complex because we can detect:
- equal
- different
- superior, superior or equal, lower, lower or equal
- belonging to an interval
- out of an interval

The condition of comparison can take into account additional signals, called **qualifiers**, taken in the system but not stored in the shift register. This enables to specify a richer trigger condition without consuming any additional memory.

Let us remark that, since the trigger event freezes the memory all the visible content of the signal happened before the **trigger** which is the **last event in memory**.
This is called ***pre-triggering*** and is the default mode of acquisition.

Trigger

block diagram: add pass and delay counters



Let us complete the trigger block diagram to introduce a **delay** between the trigger event (output of the comparator) and the moment when the memory is frozen.

This delay is carried out by two counters that are presented here in cascade:

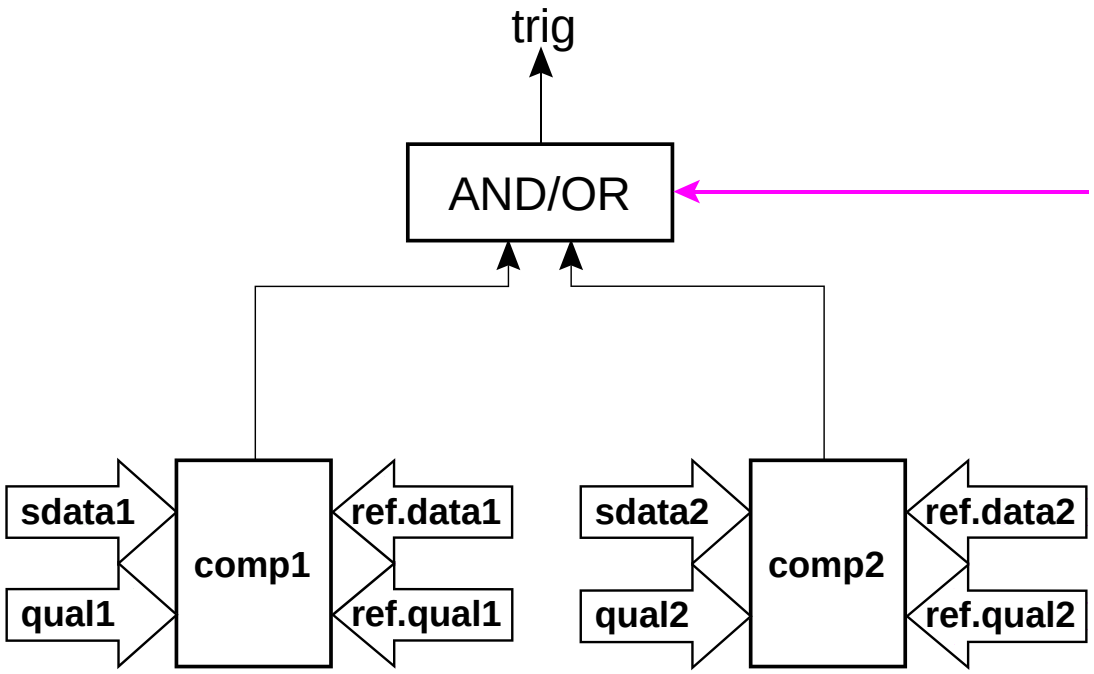
- the **pass counter** enables to freeze acquisition after N occurrences of the trigger condition; this can be useful to
  - examine of a logic circuit when a counter overflows
  - debug programs with loops

- the delay counter enables to choose the position of the trigger event in the memory; the clock of this counter is the sampling clock of the shift register
  trig_dly = 0 : all the data stored are anterior to trigger (*pre-trigger*); it is the default behaviour in the absence of counter
  0 < trig_dly < m : *trig_dly* data will be posterior to the trigger
  trig_dly > m : all the data all will be posterior to the trigger (*post-trigger*)

This last configuration makes it possible to move the window of acquisition by pages of length m in the future and thus to examine a horizon longer than the memory, in several successive acquisitions, provided the behaviour of the system is reproducible.

