Date – 05/20/2011

Minutes of the IEEE-1149.1 Working Group Friday meeting

Attendees:
Brian Turmelle,
Carol Pyron,
Craig Stephan,
Roland Latvala,
Carl Barnhart,
CJ Clark
Ken Parker
Wim Driessen
Francisco Russi
Ted Eaton
John Braden
Dave Dubberke

Excused:
Adam Ley

Meeting called to order at 8:35 am MST

Current Draft: P1149 1 Draft 20110502.pdf (_clean.pdf)
P1149 1 Draft Annex C 20110511.pdf

Agenda/Overview:
- Annex C - continued discussion
- PDL level1 vs. level0 discussion
- Misc items

Minutes:

Annex C discussion:

CJ provided an update from yesterday's 1687 meeting where he was looking for clarification to Carl's questions about PDL:
- Concept of padding: loading a 0 implies the whole register will be 0
- iScope sets the path for iCall
- 1687 wants to change iTarget to some other name.
- iComment may have a name change to iNote or iTag.
- iApply following an iApply does nothing.
- iScope followed by nothing clears iScope
- There is no such thing as "iScope ."
- There is no absolute path to iCall, only relative.

CJ also pointed out we have a need to load default values from BSDL.
Ted asked if this had been discussed with 1687 yet?
Carl and CJ said that it hasn't been discussed yet, but the issues are fairly minor, and we are pretty close on this but we are not done yet.

Carol asked if every iRead involves a compare to expected value? Carl responded yes, but it occurs at iApply.

There was a lot of discussion today about the notation for long iRead expected value strings and padding of values, and . (eg: 1000 X's), and how to best handle this.

- Ted said there is no short hand notation in TCL, you have to write it out.
- Ken suggested maybe we invent one. A command processor understands how to process commands, and they allow putting anything in quotes that says I want a thousand X's. When we are done a TCL parser should be able to read our commands even if it doesn't understand them.
- Ted responded: Yes, TCL allows writing in C, or SWIG, and it pulls this in. You can manipulate this however you want and TCL then calls the other native commands.
- Carl – A quoted string gets broken into white space delimited segments.
- Ken – TCL syntax rule 4. Between quotes the string can be as long as you want.
- Carl/Ted yes, but $ needs to be backslashed.
- CJ – We are down into the details of quotes and strings. Do we want to go back to 1687 and say we also want this capability?
- Carl – You can use the mnemonic name and a range on it. We have a way to do it. Do we need a 2nd method in PDL?
- Ken – We could also have a parameter. An iPad.. (laugh ☺)
- CJ and Ted confirmed that today they break up large registers into smaller segments and haven't had any issues to speak of to date.

Carol raised the question about whether iRead value is sticky or not. There was a lot of discussion on this topic as well:

- Ted pointed out that in 1687 iRead value is not sticky
- Carl mentioned today that it is sticky in our document
- CJ pointed out that we need rules to define constant defaults as retained, but used defined values are not.
- Ted pointed out that 1687 only defines the defaults after a reset.
- Carol agreed that once you write a register the default values are gone.
- CJ pointed out we have 3 cases:
    - Default values
    - Safe values
    - Reset values
- Ted thought these were irrelevant except for the bread crumb bits of Bypass, Device ID, and IR Captures.
- Carl – There are safe values in Boundary register. If you don't specify a safe value, we will load values by default.

- CJ – If you don’t specify an X is loaded. Not true?
- Carl – I’d like to have a fully disciplined language but might get overruled.
- Carl – Did you say iApply clears the expected value scan frame?
- CJ – It is not sticky. From one iApply to another it will change every time, so the iRead is not sticky.
- Ted – Default doesn’t mean always. 1687 has a reset value, but not a default value.
- CJ – In the register fields you can specify the iRead default and iWrite default. This goes beyond the 3 registers.
- Carl – The expected values is always defined.
- Ted – The 1687 default value is the safe value.
- Ted – We should ask 1687 if constant capture values are to be compared on every scan.
- CJ – On iApply we clear the expected data we just spoke to, but then I reload any constant capture values that need to be compared.
- CJ – Capture values after reset and more common than constant capture values.
- Carol – How do we compare a few constant bits in a field of Xs?
- Carl – In regiser fields, yes.
- Carol – A background field defaults to X, and only a subset defaults to known values on a few bits.
- Carl – You could do this in two steps

Misc items:

CJ asked for some clarification on Register Fields syntax, and will work on this this weekend.

Francisco ran a verilog synthesis and confirmed the code matches the circuits Carl has defined.

PDL level1 vs. level0 discussion:

CJ asked for input from Ted and Francisco about the choice of using level0 vs. level1 for 1149.1. How many on-chip registers can use level0 vs. level1 today? The response from Ted was that he would do everything in level1, and some limitations could be placed on the language for cases like Ken’s ICT where needed, and written out flat for such cases. Francisco was using STIL today and said he would have to look into this.

Key points of this discussion below:

- Ted – Most registers I can write literally, but why would I want to. I don’t want to write anything in level0. I think all of these internal tests like Bist benefit from a while-loop or for-loop from level1. Who wants to write linear level0 stuff? I’d go for level1 everywhere.

- CJ – I feel like we know that we all feel a need to use TCL but we are not going all the way there. Will people say this is a mistake? How can we satisfy everyone?
- Ted – Use 1687
- CJ – Ken's ate base uses BSDL. Do we ham string them from the start?
- Ted – How many of these tests can be run in ICT tester env anyway?
- Ken – I need tools that think of a collection of IC's on a board.
- Ted – Bist engines in one ASIC can be impossible to describe in level0.
- CJ – Carol has test features in her chip that cannot be described in level0
- Ted – PDL level1 could be flattened into a level0
- Ken – This is PDL level0.5 from what I hear you saying.
  - Math on arbitrary values
  - Why do I need algorithms for a single chip?
  - Why not embed some other language?
- Ted – In level1 it is a few lines of code, that can be written flat.
- CJ – Agreed, a thousand scans or a simple loop. The community is served well to go to different vendors and see the same language. TCL is a design language to build the chip, and bringing this to the test engineers supports synergy across the industry.
- Ted – To Ken's point there are $M testers that don't handle looping very well. If we want to restrict to not allow reading back.
- CJ and Ted agreed that iGetReadValue could be an issue on some ATE testers and those wouldn't have to support that command.

In closing Ken had concerns about TCL being a loose standard and resulting tool compatibility issues. Also Carl asked would people parse level0 PDL with TCL parser or be writing their own parser? CJ and Ted emphasized that a common 'free' TCL interpreter is downloaded by everyone as a baseline and then tool vendors are allowed to develop extra commands under the hood, but using the same TCL parser. In response to this Ken was a little more comfortable with the idea of TCL if we dictate a version that everyone works from.

Meeting adjourned: 10:03am MST

Action Items:

- Carl to send out an updated Annex C

Next Friday Meeting:

- Next week Friday May 27, 2011