BlueCheck: Frequently Asked Questions

Matthew Naylor, University of Cambridge

February 1, 2016

1. Can I view the test-sequences that BlueCheck is generating?

In simulation, yes: run your executable with the `+chatty` command-line option.

2. Is iterative-deepening/shrinking ever undesirable?

Yes, if the design-under-test does not properly reset to a consistent state when its reset signal is asserted, possibly resulting in false counter-examples. For more details, see the section about resettable specifications in Section II(E) of the paper. In such cases, the `blueCheck` test bench should be used, not `blueCheckID`.

Alternatively, the design-under-test should be modified to reset itself properly. Examples of components that do not reset "by themselves" are: `mkRegU` (make uninitialised register), and `mkRegFile` (make uninitialised register file). If the correctnesss of a module can be affected by changing the initial contents of a register file, then that module needs to be modified to initialise the contents explicitly. This can be done by adding an initialisation rule to the module or using a BlueCheck `pre` function to establish some initial state before each test-sequence.

3. Can I replay counter-examples found on a previous run?

Yes. In simulation, when a counter-example is found, BlueCheck saves it to a file `State.txt`. When the test bench is run with the `+replay` or `+resume` command-line options, BlueCheck will resume testing from the point at which the counter-example was found.

On FPGA, the contents of `State.txt` is produced over a UART. The first character is `1` if a counter-example was found and `0` otherwise. Counter-examples can be viewed using the `+view` option or replayed using `+replay`. Of course, a failure on FPGA may not correspond to a failure in simulation if there are any hardware components that are not accurately modelled in simulation, e.g. DRAM.

4. Why did replaying a counter-example not reproduce my bug?

When in iterative-deepening mode and the design-under-test does not properly reset itself, BlueCheck can report false counter-examples. See answer to Question 2.

5. Is BlueCheck configurable at all?

BlueCheck is configurable in various ways using the `BlueCheck_Params` structure. See `BlueCheck.bsv` for details of this structure and the `testStackIDCustom` module in `StackExamples.bsv` for an example of how to configure the structure. Note that not all combinations of configuration options are supported, e.g. shrinking is only possible in iterative-deepening mode.

6. Does shrinking work with wedge failures?

No. Iterative-deepening, on the other hand, is ideal for finding simple wedges. However, Andy Wright has a fork of BlueCheck (`acw1251` on github) which does support this feature.

7. How are values for an `enum` or `tagged union` type generated?

By generating a random bit-string and applying it to the `pack` function.

As Andy Wright pointed out to me: "Be careful using the default RNG for enumerations where the number of elements is not a power of two. The RNG will produce random values that are not actually elements in the enumeration. This can be very confusing because if you are deriving `FShow` to print the names of the elements, it will have a name that makes it look like it some element in the enumeration, but when you use the `==` operator from deriving `Eq`, it will not match that element. Instead you can derive `Bounded` for the enumeration and use an RNG that uses the `Bounded` typeclass."