

10.1 Quiz 1

Wednesday, February 28, 2001

Friday, March 2, 2001

- 12:00 - 1:00 PM, Room 50-340 - Walker
- 1 handwritten 'crib' sheet allowed (both sides).

Venue

- Problem Sets 1-3, Lab 1, Lectures 1-8
- Understand VHDL entities, simple architectures.
- Understand Boolean, If, and Case/When statements.
- You are not responsible for generating VHDL syntax.

General Topics

- Boolean Algebra and Elementary Logic
 - Expressions
 - Karnaugh Maps and Reduction
 - Minimal Realizations: MSP and MPS
 - Translation to Circuit Diagrams

10.2 Quiz 1 - More General Topics

- Combinational Logic
 - Switching Threshold for TTL
 - Tristate Busses
- Latches, etc.
 - S-R latches
 - D, JK, and T edge-triggered flip-flops
 - Transition Diagrams and Truth Tables
 - Counters, Shift Registers, Multiplexors, and Selectors
- PALs
 - General Familiarity with 20V8 and 22V10
- Finite State Machines
 - Transition Diagrams and State Tables
 - Moore: Outputs Function of State Only - Specified in state or on arcs.
 - Mealy: Outputs Function of State and Inputs - Specified on arcs.
 - Transition on NEXT rising edge of clock
 - Waveforms to be expected from FSM
 - Interpretation of CASE/WHEN VHDL statements (not syntax)

10.3 Quiz 1 - More General Topics

- Know Details (not pins) of:
 - 74LS74 Edge Triggered Flip-Flops
 - Have Asynchronous Negative True PR and CL
 - 74LS163 Synchronous 4-bit Counters
 - P and T control Counting
 - T Chains with CAR output
 - CLR and LD are Synchronous, Negative True Inputs.
 - CAR Can Have Glitches!
- Know Functions (not details) of:
 - 74LS151 Multiplexer
 - 74LS138 Decoder
 - 74LS194 Shift Register
- Synchronization
 - Always synchronize asynchronous inputs.
 - An asynchronous input results in only a single flip-flop transition.
 - Catch short pulses with a latch.
 - Turn them into an asynchronous level.
 - Synchronize asynchronous levels with a D flip-flop.
 - Use D flip-flop and SR latch to generate single pulses.

10.4 Quiz 1 - More General Topics

- Timing Considerations
 - Gate Delays (Combinational Logic)
 - Setup and Hold Times
 - Clock to Q Delays
 - Glitches
 - From combinational logic
 - From multiple flip-flop transitions
 - How to avoid glitches
 - Single flip-flop transition
 - Gate with complement of CLK
 - How to avoid trouble from glitches
 - Don't use glitchy signals for CLK, PR, CLR, S, R.
 - Clock into a register AFTER signals are stable.

10.5 VHDL Identifiers

- Case Insensitive (but best not to rely on this)
- First character must be a letter.
 - Letters, Digits, and Underscores (only)
 - ▷ Two underscores in succession are not allowed.
 - ▷ The last character cannot be an underscore.
- Using reserved words is NOT allowed.
 - Reserved words are AQUA in later versions of emacs.
 - Using reserved words usually provokes an understandable error comment.
- Legal Examples
 - CLK, Three_StateEnable, h23, Reg_12
- Illegal Examples
 - _clk, 3_State_Enable, large#num, clk_, Three__State, register, begin

10.6 VHDL Reserved Words

- Some are

| | | |
|---------|------------|-----------|
| abs | access | after |
| array | disconnect | file |
| guarded | impure | postponed |
| rem | unaffected | wait |

- In other words, there are too many to remember!
- This is another good reason for "incremental" compilation.
- Start with something that compiles and add code a block at a time!

10.7 Using Tri-State

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all; -- needed for integer + signal
entity test_tri is
    port (clk, oe, cnt_enb : in std_logic;
          data : inout std_logic_vector(7 downto 0));
end test_tri;
architecture foo of test_tri is
    signal counter : std_logic_vector(7 downto 0);
begin
    process (oe, counter)
    begin
        if (oe = '1') then data <= counter;
        else
            data <= "ZZZZZZZZ"; -- N.B. Z must be UPPERCASE!
        end if;
    end process;
    process (clk)
    begin
        if rising_edge(clk) then
            if (oe = '0') and (cnt_enb = '1') then
                counter <= counter + 1;
            end if;
        end if;
    end process;
end architecture foo;
```

10.8 Don't Care, Ampersand

- Don't cares are represented by a - (hyphen).
 - '.' single quotes for a character
 - "---" double quotes for a string (vector)
 - (others => '-') for something independent of length
- & (ampersand) to concatenate strings or signals
 - "01" & "111" is the same as "01111"
 - '0' & "1111" is the same as "01111"
- Remember, VHDL is strongly typed.
 - a + b is valid only if a and b are of the same length.
 - The result is of the SAME length as a (or b).
 - If you want it to be one bit longer, then use

```
c <= ('0' & a) + ('0' & b)
-- Of course, c must be defined to be
-- one bit longer than a.
```

10.9 Packages

- Entities need not be in the same file as the package declaration.

