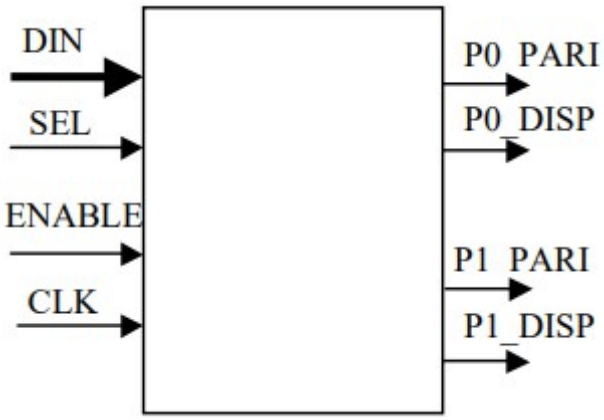
13 February 2008

Describe a circuit to generate parity bits – even and odd respectively – from 2 strings of undefined length. The circuit has a 2-bits input for the serial input streams (**DIN**), a clock signal (**CLK**), a reset signal (**RST**), an activation signal (**ENABLE**), and a selection input (**SEL**) which determines which output pair (**P0_PARI**, **P0_DISP** and **P1_PARI**, **P1_DISP**) is to be updated.

If the enable signal is equal to 0, the selected output pair must be set to high impedance. When the entity is reset, the outputs must be set to 0.



Translation Notes:
PARI = Even
DISP = Odd

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6
7   entity parity is
8      port (CLK, RST, EN, sel: in std_logic;
9            Din: in std_logic_vector (1 downto 0);
10           P0e, P0o, P1e, P1o: out std_logic);
11  end parity;
12
13
14  architecture Behavioral of parity is
15     --In order to the parity bits
16     --The values of these signals are sent to the output
17     --according to the value of "sel".
18     signal P0, P1: std_logic:='0';
19
20     begin
21
22        process (CLK, EN, RST)
23        begin
24           if (RST = '1') then
25              --General reset.
26              P0e <= '0';
27              P0o <= '0';
28              P1e <= '0';
29              P1o <= '0';
30              P0 <= '0';
31              P1 <= '0';
32           elsif (rising_edge (CLK)) then
33              --When the entity is disabled, output lines are set to high impedance.
34              if (EN = '0') then
35                 P0e <= 'Z';
36                 P0o <= 'Z';
37                 P1e <= 'Z';
38                 P1o <= 'Z';
39                 P0 <= '0';
40                 P1 <= '0';
41                 --When the entity is enabled, parity bits are computed:
42                 --the output lines are updated according to the value of "sel".
43                 --NOTE: parity bits (P0 and P1) are computed in any case, that's why there are
44                 --two instructions outside the case construction; otherwise, it is not possible
45                 --to update correctly the output lines when the value of "sel" changes.
46              elsif (EN = '1') then
47                 P0 <= P0 xor Din (0);
48                 P1 <= P1 xor Din (1);
49                 case sel is
50                    when '0' => P0e <= P0 xor Din (0);
51                                P0o <= not (P0 xor Din (0));
52                    when '1' => P1e <= P1 xor Din (1);
53                                P1o <= not (P1 xor Din (1));
54                    when others => null;
55                 end case;
56              end if;
57           end if;
58        end process;
59
60  end Behavioral;
61
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--2 serial input lines, DIN(0), DIN(1) and 2 pairs of serial output: P0_ODD, P0_EVEN
--P1_ODD and P1_EVEN.
--SEL (1 bit) allows to select which output pair is
--updated;
--P_ODD: odd parity bit of DIN;
--P_EVEN: even parity bit of DIN.
--DIN bitstream is continuous; parity bits should be computed according to
--the present input bit and the previous ones, each time.

entity parity is
   port(CLK, RST, EN, SEL: in std_logic;
        DIN: in std_logic_vector (1 downto 0);
        P0_EVEN, P0_ODD, P1_EVEN, P1_ODD: out std_logic);
end parity;

architecture Behavioral of parity is

   signal buff0, buff1: std_logic:='Z';

begin

process(CLK, RST)
begin
   --1) Reset:
   if (RST = '1') then
      buff0 <= '0';
      buff1 <= '0';
   elsif (EN = '1' and rising_edge(CLK)) then
      --2) SEL = 'i': DIN(i) parity bits computation:
      --even parity = (previous parity bit) XOR (new DIN(i) bit);
      --odd parity = NOT (even parity).
      --when an output pair is selected, the other one should store its present value;
      --when SEL switches, new parity bit is computed on the last value which ha been
      --stored;
      --actually, the entity can always compute both parity bits, and transfer to the output
      --just the ones selected by SEL, but that will cause unnecessary commutations.
      case (SEL) is
         when '0' => buff0 <= buff0 xor DIN(0);
         when '1' => buff1 <= buff1 xor DIN(1);
         when others => null;
      end case;
   end if;
end process;

P0_EVEN <= buff0 when (EN = '1') else 'Z';
P0_ODD <= not(buff0) when (EN = '1') else 'Z';
P1_EVEN <= buff1 when (EN = '1') else 'Z';
P1_ODD <= not(buff1) when (EN = '1') else 'Z';

end Behavioral;
```