# AMChip test stand and chip tests

Description of test strategy

Roberto Beccherle, INFN - Pisa, Email: Roberto.Beccherle@pi.infn.it

## AMChip05 Mini@sic

- Yesterday we submitted a new version of the AMChip.

- Mini@sic to test some new features to be implemented in the final design:
  -> New implementation of the pattern matching mechanism that uses less power.
  -> New commercial high speed serializers that run up to 2.4 Gbit/s.
  -> New commercial LVDS pads.
  -> New serial protocol which runs in parallel to JTAG.
  -> New test features to address testability issues of the final chip.

- Final chip design will have to address testability of both the chips and the system.

- A "good" test system allows to fully decouple Chip issues from System issues

- Two main testability areas have to be addressed:
  -> Able to test chips in order to select good ones and detect faults.
  -> Able to detect malfunctioning chips/boards during operation.

# Ideal Test Features

- Ideally one would like to have JTAG and Boundary Scan for all chip pads
  -> Allows to check connectivity of devices on mounted boards.
  -> Allows system initialisation.
  -> Simple protocol fully supported by design tools.

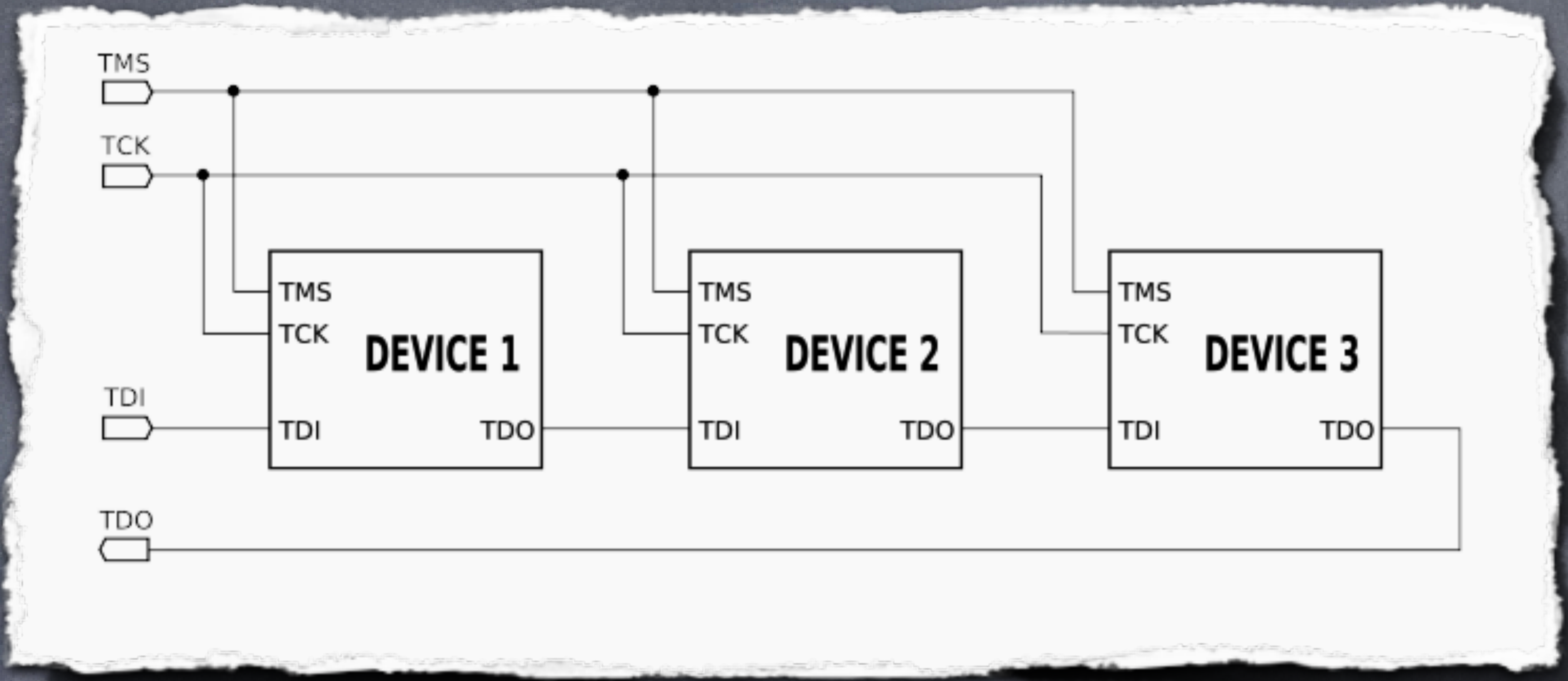
- Scan Chain built in inside the chip allowing observability of "all" internal FF's
  -> Able to check most fabrication faults inside the chip using a very limited number of test vectors
  -> Also fully supported by all design tools.

