ClockResetPkg Package User Guide

User Guide for Release 2024.11

By

Jim Lewis

SynthWorks VHDL Training

Jim@SynthWorks.com

http://www.SynthWorks.com

Table of Contents

1 ClockResetPkg Overview

ClockResetPkg provides testbench utilities for creating clock and reset.

2 CreateClock

Create a clock with a specified period and specified duty cycle. ClkActive is the active edge value. Offset is the delay from time 0 at which the active edge value happens. Designed so that after the first clock transition that Clock always transitions on delta cycle 0. Designed so that the resulting clock period will match the Period parameter and that any rounding due to DutyCycle and simulator resolution will only impact the effective DutyCycle.

```
procedure CreateClock (
  signal   Clk            : inout std_logic ;
  constant Period         : in    time ;
  constant DutyCycle      : in    real := 0.5 ;
  constant Offset         : in    time := 0 sec ;
  constant ClkActive      : in    std_logic := CLK_ACTIVE
) ;

procedure CreateClock (
  signal   Clk            : inout std_logic ;
  signal   Enable         : in    boolean ;
  constant Period         : in    time ;
  constant DutyCycle      : in    real := 0.5 ;
  constant Offset         : in    time := 0 sec ;
  constant ClkActive      : in    std_logic := CLK_ACTIVE
) ;
```

3 CreateJitterClock

Create a clock in a similar fashion to CreateClock, but have a 10% jitter in the clock period. The Jitter is controlled by CoverID and the bins in CoverID can be changed after time 0 sim cycle 1.

```
procedure CreateJitterClock (
  signal   Clk            : inout std_logic ;
  signal   CoverID        : inout CoverageIdType ;
  constant Name           : in    string ;
  constant Period         : in    time ;
  constant DutyCycle      : in    real := 0.5 ;
  constant Offset         : in    time := 0 sec ;
  constant ClkActive      : in    std_logic := CLK_ACTIVE
) ;
```

4 Creating Reset

Create reset that is asserted tpd after the first active edge of clock and deasserts tpd after the clock edge that is Period in time from the first active edge of clock. ClkActive is the active edge value – its default is '1' (rising edge).

```
procedure CreateReset (
  signal   Reset       : out std_logic ;
  constant ResetActive : in  std_logic ;
  signal   Clk         : in  std_logic ;
  constant Period      :     time ;
  constant tpd         :     time ;
  constant ClkActive   : in  std_logic := CLK_ACTIVE
) is   . . .
```

5 Checking the Clock Period

The procedure CheckClockPeriod check that the clock period matches the Period parameter. It checks the first HowMany clocks.

```
procedure CheckClockPeriod (
    constant AlertLogID : in  AlertLogIDType ;
    signal   Clk        : in  std_logic ;
    constant Period     : in  time ;
    constant ClkName    : in  string := "Clock" ;
    constant HowMany    : in  integer := 5 ;
    constant ClkActive  : in  std_logic := CLK_ACTIVE
  ) ;

  procedure CheckClockPeriod (
    signal   Clk        : in  std_logic ;
    constant Period     : in  time ;
    constant ClkName    : in  string := "Clock" ;
    constant HowMany    : in  integer := 5 ;
    constant ClkActive  : in  std_logic := CLK_ACTIVE
  ) ;
```

6 Checking the Reset

LogReset creates logs when the Reset signal changes.

```
procedure LogReset (
  constant AlertLogID  : in  AlertLogIDType ;
  signal   Reset       : in  std_logic ;
  constant ResetActive : in  std_logic ;
  constant ResetName   : in  string := "Reset" ;
  constant LogLevel    : in  LogType := ALWAYS
) ;

procedure LogReset (
  signal   Reset       : in  std_logic ;
  constant ResetActive : in  std_logic ;
```

```
        constant ResetName   : in  string := "Reset" ;
        constant LogLevel    : in  LogType := ALWAYS
      ) ;
```

7 About ClockResetPkg

ClockResetPkg is part of the OSVVM Utility library.

ClockResetPkg was created by Jim Lewis of SynthWorks. Please support our work by buying your VHDL Training from SynthWorks.

ClockResetPkg.vhd is a work in progress and will be updated from time to time. Caution, undocumented items are experimental and may be removed in a future version.

8 Compiling and Using OSVVM Utility Library

Reference all packages in the OSVVM Utility library by using the context declaration:

```
    library OSVVM ;
      context osvvm.OsvvmContext ;
```

Compilation order for OSVVM Utility Library is in OSVVM_release_notes.pdf. Rather than learning this, we recommend using the OSVVM compilation scripts.

OSVVM Utility library is released under the Apache open source license. It is free (both to download and use - there are no license fees). You can download it from osvvm.org or from our development area on GitHub (https://github.com/OSVVM/OSVVM).

If you add features to the package, please donate them back under the same license as candidates to be added to the standard version of the package. If you need features, be sure to contact us.

We also support the OSVVM user community and blogs through http://www.osvvm.org. Interested in sharing about your experience using OSVVM? Let us know, you can blog about it at osvvm.org.

9 About the Author - Jim Lewis

Jim Lewis, the founder of SynthWorks, has thirty plus years of design, teaching, and problem solving experience. In addition to working as a Principal Trainer for SynthWorks, Mr Lewis has done ASIC and FPGA design, custom model development, and consulting.

Mr. Lewis is chair of the IEEE 1076 VHDL Working Group (VASG) and is the primary developer of the Open Source VHDL Verification Methodology (OSVVM.org) packages. Neither of these activities generate revenue.

Please support our volunteer efforts by buying your VHDL training from SynthWorks.

If you find bugs these packages or would like to request enhancements, you can reach me at jim@synthworks.com.