Trigger

modular *trigger* with combinatorial logic

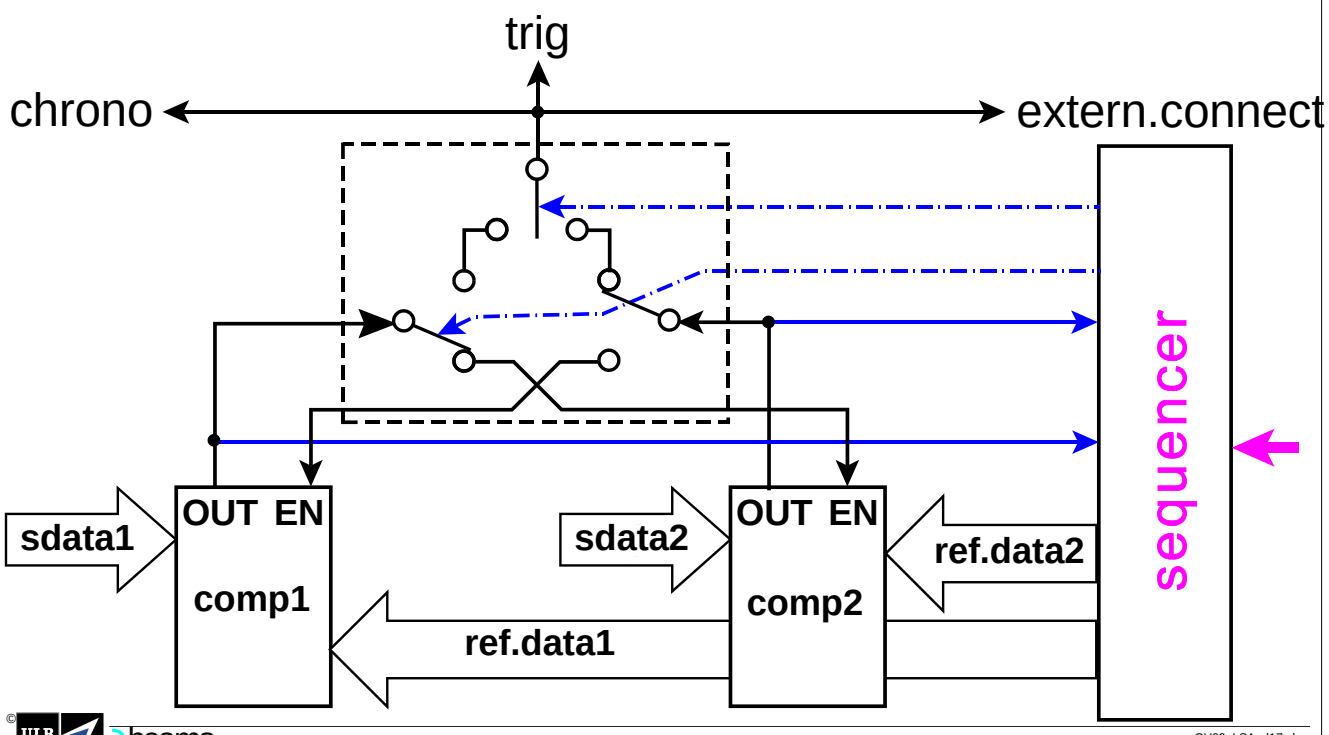


The trigger system can be modular: several comparators are used to compare same or different input signals to different reference data.

The outputs of these comparators can then ANDed/ORed by combinatorial programmable logic (CPLD)

Trigger

modular trigger with sequential logic



The following step is obtained by introducing a **programmable sequencer**. The comparators must have an ENable input. If this entry is not active, the comparator is not armed and its output is inactive.

We can thus build trigger conditions like: trigger if event_1 THEN event_2, THEN event_3... THEN event_k.

At least two comparators are required to build the sequence. Remark: the qualifiers have been omitted to simplify the figure).

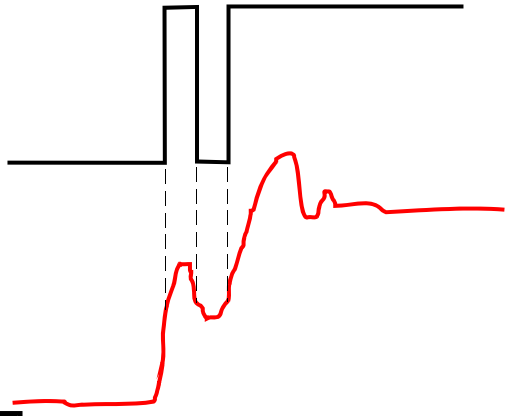
- the sequencer presents the event_1 on the comparator_1 and the event_2 on the comparator_2; the output of the comparator_1 is connected to the ENable of the comparator_2, which is thus inactive at this moment
- when the comparator_1 recognizes the event_1, the state machine goes to the next step: the comparator_1 is loaded with the event_3 and the output of comparator_2 is connected to the ENable of the comparator_1
- ...and so on
- at the k^{th} step, the output of the last active comparator is directed to the trigger and the state machine stops (and the acquisition too, of course)

REM the signal "trig" at the output of the last comparator of the sequence can be used for 3 purposes (separate or simultaneous)
- **freeze** the acquisition (most obvious and default action)
- start/stop internal stopwatches to measure execution times
- cause a transition on an external connector to trigger an oscilloscope, another analyser or an emulator.

Trigger

options : coupling with an oscilloscope

- logic analyser => information on the signal is limited to the logical levels
- if strange behaviour => trigger an oscilloscope on a logical event detected by the LA



The interest of coupling the logic analyser with an oscilloscope appears on this figure. The logical analyzer provides the time of transitions, but a very coarse representation of their actual shape, which is idealized as rectangular pulses.

When an abnormal logic event is captured (here a very short negative pulse), we can trigger an oscilloscope to see the exact shape of the signal, which shows for example an oscillation around the threshold.

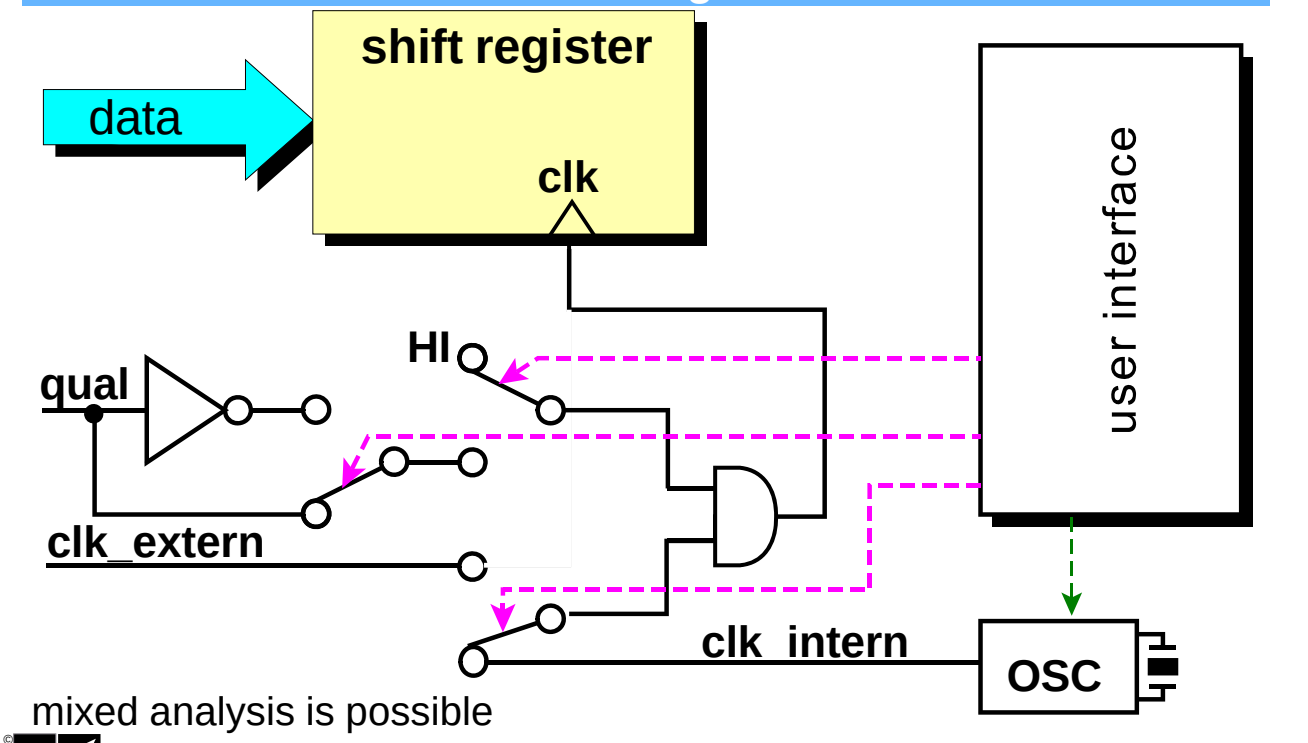
Logic analysers

CONTENTS

- Introduction
- **Classical logical analyser**
  - principles of memorisation
  - trigger
  - **clock**
  - harnessing and connectors
  - display
  - additional functions
- Specialized analysers
- Conclusions

