Library to generate random number and PRBS (Pseudo-Random Binary Sequence)

The cores in this library use the psC language. This C++ like language abstracts the low-level languages VHDL or Verilog, as the C++ abstracts the assembly language. Whatever you can code with VHDL or Verilog can be coded using psC, with all the advantages of a high-level language.

The library is available on FreeCores.

Library description

Testing communication systems and digital processing systems requires random numbers. This library includes two cores. The first generates a PRBS (Pseudo Random Binary Sequence) based on LFSR (Linear Feedback Shift Registers) and the second generates 32 bits random numbers. The second core is based on a known random number generator algorithm, ***xoroshiro128+ 1.0***.

Choosing

There are two criteria for selecting a random number generator: resources usage and quality of random numbers.

- PRBS is simpler and uses less resources.
- Random (xoroshiro128+ 1.0) provides better formal qualities but uses more resources.

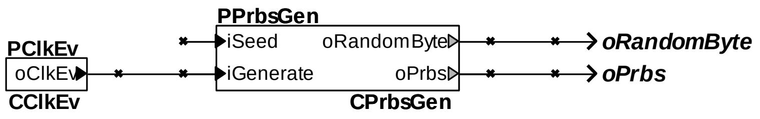
Test bench

Basically, the cores need a clock to generate a value. The test will consist in providing the clock, running the simulation, and observing the resulting sequences.

The psC test programs

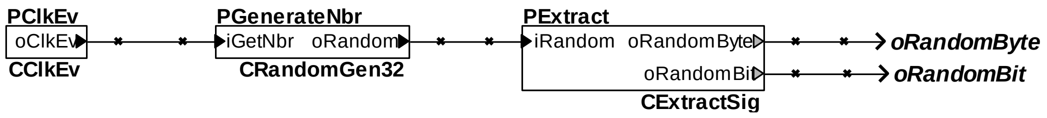
PRBS generator code

The test program consists of two components, a clock generator ***CClkEv***, and the core to be tested. The clock sends an event at every step. On each event on ***iGenerate***, a new value ***oRandomByte*** and a new bit ***oPrbs*** are produced.



Random number generator code

The test program consists of thre components, a clock generator ***CClkEv***, the core to be tested and a component to extract a bit sequence and a byte sequence. This last component is not necessary, but will help making the comparison between the two algorithms. As before, the clock sends an event at every step. On each event on ***iGetNbr***, a new 32 bits random value ***oRandom*** is produced. The last component extracts a bit ***oRandomBit*** and a byte value ***oRandomByte***.



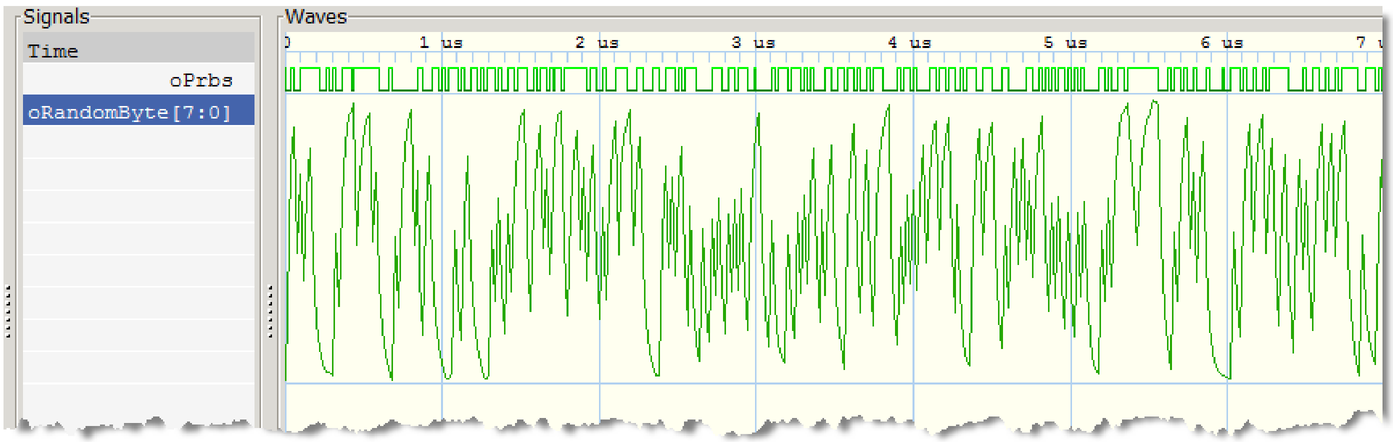
Compiling and running the application

You can compile and execute both programs in the same manner, then use the ***Signal Viewer*** to look at the random signals.

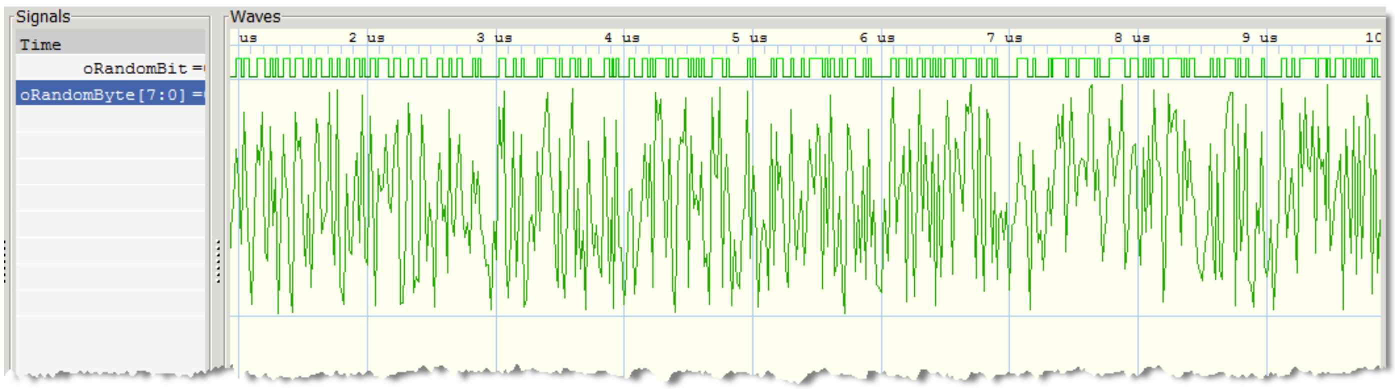
1) Start Novakod Studio with a double-click on the ***main.prj***.
2) Select the menu ***Run→ Run N Steps*** to simulate.
3) Double-click on ***targets\Simulation.evo*** to start the **Signal Editor**.
4) Select menu ***File→ View Signals*** or click the shortcut . The GTK Wave windows appears. GTK Wave has been pre-configured to show the desired signals.

The signals for both random number generator cores are:

PRBS signals (CPrbsGen)



Random signals (CRandomNumbers)


Signals
Time
oRandomBit =
oRandomByte[7:0] =
Waves
2 us
3 us
4 us
5 us
6 us
7 us
8 us
9 us