Counters
Tom Kelliher, CS 220
Nov. 12, 2003

1 Administrivia

Announcements

Exam Friday.

Assignment

Read 6-1,6 for Monday.

From Last Time

Shift registers.

Outline

1. Ripple counters.

2. Synchronous counters.

3. VHDL.
2 Ripple Counters

The increment ripples — propagation delay problems.

Slow counters.

Basic idea:

1. Each flip-flop’s !Q is fed back to D. What does this accomplish?

2. Flip-flop i’s !Q is used to clock flip-flop i + 1. What does this accomplish?

3. Trace the propagation delay of the clock if the count is currently 1111 and a rising clock edge is applied.

4. Through what sequence, starting with 0000, does the counter count?
3 Synchronous Counters

1. All flip-flops receive same clock signal.

2. Still have some rippling. (Where?)

3. Inputs: clk, enable.

4. Outputs: count, carry output (for cascading).

5. After state table minimization, input equation for bit $i$:
   \[ C_i \oplus (C_0 \cdot C_1 \cdot \ldots \cdot C_{i-1} \cdot EN) \]

6. One bit slice for serial gating:

7. Serial vs. parallel gating.

4 VHDL

32 bit up counter with enable and reset.

-- Up counter with enable and reset
--
-- Note how en must be handled after the flip-flop generating
-- code.

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter is

port (
    clk, reset_n, en : in std_logic;
    q : out std_logic_vector (31 downto 0);
    co : out std_logic);

end counter;

architecture behavioral of counter is

    signal count : std_logic_vector (31 downto 0);

begin -- behavioral

    q <= count;

    state: process (clk, reset_n)
    begin -- process state
        if reset_n = '0' then
            count <= X"00000000";
        elsif clk'event and clk = '1' then
            if en = '1' then
                count <= count + X"00000001";
            end if;
        end if;
    end process state;

    carry_out: process (count, en)
    begin -- process carry_out
        if count = X"FFFFFFFF" and en = '1' then
            co <= '1';
        else
            co <= '0';
        end if;
    end process carry_out;

end behavioral;