Clocks

block diagram



How is the sampling clock chosen? Two cases are possible:

- a **synchronous** clock taken in the system under test; this only applies of course to systems which generate their own clock, i.e. synchronous logics and the microprocessors

- an **asynchronous** clock provided by a programmable oscillator within the analyser; the frequency is fixed by the operator, in a range from a few MHz to several GHz; this internal clock enables
  - the analysis of asynchronous logics
  - the analysis of the events between 2 clock ticks of a synchronous system
  - the resolution of the problems of timing (measurements of execution times, access time, setup&hold time)

The operator can thus choose:
- between a synchronous and an asynchronous clock
- the frequency of the asynchronous clock
- which clock edge is active (rising, falling, or both)

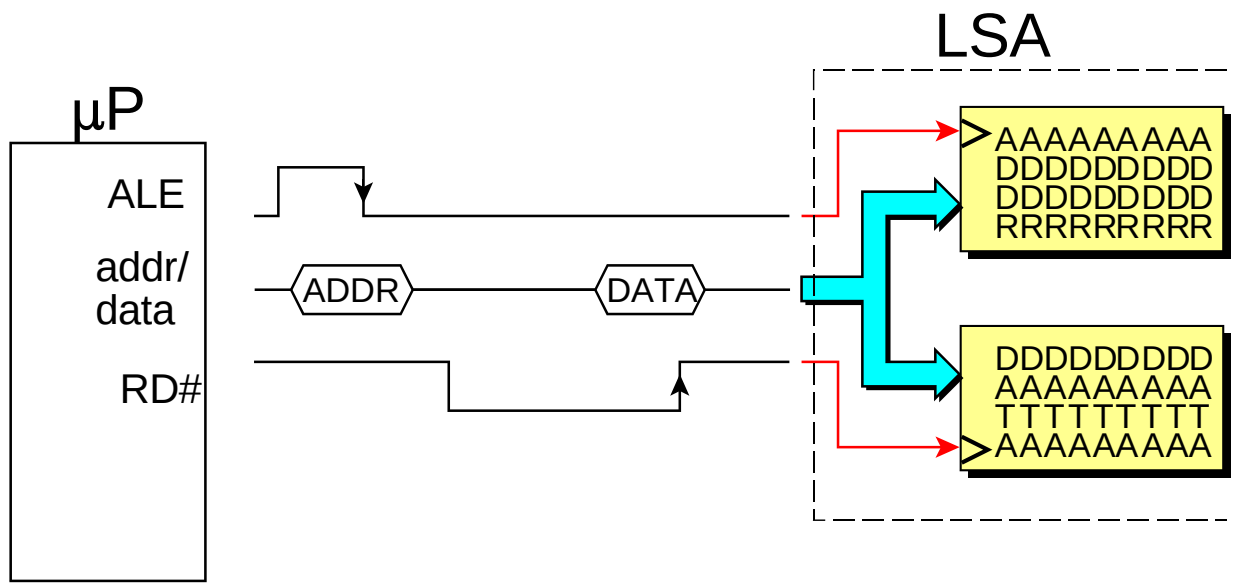
We can also take several clocks in the system and combine them (see next slide on the demultiplexing)

As for the trigger, we can take **clock qualifiers** in the system, i.e. signals which are not memorized, but are used to mask certain clock cycles ; this avoids filling the shift register with unwanted cycles, which contributes to saving memory and to the clearness and the ergonomics of the display.
We can for example use as qualifier the RD# of the microprocessor, which will restrict the acquisition to read cycles, or the Chip Select of a peripheral case to record only the exchanges with this one.

We can finally use a **mixed analysis** to detect an anomaly of timing in a synchronous system
- the synchronous analysis gives a trigger when the program reaches at a precise point (addresses, data, ctl)
- this trigger starts an asynchronous analysis with a faster clock to make a finer analysis from the current cycle

Clocks

demultiplex a time-multiplexed bus



This figure illustrates how we can pick-up two clock signals in a synchronous system to demultiplex a multiplexed address/data bus (e.g. a microprocessor from the 8051 family).

The multiplexed bus is connected to the inputs of the analyser and given to two modules of shift register (this connection is internal, there is no need to use two sets of probes).

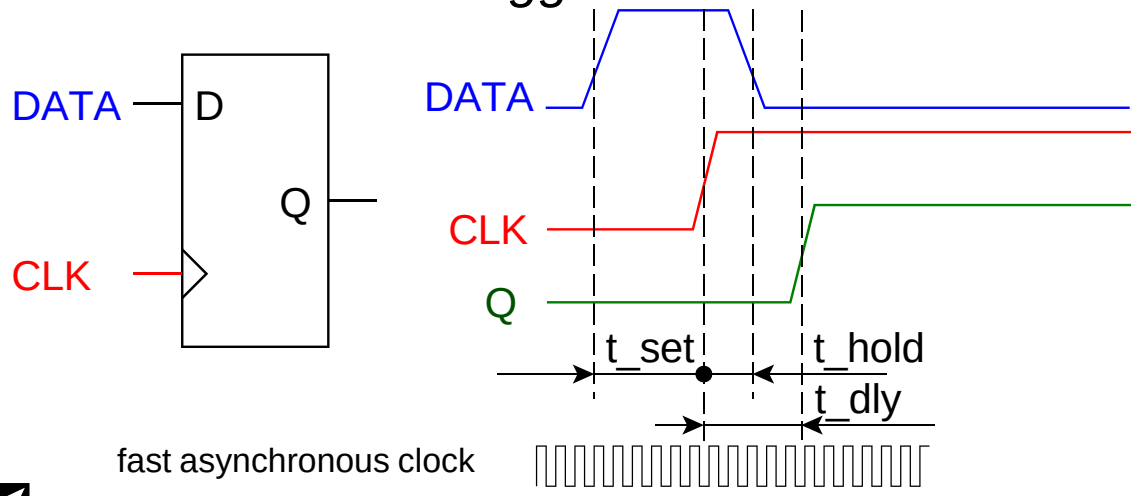
Each shift register is given its own clock:
- the falling edge of ALE to sample the addresses
- the rising edge of RD# to sample the data

On the display, the signals of the two modules will be gathered to present the addresses and the data in the same machine cycle.

