Using SystemVerilog Assertions and Zocalo Zazz to Improve IP Quality

Eric Deal
Cyclic Design
eric.deal@cyclicdesign.com

Overview

Design and verification engineers detect simulation problems and pinpoint failure points within a design using assertions. Despite the added value of SystemVerilog assertions (SVAs), barriers to implementing assertions have prohibited wider adoption throughout the industry.

In particular, IP providers and users could benefit greatly from the inclusion of SVA within the IP delivery. So why aren't more companies using assertions as part of their design flow?

The problem lies in the complexity of implementing assertions. Designers must learn new language constructs and overcome the daunting task of even knowing where to start.

What are Assertions?

Assertions are “rules” coded into a design that monitor the activity of signals to check for non-compliant behavior. They can be viewed as an enhancement of documentation that may (or may not) accompany a design. During simulation, these assertions are checked and violations reported to the user.

Example: When valid goes active (high), it must remain high until ready is asserted (or more simply: if valid is high and ready is not asserted, valid must remain high the following cycle):

```
valid_until_ready: assert property ( @(posedge clk)
    valid & ~ready |=> valid
);
```

Groups of these simple rules, relating how signals interact, can be used together to check complex protocols.

Assertion Use in Design Teams

Logic designers understand the implicit protocol rules used by low-level signals, so the use of low-level signals is often not formally documented. Assertions ensure the proper operation of the logic during testing and check that the logic continues to operate properly after design modifications.

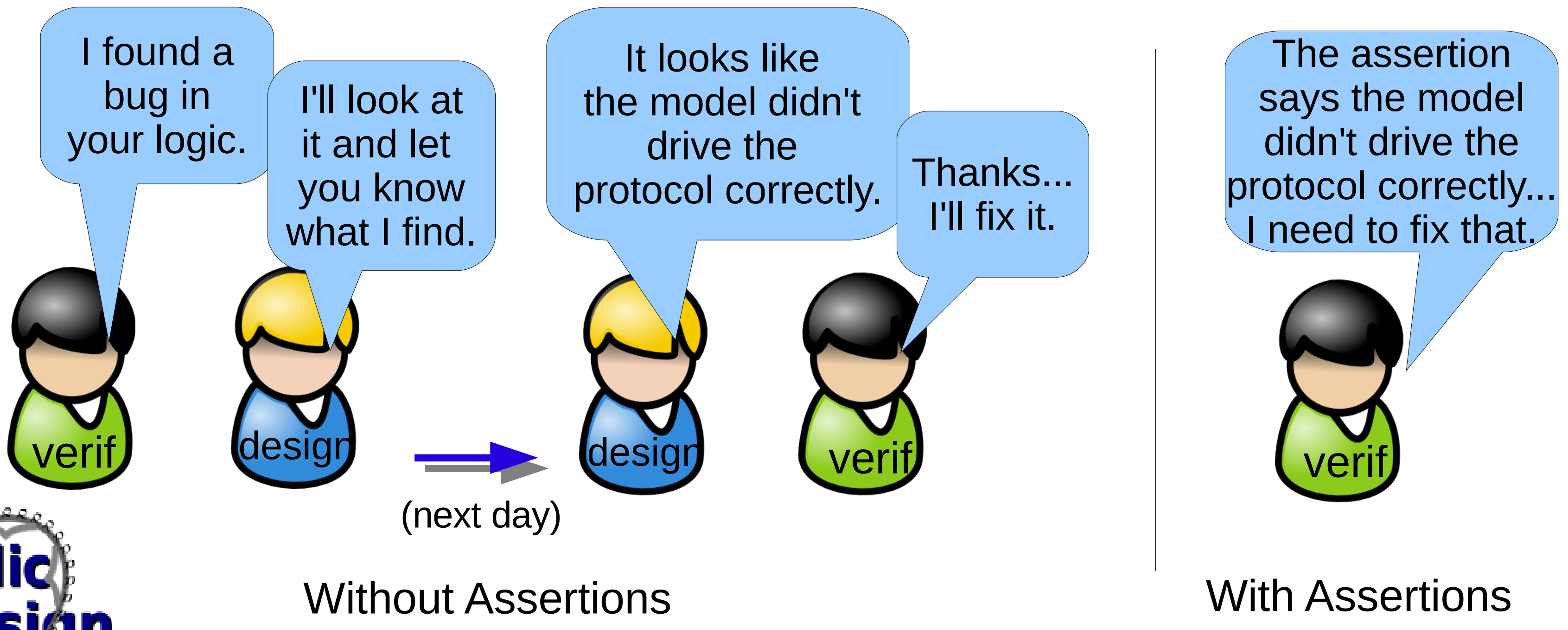
Assertions also help flag and identify bugs at the source, saving debug time. Furthermore, they can also identify issues that might not be detected by a particular test.

