REAL INTENT
Accelerating RTL Sign-off

A Practical Solution to Fixing Netlist X-Pessimism

Most functional verification for SoC and FPGA designs is done prior to RTL hand-off to digital synthesis, since gate-level simulations take longer to complete and are significantly harder to debug. However, gate-level simulations are still needed to verify some circuit behavior. Ideally, the output of the RTL simulations will match the output of gate-level netlist simulations on the same design after synthesis. And why wouldn't they? Besides the obvious things that are being verified in your gate-level simulations, there are also unknown values (X's) that were not seen in RTL due to X-optimism, and additional X's in the gate-level simulations due to X-pessimism. This paper focuses on the issues of X-pessimism at the netlist level. X-pessimism is described, current solutions are discussed, and an Ascent XV- Netlist solution is presented.

X-pessimism and X-optimism Defined

The presence of X's can cause both X-optimism in RTL simulations and X-pessimism in netlist simulations. X-optimism can result in the failure to detect functional bugs at RTL. X-pessimism typically makes it hard to get netlist simulations up and running quickly.

X-pessimism occurs in gate-level designs when an X signal at the input of some digital logic causes the simulation output to be an X value, even though in real hardware the value will be deterministic, e.g. a 1 or 0. Figure 1 shows a very simple example. When the value of **in1** and **in2** are both 0, the simulation value at the output is 0, as it would in hardware. But when the "input" value is an X, and the values of **in1** and **in2** are both 1, the simulation value at the output is X in simulation but is 1 in real hardware. This behavior is called X-pessimism because the known value simulates as an unknown. More specifically, we say it is 1-pessimistic because the output should have been a value of 1.

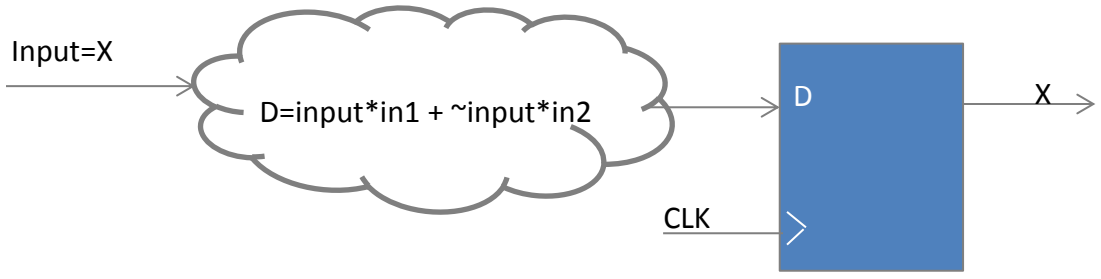

Figure 1. X-pessimism example showing an X results when the values of signals in1 and in2 are both 1.

X-optimism is the opposite, when an unknown value is simulated as though it is a known value in hardware. Consider the example shown in figure 2 below. If the "input" signal is an X value,

that means that “input” could be either a 0 or a 1 value in real hardware because real hardware does not have an X value. So in real hardware, signal “D” might also be a 0 or a 1 value. However, in simulation, the output “D” would always show as a 1 value. It is called “optimism” because the unknown was resolved as a known value. This can cause functional bugs to be missed in RTL simulations, though in netlist the X would always be properly propagated.

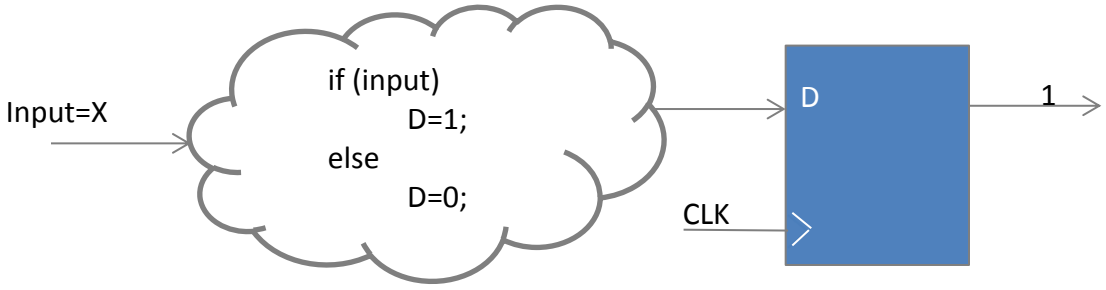

Figure 2 X-optimism example showing an input value of X produces a 1 result.

