

VHDL 200X Fast Track

By Jim Lewis
SynthWorks VHDL Training
Team Leader VHDL-200X Fast Track
jim@synthworks.com

VHDL-200X Fast Track

SynthWorks

- Original Intent
 - Make critical, non-contentious updates quickly (1 yr)
 - Intended as an enabler for other VHDL standards work
 - Both 200X and 1164/1076.3
 - Only took on items we thought we could complete within our time constraints (1 yr)
- Result
 - All items of high value, but
 - might not be your #1 item because of the original limited time scope of fast track.

Status At a Glimpse

- Items to be completed
 - FT 32: Disambiguate Path Names wrt VHPI & Assertions
 - FT 7: Methodology for Hierarchical Reference
- Items Needing Further Review
 - FT 5 / 8: to_String / hwrite, hread
 - N-008 & N-009: Fixed and Floating Point Packages
- All other work items
 - "User LCS" means taken as far as possible by the WG
 - "Intent" means detailed the change to the point where an LRM writer can finish it.
- Details at:
 - <http://www.eda.org/vhdl-200x/vhdl-200x-ft/proposals/proposals.html>

Value Statement

- As FT chair, it is hard to be objective
 - All items are high on my list
 - All work was done on a volunteer basis -
 - It is high on someone else's list too.
 - For most work, the only requirement to complete it is funding
 - LRM completion is currently being delayed by funding
 - Fast Track presentations have been well received at DVCon, DesignCon, and at Mentor's users group meetings
 - If available today, many of the features would be immediately adopted. Many have already tried (accidentally).

High Value FT Items (My List):

- Arrays and Records with unconstrained arrays (FT 14)
- Fixed and Floating Point Packages (N-008, N-009)
- Hierarchical references of signals (FT 7)
- Case Statement Updates (FT 22, 23, 25)
- Conditional Expression Updates (FT 18)
- Process(all) (FT 19)
- Improved IO: hwrite, hread, (FT 8) and to_string, to_hstring (FT 5)
- Sized bit string literals (FT 9)
- Expressions in port maps (FT 11)
- Read out ports (FT 12)
- Conditional and Selected assignment in sequential code (FT 10A)
- If Expressions (FT 10B)
- Operator Updates (FT1, FT2, FT3, FT4, FT28)
- Numeric_Std (N-*) and Std_logic_1164 Updates (CP*)
- Stop and Finish (FT 13)
- Slices in array aggregates (FT 20)

High Value FT Items (My List):

- Arrays and Records with unconstrained arrays - FT14
 - New Feature
 - Building block for Vector (array of numbers) and Matrix math
- Fixed and Floating Point Packages - N-008, N-009
 - New Feature
 - Packages already in use
 - Need review and finalization
- Hierarchical references of signals - FT7 (to be completed)
 - New Feature
 - Vendor specific packages already in use

Case Statement Updates - FT22, 23, and 25

- Revision to current feature - ease of use
- Bring VHDL's capability up to par with other languages

```

constant ONE1      : unsigned := "11" ;
constant CHOICE2   : unsigned := "00" & ONE1 ;
signal A, B        : unsigned (3 downto 0) ;
. . .
process (A, B)
begin
  case (A xor B) is
    when "0000"      => Y <= "00" ;
    when CHOICE2     => Y <= "01" ;
    when "0110"      => Y <= "10" ;
    when ONE1 & "00" => Y <= "11" ;
    when others      => Y <= "XX" ;
  end case ;
end process ;

```

Conditional Expression Updates - FT 18

- Revision to current feature - ease of use
- Bring VHDL's capability up to par with other languages
- Applies to conditional of: if, elsif, wait until, when, while

```

if (Cs1 and not nCs2 and Addr=X"A5") then
if (Cs1 and nCs2='0' and Addr=X"A5") then
if (not nWe) then

```

- Current VHDL syntax (proposal is backward compatible):

```

if ((Cs1 and not nCs2)='1' and Addr=X"A5") then
if (Cs1='1' and nCs2='0' and Addr=X"A5") then
if (nWe = '0') then

```

Process (all) - FT 19

- New Feature
- Removes common design error.
- Creates a sensitivity list with all signals on sensitivity list

```
Mux3_proc : process(all)
begin
  case MuxSel is
    when "00" =>      Y <= A ;
    when "01" =>      Y <= B ;
    when "10" =>      Y <= C ;
    when others =>    Y <= 'X' ;
  end case ;
end process
```

- Benefit: Reduce mismatches between simulation and synthesis

Improved IO - FT 5 and FT8

- New and expanded IO capabilities - VHDL currently weak
- To_String (FT 5)
- Hread / Hwrite (FT 8)

- Use of to_string with VHDL's built-in write

```
write(Output, "%%ERROR data value miscompare." &
  NL & "  Actual value = " & to_hstring (Data) &
  NL & "  Expected value = " & to_hstring (ExpData) &
  NL & "  at time:  " & to_string (now, right, 12)) ;
```

Sized Bit String Literals - FT9

- Currently hex bit string literals are a multiple of 4 in size

```
x"AA" = "10101010"
```

- Allow specification of size (and decimal bit string literals):

```
7X"7F" = "11111111"
7D"127" = "11111111"
```

- Allow specification of signed vs unsigned (extension of value):

```
9UX"F" = "000001111"   Unsigned 0 fill
9SX"F" = "111111111"   Signed: left bit = sign
9X"F"  = "000001111"   Defaults to unsigned
```