Clocks

asynchronous clock: measure time and delays

- aim: to comply with the timing prescriptions
  - access time to a memory
  - *setup & hold time* of a bistable
- best LA : automatic *trigger* if violation



This slide illustrates the measurement of setup and hold times of a flip-flop.

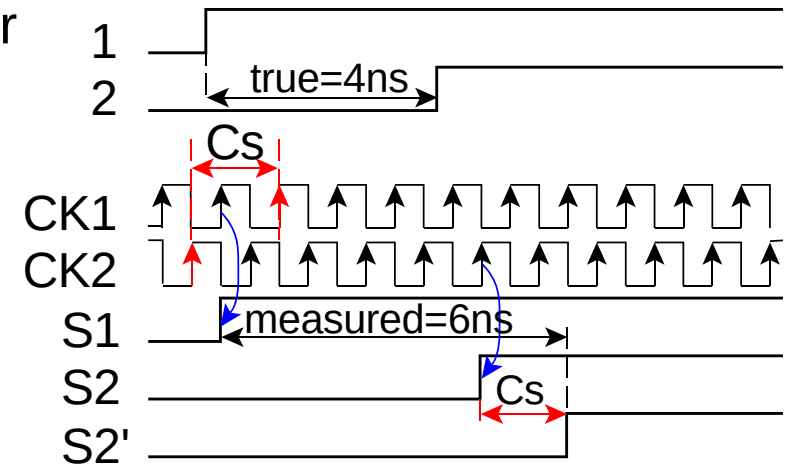
In top-of-the-range analysers, we can enter the prescripted values and the LA will trigger as soon as there is violation of these constraints, which saves a considerable time in the tracking of fugitive events.

To obtain a good accuracy, it is obviously necessary to have an asynchronous clock whose period is definitely lower than the delays to measure. We can now understand the need for sampling frequencies of several GHz to measure characteristic times which are sometimes as low as 1ns.

Clocks

precision of the measurements

- factors of inaccuracy
  - *clock jitter* = irregularity of frequency (negligible)
  - Ts = sampling period
  - Cs = channel skew
- measurement error
  - e = ± (Ts + Cs)



Like any measuring instrument, the logical analyser is prone to some **errors**.

- the best-controlled error is the **jitter** on the clock period, i.e. random fluctuations of the period around its average
- the second error is incompressible and results from the **time quantification**: each transition is displaced to the next active edge of the clock
- the third error appears especially in very fast logics in the form of a phase shift between the clocks of various modules called **clock skew**. This shift can be due to differences in lengths of connections. Indeed, the travel speed of the signals in wires is about 25cm/ns; therefore if we sample at 5 GHz (Ts=200ps), a difference of 5cm causes a shift of one sampling period.

This figure illustrates here a measurement error due to a 2 periods shift between clocks CK1 and CK2.

Clocks

which frequency to choose?

- asynchronous : Fs ≥ 4*Fmax
- synchronous : Fs < CLK(µP)
  - many µP have an internal PLL (clock multiplier)
  - Fs = frequency of the valid and stable data
  - ex : PC
    - frequency of memory buses = 100...1000 MHz
    - frequency PCI = 33 MHz or 66 MHz
    - high speed serial buses (k.100MHz)

The choice of the clock and its frequency are thus crucial.

For **asynchronous** systems, a good practice is to choose a sampling rate at least 4 times higher than the highest frequency to be analysed.

For the **synchronous** systems, in particular the fast microprocessor-based systems, the current frequencies of some GHz only exist in the processor kernel. The external oscillator is multiplied by a PLL (Phase Locked Loop) in the processor.

The highest frequencies to observe are those of the buses for memories, and serial I/O buses like LVDS.

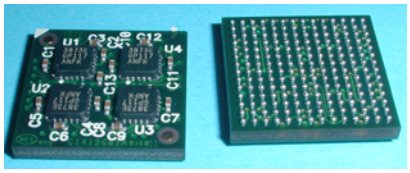
Logic analysers

CONTENTS

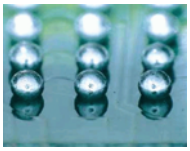
Harnessing

need for special connectors

- problems:
  - many pin, small pitch (≤1/20")
  - inaccessible pins (*ball-grid array*)



- solutions:
  - **/!\ avoid soldering wires**
  - special clips for gripping ot touching legs
  - special sockets with flexible printed circuit
  - special probe replacing the µP
  - high-density connectors on the PCB with adapted terminations (!specification for routing of the tracks)



- **foresee the clearance to connect the analyzer**
  - **surface on the PCB**
  - **thickness**

Modern cases considerably complicate the connection to the logical analyser; indeed, even when the pins are visible, their spacing keeps shrinking; in the BGA (Ball Grid Array) cases, pins are replaced by studs of solder at the bottom side of the circuit and are thus inaccessible (see figure).