The above are very simplistic examples. In real designs the logic cone of the “cloud” is often very complex, with the selected output being driven by a logic expression and the “input” also being a logic expression. In this case, sometimes the output X value is a result of pessimism and sometimes it is simply the propagation of an X that was optimistic in RTL. One can be artificially corrected without harm, but the other could be a bug lurking to be discovered. Refer to “X-Propagation Woes – A Condensed Primer”[1] for more details.

Commonly Used Quick Fixes

There are three approaches that we hear being used for addressing X-pessimism, all with downsides. The approaches are 1) eliminating all X’s in the design, 2) artificial random initialization of uninitialized flops, and 3) manual drudgery.

The first common approach for eliminating all X’s is to add a reset to all memory elements. This eliminates the most common source of Xs – uninitialized flops. However, synchronous resets can sometimes cause pessimism issues to be introduced during the synthesis process. Also, other sources of X’s do exist in a design, such as explicit X-assigns for flagging illegal states, bus contention, and out of range references, among others. A more significant issue is that extra resets eat into power, size, and routing budgets. Resettable flops are larger, more power hungry, and require additional routing resources. Resetting all flops is practical only for smaller designs, and does not address all sources of X.

The second common approach is to artificially randomize the initialization of uninitialized memory elements at time 0. The issue with this approach is that while it will help simulations, it will not necessarily match real hardware. It also does not address all sources of X in a design. X-optimism bugs that were masked in RTL will likely be artificially masked again in netlist simulations. This risk of missing a bug that was masked by optimism in RTL and artificially

removed at netlist, may result in a critical bug being missed. These days, with the high costs of manufacturing and heaven forbid, recalls, the downside for a failure can be very expensive.

The third approach is to manually analyze the root cause of all X-differences between the outputs of RTL and gate-level simulation, and then determine the correct value and time to force and release pessimistic nodes. This can be very difficult to do because a gate-level design is a transformation of an RTL design into something that is more complex, has a different structure, and has unfamiliar signal names. It can be very time consuming to chase down those pessimistic X's in the netlist simulation. The issue is exacerbated when there is a mixture of X differences from both optimism and pessimism, with pressure to get it fixed as soon as possible and not accidently miss a real bug. For large designs, this effort can take months.

Existing approaches fall short in that they do not address pessimism caused by X's from all sources, they can mask issues in the gate-level simulations due to X-optimism in RTL simulations, and they are insufficient to handle the largest SoC designs.

Introducing Ascent XV – Netlist

If a node is determined to be 1(or 0) pessimistic, that means its real circuit value is 1(or 0), but simulation produces an X. A pessimistic simulation value can be corrected by forcing a 1(or 0) on the node until the conditions for pessimism no longer hold, at which time, it is released. This does not mean that all X's can be arbitrarily forced to a known value. Only X's that result from pessimism should be forced, and they must be forced to represent the deterministic value that real hardware would see and released immediately when the pessimism stops.

Ascent XV-netlist makes your simulation hardware accurate by appropriately correcting pessimism. Ascent XV statically identifies the potentially pessimistic nodes and then uses that information to create SimPortal files that augment gate-level simulation to correct X-pessimism on the fly. By doing the analysis statically before the simulation starts, the number of nodes that must be analyzed during simulation is significantly reduced. Also, the X-analysis during simulation can be reduced to a table look-up when the potentially pessimistic node has an X-value. The SimPortal files monitor the potentially pessimistic nodes in the design on the fly, independent of the testbench.

