Electrical and Computer Engineering

Register - Synchronous Storage

By Prawat Nagvajara

Synopsis

This note presents a design and implementation of n-bit register with parallel load input or end-around right shift (see Fig. 1). The register loads new contents when the load enable, LD_en = '1' and shifts whne LD_en = '0'.

Schematic

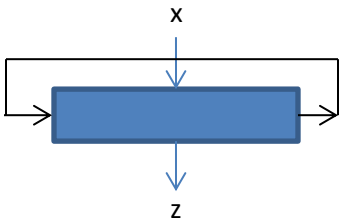

Fig. 1 End-Around Shift Register

Code

```
-------------------------------------------------------
-- Module Name: end_around_sh_reg
-- Description: Behavioral
-- Parallel load register when LD_en = '1'
-- end-around right shift when LD_en = '0'
-------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity end_around_sh_reg is
generic (n: natural);
Port ( X : in std_logic_vector(n-1 downto 0);
 Z : out std_logic_vector(n-1 downto 0);
 CK, LD_en, btn0, btn1 : in std_logic );
end end_around_sh_reg;

architecture Beh of end_around_sh_reg is
signal en: std_logic; -- single step signal
-- declare 4-bit storage as internal signal
signal temp:std_logic_vector(3 downto 0);
begin

z <= temp; -- assign output ports

-- single step, debounce (db)
-- btn0 to enter and btn1 to reset
process(ck)
type db_state is (not_rdy, rdy, pulse);
variable db_ns: db_state;
begin
if ck='1' and ck'event then
  case db_ns is
   when not_rdy => en <= '0';
   if btn1 = '1' then db_ns := rdy; end if;
   when rdy => en <= '0';
   if btn0 = '1' then db_ns := pulse; end if;
   when pulse => en <= '1';
   db_ns := not_rdy;
   when others => null;
  end case;
end if;
end process;

-- end around shift reg
process(en) -- sync to single-step pulse
begin
if en='1' and en'event then -- sync fence
 if ld_en = '1' then temp <= x; else --parallel load
   temp(3) <= temp(0);
   for i in 2 downto 0 loop
            temp(i) <= temp(i+1);
            end loop;
-- shift using concatenate in one line assignment
--          temp <= temp(0)&temp(3 downto 1);
 end if;
end if;
end process;

end Beh;
```

Wrapper

Below shows a wrapper for n=4.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity shift_reg_4  is
Port (x : in     std_logic_vector(3 downto 0);
        z : out std_logic_vector(3 downto 0);
 CK, LD_en, btn0, btn1 : in std_logic );
end shift_reg_4;

architecture struc of shift_reg_4 is

component end_around_sh_reg
generic (n: natural);
Port ( X : in std_logic_vector(n-1 downto 0);
 Z : out std_logic_vector(n-1 downto 0);
 CK, LD_en, btn0, btn1 : in std_logic );
end component;

begin

U2: end_around_shift_reg
generic map(4)
port map(x=>x, z=>z, ck=> ck, LD_en=> Ld_en,
            btn0 => btn0, btn1 => btn1);

end struc;
```

Simulation

Figure 2 shows the waveforms from the simulation of the wrapper (n=4). The single step signal en appears every 3 clock cycles. When LD_en = '1' the input "1101" was loaded. When LD_en = '0' the contents of the register – signal temp end-round shifted right synchronized with the single-step signal rising edge. The content temp wires to the output port z.

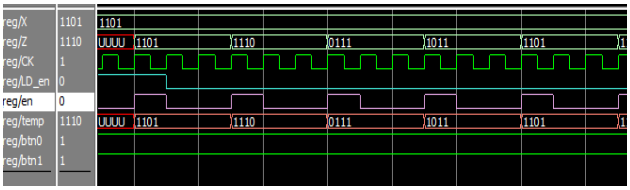

Fig. 2 Simulation

Exploration

Design variation of shift registers such as a register that serial loads the input when enable or end-around shifts when not load enable.