Several solutions are possible:

- the **worse is to solder wires** to the PCB or the pins of the IC; the risks are
  - short-circuits between pins or between tracks
  - thermal shocks
  - destruction of the pads of the printed circuit
  - antenna effects associated with these wires

- when the pins are soldered and accessible, special **grips** can be used (see next slide)

- when the circuit is in a socket, it can be removed and replaced by a **special probe** or a printed circuit carrying of the connectors

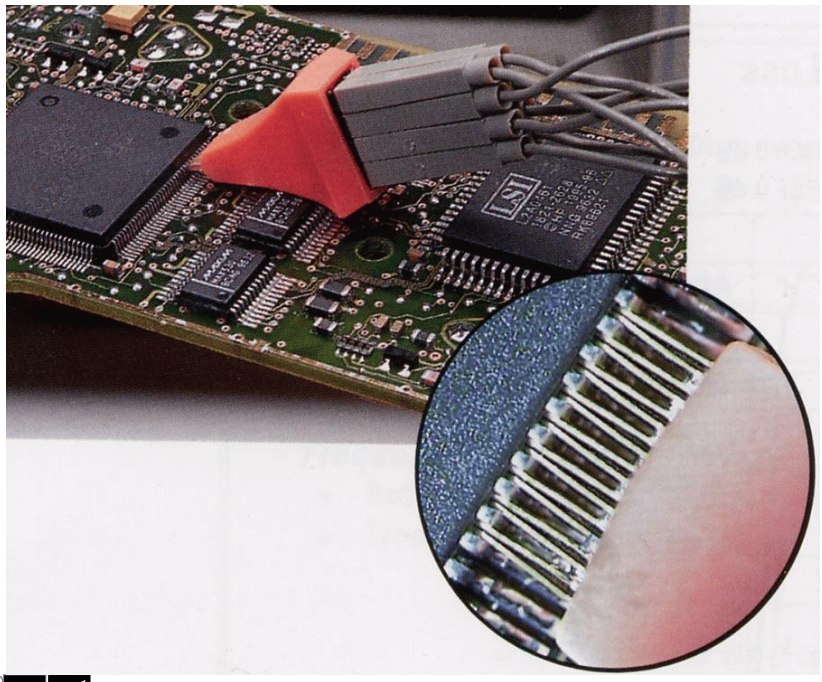
- finally, the easiest (but also the most expensive solution) is to place on the PCB special **high-density connectors**. Prescriptions of the manufacturer have to be respected carefully to route the tracks between the circuits and the connectors. In the last generations known as "soft-touch", these connectors are simplified in the form of miniature springs which come in contact directly on pads of the PCB.

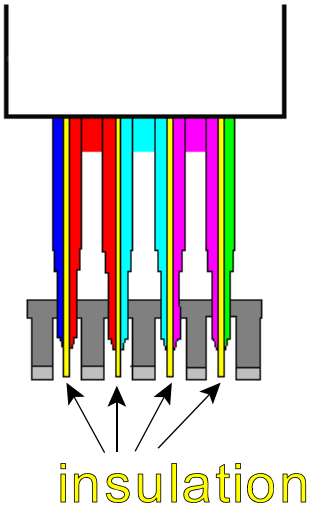
The problems of **ground connections** are delicate; in fast logics (<5ns), a ground wire is required for each signal wire.

When designing a PCB and the rack in which it will be inserted, always foresee **enough clearance** (in surface and height) to connect the analyser.

Harnessing

grips for lateral pins of SMD circuits



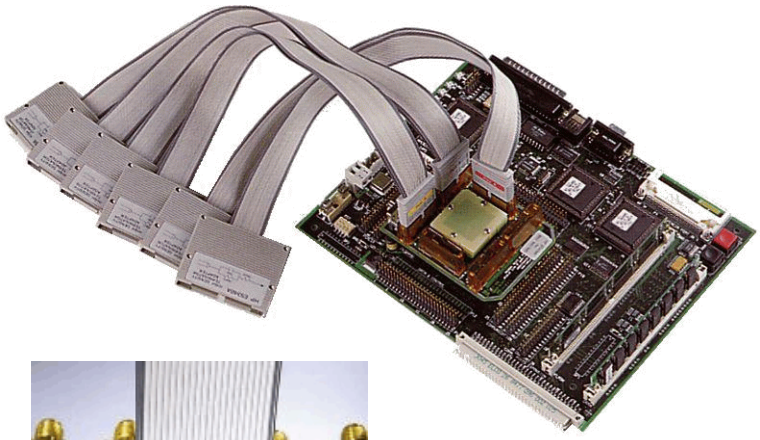


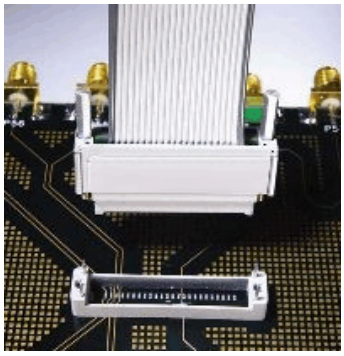
This slide shows probes adapted to surface-mounted IC whose pins are accessible laterally.

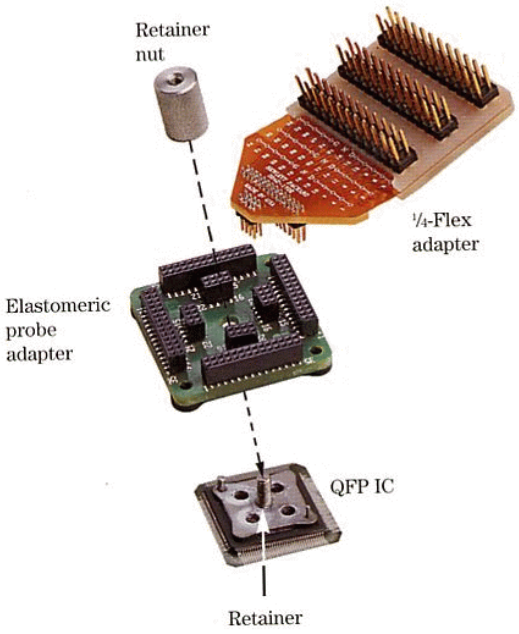
The geometry of the grip has been designed to avoid creating short-circuits, thanks to insulators (coloured in yellow on the figure).

Harnessing

special connectors







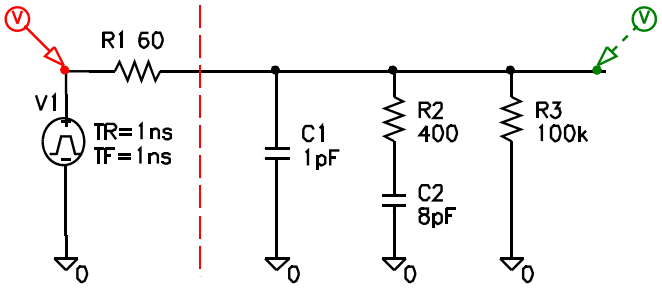
We have here examples in which the circuit to be analysed is equipped with connectors facilitating the connection of the logical analyser.