- Built In Self Test (BIST) to test internal memories, full custom components
  -> For memories there are some standars supported by the tools.
  -> For Full Custom blocks one has to develop his own test strategy.

- Send chips to commercial company to perform wafer tests to determine yield.

- Simulation environment connected to hardware Test Stand in order to allow detailed chip debugging during Lab measurements.

# JTAG - Boundary Scan

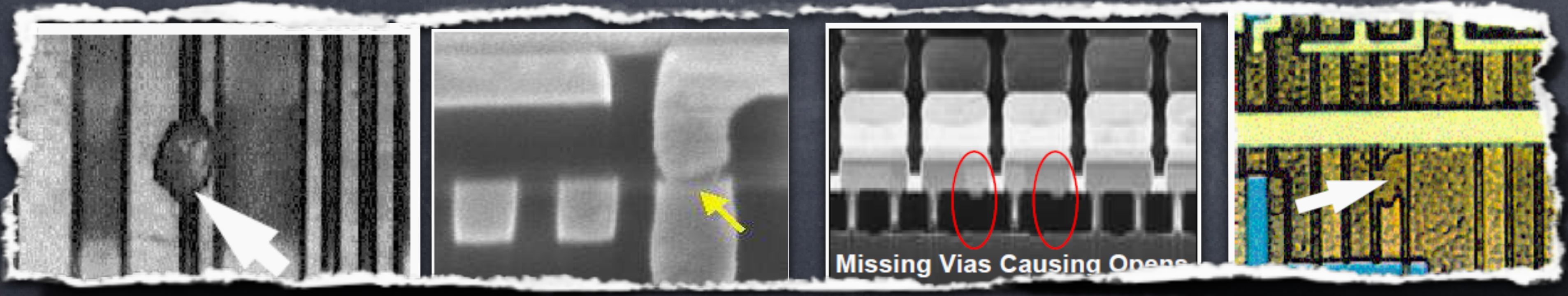


- All chips on the AMBoard will be connected by JTAG

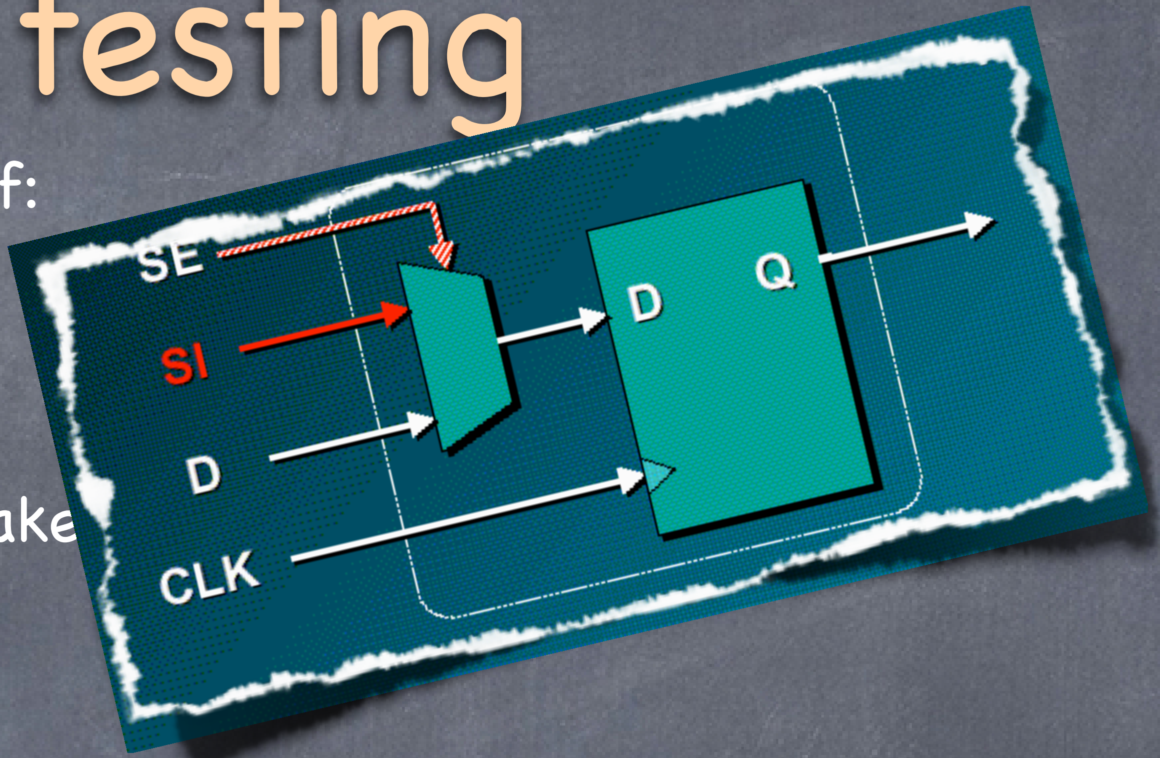
- Unfortunately not all PADS will be part of Boundary Scan:
  -> LVDS Pads allow to be added to Boundary Scan.
  -> SERDES Ip blocks do NOT provide this feature.
  -> We will have to deal with self made tests (see Matteo's talk), but we will NOT be able to fully decouple board issues from chip issues.

