DVClub

SystemVerilog Assertions (SVA) in the Design/Verification Process

Benefits of Using Assertions

I have found that adding assertions shorten verification time and improve the quality of my designs:

- Less time to identify and debug failures
- Improve designer/verification communication
- Document design behavior
- Detect unobservable faults
- Ease integration of reused/IP modules

Incorporate them early in the design cycle (during RTL creation) to benefit throughout the verification process

Cyclic Design

Background

I began using OVL assertions when they were standardized by Accellera:

- Simple, but fairly inflexible
- Made the design “messy” since instantiated as modules

Learned SVA when evaluating formal tools

- Provided power, flexibility, concise syntax
- Clean way of creating simple assertions
- Inlined in RTL, but less messy than OVL

Problems using SVA

SVA presented its own set of problems:

- Difficult to construct anything besides relatively simple assertions
- Didn't use the more complex operators
- Attempts to create moderately complex assertions, often resulted in them triggering incorrectly; subsequently I would remove these since the time to debug them was not worth the effort.

Zocalo Zazz

Approached by Zocalo to provide feedback on their new tool Zazz

- Visual SVA addressed the creation issues, allowing me to create complex assertions without becoming an expert in SVA syntax
- More importantly, Zazz provides a methodology to debug assertions at the time of creation

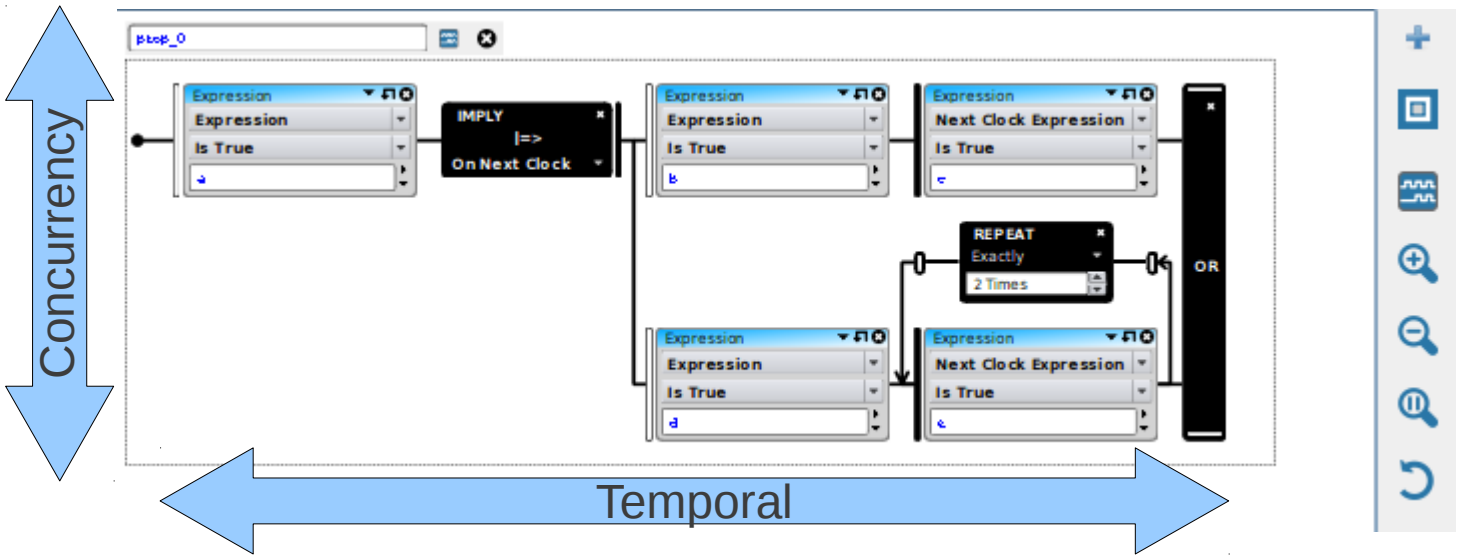
Debugging assertions prior to use is critical

- *This is why most companies use predefined assertion templates (like OVL) – not flexible, but debugged.*

Cyclic Design

Visual SVA

Zazz solves the creation problem by representing assertions graphically on a 2-dimensional canvas which conveys the temporal (left to right) as well as the concurrent (top to bottom) nature of assertions.