Assertions allow verification engineers to more quickly diagnose test failures and issues resulting from behavioral models which violate minimally documented protocols.

More Efficient Development

Ultimately, assertions reduce communication time between design and verification engineers, speeding development by:

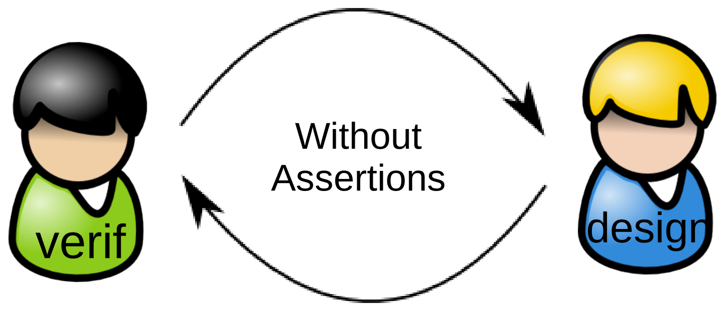
- Isolating design bugs more quickly
- Identifying and fixing testbench behavioral issues with less interaction between design and verification.



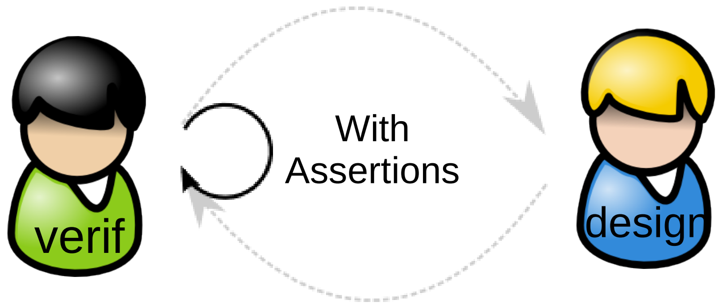
I found a bug in your logic.
verif
I'll look at it and let you know what I find.
design
(next day)
It looks like the model didn't drive the protocol correctly.
design
Thanks... I'll fix it.
verif
Without Assertions
The assertion says the model didn't drive the protocol correctly... I need to fix that.
verif
With Assertions

Communication Loop

Interactions between the designer and user can be viewed as a communication loop.

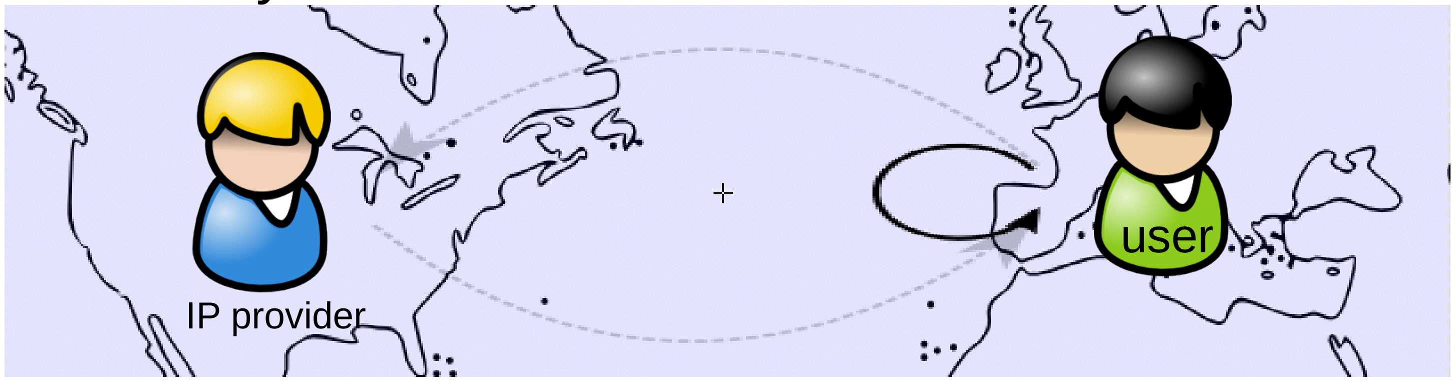


Assertions allow the loop to be short-circuited for a large subset of interactions, improving design and verification efficiency.



