

## VHDL Quick Reference

### 1. VHDL Designs

A design is partitioned into modular blocks. Each block in the design is created with an entity and architecture. Each block is coded in a separate file.

Each entity and architecture is compiled into a library. Entity names within a library must be unique. The architecture statement repeats the entity name, so the architecture name typically indicates the type of code it contains: RTL, structural, or testbench.

### 2. Entity = IO of a Design

```
library ieee ;
use ieee.std_logic_1164.all ;

entity MuxReg is
port (
    Clk : In std_logic ;
    Sel : In std_logic ;
    A : In std_logic_vector(7 downto 0);
    B : In std_logic_vector(7 downto 0);
    Y : Out std_logic_vector(7 downto 0)
);
end MuxReg ;
```

### 3. RTL Architecture = Implementation

RTL code creates hardware and/or logic. RTL code contains assignments and process statements.

```
architecture RTL of MuxReg is
    -- Declarations
    signal Mux:
        std_logic_vector(7 downto 0);
begin
    -- Code
    Mux <= A when (Sel = '0') else B ;

    RegisterProc : process (Clk)
    begin
        if rising_edge(Clk) then
            Y <= Mux ;
        end if ;
    end process ;

end RTL ;
```

### 4. Structural Architecture = Connectivity

Structural code connects lower levels of a design. Structural code has three pieces: component declarations, signal declarations, and component instances (creates the connectivity).

```
architecture Structural of MuxReg is
    -- Component Declarations
    component Mux8x2
    port (
        Sel : In std_logic ;
        I0, I1 : In unsigned(7 downto 0);
        Y : Out unsigned(7 downto 0)
    );
    end component ;

    component Reg8
    port (
        Clk : In std_logic ;
        D : In unsigned(7 downto 0);
        Q : Out unsigned(7 downto 0)
    );
    end component ;

    -- Signal Declarations
    signal Mux : unsigned(7 downto 0);
begin
    -- Component Instantiations
    -- Named Association
    Mux8x2_1 : Mux8x2
        port map (
            Sel => Sel,
            I0 => A,
            I1 => B,
            Y => Mux
        );

    -- Positional Association
    Reg8_1 : Reg8
        port map (Clk, Mux, Y);

end Structural ;
```

### 5. Common Packages

Usage	Abbr.	Source
use std.standard.all ; -- *	std	IEEE
ieee.std_logic_1164.all ;	1164	IEEE
use ieee.numeric_std.all ;	ns	IEEE
use ieee.numeric_std_unsigned.all ;	nsu	IEEE
use ieee.std_logic_arith.all ;	sla	Shareware
use ieee.std_logic_unsigned.all ;	slu	Shareware
use std.textio.all ;	textio	IEEE
use ieee.std_logic_textio.all ;	-	Shareware

VHDL-2008 adds packages for fixed and floating point.

libraries work and std are implicitly referenced  
\* package std.standard is implicitly referenced

### 6. Common Synthesizable Types

Type / Abbreviation	Value	Package
<b>std_logic</b> / sl	U X 0 1 Z W L H -	1164
<b>std_logic_vector</b> / slv	array of std_logic	1164
<b>signed</b> / sv	array of std_logic	ns, sla
<b>unsigned</b> / uv	array of std_logic	ns, sla
<b>boolean</b> / bool	(False, True)	std
<b>integer</b> / int	-(2 <sup>31</sup> - 1) to 2 <sup>31</sup> - 1	std
<b>natural</b> / int0+	0 to 2 <sup>31</sup> - 1	std
<b>line</b>	access string	textio

Enumerated **type** StateType is (S0, S1, S2, S3) ;

### 7. Assigning Values

```
A_sl <= '1' ; -- Character literal
B_slv <= "1111" ; -- string literal
C_slv <= X"F" ; -- hex. 4 bits per character
E_slv <= (others => '1') ; -- aggregate
L_int <= 15 ; -- universal integer
M_int <= 16#F# ; -- base literal (16 = base)
N_bool <= TRUE ; -- boolean only true or false
```

### 8. VHDL Operators

<b>Logic</b>	and, or, nand, nor, xor, xnor
<b>Comp</b>	=, /=, <, <=, >, >=
<b>Shift</b>	<u>sl</u> , <u>slr</u> , <u>sla</u> , <u>sra</u> , <u>rol</u> , <u>ror</u>
<b>Add</b>	+, -
<b>Sign</b>	+, -
<b>Mult</b>	*, /, mod, rem
<b>Misc</b>	** <u>, abs, not, and, or, nand, nor, xor, xnor</u>

Precedence increases from logic to misc. Underlined items are VHDL-2008.

### 9. Concurrent Statements

Concurrent statements are coded in the architecture.

#### 9.1 Signal Assignments

Expression is evaluated immediately. Value is assigned one delta cycle later.

#### 9.2 Simple Assignment =logic and/or wires

```
Z <= AddReg ;
Sel <= SelA and SelB ;
YL <= A(6 downto 0) & '0' ; --Shift Lt
YR <= '0' & A(7 downto 1) ; --Shift Rt
SR <= SI_sl & A(7 downto 1) ; --Shift In
```

#### 9.3 Conditional Assignment = Concurrent IF

```
Mux2 <=
    A when (Sel1 = '1' and Sel2 = '1')
    else B or C ;

ZeroDet <= '1' when Cnt = 0 else '0' ;
```

The conditional expression must be boolean. Also see the if statement.