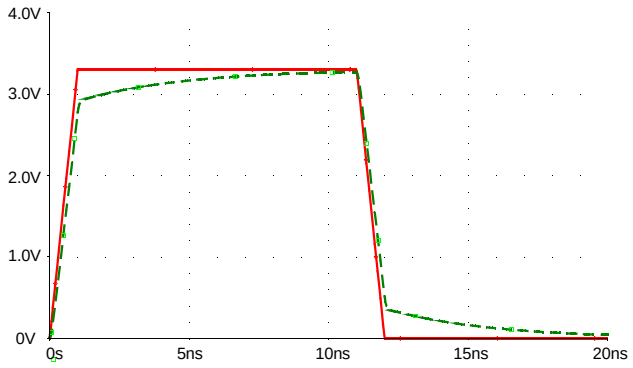
- in the top left figure, the IC has been removed and replaced by a PCB with the same IC, plus the cables towards the analyser
- in the right-side figure, a PCB with elastomeric contacts is pressed on the IC and picks-up the signals on all pins; to ensure very precise placement, a gauge is glued on the IC with a threaded mechanical part that will help to centre the PCB and press it by screwing a nut.
- the left-bottom figure illustrates a "soft-touch" connector whose contacts are miniature springs; the female connector on the PCB consists of solder pads plus a plastic framework to guide the male probe

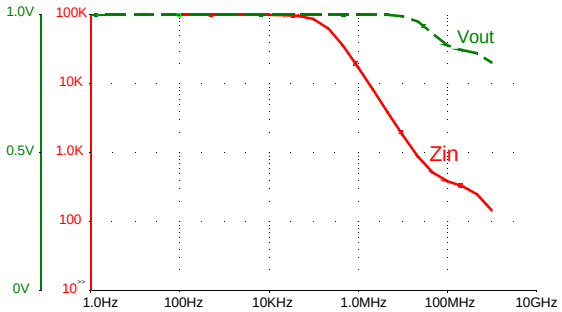
Probes

time and frequency response of the probes

- tr,tf = 1ns
  - BP=350MHz @-3dB
  - good precision on the time
  - probe = weak load
  - ! quality of the probes







The figure on the left shows us the equivalent diagram of a high-quality probe for logic analysers.

In the right-bottom figure, we see the small signal AC response; the input impedance falls appreciably beyond 1MHz, while the bandwidth is about 350MHz.

The most important quality is the fidelity of the step response.

The top-right figure shows the simulation of the response to a pulse with a rise-time of 1ns and a source impedance of 60Ω, which is realistic for a fast logic circuit; the deformation of the signal is visible, but not awkward, because the variation of the pulse width at the logic threshold (50% of the amplitude of the pulse) is negligible.

The probes must present a weak capacitive load (1pF to some pF) for the measured circuits, to avoid the degradation of the transition times.

Logic analysers

CONTENTS

Display

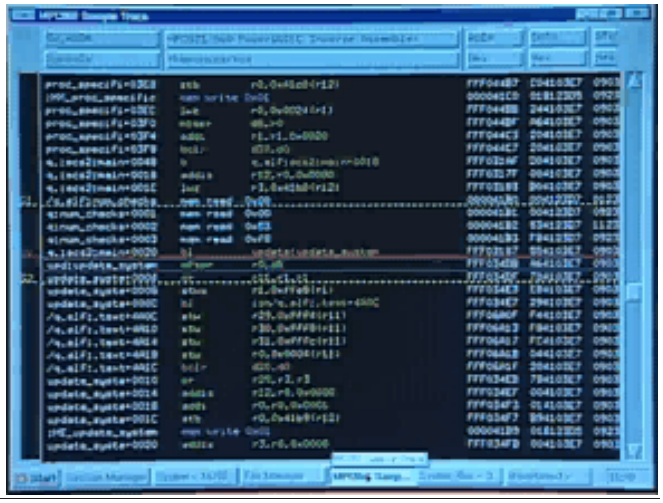
types of screens

- after the freezing of the memory, data is post-processed; then it can be displayed in 3 ways :
  - listing
  - time
  - performance

Display

state analyser (listing)

- debug and maintenance in synchronous logic or µP
- chronological listing of the memory
- selection of the channels to be displayed
- logical grouping of the channels (addresses, data,...)
- display
  - BIN
  - HEX
  - OCT
  - mnemonics (ASM)
  - source code HLL
  - ASCII/Unicode



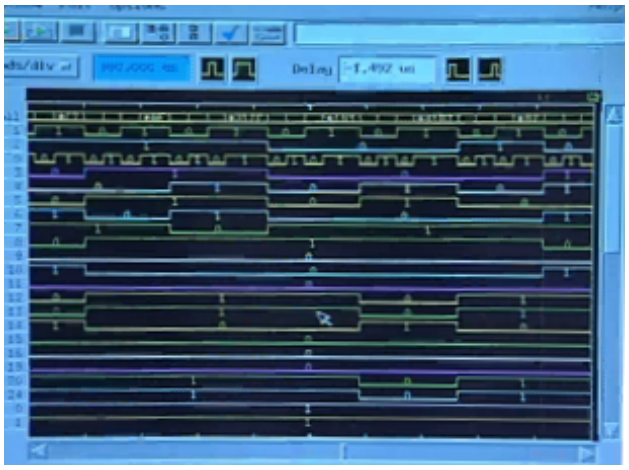
The **state analyser** is the display mode which looks the most like the screen of usual debuggers, i.e. the form of a listing, generally used for the software development of the microprocessor-based systems. In this case, the display groups clearly the channels belonging to the same bus and each column has got an explicit label.

The user will wants obviously to clarify the display by selecting only relevant data as well as the best type of representation:
- logical data, such as input/output ports are generally displayed in binary
- numerical data, like the address buses, or the value of a counter in hexadecimal
- strings are in readable text format
- execution cycles of a processor can be visualized in assembly language, but also linked directly to the HLL source code, which can sometimes require a very sophisticated post-processing of the data.