IP Has a Long Communication Loop

For IP, the designer and user are likely separated by different locations, timezones, companies, and possibly language barriers. Any reduction in round-trip communications required saves a lot of time and money.



Assertions accomplish this by notifying the IP user of an integration error immediately. It's like having an on-site application engineer watching every simulation.

The customer benefits from improved IP quality, and the IP provider benefits from reduced support requests and more satisfied customers.

Enabling Assertions

So what can be done to empower assertion usage within IP?

Zocalo approached Cyclic Design to get feedback on an upcoming product called Zazz, which aimed to ease the use of assertions.

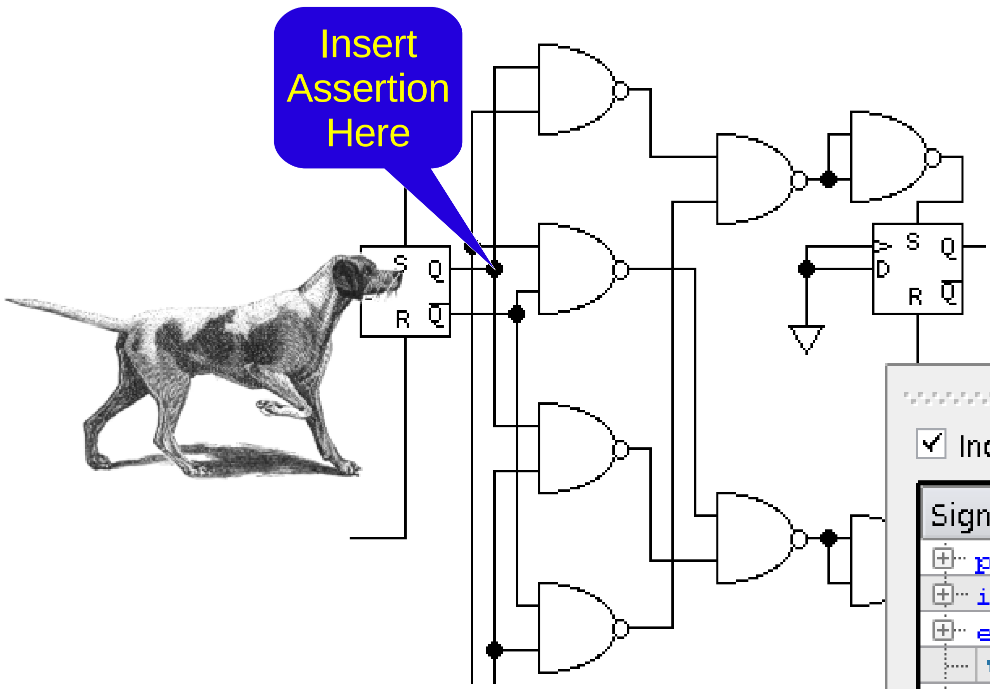
The resulting collaboration between Zocalo and Cyclic Design to identified and solved two major problems designers have with implementing SystemVerilog assertions:

- Where to start
- How to code assertions

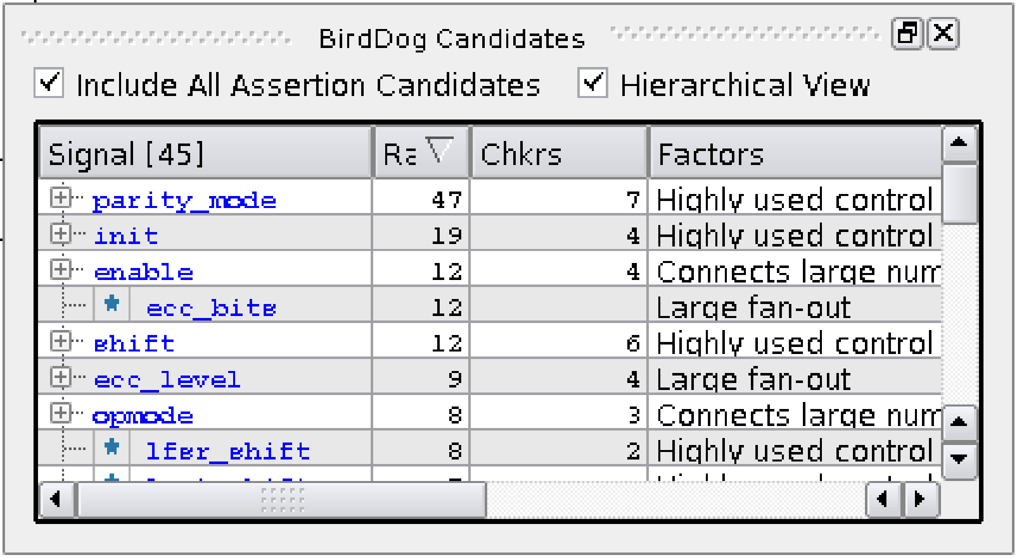
Cyclic Design

Zazz - BirdDog

Where to start: Assertion placement is difficult for many engineers. A Zazz feature called Bird Dog identifies and ranks candidate signals for assertions.



Based on initial manual assertion placement in the Cyclic Design IP, we developed an algorithm that mimics, and improves upon, the process an engineer would use to identify candidates for assertions.



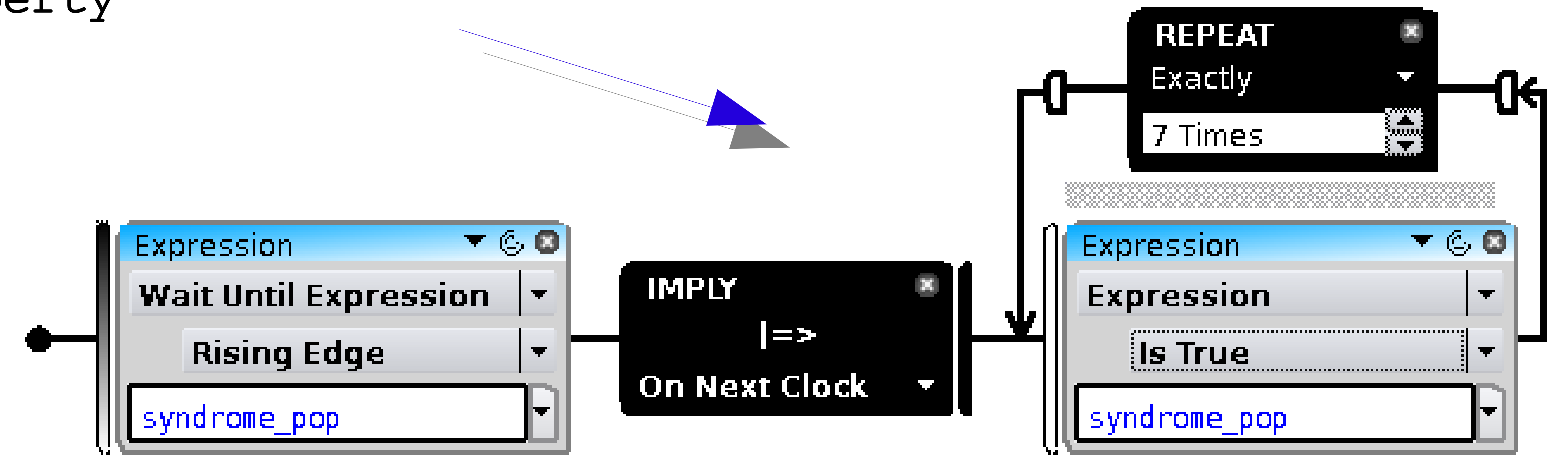
[✓] Include All Assertion Candidates [✓] Hierarchical View

Signal [45]	Ra	Chkrs	Factors
parity_mode	47	7	Highly used control
init	19	4	Highly used control
enable	12	4	Connects large num
ecc_bits	12		Large fan-out
shift	12	6	Highly used control
ecc_level	9	4	Large fan-out
opmode	8	3	Connects large num
lfsr_shift	8	2	Highly used control

Zazz - Visual SVA

How to code assertions: While discussing assertion coding complexity, we realized that diagrams on a whiteboard most effectively illustrate the complex event structure of SVAs. Why not incorporate this functionality into a design tool?