A bottoms-up hierarchical static analysis can also be done at the block level. When all the blocks are integrated for full chip simulations, a very scalable solution is achieved. The SimPortal is designed for performance, and also minimizes compile time and memory overhead. You can control the verbosity at simulation time, and can choose to drop back to simple monitoring or even turn off both the correction and monitoring at any point in time. The flow and methodology is shown below in Figure 3.

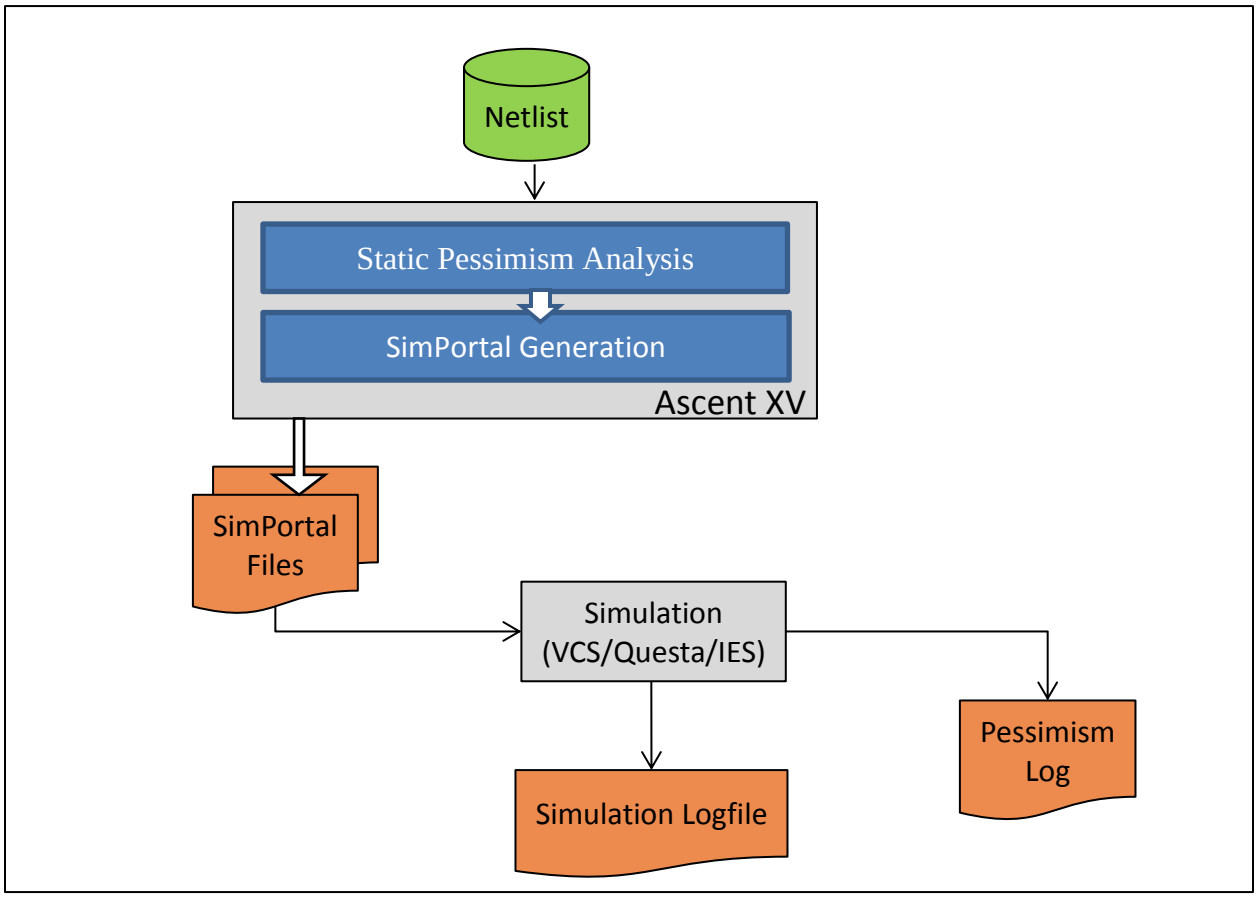

Figure 3 Ascent XV X-Pessimism Flow and Methodology

Ascent XV X-pessimism Flow and Methodology:

1. Run static analysis to determine which data input values can cause monitored nodes to exhibit pessimism. Generate design-specific SimPortal data files.
2. Run SimPortal simulation to find out which nodes experienced the input combinations that cause pessimism.