Display

time analyser

- screen looks like multi-channel oscilloscopes
- debug timing problems
  - access time
  - race conditions (set-up and hold time)
- cursors to measure time and delays
- zoom, pan



The **time analyser** looks like a digital oscilloscope with a much larger number of channels.

Channels related to buses can be grouped and labelled (address, data, ...) and the hex value is superimposed to the timing diagram.

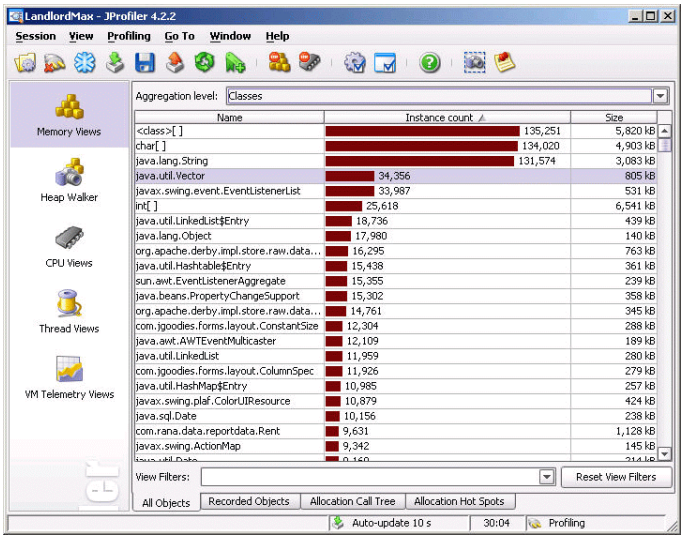
It is obviously used to solve the **timing problems**: race condition, access time to the memories, setup & hold time of bistables.

Time measurements are based on **cursors** and the principal commands of visualization are *pan* (horizontal scrolling) and *zoom* (adjust the time scale).

Display

performance analyser (code profiling)

- detection of functions which are most responsible for the execution time
- detection of "dead" code, never executed
  - useless or
  - non-exhaustive test
- histograms of the memory map



A useful postprocessing of the data is to make statistics of execution of a code on the microprocessor to determine:

- the portions of the code in which the most time is spent, and should therefore be optimized

- the portions of code which are never accessed because:
  - some branch conditions have never been true; if you are in a test phase, it can mean that the test is not exhaustive
  - if the test is exhaustive there is "dead" code that can be wiped-out

The screen is presented in the form of a histogram giving the CPU time occupied for all the functions.

Logic analysers

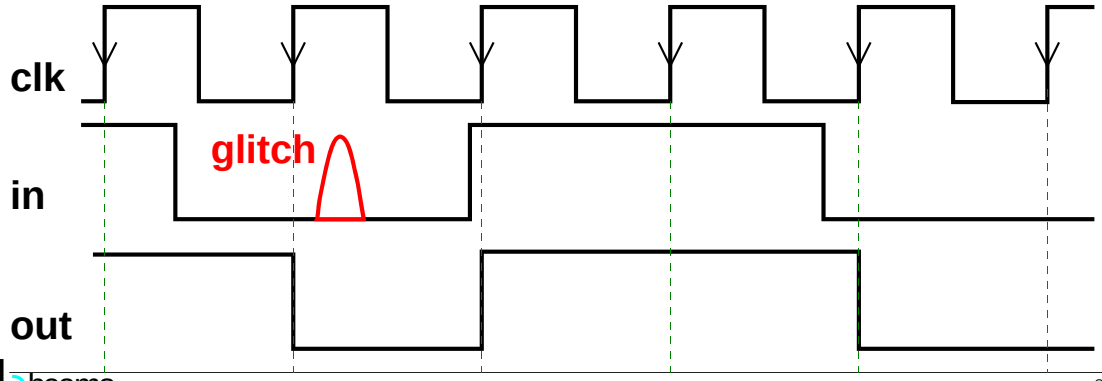
CONTENTS

- Introduction
- **Classical logical analyser**
  - principles of memorisation
  - trigger
  - clock
  - harnessing and connectors
  - display
  - **additional functions**
- Specialized analysers
- Conclusions

Additional functions

glitch detection

- *glitch* = short parasitic pulse
  - level and energy can be sufficient to perturbate the system
  - origin : race condition, crosstalk, power supply
  - width < Ts => difficult to detect
- detection
  - 2 flip-flops in tandem
  - permanent high frequency asynchronous analysis (GHz+)



The glitch is a very short parasitic pulse whose level and energy are large enough to be interpreted as a change of state at the input of a logic circuit.

During an analysis, it can happen (specially in synchronous analysis) that the sampling period is too long to detect such pulses.

In midrange logic analysers, detection of the glitches can be done by using 2 high-speed flip-flops in tandem, and reset alternatively at each clock period. The output of the active flip-flop is sampled in the shift simultaneously with the data. Hence glitch detection consumes half of the memory, since we store 2 bits per channel (data+glitch).

In the high-speed modern logical analysers, whose sampling rate can reach several GHz, the concept of glitch disappears, thanks to a permanent fast asynchronous analysis.

REM: to avoid glitches, prefer synchronous logic

Additional functions

reference memory

- reference memory is loaded with
  - acquisition on a good working system
  - reference file
- analysis of a suspect system
  - **no trigger condition to write** ☺
  - a trigger is generated automatically at the first difference
  - differences are highlighted on the screen
- remarks
  - very much used in production (ATS Automatic Test Systems) and in service
  - analyze "logical signature" instead of exhaustive test

Additional functions

special functions for µP

- ASM debug
  - acquisition of buses ADR + CTRL + DATA
  - display of disassembled code
  - symbolic debug (link to tables on development system)
- HLL debug
  - sophisticated probes
  - coupling with a PC to process data
- multi-processor analysis
  - problem of synchronization
  - composite *trigger*

The analysis of microprocessor-based systems is certainly one of the most complex tasks, in particular because we like to see on the screen of the logical analyser a listing close to the source code written by the programmer.

After having acquired the signals on the buses of a processor, the analyser post processes the data which means:

- **disassemble** the machine code to produce a listing in assembly language
- enrich this listing with the **symbols**, which implies a connection between the logic analyser and the host on which the code has been produced and to get the table of symbols and the memory mapping
- a correlation with the **source code in high-level language**; this last operation is very complex and often requires a very heavy postprocessing, plus a connection with the integrated development environment. More and more, this processing takes place on the host PC connected to the analyser by the network, which also enables remote diagnosis

Finally for complex processors, it is sometimes necessary to make a **preprocessing** of the signals acquired on the buses before storing them in the memory of the logic analyser; this preprocessing is carried out by a personalized probe which replaces the processor.