```
library ieee;
use ieee.std_logic_1164.all;
entity mux2to1 is port (
  a, b, sel: in std_logic;
  c: out std_logic);
end mux2to1;
architecture archmux2to1 of mux2to1 is
begin
  c <= (a and not sel) or (b and sel);
end archmux2to1;
library ieee; -- note repeated library and use statements
use ieee.std_logic_1164.all;
package mymuxpkg is
  component mux2to1 port (
    a, b, sel: in std_logic; -- identical port list
    c: out std_logic);
  end component;
end mymuxpkg;
```

10.10 Toplevel

```
library ieee;
use ieee.std_logic_1164.all;
entity toplevel is port (
  s: in std_logic;
  p, q, r: in std_logic_vector(2 downto 0);
  t: out std_logic_vector(2 downto 0));
end toplevel;
use work.mymuxpkg.all;
architecture archtoplevel of toplevel is
  signal i: std_logic_vector(2 downto 0);
begin
  -- the first two instantiations are named associations
  m0: mux2to1 port map (a=>i(2), b=>r(0), sel=>s, c=>t(0));
  m1: mux2to1 port map (c=>t(1), b=>r(1), a=>i(1), sel=>s);
  -- the last instantiation is a positional association
  m2: mux2to1 port map (i(0), r(2), s, t(2));
  i <= p and not q;
end archtoplevel;
```

- Exercise - rewrite muxpkg.vhd and toplevel.vhd to make use of generics to specify the length of signal vectors for a 2:1 mux.

10.11 Predefined Attributes

- s'event is read as "s tick event" where s is a signal name.

rising_edge(event) is the same as (s'event and event = '1')
as synthesis only worries about 1 and 0.

- Transactions occur when a signal is evaluated, whether or not the signal value changes.

- Time, if not specified, is in deltas.

```
s'event      - true only if an event occurred in the
              current delta cycle
s'active     - true only if a transaction occurred on s
s'last_event - time elapsed since the last event on s
s'last_value - value of s before the previous event on s
s'last_active - time elapsed since the previous
               transaction on s
```

10.12 Predefined Attributes cont'd

- Signal attributes that create a signal

○ Mainly used for simulation

```
s'delayed[(time)] - creates a signal the same
                  as s but delayed by the
                  specified time

s'stable[(time)]  - a boolean signal that is
                  true if s had no events
                  for the specified time

s'quiet[(time)]   - a boolean signal that is
                  true if s had no transactions
                  for the specified time

s'transaction     - a signal of type bit that
                  changes for every transaction on s
```

10.13 Array Attributes

```
Signal s : std_logic_vector( 7 downto 3)

    s'left  = 7          s'high = 7

    s'right = 3          s'low  = 3

    s'length = 5

type rom is array (0 to 6, 3 down to 0) of std_logic;
signal r : rom;

    r'left(1) = 0          r'high(1) = 6
    r'left(2) = 3          r'high(2) = 3

    r'right(1) = 6         r'low(1)  = 0
    r'right(2) = 0         r'low(2)  = 0

    r'length(1) = 7
    r'length(2) = 4
```

10.14 User Defined Attributes

```
type state_type is (idle, state1, state2);
attribute state_encoding of state_type: is sequential;
-- or one_hot, zero_hot, gray

attribute enum_encoding of state_type: is "11 01 00";
-- or whatever assignment you want to make

-- within an entity
attribute pin_numbers of counter:Entity is
    "clk:13 reset:2 &
    " count(3):3;
-- Note the space before count(3) above

-- within an entity
attribute pin_avoid of mydesign: entity is "21 24 26";

-- the following are not recommended

attribute lab_force of mysig: signal is a1;
attribute node_num of buried: signal is 202;
attribute low_power of mydesign: entity is "b g e";
attribute slew_rate of count(3): signal is slow; -- or fast
```

10.15 Two More that Are Useful Sometimes

Sum splitting occurs because of too many product terms.

- Balanced (default) has better timing but uses more macrocells.
- Cascaded uses fewer macrocells and is slower.

```
attribute sum_split of mysig: signal is cascaded;
```

The synthesis_off attribute is used to make the signal a factoring point.

- Making a signal a factoring point can result in a reduction of product terms for a subsequent signal.

Registered equations are natural factoring points so only use synthesis_off on combinational signals.

```
attribute synthesis_off of sel: signal is true;
```