VHDL in Motion

Chapter 4
Macro and micro time

Last time

- Delay Models
  - Inertial
  - Transport
- Waveform Updating Algorithms

Short Pulses

architecture SHORT of pulse_test is
signal A, B, S_INERTIAL, S_TRANSPORT: Bit;
begin
  S1: A <= '0', '1' after 1 ns, '0' after 5 ns, '1' after 20 ns, '0' after 30 ns;
  S2: B <= '0';
  S3: S_INERTIAL <= A or B after 10 ns;
  S4: S_TRANSPORT <= transport A or B after 10 ns;
end SHORT;

Additional Waveform Updating for Inertial Delay with reject clause

- B1. All of the new transactions are marked.
- B2. An old transaction is marked if the time at which it is projected to occur is less than the time at which the first new transaction is projected to occur minus the pulse rejection limit.
- B3. An old transaction is marked if it immediately precedes a marked transaction and its value component is the same as that of the marked transaction.
- B4. The transaction that determines the current value of the driver is marked.
- B5. All unmarked transactions (all of which are old transactions) are deleted from the projected output waveform.

An example

- Consider the following signal driver for an integer S:
  Time 5ns 13ns 22ns 23ns 30ns 52ns
  Value 1 2 2 6 5 8
- At time 10ns, the following assignment occurs:
  S <= reject 15 ns inertial 12 after 20 ns, 18 after 41 ns;

Macro and Micro Time

- Macro Time
  - standard time units (ns, ps, seconds, etc.)
- Micro Time
  - “delta delay” time
  - slightly longer than instantaneous time
  - can have multiple Micro times within Macro times
Macro and Micro Time

Delta delays

- If no propagation delay is specified, then the delay is assumed to be “delta”...
- Why not specify a delay?
  - To check the functional correctness but not timing.
  - Might add timing information later, after synthesizing hardware...

Delta delays

- Why does the simulator use delta delays?
- The simulator evaluates all the right hand sides before updating the left hand sides to properly order events.
- Delta delays are the mechanism for doing this.
  - Updates occur delta time after evaluations...so all evaluations are finished before moving on to the updates.

If we DIDN’T have delta delays...

Seven Examples

- Standard buffer whose output follows its input
- entity BUFS is
  port (X : in Bit; Z : out Bit);
end BUFS;
Input X: X \rightarrow Z

Buffer One

architecture ONE of BUFS is
begin
X := "1" after 1 ns;
begin
Y1 := X;
Z <= Y1 after 1 ns;
end process;
end ONE;

architecture ONE of reg is
SIGNAL b, c : bit;
BEGIN
b <= NOT(a);
c <= NOT(clock AND b);
d <= c AND b;
END test;

If no propagation delay is specified, then the delay is assumed to be “delta”...
- Why not specify a delay?
  - To check the functional correctness but not timing.
  - Might add timing information later, after synthesizing hardware...

If we DIDN’T have delta delays...

ARCHITECTURE test OF reg IS
SIGNAL b, c : bit;
BEGIN
b <= NOT(a);
c <= NOT(clock AND b);
d <= c AND b;
END test;

Assume clock = '1' and a changes from '1' to '0'
Buffer Two

architecture TWO of BUFF is

signal Y2 : Bit;

begin
X <= "1" after 1 ns,
'0' after 4 ns;
Y2 <= X;
Z <= Y2;
end TWO;

<table>
<thead>
<tr>
<th>ns</th>
<th>X</th>
<th>Y2</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0+1Δ</td>
<td>--</td>
<td>--</td>
<td>0</td>
</tr>
<tr>
<td>0+2Δ</td>
<td>--</td>
<td>--</td>
<td>0</td>
</tr>
</tbody>
</table>

Buffer Three

architecture THREE of BUFF is

signal Y3 : Bit;

begin
X <= '1' after 1 ns,
'0' after 4 ns;
Y3 <= X;
Z <= Y3 after 1 ns;
end THREE;

<table>
<thead>
<tr>
<th>ns</th>
<th>X</th>
<th>Y3</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0+1Δ</td>
<td>--</td>
<td>--</td>
<td>0</td>
</tr>
<tr>
<td>1+1Δ</td>
<td>--</td>
<td>--</td>
<td>1</td>
</tr>
</tbody>
</table>

Buffer Four

architecture FOUR of BUFF is

signal Y4 : Bit;

begin
X <= '1' after 1 ns,
'0' after 4 ns;
Y4 <= X after 1 ns;
Z <= Y4 after 1 ns;
end FOUR;

<table>
<thead>
<tr>
<th>ns</th>
<th>X</th>
<th>Y4</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1+1Δ</td>
<td>--</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>1+2Δ</td>
<td>--</td>
<td>--</td>
<td>1</td>
</tr>
</tbody>
</table>

Buffer Five

architecture FIVE of BUFF is

signal Y5 : Bit;

begin
X <= '1' after 1 ns,
'0' after 4 ns;
Y5 <= X;
Z <= Y5 after 1 ns;
process (X)
begin
Y5 <= X;
Z <= Y5;
end process;
end FIVE;

<table>
<thead>
<tr>
<th>ns</th>
<th>X</th>
<th>Y5</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0+1Δ</td>
<td>--</td>
<td>--</td>
<td>0</td>
</tr>
<tr>
<td>4+1Δ</td>
<td>--</td>
<td>--</td>
<td>1</td>
</tr>
</tbody>
</table>

Buffer Six (Five_A)

architecture FIVE_A of BUFF is

signal Y5A: Bit;

begin
X <= '1' after 1 ns,
'0' after 4 ns;
process (X, Y5A)
begin
Y5A <= X;
Z <= Y5A;
end process;
end FIVE_A;

<table>
<thead>
<tr>
<th>ns</th>
<th>X</th>
<th>Y5A</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0+1Δ</td>
<td>--</td>
<td>--</td>
<td>0</td>
</tr>
<tr>
<td>1+1Δ</td>
<td>--</td>
<td>--</td>
<td>1</td>
</tr>
</tbody>
</table>

Buffer Seven (Five_B)

architecture FIVE_B of BUFF is

signal Y5B: Bit;

begin
X <= '1' after 1 ns,
'0' after 4 ns;
process (X)
begin
Y5B <= X;
Z <= Y5B;
end process;
end FIVE_B;

<table>
<thead>
<tr>
<th>ns</th>
<th>X</th>
<th>Y5B</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0+1Δ</td>
<td>--</td>
<td>--</td>
<td>0</td>
</tr>
<tr>
<td>4+1Δ</td>
<td>--</td>
<td>--</td>
<td>0</td>
</tr>
</tbody>
</table>
Summary

- Macro and micro time

- Delta delays: Ensure correct ordering of zero time events

- Next time: Packages and libraries, pp. 96-114