One of the most complex problems remains multiprocessor analysis, where it is necessary to couple several analysers and to define composite trigger conditions, combining several asynchronous events associated the different processors.

Additional functions

pattern generator

- provides programmable logical outputs
- allows to excite the inputs of the system under test outside its environment
- frequently used in combination with the reference memory for
  - automatic production tests
    - PCB
    - integrated circuits
  - service

When we must test a board or an integrated circuit in production or for maintenance, we are outside the normal environment; it is thus necessary to produce test sequences on the inputs of the system and measure the corresponding outputs.

In option, a "**pattern generator**" can be added to some logic analysers; complex sequences can be programmed providing the test vectors.

REM: the test vectors have generally been created in the **simulation** step of the design.

Logic analysers

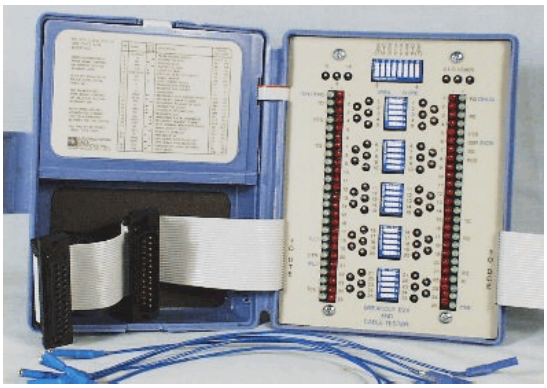
CONTENTS

- Introduction
- Traditional logical analyser
- **Specialized analysers**
  - **serial transmissions / networks**
  - **busses**
- Conclusions

Specialized analysers

serial transmissions - networks

- hardware level
  - serial-// conversion
  - detect format, baudrate
  - simulate a node
  - *breakout box*
    - open lines
    - impose level of control signals
    - permute signals
  - anomalies of wiring (reflexions?)
  - determination of the load and error rate of the medium
- software
  - protocol analysis (PDUs between peer layers)
  - I²C, SPI, CAN, Ethernet, ...



Data are more and more transferred via **serial transmission**; the manufacturers of instrumentation thus developed analysers (or software pre/post processor modules) adapted to the various communication standards from the simple RS-232 to the fastest networks.

These analysers are able to detect the hardware problems in the physical layer PHY

- by reconstituting the **characters** of the frames by a serial-to-parallel conversion
- by detecting the **format** and the effective **bitrate**
- by simulating one of the nodes of the transmission
- by providing a help for **wiring**; the figure shows a *breakout box* for RS-232 transmission which can:
  - open some wires
  - set the level of some control signals
  - permute signals
  - detect wrong wiring (abnormal impedance, reflexions,...)
- by making the statistics to determine of the **load** and the **BER** (Bit Error Rate) of a network

Compared to the various levels of the OSI model, some analysers allow to check the respect of the protocols of the various PDU (Protocol Data Units) exchanged between peer layers; they are called **protocols analysers**.

Even very simple and low-cost logic analysers are sold with software modules to debug simple serial communications and networks like SPI, I²C, CAN (see labs)

Specialized analysers

normalized buses

- GPIB (IEEE488), ISA, PCI, SCSI, AGP, DDR, USB, IEEE1394
- traditional analyzer + adapter/preprocessor + software
- development of the motherboards
- detection of the conflicts on multiprocessors busses

The interconnection between the various actors of a system relies upon **standardized buses**. The majority of the manufacturers of logic analysers, as well as satellite firms, propose accessories to connect (more or less directly) the logical analyser to these buses.

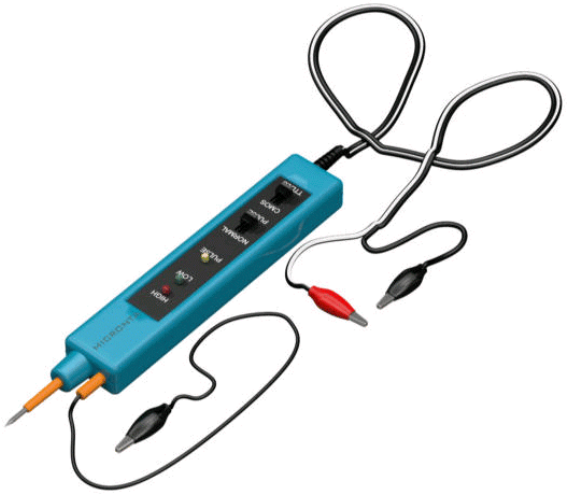
Probes are provided with the connector specific to each bus and generally contain a preprocessor. This slide shows a probe adapted to PCI; the interface is a mini PCI board, visible in green at the foreground.

A software postprocessing produces an ergonomic display in which the various events and protocol elements of the bus (initialization, burst, error) are clearly indicated.

Specialized analysers

simple logic probes

- very cheap
- only one signal
- functions
  - adjustable threshold (TTL/CMOS)
  - levels HI, LO, TRISTATE
  - detection of pulses
  - detection of periodic signals with coarse indication of the duty-cycle



At a more modest level, you should not underestimate the utility of **very cheap probes** measuring **only one logic signal**; the display is reduced to a few LED and can indicate

- the logic state: HI, LO, HiZ
- an approximate idea of the frequency and of the duty-cycle
- the occurrence of very short, possibly parasitic pulses

There are also some more complex probes for 2 or 3 signals, with a more elaborated display of voltages and frequencies.

Logic analysers

CONTENTS

- Introduction
- Traditional logical analyser
- Specialized analysers
- **Conclusions**

Conclusions

- start with simple means
  - test program (loops)
  - small logic probe / digital oscilloscope
- logic analyzer
  - irreplaceable in the difficult cases
  - little influence on the circuit
  - complex (trigger condition)
  - price 100€...25k€ depending on performances
- currently
  - frequency and memory size >>
  - coupling and synchro with fast digital scope
  - reconfigurable modular devices
  - support of HLL
  - coupling with the emulators