VCS Xprop and SimXACT are Complementary, Not Mutually Exclusive!

Kai-hui Chang
Avery Design Systems, Tewksbury, MA, USA

Abstract

Both VCS Xprop and SimXACT deal with X-issues. However, the issues they address are orthogonal and the tools are complementary instead of mutually exclusive: the former eliminates X-optimism in RTL simulation, allowing real Xs to propagate in RTL simulation at the expense of possible false alarms. The latter eliminates X-pessimism in gate-level simulation so that X propagation becomes hardware accurate and no unreal, false Xs can propagate. In this paper we compare how they are different and explain why a complete verification methodology and its outcomes can benefit from employing both methods.

1. Background

In this section we provide necessary background for understanding the X issues that different tools solve by explaining what X optimism and X pessimism are.

1.1 X-Optimism Problems

X optimism occurs when logic simulation produces a non-X simulation result while in real hardware the value can be an arbitrary non-deterministic value, typically denoted as X. The following is a simple example:

```
always @(posedge clk)
    if (cond)
        b <= 0;
    else
        b <= 1;
```

Assume that “cond” is X, in logic simulation according to Verilog standard, only the else branch will be executed, and “b” will become 1. But in real hardware “b” can be either 0 or 1, which means “b” should really evaluate to an X. In this case an X is masked, thus the name X-optimism: simulation result is too optimistic about the X in that a non-X value is produced instead of an X.

X optimism typically occurs in if/case constructs and is a serious problem in RTL simulation that uses these construct because real issues can be masked during RTL simulation and only found in gate-level simulation or at Silicon. Additionally, any clock going 0X0X can also create X-optimism issues for @(posedge clk) because logic simulation can only assume the edge is valid and executes the statements under this event, while in real hardware “clk” may remain idle and no value propagation using this clock should occur.

1.2 X-Pessimism Problems

X-pessimism occurs when false Xs are created where simulation result should actually resolve to a known value. The following is a simple example:

```
assign o = s & a | ~s & b;
```

Assume that “a” is 1, “b” is 1 and “s” is X. In real hardware “o” should be 1. But logic simulation produces X instead because “X & 1” is X, “~X & 1” is X, and “X | X” is still X. X pessimism does not mask issues but creates false alarms that make design verification and debugging difficult. X pessimism issues can occur both in RTL and gate-level simulation but are more serious in gate-level simulation because almost all gates are modeled using primitives or continuous assignments that are prone to X-pessimism issues. RTL X-pessimism in simulation is common with logical and arithmetic comparison operators and unknowns in address and bit selects. On the positive side, because if/case statements are rarely used in gate-level netlists, Gate-level simulation does not suffer from X-optimism issues other than 0X0X transitions at clocks.

2. Solutions for Handling X Optimism and X Pessimism

As described in Section 1, X-optimism issues are more serious in RTL simulation and X-pessimism issues are more serious in gate-level simulation. As a result, X-optimism removal tools like VCS Xprop target RTL simulation while X-pessimism elimination tools like SimXACT target gate-level simulation.

2.1 X-Optimism Removal

To address X-optimism problems, X-optimism removal tools, such as VCS Xprop, have been developed and similar features are offered in all major simulators. VCS Xprop works by injecting Xs whenever Xs can potentially propagate. It has two different modes. In the pessimistic mode, whenever an if/case condition is X, every variable on the left-hand-side of assignments on all branches are corrupted to X. For the example shown in Section 1.1, “b” will always be corrupted to X no matter what values on the right-hand-side of the assignments are. In the more accurate mode, right-hand-side values are considered. For the example below, in the pessimistic mode X will be assigned to “b” when “cond” is X, while in the more accurate mode 0 will be assigned to “b”.

```
always @(posedge clk)
       if (cond)
           b <= 0;
       else
           b <= 0;
```

Needless to say, pessimistic mode can create a tremendous amount of false alarms because Xs that are not real are injected and propagated. Running simulation using this mode can catch most real X issues, but this may result in unnecessarily large RTL debugging time or sub-optimal final design due to added reset signals that address unreal X issues. As a result, most people use the more accurate mode where right-hand-side values are considered. However, unreal Xs can still be injected in this mode when if/case statements are nested or when the variable on the left-hand-side of an assignment in an if/case statement becomes the right-hand-side of assignments in another if/case statement. Additionally, 0X transitions at clocks due to clock gating as well as X-pessimism issues in RTL simulation can still create additional