## 9.4 Selected Assignment = Concurrent Case

See case statement for rules.

```
with MuxSel select
Mux41 <=
  A   when "00",
  B   when "01",
  C   when "10",
  D   when "11",
  'X' when others ;
```

## 9.5 Process = Container of Sequential Code

Must have either a sensitivity list or wait statement.

Combinational logic requires all inputs (signals read in the process) to be on the sensitivity list. The "is" following the sensitivity list is optional.

```
Mux : process (MuxSel, A, B, C, D) is
begin
  case MuxSel is
  when "00" => Y <= A ;
  when "01" => Y <= B ;
  when "10" => Y <= C ;
  when "11" => Y <= D ;
  when others => Y <= 'X' ;
  end case ;
end process ;
```

## 10. Sequential Statements

Contained in processes and subprograms.

### 10.1 Signal Assignment

```
Z <= AddReg ;
Sel <= Sel1 and Sel2 ;
```

Note: VHDL-2008 allows conditional and selected assignments in sequential statements.

### 10.2 Variable Assignment

Expression is evaluated and assigned immediately.

```
MuxSel := S1 & S0 ;
```

### 10.3 IF Statement

```
if (in1 = '1') then
  NextState <= S1 ;
  Out1 <= '1' ;
elsif (in2 = '1' and in3 = '1') then
  NextState <= S2 ;
elsif (in4 and in5) = '1' then
  NextState <= S3 ;
else
  NextState <= S4 ;
end if ;
```

An IF statement can have one or more signal assignments per branch. Prior to VHDL-2008, the conditional expression must be boolean. With VHDL-2008 it may also be bit or std\_ulogic (std\_logic).

<http://www.SynthWorks.com> jim@SynthWorks.com

## 10.4 Case Statement

```
Mux : process (S1, S0, A, B, C, D)
  variable MuxSel :
    std_logic_vector(1 downto 0) ;
begin
  MuxSel := S1 & S0 ;
  case MuxSel is
  when "00" => Y <= A ;
  when "01" => Y <= B ;
  when "10" => Y <= C ;
  when "11" => Y <= D ;
  when others => Y <= 'X' ;
  end case ;
end process ;
```

A case statement can have zero or more assignments per target. The others choice must be last and is required if all conditions are not covered. Since std\_logic has 9 value, others is almost always required for std\_logic and std\_logic\_vector,

The case expression must have locally static type. Prior to VHDL-2008, this typically means use either a signal or variable name or a slice of a signal or variable.

Regular case statement does not use '-' as don't care.

### 10.5 Asynchronous Reset Flip-Flop

Asynchronous reset is specified before the clock. Clock and reset must be on the sensitivity list.

```
RegProc : process ( Clk, nReset)
begin
  if (nReset = '0') then
    AReg <= '0' ;
    BReg <= '0' ;
  elsif rising_edge(Clk) then
    if LoadEn = '1' then
      AReg <= A ;
      BReg <= B ;
    end if ;
  end if ;
end process ;
```

### 10.6 Synchronous Reset Flip-Flop

Synchronous reset is specified after the clock. Only clock must be on the sensitivity list.

```
RegProc : process (Clk)
begin
  if rising_edge(Clk) then
    if (nReset = '0') then
      AReg <= '0' ;
    elsif LoadEn = '1' then
      AReg <= A ;
    end if ;
  end if ;
end process ;
```

## 10.7 Flip-Flop with Wait Until

Flip-flops with wait until only support synchronous reset.

```
RegProc : process
begin
  wait until Clk = '1' ;
  AReg <= A ;
  BReg <= B ;
end process ;
```

## 10.8 For Loop

```
RevAProc : process(A)
begin
  for i in 0 to 7 loop
    RevA(7 - i) <= A(i) ;
  end loop ;
end process ;
```

Loop index can be any identifier and does not need to be declared. For synthesis, loop index must be integer.

## 10.9 Wait Until and after

Wait stops a process for at least a delta cycle. Wait until Clk = '1' finds the next rising edge of clock and is used extensively in testbenches.

Signal assignments using "after" always project a value on a signal. "After" never causes a process to stop.

```
TestProc : process begin
  wait until Clk = '1' ;
  Addr <= "000" after tpd_Clk_Addr ;
  wait until Clk = '1' ;
  Addr <= "001" after tpd_Clk_Addr ;
  -- and so on ...
  wait for tperiod_clk * 5 ;
  report "Test Done" severity failure ;
end process ;
```

## 10.10 VHDL-2008

VHDL-2008 simplifies case statement rules, allows std\_logic and bit in a conditional expression (if, while, ...), allows selected and conditional assignment for signals and variables in a sequential code and more. See SynthWorks' website for papers on VHDL-2008.

Let your vendors know you want these updates.

© 1999 – 2014 by SynthWorks Design Inc. Reproduction of entire document in whole permitted. All other rights reserved.

**SynthWorks Design Inc.**  
VHDL Hardware Synthesis and Verification Training  
11898 SW 128<sup>th</sup> Ave. Tigard OR 97223 (800)-505-8435  
<http://www.SynthWorks.com> jim@synthworks.com