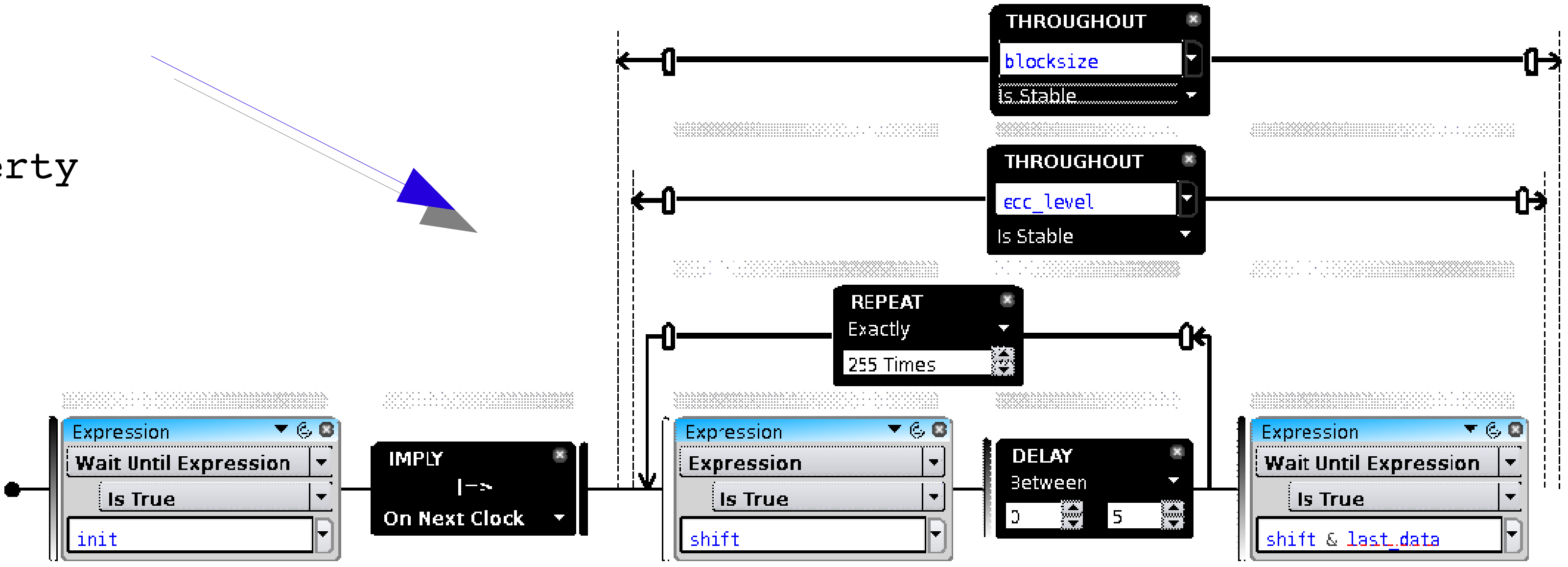
```
property userSVA;
 (   $rose(syndrome_pop) [->1] )   |=>
    ( ( syndrome_pop   ) [*7]   );
endproperty
```



Zazz - Visual SVA

The Visual SVA Zazz feature enables the creation of very complex assertions without requiring expert knowledge of the SVA language.

```
property Stable_blocksize_and_ecc;
  ( init [->1] ) |=>
  ( $stable(blocksize) throughout
    (  $stable(ecc_level) throughout
      ( ((  (shift)  ##[0:5]  ) [*255]  )
        ( (shift & last_data) [->1] )
      )
    )
  );
endproperty
```



Conclusion

Zazz enabled Cyclic Design to incorporate more complex assertions into its IP and to identify areas where additional assertions could be applied.

Inclusion of these assertions has minimized the support required for the IP; customers need little integration support and are able to self-diagnose problems in the interface between the controller and IP.

The assertions have identified several integration issues that would have taken much longer to discover or possibly not have been discovered manually. Ultimately, Cyclic Design's customers indicate that the assertions gave them more confidence in both the IP and their use of it.