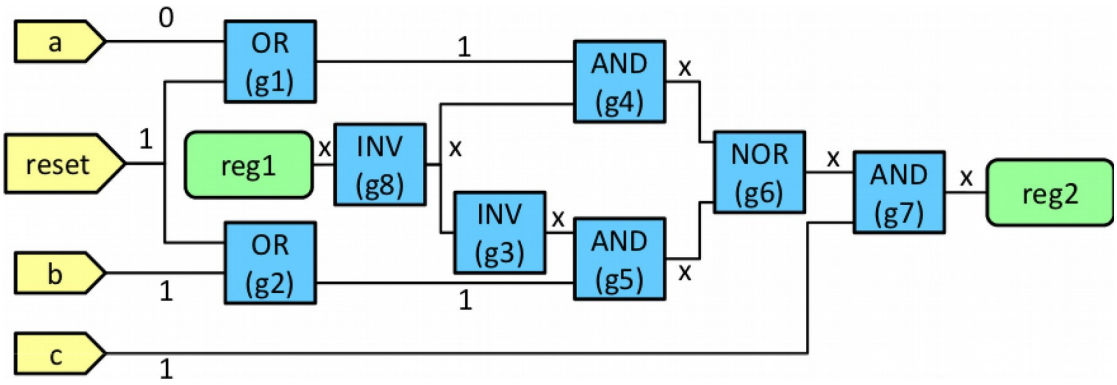
unreal Xs that makes debugging challenging. The end result is that VCS Xprop helps expose real X issues in RTL simulation but noise ratio due to false X propagations can be a serious drawback: if noise ratio is too high, designers may either add too many resets to eliminate the Xs which produces sub-optimal design, or spend too much timing analyzing the results thus missing project deadline. As a VCS Xprop user commented in ESNUG 580, “automatically adding RESETS to everything VCS Xprop doesn’t like will cause excessive power/size use in our design.” An even worse situation is designers may simply ignore the Xs by adding deposits without correct analysis that may mask real issues.

2.2 X-Pessimism Removal

SimXACT is a unique tool that uses patented formal analysis methods to eliminate all combinational false Xs in gate-level simulation thereby correcting simulation results so they match real hardware. It is a simulator add-on using the PLI interface that is easy to setup and use. The tool works as follows:

1. At the start time it checks Xs in register data inputs (typically denoted as “d”) to determine if they are false.
2. For each false X, it traces the fan-in cone of the register input to find a portion of the cone, called a sub-circuit, whose inputs are real Xs and whose output is a false X.
3. It then generates a false X elimination solution, called a fix, and auxiliary code based on the sub-circuit to eliminate such Xs.
4. The generated fixes are applied immediately and then throughout simulation to repair the incorrect simulation values whenever the same false X condition appears.
5. Throughout simulation, if the X at any register input has fanin change, it is reanalyzed to make sure no false Xs are missed.

The following example shows a circuit suffering from X-pessimism issues: reg2 should latch 0 instead of X.



SimXACT can prove that the X is false and identify a small portion of the logic that is responsible for the false X (gate g3, g4, g5 and g6). These gates form a MUX with output inverted and when both inputs are 1 and select (g8.o) is X, the output should be 0. It then generates a compact Verilog fix that eliminates the X and restore simulation value to the correct hardware value. The fix code is shown below:

```
always @(g1.o or g2.o or g8.o)
if (g8.o === 1'bx && g1.o === 1'b1 && g2.o === 1'b1) force
    g6.o = 1'b0;
else
    release g6.o;
```

After SimXACT eliminates all false Xs, remaining Xs are all real. If the Xs propagate to key signals and make tests fail, the issue needs to be debugged and fixed. Because all false Xs are eliminated, tracing the remaining Xs will lead directly to the real X sources, greatly reducing the debugging effort.

3. Myths about VCS Xprop

VCS Xprop and SimXACT solve different problems in different phases of a design flow. They are complementary to each other and both can help improve the design flow. There are misconceptions that using VCS Xprop alone can resolve all X issues in a design flow, which is not correct. In this section we explain why VCS Xprop alone is not sufficient.

3.1 Myth 1: Using VCS Xprop Requires Little Effort

It is true that turning on VCS Xprop is simple and using it is no different from running simulation with a modified X-handling algorithm. However, analyzing the result is not as simple as one may expect.

VCS Xprop resolves X-Optimism issue by injecting Xs whenever Xs can potentially occur, which tends to create serious X-pessimism issues in RTL simulation and can generate a tremendous amount of false Xs that make RTL simulation results difficult to analyze. This high noise ratio, alone with performance degradation in simulation, are the two most common issues that designers report when they use VCS Xprop in their RTL simulation.

3.2 Myth 2: VCS Xprop Resolves X-Pessimism in Gate-level Simulation

The biggest misconception regarding VCS Xprop is that people think if RTL simulation passes with VCS Xprop, there is no need to use SimXACT because there will not be any false X propagation in gate-level simulation. This is not correct. Take the following simple RTL code as an example:

```
always @*
    if (sel)
        o = d1;
    else
        o = d2;
```

After logic synthesis, the MUX might become the following gates:

```
or g1(o, w1, w2);
and g2(w1, sel, d1);
not g3 (sel_n, sel);
```

```
and g4(w2, sel_n, d2);
```

If d1=d2=1 and sel=X, in RTL simulation VCS Xprop propagates 1 at “o”, which matches hardware value. In gate-level simulation, because both “w1” and “w2” are Xs, simulation value for “o” remains X, which is incorrect and is a false X.

