Announcements/Review

• Project 1 due Tuesday.
  • As always, start early, start often.

• Last time: LDV overview
  • Analyzing, elaborating, simulating

Today

• Entity, architecture, process
• Nature of variables in VHDL
• Nature of signals in VHDL
• Transactions, events
• Transaction queue
• Processes
• VHDL simulation

Example: Ones Counter

• Counts the number of 1’s in an input vector of length 3
• Interface description:
  entity ONES_COUNTER is
  port (A: in Bit_vector (2 downto 0);
       C: out Bit_vector (1 downto 0));
  end ONES_COUNTER;

Main Parts of a VHDL Model

• Entity
  • Interface to other devices
    • View from the outside
    • Analogous to a schematic symbol
• Architecture
  • Behavioral
    • Output in response to inputs
  • Structural
    • Internal structure

Behavioral Domain

Algorithmic Architecture

architecture behavior of ones_counter is
begin
  process (A)
  variable Num: integer range 0 to 3;
  begin
    Num := 0;
    for i in 0 to 2 loop
      if A(i) = ’1’ then
        Num := Num + 1;
        end if;
      end loop;
      case Num is
        when 0 =>
          C <= "00";
        when 1 =>
          C <= "01";
        when 2 =>
          C <= "10";
        when 3 =>
          C <= "11";
      end case;
    end process;
  end
end behavior;
**Nature of VHDL Signals and Variables**

- VHDL Variables
  - Similar to variables in C
  - Value changes instantaneously
- VHDL Signals
  - Have a time dimension
  - NEVER change instantaneously

**VHDL Signal Assignment**

- General Format
  - cout <= "0000" after 10 ns;
- Default Format
  - cout <= "0001";
  - Default time is a very small time (infinitesimal), but NOT zero, called delta time.
  - Cout <= "0001" after 0 ns;

**Nature of VHDL Variable Assignment**

- Suppose that a variable V has an initial value of '0'.
- Suppose the following two statements are executed sequentially inside a process
  
  \[
  V := '1';
  \]
  
  If \( V = '1' \) then (Statement 1);
- Will Statement 1 be executed?

**Transaction Queue**

- Statement 1: cout1 <= "0000" after 10 ns;
- Statement 2: cout2 <= "0001";
- Statement 3: count2 <= "0001" after 0 ns;

<table>
<thead>
<tr>
<th>Statement</th>
<th>Signal</th>
<th>Time (ns)</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>cout1</td>
<td>CT+10</td>
<td>&quot;0000&quot;</td>
</tr>
<tr>
<td>S2, S3</td>
<td>cout2,</td>
<td>CT+delta</td>
<td>&quot;0001&quot;</td>
</tr>
<tr>
<td></td>
<td>count2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Nature of VHDL Signal Assignment**

- Suppose that a signal S has an initial value of '0'.
- Suppose the following two statements are executed sequentially inside a process
  
  \[
  S := '1' \text{ after } 10 \text{ ns};
  \]
  
  If \( S = '1' \) then (Statement 1);
- Will Statement 1 be executed?
Nature of VHDL Signal Assignment

- Suppose that a signal $S$ has an initial value of '0'.
- Suppose the following two statements are executed sequentially inside a process
  
  $S <= '1'$ after 0 ns;
  
  If ($S = '1'$) then (Statement 1);
  
  Will Statement 1 be executed?

Terminology

- Signal
- Changes are scheduled for the future
- Transaction
- Event
- Propagation Delay
- Transaction queue

Simulation System

- Model is created
- Stimulus is applied to model input
- Simulator services queue entries
- Events cause model processes to execute
- New transactions are added to queue
- Simulation continues until queue is empty or breakpoint is reached

Simulation Cycle (Part 1)

- Advance simulation time to time of next entry in the transaction queue. If no entries, stop.
- Update all signals that have entries at the current simulation time and remove those entries from the transaction queue.

Simulation Cycle (Continued)

- For each signal event, execute all processes having the signal in the process sensitivity list

Half Adder with Unequal Delays

Half_Adder: process (A, B)

  Sum <= A xor B after 8 ns;
  Carry <= A and B after 5 ns;
end process;

\[
\begin{align*}
A & \\
B & \\
\text{Tprop} = 5 \text{ ns} & \quad \text{Carry} \\
\text{Tprop} = 8 \text{ ns} & \quad \text{Sum}
\end{align*}
\]
**Events (cont’d)**

<table>
<thead>
<tr>
<th>Time</th>
<th>A</th>
<th>B</th>
<th>Sum</th>
<th>Carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0ns</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5ns</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8ns</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10ns</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>15ns</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18ns</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25ns</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Devices Represented by Processes**

- DFF: process (CLK, R)
  - begin
  - if r='0' then
    - Q <= '0';
  - elsif CLK'EVENT
  - and CLK='1' then
    - Q <= D;
  - end if;
  - end process;