The Ascent XV solution is characterized as follows:

- Performance
  - Gate-level simulation overhead is as low as 2x-2.5x
  - Memory overhead is 0.5x
  - Negligible overhead to compilation time
  - Simulation time configuration of verbosity from totally quiet to full details
  - Can choose to turn off correction at any point in time (such as after reset)
- Capacity
  - Unique approach easily handles next generation full chip netlists (billion gate SOCs)
- Accuracy
  - Only does forces when pessimism is occurring.

    - The value forced is the value that will be seen in real hardware.
- Ease of Use
    - No setup required
    - Testbench independent static analysis
    - No need to touch the existing design or testbench, only the simulation script

In RTL, X's can hide functional bugs due to X-optimism. These bugs will be brought to light in netlist gate simulations. Unfortunately, X's also cause X-pessimism in netlist simulations, making it difficult to determine whether a functional mismatch is due to X-optimism, X-pessimism, or something else entirely. Ascent XV – Netlist will remove X's caused by X-pessimism, removing the major source of simulation RTL and netlist simulation differences.

Related Considerations

Reliable correcting of pessimism at the netlist has become very feasible, thanks to Ascent XV. But there is additional analysis that can be done early in the RTL development process to prevent potential X-issues. This will benefit the post-RTL handoff, whether it is gate-level simulations or FPGA modeling of your design, so you are not debugging X-optimism issues in hard to debug environments.

Ascent XV- Reset Optimization minimizes significant X's in the design that result from incomplete initialization. Ascent XV – Reset Optimization will do a hardware accurate reset analysis that will report where additional resets are needed; as well suggest where resets can be removed. It ensures complete initialization, taking into account the propagation of known values to avoid adding extraneous resets. The goal is to minimize X-issues during the design of the RTL. In the case of simulations, fewer occurrences of pessimism will speed your simulation.

Ascent XV-RTL Optimism analyzes where the X-sources of a design are, as well as where they can cause X-optimism. This ensures hardware accurate simulations at RTL either by eliminating the X-source, or through coding for X-accuracy. Hardware accurate RTL simulations will make the RTL-netlist simulation outputs easier to compare, but more significantly, it will make FPGA-based modeling easier to get up and running.

Summary

Once a design is synthesized, the immediate goal is to get gate-level simulations up and running fast. Unfortunately, X's in gate-level simulations can cause differences in the RTL simulation output and the gate-level simulation output. X's generally exist in all designs – it can be difficult to prevent this for practical reasons. Simulation results may be different because of X's that are hidden in the RTL simulation by X-optimism, or additional X's may exist due to X-pessimism in gate-level simulations. Pessimism can be fixed by overriding the simulator because you know that real hardware would always resolve to a deterministic value. The challenge is confirming that the X value is a result of X-pessimism and not simply X-propagation, and then forcing it to the right value at the right point in time so the simulation matches that of real hardware.

Ascent XV- Netlist Pessimism corrects X-pessimism on the fly so the simulation is hardware accurate. Use of Ascent XV saves in the time required to get gate-level simulations started by an order of magnitude. It is proven to be superior to alternative approaches in the marketplace in terms of performance, memory, and accuracy. Its ease of use and capacity, make it the only practical solution for large SOCs.

Author

Lisa Piper is Senior Manager of Technical Marketing at Real Intent.

References

1. X-Propagation Woes – A Condensed Primer. Lisa Piper, Real Intent.
2. X-Propagation Woes: Masking Bugs at RTL and Unnecessary Debug at the Netlist. Lisa Piper, Vishnu Vimjam. DVCon 2012.