In this case, RTL simulation works perfectly fine without any X propagation, but gate-level simulation for the same logic will fail due to a false X!

Another reason why VCS Xprop alone cannot solve X-pessimism issues in gate-level simulation is that in addition to real X issues from RTL code, there can be other sources of Xs in gate-level simulation. For example, powered off blocks in power-aware simulation when UPF is applied will produce Xs. Xs from third-party hard IPs will show up in gate-level simulation as well. Since gate-level simulation is the last guard against X issues before tapeout, a flow that contains SimXACT to effectively address X issues in gate-level simulation is essential for design correctness.In summary, here is a list of X sources that can generate false Xs in gate-level simulation that VCS Xprop cannot resolve:

1. Logic/physical synthesis induced false Xs involving reconvergent path along multiple library cell instances
2. Netlist ECO modifications that can change how Xs propagate
3. Xs from UPF constructs
4. Xs from DFT constructs in post-layout netlist
5. 3rd party hard IPs that only exist in netlist form

3.3 Myth 3: VCS Xprop Handles All X-optimism Issues at the RTL

VCS Xprop handles most Verilog constructs that suffer from X-optimism issues like if/case statements. However, not all such constructs are handled, and X bugs can still escape RTL simulation even when Xprop is used.

One such construct is “var1[idx] = var2” when idx is X. In this case, an arbitrary bit in var1 will be updated to var2 because idx is X. As a result, for a bit in var1 that has a value different from var2, its value should be X. However, Verilog simulator does not corrupt var1 to X even when Xprop is used. As a result, X bugs can still escape.

4. Safe X Handling Flow with SimXACT

Typically one can use SimXACT without VCS Xprop because all missed X issues after regular RTL simulation can always be caught in gate-level simulation. But this flow is practical only when SimXACT is applied because unless false Xs are removed, the noise ratio due to false Xs in gate-level simulation can be too high for the verification team: without SimXACT to eliminate the false Xs, the number of Xs that exist in gate-level simulation can be overwhelming. The net result is that the team may choose to ignore the Xs by doing some form of deposits to all DFFs, which can miss real issues. Or the team nay need to spend a large amount of time trying to come up with manual deposits after analyzing the X sources and determine that the Xs are harmless. Adding SimXACT into gate-level simulation flow solves these issues.

4.1 Advantages of Using SimXACT

SimXACT has the following advantages that make it easy to use:

1. Formally and comprehensively eliminates all combinational false Xs in gate-level simulation without any false alarm and zero noise
2. Handles any X source in netlist including Xs from UPF constructs, DFT constructs, 3rd party IPs, ECO changes and logic/physical synthesis optimizations
3. Easy to use: is a simulator add on and little engineer involvement is needed other than simple makefile and testbench modifications
4. Cleans all false Xs; if Xs do cause issues, the Xs are real and debugging will lead directly to the real X sources
5. Accurately addresses clock-gater induced X-pessimism (0X0X at clock) at each FF that is difficult for VCS Xprop

In addition to resolving false Xs, SimXACT also offers a rich set of features for bringing up gate-level simulation including: pseudo-SDF for resolving racing conditions in 0-delay simulation; XTDB formal-based X backtracing including Verdi XTDB Viewer App for interactive debugging; selective deposit methods based on module name, design hierarchy or cell type for quick simulation bring up for sanity check; and checks for common X problems including undriven nets, SUDP modeling issues, X glitches, etc. These features are not provided by existing simulators or waveform viewers.

4.2 Using VCS Xprop and SimXACT Flow

VCS Xprop is useful in a design flow because it can help designers find real X issues earlier in the design flow. By resolving real X issues earlier, fewer issues will escape to gate-level simulation. This also helps a flow with SimXACT because when real X issues do escape, they become real Xs in gate-level simulation and SimXACT needs to spend extra effort formally analyzing the Xs to verify if the Xs are false or not. By applying VCS Xprop in addition to SimXACT, design verification can be performed more smoothly. As a result, VCS Xprop and SimXACT are complementary instead of mutually exclusive.

5. Conclusions

VCS Xprop and SimXACT both address X issues but have different purposes and are for different stages in a design flow. While VCS Xprop can help identify real X issues earlier in the design flow and reduce the number of real Xs escaping RTL design phase, it does not prevent false Xs from being generated in gate-level simulation. Synthesis optimizations, UPF constructs, third party IPs, ECO change as well as powered-down blocks in gate-level simulation can all create false Xs that do not exist in RTL simulation. Without SimXACT to eliminate false Xs, debugging Xs in gate-level simulation will always be a challenging, error-prone and time-consuming task.