Date – 02/18/2011

Attendees:
Carl Barnhart,
John Braden,
CJ Clark,
Dave Dubberke,
Ted Eaton,
Heiko Ehrenberg,
Roland Latvala,
Adam Ley,
Ken Parker,
Carol Pyron,
Mike Ricchetti
Francisco Russi,
Brian Turmelle,

Agenda:

- CJ: Carl proposed a "clean-up" instruction instead of blocking Reset* and I could accept that, but feel we already meet the requirements without separate blocking of Reset*
- Carl: Would having it a Standard instruction help the downstream test engineer?
- CJ: Maybe. The problem I was trying to solve is that we have never had guidance before about user instructions having to manage the I/O. With CLAMP_HOLD, we need that and I don't want to provide a way to avoid that.
- Carol: We currently will have a chip in system go "busy" so we can test it. We then clear the registers with the system reset on the chip.
- Mike: I see no need to block Reset*, we use POR/TRST* to clear critical registers. We would use the Test-Mode Persistence controller and IC_RESET as we have similar private instructions now.
- Carol: Do you have any TCK to mission clock domain crossings?
- Mike: We exit persistent test modes with POR/TRST*.
- Ted: What about the rest of the system?
- Mike: We don't support our private tests in system, so we don't have to worry about pin control.
- Ted: What about TLR? What about 1500 wrappers?
- Mike: All 1500 wrappers are private instructions.
- Ted: Both TLR and TRST* are allowed to reset TDRs. I use TLR and need TLR blocking with and without CLAMP behavior.
- Carol: There is a permission for local control of the mode signal in Test-Mode Persistence.
- Ted: I have scenarios with pins clamped and without pins clamped, but still need to hold TLR.
- CJ: I think there is just a misunderstanding. CLAMP_HOLD is only about the "mode" signal and controlling the I/O. What is critical is that when you design your TDRs, you need to take into account and provide any I/O control required.

- Carol: The blocking of Reset* with the Persistence controller right now is for controlling the I/O, we do not talk about other TDRs.
- Ted: Example: many other registers may need to interact with the reset-select register. If we need Reset* blocking there, we will need it in the registers that interact with it.
- CJ: The Persistence controller sets the "mode" signal, and we give you the capability to override it locally. That is what you need.
- Ted: What I need is control of Reset*. CLAMP_HOLD is a boundary register enabling instruction and is orthogonal to the issue of controlling Reset*.
- CJ: User instructions can manage the I/O if they want, up to and including total override of the I/O clamping.
- Ted: you are missing my point. I need separate control of Reset* independent of any clamping behavior.
- Carol: Reset blocking is orthogonal to clamping.
- Ted: Yes. I must have Reset* to all TDRs, and need separate control.
- Carl: Blocking can lead to other problems. I keep hearing a need for three levels of reset: TLR based Reset*, POR/TRST*, and a new reset controlled by the test sequence, so I proposed a sort of soft TRST* driven by a new instruction to replace Reset* on all those TDRs that would need a blocked Reset*. (See emails for details.)
- Ted: The problem is that a fault on TRST* could prevent the chip from coming up functionally. Reset* (which TRST* will also assert) is taken to all TDRs so that the 5-TMS reset sequence will allow the chip to come up, ensuring that test logic does not interfere with functional logic.
- Carl: Finally, I understand why you have Reset* to all TDRs, which is not common practice.
- CJ: What about a POR?
- Ted: On-chip POR is not reliable and not allowed in our chips.
- CJ: What about using a PLL lock or count?
- Ted: Also unreliable and PLLs must be programmed to work. I've proposed two ways of providing a reset-block. It's not difficult.
- Carol: We could put two bits in the Persistence controller.
- CJ: I think your reliability constraints are beyond what most of the industry needs. Would you be willing to leave this issue as something that you can implement with your own instruction rather than a Standard instruction?
- Ted: If no one else needs it, I can live with that.
- Ken: New question: when does the local override of the "mode" signal go away?
- Carl: It is based on the instruction decode, so I assume it goes away when the instruction is no longer active (Note: I added a rule to that effect, we will need to review it when we get to that clause.)
- Ken: I want to avoid a user instruction leaving behind an override that would require a power cycle to remove.
- CJ: We could use a TLR or other instruction-decode to clean up. Perhaps cancel the override any time a Standard instruction is loaded.

Meeting adjourned: 11:05am MST.

Action Items:

- Nothing new.

Next Working Group Meeting:

- Next meeting Feb 25, 2011