# Functional vs Structural testing

- Functional testing verifies that a circuit fulfils the desired specs
- Functional testing not feasible for exhaustive tests
  -> An example: 32-bit adder requires $2^{65} \approx 4*10^{19}$ test vectors!
- Structural test focuses on circuit structure and can cover manufacturing defects that may not have been detected with functional testing:
  -> Power or ground shorts.
  -> Open interconnection on the die
  -> Short-circuited drain or source of the transistor,
  caused by metal-spike through



# Scan Chain testing



- Typical Scan Chain Design flow consists of:

  I. Replace FF's with scannable FF's

  II. Isolate internal blocks in order to make them fully observable

  III. Create test patterns that allow to detect faults

  IV. Optimise and minimize number of test patterns.

- In the AMChip we have full custom cells that provide a very high pattern density inside the chip and therefore we cannot allow to make a full scan chain replacement inside the chip, especially with cell isolation, but we plan to implement at least partial scan chain replacement in order to find the best detectability / pattern density decrease factor.

- For the Mini@sic submission we did NOT use this test methodology, but it will definitely use it in the final AMChip05.

# Built In Self Testing

- In order to be able to fully exploit the memory writing and Pattern matching mechanism we developed a custom test structure in order to be able to address testability of these blocks

- Basic idea is the following:
  -> Initialise the internal memory.
  -> Generate random data be written to the memory
  -> Generate random data in order to simulate data coming from the detector
  -> Read back matched patterns
  -> Calculate a Cyclic Redundancy Check (CRC) on the output patterns
  -> Check that the CRC is the same as the precomputed one

- This approach would allow us to test the correct functional behaviour of the memory without having to check data offline

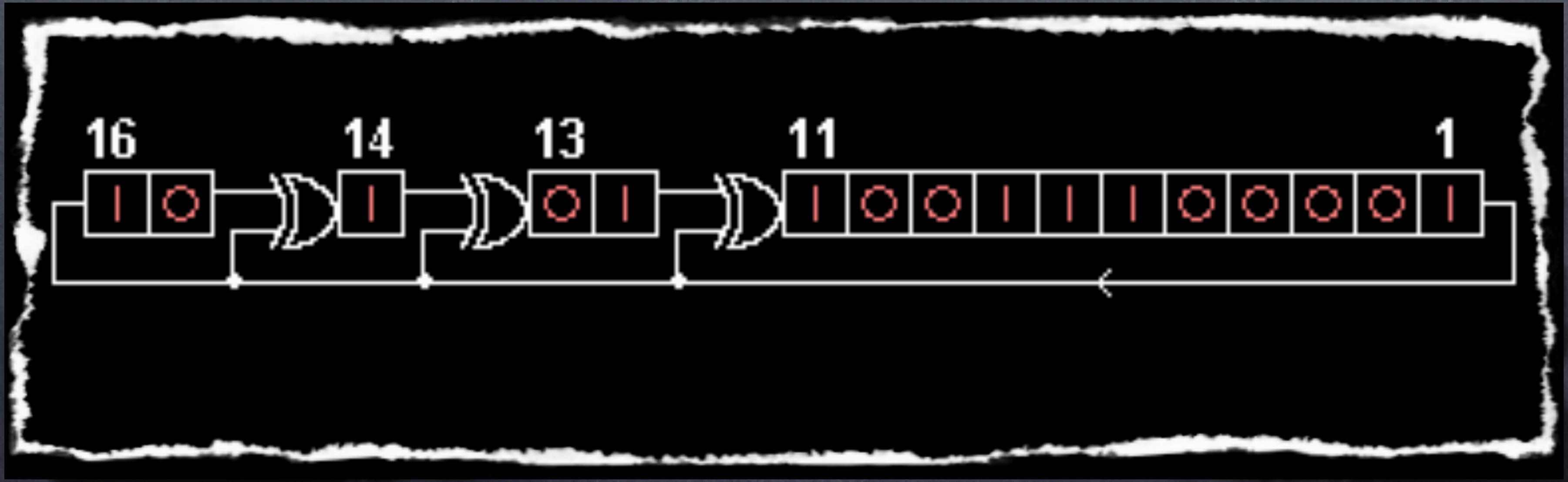
- Another feature of this test would be the ability to run power consumption tests in a very easy way

