1 Administrivia

Announcements

Assignment

Read 5-1,2.

From Last Time

Sequential circuit design.

Outline

1. Modified serial comparator.

2. VHDL for serial comparator.

3. Exercise.

Coming Up

Registers
2 Modified Serial Comparator

Inputs: A, B, (no more msb). A and B are received least significant bit first. Output 0 if A \geq B, otherwise 1.

Reset to S0 on reset.

State diagram:

```
S0: a \geq b  S1: b > a
00/0  10/0  01/1  11/0  00/1  11/1
```

3 VHDL for Serial Comparator

Things to observe:

1. Flip-flop implementation: reset priority, event, rising edge sensitive.

2. If and case — sequential statements — are valid only within a process.

3. Concurrent assignment is a “process.”

4. Semantics of a process: sensitivity list, assignments:

   ```
   b <= a;
   c <= b;
   ```

   does not behave as it would in C.

5. VHDL architecture broken into three processes:

   (a) State storage.
(b) Next state generation.

(c) Output generation.

Compare process inputs to sensitivity lists.

-- VHDL for serial comparator. The inputs a and b are input lsb first.
-- The Mealy machine uses rising edge sensitive flip-flops and an
-- asynchronous active low reset.
--
-- The output is 1 if b > a, otherwise 0.
--
-- Note: The reset will not simulate properly in functional simulation.
-- It will simulate properly in timing simulation, once an
-- implementation has been created.

library ieee;
use ieee.std_logic_1164.all;

entity comparator is
  port
    (a, b, clk, reset : in std_logic;
     o : out std_logic);
end comparator;

architecture process_defn of comparator is

  -- Two states needed.
  type state_type is (S0, S1);
  -- State assignment.
attribute enum_encoding : string;
attribute enum_encoding of state_type :
  type is "0 1";

signal state, next_state : state_type;

-- For convenience, concatenate a and b.
signal inputs : std_logic_vector (1 downto 0);

begin

  -- Concurrent assignment executes the rhs changes.
  -- Concatenate a and b into inputs.
  inputs <= a & b;

  -- Processes execute whenever something on their sensitivity list
  -- changes. All assignments take place when the process exits.
  --
  -- This process implements the D flip-flop.

  state_register : process (clk, reset)
  begin
    -- If/else construct only valid within a process.
    if (reset = '0') then
      state <= S0;
      elsif (clk'event AND clk = '1') then
        state <= next_state;
    end if;
  end process;

  -- This process computes the next state.

  next_state_process : process (inputs, state)
  begin
    case state is
      when S0 =>
        if (inputs = "01") then
          next_state <= S1;
        else
          next_state <= S0;
        end if;
      when others =>
        next_state <= S0;
    end case;
  end process;

end;
when S1 =>
    if (inputs = "10") then
        next_state <= S0;
    else
        next_state <= S1;
    end if;
end case;
end process;

-- This process computes the output.

output_process : process (inputs, state)
begin
    case state is

    when S0 =>
        if (inputs = "01") then
            o <= '1';
        else
            o <= '0';
        end if;

    when S1 =>
        if (inputs = "10") then
            o <= '0';
        else
            o <= '1';
        end if;

    end case;
end process;

end process_defn;

4 Exercise

Serial comparator. Inputs: A, B. A and B are received most significant bit first. Reset to initial state on reset. Output 0 if A ≥ B, otherwise 1.