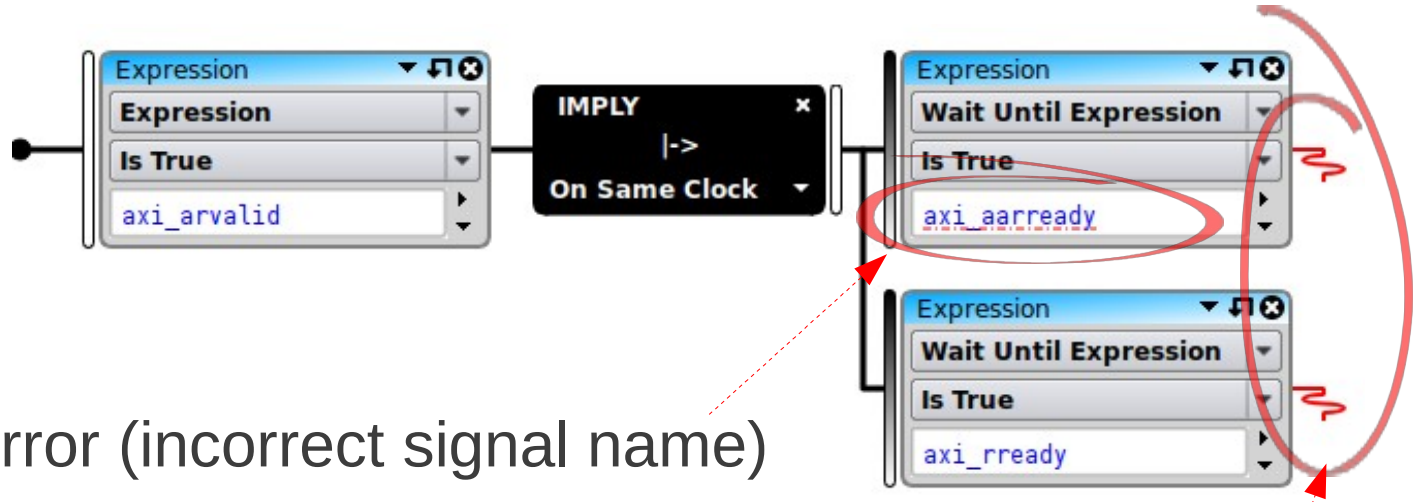


This mimics the way that I like to think and makes it easy to create assertions and understand the relationships between the operators

Cyclic Design

Visual SVA

Visual SVA also helps in creating structurally and syntactically correct SVAs:



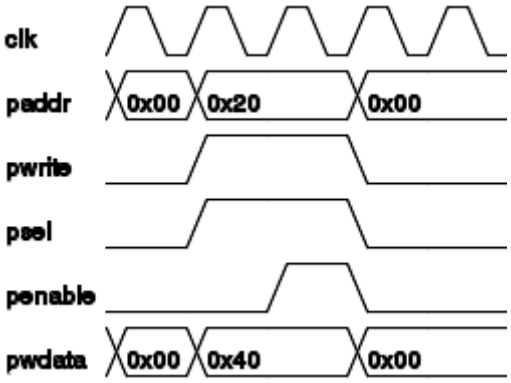
syntax error (incorrect signal name)

structural error – unterminated sequence

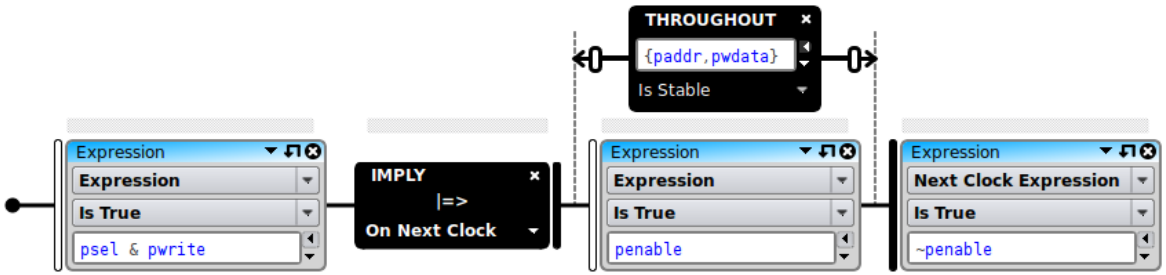
Cyclic Design

Debugging Assertions

Visual SVA closes the loop between design intent and actual SVA behavior by creating a constrained-random testbench around each assertion.



