SJTAG

Here are my thoughts following on from Monday's meeting regarding SJTAG fault reporting – hope they make sense, and stimulate some discussion.

One of the greatest problems facing the adoption of SJTAG is how faults are reported/ logged once identified. There is an overriding need to ensure that accurate fault details are fed back to the repair depot to reduce NFF and ensure that the correct fault data pertaining to the system test is available.

At this time we, SJTAG working group, should restrict ourselves to the issues of failures detected while running these specific system tests. The fault catalogue of tests at system level is not necessarily the same as when performed in production and therefore there should be a higher level of single incidence faults compared to production test runs. This should make it easier to identify the actual fault detected. It is also necessary to take into consideration what level of detection is necessary at the system level

- Do we really need to know exactly which device/pin is failing?
- Is it enough to detect to a subsystem level and "swap out" the faulty subsystem?

Either way it is important that all the relevant data is transported with the faulty assembly and easily readable by the repair depot.

SJTAG is a vector based test system and all faults are reported as vectors and then analysed/translated by the test manager into meaningful fault data. However each JTAG vendor has their own method of translation and correspondingly the interchange of this data between vendors is not that simple. It may well be the case that a system comprises of several subsystems that are multi-vendor and consequently can be tested on a variety of boundary scan platforms.

I would suggest that the SJTAG test manager therefore has a role to play in the formation of fault data and the subsequent storage of fault data.

- Can the test manager be used to reformat the test data into a predefined format, create a write vector containing this data and write directly to the faulty subassembly?
- Can the device USERCODE REG be used to store failure data (or at least part of it) – again this would need some form of definition to be consistent and then any vendor toolset can read this data?
- Does each subsystem require a non-volatile storage device that can be written to by the SJTAG test manager?
- When the failure is across the boundary of more than one subsystem this is more problematic and the test manager may need to perform additional tests to isolate to a single subsystem, but then SJTAG is about diagnostics in any case
- Would there be a need for an additional system fault store whereby that fault data is also available in the case when the complete system is the repair item rather than a subsystem? This may need the inclusion of a non-volatile storage device where a specific area is reserved for SJTAG fault data. We would need to bear in mind however that in many applications a device specifically for fault data is an added cost burden which some designers may not be prepared to carry.

- Once we resolve this then the test manager could possibly be used to read fault data from other system level tests (BIT for example) and then write this data in to non-volatile memory.

In some cases the USERCODE REG is used to store device configuration data but does this need to use all of the register contents or can several bits be used/specified for fault data regarding that device? There is an argument that says configuration data should be controlled by the subsystem drawing set and any one device changing its configuration changes the drawing revision. If this is case then configuration control still needs to be available to SJTAG to ensure that correct test vectors are applied for the relevant system configuration.

Not all devices give access to a USERCODE REG – I can't remember if this is mandated in 1149.1 but I will look it up!

Well this is a start – may not be totally accurate but better than nothing ☺

Jim Webster

Comments from Brad (further comments by Jim in italics)

I am beginning to understand what you were talking about. I wonder if your proposal of using the USERCODE register warrants the optional addition of a separate dedicated DIAGNOSTIC register in order to not break the semantics of the original purpose of the USERCODE register.

I don't think that we should be trying to change or modify the 1149 standard here

Ideally, we need to consider the scope at which each test is applied. If it is a device BIST test the scope is at the device level. If the test is a board interconnect test, the test is scoped at the board level. I do not see a way of storing the board level test information in the scope of the device level since the device level view is based on the instance of that device in the overall netlist. Back in 1993, Rod Tullos and I looked into developing an SVF-II language that would incorporate diagnostic information that may be migrated to any type of tester that could apply boundary scan tests. This whole issue of test scope was the primary roadblock for coming up with a unified language and single test diagnostic representation.

I came up with the boundary-scan plug'n'play architecture that I presented at ITC 2005 for the very reason that board level tests should be scoped at the board level and accessible independently of that level but applicable for that level. The difficulty come with what data does one need to accurately diagnose the given test at the required stage of the product life cycle when the test is applied. It used to be thought that GO/NO-GO diagnostics in the field were good enough to identify the FRU, but current demands require the identification of root cause of the failure to determine trends in failures that might indicate a design issue or a manufacturing process issue. It is no longer the case that GO/NO-GO is good enough in telecommunications.

Further, the same system level tests are used throughout the product life cycle, including functional test, environmental stress test, and system integration testing. At each of these phases, the same tests require more granularity of diagnostics than just GO/NO-GO.

I don't believe that the fault catalogue at system level is the same as required in production – in many cases during part of production testing programmable devices (FPGA CPLD Flash etc) require to be configured and I don't think that a diagnostic field test should be reconfiguring such devices. Furthermore the production tests tend to look for manufacturing failures whereby system level test should be looking for functional failures with interacting subsystems.

I think what you have is a good start for the discussion, but I would like to see addressed the issue of diagnostics of the same test over the entire product life cycle.

See comment above

What I think is coming out of this avenue is the need for another standard group to identify system level diagnostic data representation for both boundary-scan and functional testing.

Do we really need to develop yet another standard? That tends to suggest that what we a proposing has shortcomings that need to be addressed elsewhere. Anything we develop should, in my opinion, be self supporting for that task undertaken. What we should be trying to do is includein our work a data format, based on SJTAG test applications, that is vendor independent

Perhaps some of the IEEE 1450 work could be extended?

We could certainly look at this, but yet another standard to attach ourselves to, especially when we don't control any updates/modifications that may have a knock-on effect to what we are doing.