- Allow Replication of X and Z

```
7X"XX" = "XXXXXXXX"
7X"ZZ" = "ZZZZZZZZ"
```

Signal Expressions in Port Maps - FT 11

```
U_UUT : UUT
  port map ( A, Y and C, B) ;
```

- Needed to avoid extra signal assignments with OVL
- If expression is not a single signal, constant, or does not qualify as a conversion function, then
 - convert it to an equivalent concurrent signal assignment
 - and it will incur a delta cycle delay

Read Output Ports - FT 12

- Read output ports
 - Value read will be locally driven value
- Assertions need to be able to read output ports

Conditional and Selected Assignments for Sequential Code - FT 10A

- Statemachine code:

```

if (FP = '1') then
    NextState <= FLASH ;
else
    NextState <= IDLE ;
end if ;

```

- Simplification (new part is that this is in a process):

```

NextState <= FLASH when (FP = '1') else IDLE ;

```

- Also support conditional variable assignment:

```

NextState := FLASH when (FP = '1') else IDLE ;

```

IF Expressions - FT10B

- Similar to conditional signal assignment ...

```
Y <= A and B if S = '1', C and D ;
Y <= (A and B) if S = '1', (C and D) ;
```

- ... except it is an expression:

```
Y <= A and (B if S = '1', C) and D ;
```

- And it can be used anywhere an expression can:

```
Signal A : integer := 7 if GEN_VAL = 1, 15 ;

with MuxSel select
  Y <= (A if Asel='1', B) when '0',
       (C if Csel='1', D) when '1',
       'X' when others ;
```

Operator Updates +

- Unary Reduction Operators - FT 2

```
Parity <= xor Data ; -- Data(7) xor Data(6) xor ...
```

- Array Scalar Logic Operators - FT 3

```
signal Sel : std_logic ;
signal T, A : std_logic_vector(2 downto 0) ;
. . .
T <= (A and Sel) ;
-- (A(2) and Sel, A(1) and Sel, A(0) and Sel) ;
```

- Explicitly declared subprograms overload implicitly declared subprograms - FT 1
 - Good for numeric_std_unsigned (and comparison ops)
 - Good for backward compatibility when adding new implicitly defined operators

Numeric Std and Std Logic 1164

- Implementation of Shift Operators (rol, ror, sll, srl, sla, sra)
- Unary reduction operators (See previous slide)
- Array Scalar Logic Operators (See previous slide)
- Hread, Hwrite, to_string, ...
- To_X01, ... IS_X
- Unsigned package for std_logic_vector / bit_vector (numeric_std_unsigned / numeric_bit_unsigned)
- Array Scalar Math Operations - N-001

```

signal Y           : unsigned (8 downto 0) ;
signal A,          : unsigned (7 downto 0) ;
signal CarryIn    : std_logic ;
. . .
Y <= ('0' & A) + ('0' & B) + CarryIn ;

```

Stop and Finish - FT13

- Add procedures STOP and FINISH:
 - STOP - Stops like breakpoint
 - FINISH - Terminate the simulation

```

procedure STOP;
procedure FINISH;

```

- Benefit:
 - Just stop. No annoying messages.

Slices in Array Aggregates - FT 20

- Allow slices in an Array Aggregate

```

Signal A, B, Y      : unsigned (7 downto 0) ;
signal CarryOut    : std_logic ;

. . .

(CarryOut, Y)  <= ('0' & A) + ('0' & B) ;

```

- Currently, this would have to be written as:

```

(CarryOut, Y(7), Y(6), Y(5), Y(4), Y(3), Y(2), Y(1), Y(0))
  <= ('0' & A) + ('0' & B) ;

```

VHDL-200X-FT, Summary

- Most work is ready for hand-off to LRM writer
 - Finalization of FT 7 and FT 32 done under ?IEEE?
 - Final review done under ?Accellera?
 - Only want to finalize items that will be funded
- FT is done (mostly) - so lets finish it and fund it.
 - Funding would be a morale enhancement to the current FT group who have been working 2 years on these tasks.
- Funding FT will free up the FT team to work heavily on the new features:
 - Constrained Random, Interfaces, Functional Coverage, DPI, ...

Increase Default Context - FT 29

- Include library IEEE and package std.textio in default context
- Resulting default context will be:

```
Library std, work, ieee ;
  use std.standard.all ;
  use std.textio.all ;
```

Context Unit = Primary Design Unit - FT16 SynthWorks

- Allows a group of packages to be referenced by a single name

```
Context project1_Ctx is
  use std.textio.all ;
  library ieee ;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all ;
  use ieee.fixed_pkg.all ;
end ;
```

- Reference the named context unit

```
Library Lib_P1 ;
  context Lib_P1.project1_ctx ;
```

- Benefit increases as additional standard packages are created
 - Fixed Point, Floating Point, Assertion Libraries, . . .

Case With Don't Care - FT 24

- Allow use of '-' in targets provided targets are non-overlapping

```
-- Priority Encoder
process (Request)
begin
  case? Request is
    when "1---" => Grant <= "1000" ;
    when "01--" => Grant <= "0100" ;
    when "001-" => Grant <= "0010" ;
    when "0001" => Grant <= "0001" ;
    when others => Grant <= "0000" ;
  end case ;
end process ;
```

Note: Only '-' in the case target is treated as a don't care.
A '-' in the case? Expression will not be treated as a don't care.