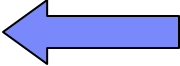




Automated Assertion Checking in Static Timing with IBM ASICs

Nathan Buck (nbuck@us.ibm.com)
Bill Rose (billrose@us.ibm.com)

Working hardware requires good static timing assertions

- Static timing assertions must model chip behavior, or hardware failure may occur
- Multiple methods are necessary to help test logic and assertion correctness
 - Tools to check clock domain crossings
 - Important to do. Tool must be purchased or created in-house.
 - SDF simulation
 - Important to do. However, it is difficult to get working before netlist is timing closed.
 - Manual review
 - Important to do. Catches many defects, but cannot guarantee all defects were found.
 - IBM ASICs automated assertion checks  **Our method**
 - Offers quick and valuable assertion correctness feedback throughout the design cycle.

Automated Assertion Checking checks for things like:

- Phase existence and slew requirements on the chip boundary are complete and consistent with the assertions
- No timing tests on one clock domain are excluded from timing tests with itself.
- Flop data pins have data phases
- Flop clock pins have clock phases
- Jitter has been modeled on all clock phases
- Timing rules exist for every cell on the chip
- Partial and complete clock domain crossings are identified
- Flops with no timed phases on their data inputs or outputs
- Summarize all the unique phases present on flops and MUXes
- Verify phase-based assertions corresponded to the actual found phases to ensure the intended assertions were applied.

Timing utility to check assertions can be run quickly

- Works within IBM's EinsTimer tool that customer is already familiar with and using
- Mostly cares about relationships of phases, missing phases, etc.
- Does not have to be a complete variation-aware or statistical timing run with all those bells and whistles
- Do not have to load parasitics, voltage files, etc.
- Can use simple Zero Wire Load (ZWL) timing
- Each check's output includes the documentation of that check, so no user guide needs to be consulted

Assertion checks can be throughout the design cycle with netlists of different maturity and scope, and with assertions of different maturity

- Can be run with top level padding inserted or not
- Can be run with test logic present or not
- Can be run with simple Zero Wire Load (ZWL) timing, or any other wired load model
- Can be run with Steiner (estimated) parasitics or the most accurate Spice-extracted parasitics
- Can be run on the first generated netlists, intermediate netlists, or final netlists.
- Can be run on netlists of different scope, including the whole (flat) chip, just the top-level, or any large block of logic (RLM) by itself.
- Can be run with any level of assertion maturity, and can be used to help judge assertion maturity.

Picked and run like this from within the IBM ASICs Timing Methodology

```
△ OPT Optional EinsTimer Timing Assertion Checking (Zero Wireload Timing)
  OPT (a) Assertions on A and Tristate-enable pin on chip output I/O cells
  OPT (b) Assertions on chip output ports (unacceptable in ASICs methodology)
  OPT (c) Slew requirements specified on A and Tristate-enable pin on chip output I/O cells
  OPT (d) Slew requirements specified on chip output ports
  OPT (e) Verify no phase is phase-pair excluded with itself
  OPT (f) Check clock group phases to verify their phases are not phase-pair excluded
  OPT (g) Check differential I/O inputs have phase on P or N pin not both
```

Each of the 7 single step examples above executes an assertion check within a ZWL timing run.

The assertion checks can be selected one at a time, or groups of checks can be run with no manual intervention.

...or anytime timing has already been run

Assertion checks have their own namespace, and can be called from within any type of timing run.

```
echo -prefix assertion_check -information "(e) Verify no phase is phase-pair excluded with itself"  
# If a phase is phase-pair excluded with itself then no static timing tests are performed within its domain.  
# This is sometimes intended, but usually an error.  
::assertion_checks::check_phase_pair_excluded_with_self -sOutputFileName /afs/btv/data/aework/bof_hier/guide_work/node1/out
```

Assertion check example – incorrect or no phase on flop clock pin – documentation of assertion check is shown first

