SynthWorks

VHDL Training Experts

VHDL Quick Reference

1. VHDL Designs

A design is partitioned in to a 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).

Before compiling a structural architecture, the lower level blocks must be compiled into the library first.

```
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
use ieee.**std_logic_1164**.all ;	1164	IEEE
use ieee.**numeric_std**.all ;	ns	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 ;	sltextio	Shareware

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

6. Common Synthesizable Types

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

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

Precedence increases from logic to misc. Shift operators are not yet implemented for std_logic_vector. See also VHDL Types and Operators Quick Reference.

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

Conditional and selected signal assignments are not supported in sequential statements.

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

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. The conditional expression must be boolean. Use comparison operators as shown above to accomplish this.

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 be locally static. Typically this means it is either a signal name or a slice name. To form expressions, a variable is commonly used.

The case expression 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

Using wait until to create a register. Only supports synchronous resets.

```
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, Wait For, and Assert

Wait until and wait for are used in a testbench to find a particular event or point in time. Wait stops a process for at least a delta cycle. Report with severity failure stops a simulation.

Signal assignments using "after" always project a value on a signal and 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 brings us simplified case statement rules, allows std_logic and bit in a conditional expression (if, while, …), 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.

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