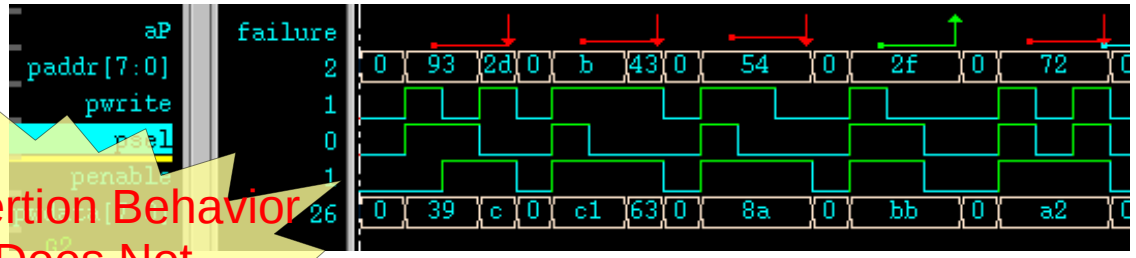
Designer Intent



Created Assertion

Debug Assertions Before They Fail in Simulation

Assertion Behavior Does Not Match Intent.



Actual Behavior Allowed by Assertion

Cyclic Design

Impact to Cyclic Design

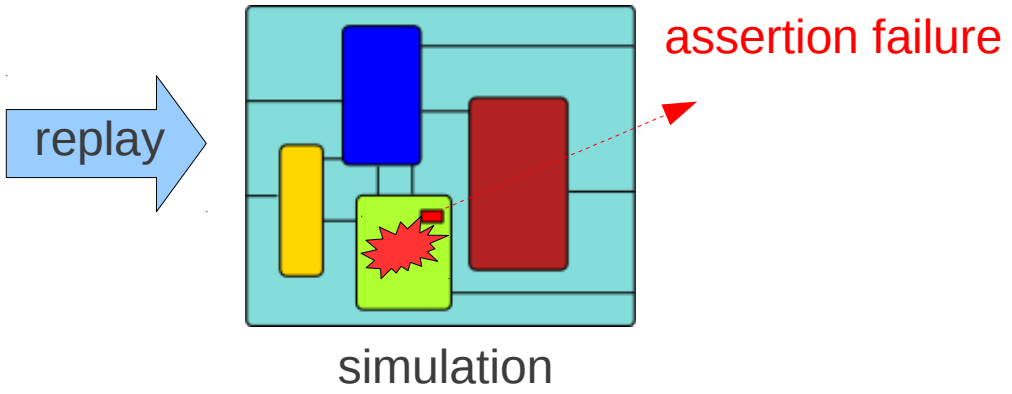
Ability to create quality assertions has a two-fold impact on Cyclic Design:

- Improved my internal verification and debug by quickly identifying both time and location of errors in simulation. Assertions also identified corner-case errors that do not propagate to testbench failures.
- More importantly, they improve my customer's experience when using the IP.

Example: FPGA Regression

During regression of Cyclic Design's ECC IP, I use an FPGA to run billions of correction operations.

When a test fails, I replay the vector in simulation. In nearly all cases, an assertion fired indicating the root cause of the failure, saving a tremendous amount of debug time and effort.



It should be noted that these were extreme corner-case bugs found only after billions of correction operations

Cyclic Design

Example: Assertions in IP

Cyclic Design's BCH ECC IP is parametrized to support different maximum ECC levels.

A customer configured the logic for their application for 60-bit ECC and ran the included testbench, which resulted in an error. The problem was that the testbench was running a test using 64-bit ECC.

Once assertions were enabled, three unrelated assertions immediately flagged this particular problem.

End-users should be educated of the values of assertions already present in the design and be instructed on how to enable them.

Cyclic Design

Customer Feedback

Did the assertions provide additional insights into the use of the signals that was not covered in the User's Guide?

“We had some assertion violations after integration, so we have modified design to clear those violations.”

Did the assertions flag real violations on the port interfaces or internals?

"We had at least one real violation flagged by assertion.”

“We fixed some issues that might have created failure at a later stage in regression.”

Overall, did you feel that the inclusion of the assertions represented additional value in the IP?

“Assertions are very powerful. It’s very effective to have in IP; apart from less debugging time, they give us more confidence.”

Cyclic Design