# Linear Feedback Shift Register

- Random pattern generator is implemented using an 8bit LFSR

- The circuit allows to generate 256 pseudo-random numbers.
  -> We use an 8-bit shift register with a certain number of Taps (XOR gates) in order to generate random order of the output numbers.
  -> The arrangement of taps for feedback in an LFSR can be expressed in finite field arithmetic as a polynomial mod 2.
  -> One has to choose a maximum length polinomial.
  -> A seed allows to start the sequence from a well known value.
  -> Output streams are deterministic.

- In final version of AMChip05 we will use a 16 bit LFSR



# BIST State Machine

- We implemented a very configurable state machine in order to allow full memory/pattern testing

- There are eight 18bit input buses to address and 64K memory locations

- State machine allows following variable parameters:
  -> Seed: Initial seed for the LFSR.
  -> NPatt: Number of Patterns to be written.
  -> AddrOffset: Address offset [0, 1, 2, 4, 8, 16, 32, 64].
  -> StartingAddress: Initial address of the memory to be written [0...127].
  -> DataSel: Allows to connect the buses in two distinct configurations
  -> ClkSel: Allows to choose the frequency of Data, simulating detector data, to be written to the memory. Clock frequency is 1, 2, 4 or 8 times smaller than original clock frequency.

- The State Machine has three main states, all individually selectable:
  -> Init: Clear all patterns.
  -> Write_Memory: Fill memory with random patterns.
  -> Write_Data: Simulate data coming from the detector.

# CRC-32C Generator

- Error detecting codes are used to detect accidental changes to raw data.
  -> Usually used in digital data transmission protocols like Ethernet.
  -> A CRC is a short checksum calculated on original data and appended to the data itself. On the receiving end one recalculates the CRC on received data and if the result is the same of the transmitted ones no data corruption is assumed.
  -> This is performed dividing original data by a fixed number and the CRC is the reminder of the operation.
  -> This technique is able to detect single, double bit flips and burst errors
  -> a "good" CRC is one that allows to detect as many as possible errors and that has a "low" probability that two different inputs generate the same CRC.
  -> There are many "standard" CRC's on the market, like CRC-32 used in Ethernet, Serial ATA, GZip, Bzip2 etc and CRC-32C used in iSCSI, SCTP and ext4

- We chose CRC-32C being a more powerful CRC with respect to CRC-32 even if its hardware implementation is a little bit more complex.

# Practical Implementation

- The idea on how to use this in the AMChip05 is the following:
  -> CRC calculator can be reset to a known initial value.
  -> CRC calculation is performed on all Data coming from the Memory.
  -> The calculated CRC will be stored in a configuration Register.
  -> A new value will be calculated on a different set of Data.
  -> At the end a simple comparison of calculated with expected CRC's will be performed allowing a "fast check" of faults in the memory.
  -> In case of errors one can still enable standard Data outputs in order to perform deeper debugging of the causes of errors.

- Unfortunately this block did NOT make it into the Mini@sic but it will be surely implemented in the final version of the chip.

# Conclusions

- New Mini@sic has just been submitted.
- This new implementation contains many improvements over AMChip04 design.
- With a full scale chip production one cannot relay on "in lab" tests.
- We definitely have to use test methodologies commonly used in industry.
- We have to be able to test chips using an external company.
- On such a complex system, in order to address reliability of the whole system, test methodology has to be part of the design cycle of both the chip and the boards.

## Still to be done

- We managed to add only some of the desired test features.

-> Implement ScanChain and CRC checks.

-> Integrate tests at the system level and prepare for testing the Mini@sic.