SynthWorks

VHDL Training Experts

TbUtilPkg Quick Reference

1. ZeroOneHot, OneHot

```
signal SA, SB, SC, SD : std_logic;
signal ZOH, OH : Boolean ;
```

ZeroOneHot returns true when the std_logic_vector input either has a single one or is all zero.

```
ZOH <= ZeroOneHot((SA & SB & SC & SD)) ;
```

OneHot returns true when the std_logic_vector input either has a single one.

```
OH <= OneHot((SA & SB & SC & SD)) ;
```

2. Transaction Handshaking

Used for handshaking between the client side of a transaction interface and the model.

2.1 RequestTransaction

Request a transaction from the client side (TestCtrl).

```
procedure DoTransaction(
  ModelRec : inout ModelRecType ;
  DataIn     : in DataType
)_ is
begin
  ModelRec.Data <= DataIn ;
  RequestTransaction(
    Req => ModelRec.CmdReq,
    Ack => ModelRec.CmdAck ) ;
end procedure DoTransaction ;
```

2.2 WaitForTransaction

Model side control to wait for client side (TestCtrl) to request a transaction

```
ExecuteTransProc : process
begin
  WaitForTransaction(
    Clk => Clk,
    Req => ModelRec.CmdReq,
    Ack => ModelRec.CmdAck
  ) ;
  -- decode and execute transaction
```

2.3 WaitForTransaction with Timeout

Model side control to wait for client side (TestCtrl) to request a transaction with Timeout. The timeout is to handle applications where the flow needs to be disrupted by an alternate stream of transactions (such as an interrupt handler).

```
ExecuteTransProc : process
begin
  WaitForTransaction(
    Clk => Clk,
    Req => ModelRec.CmdReq,
    Ack => ModelRec.CmdAck,
    TimeOut  => InterruptReq,
    Polarity => '1'
  ) ;
```

2.4 Finish Transaction Handshaking

Used when handshaking with multiple streams of transactions.

```
Finish(Ack => ModelRec.CmdAck) ;
```

2.5 TransactionPending

Returns true when a steam of transaction has a transaction ready.

```
If TransactionPending(Rdy =>
ModelRec.CmdRdy) then
  . . .
```

3. Process to Process Synchronization

One process toggles a signal using the toggle procedure, the other process waits until the signal changes.

3.1 Toggle

Toggle a signal between 0 and 1. Signal type dcan be either bit or std_ulogic.

```
Toggle(Sync1) ;
Toggle(Sync2, 2*tperiod_Clk);
```

3.2 WaitForToggle

Wait until a signal changes. Signal type can be either bit or std_ulogic.

```
WaitForToggle(Sync1) ;
```

4. Barier Synchronization

All processes stop until all processes have reached the barrier and called WaitForBarrier.

```
WaitForBarrier(Sync1) ;
WaitForBarrier(Sync2, 25 ms) ;
```

Type of Sync1 and Sync2 may be either std_logic or integer_barrier.

```
signal sync1 : std_logic := '0' ;
Signal sync2 : integer_barrier := 1 ;
```

5. Waiting for Clock

Wait for clock periods specified in either time units or an integral number of clock cycles. Is aligned to clock when it finishes.

```
WaitForClock(Clk, 5 * Tperiod_Clk);
WaitForClock(Clk, 5) ;
```

6. Wait for Level

Wait until a signal is at a level.

```
WaitForLevel(A, '1') ; -- A='1'
WaitForLevel(Bool) ;   -- TRUE
```

7. Create Clock

Create clock with designated period and duty cycle.

```
CreateClock(
  Clk       => Clk,
  Period    => 10 ns,  -- 100 MHz
  DutyCycle => 0.5     -- Default
) ;
```

8. Create Reset

Create clock with designated period and duty cycle.

```
  CreateReset (
    Reset       => nReset,
    ResetActive => '0', -- active low
    Clk         => Clk,
    Period      => 5 * Tperiod_Clk,
    tpd         => 2 ns
  ) ;
```

9. Clock Polarity

Clock polarity is controlled by the constant CLK_ACTIVE. This will be changed to a generic in a future revision.

```
constant CLK_ACTIVE : std_logic := '1' ;
```


SynthWorks Design Inc.

VHDL Design and Verification Training

11898 SW 128th Ave. Tigard OR 97223 (800)-505-8435

http://www.SynthWorks.com jim@synthworks.com