- #
- # EinsTimer - Assertion Check - Clock pins of flop/latch - no phase - bad phase - or voltage
- #
- # The purpose of this assertion check is to determine if the clock input pin of
- # each flop/latch has an EinsTimer data phase. EinsTimer data phases have a
- # suffix of '@L' or '@T' (e.g. CLKX@L). EinsTimer clock phases have a suffix
- # of '+' or '-'.
- #
- # Most flop/latch cells will have a clock phase on their clock pin(s), but sometimes their
- # are exceptions as might be the case where a flop/latch is used for certain kinds of IBM
- # test purposes only.
- #
- # This check is not run on set/reset pins, if any. No phase checks are made on set/reset pins.
- #
- # If the instance names of flops/latches that are exceptions to this check follow a
- # consistent naming convention then it is easier for the user to review the output and
- # quickly verify the printed messages are acceptable for this chip.
- #
- # Four clock pin phase checks are done and the user is notified if any check is violated:
- #
- # 1) No clock pin is connected to VDD or GND
- # 2) A clock pin has a phase, but is not '+' nor '-'
- # 3) A clock pin has no phases
- # 4) No clock pin is found on the flop/latch
- #

Assertion check example (continued) – incorrect or no phase on flop clock pin – sample output and summary

- NO CLOCK PHASES: 'iBLK/iCLKG/gSPIpll[1].iSomeCell/iPLL/LAT_OBSERVE0', pin 'B' (Type LTL0001X2A12TH). Skipped.
 - NO CLOCK PHASES: 'iBLK/iCLKG/gSPIpll[0].iSomeCell/iPLL/LAT_OBSERVE0', pin 'B' (Type LTL0001X2A12TH). Skipped.
 - NO CLOCK PHASES: 'iBLK/iCLKG/iCORE/iPLL/LAT_OBSERVE0', pin 'B' (Type LTL0001X2A12TH). Skipped.
 - BAD CLOCK PHASE: NO TRAILING [+]: 'iBLK/iCLKG/iTWGDSK64QTR/VM/IBM_USEROFF_in_reg', pin 'B' (Type LTL0001X1A12TR), Clock phases 'LBIST_CLK@L'. Skipped.
 - BAD CLOCK PHASE: NO TRAILING [+]: 'iBLK/iCLKG/iTWGDSK64QTR/VM/IBM_WIDTH_in_reg_0', pin 'B' (Type LTL0001X1A12TR), Clock phases 'LBIST_CLK@L'. Skipped.
 - BAD CLOCK PHASE: NO TRAILING [+]: 'iBLK/iCLKG/iTWGDSK64QTR/VM/IBM_WIDTH_in_reg_1', pin 'B' (Type LTL0001X1A12TR), Clock phases 'LBIST_CLK@L'. Skipped.
-
- Registers on the chip: 1463041
 - Cross01 Registers that are scan only: 0
 - Cross02 Registers with no clock pin: 0
 - Cross03 Registers with no clock phase: 3
 - Cross04 Registers with all untimed data phases:* 27417
 - Cross05 Registers with bad clock phase:* 800
 - Cross06 Registers with no data pin:* 0
 - Cross07 Registers with no data phase:* 411
 - Cross08 Registers with bad data phase:* 32
 - Cross09 Registers with data pin voltage:* 33372
 - Cross10 Registers with clock pin voltage:* 0
 - Cross11 Registers with partial phase mismatch:* 196117
 - Cross12 Registers with complete phase mismatch:* 61060
 - Cross13 Registers with complete phase match:* 1297712
-
- Cross14 Registers with data pin constant 0 or 1:* 3721
 - Cross15 Registers with clock pin constant 0 or 1:* 28671

Has automated assertion checking caught real errors?

- YES! These scripts have caught errors in assertions that would have resulted in bad hardware
- If an assertion defect causes a hardware failure or is caught just in time before tape out then a new assertion check is added to prevent a similar defect from escaping detection on other chips.
- 31 checks (and counting..)