|  |
| --- |
| library IEEE; |
|  | use IEEE.STD\_LOGIC\_1164.ALL; |
|  | use IEEE.STD\_LOGIC\_arith.ALL; |
|  | use IEEE.STD\_LOGIC\_unsigned.ALL; |
|  |  |
|  | -- Uncomment the following library declaration if using |
|  | -- arithmetic functions with Signed or Unsigned values |
|  | --use IEEE.NUMERIC\_STD.ALL; |
|  |  |
|  | -- Uncomment the following library declaration if instantiating |
|  | -- any Xilinx primitives in this code. |
|  | --library UNISIM; |
|  | --use UNISIM.VComponents.all; |
|  |  |
|  | entity VotingMachine is |
|  | Port |
|  | ( |
|  |  |
|  | clk : in std\_logic; |
|  | reset : in std\_logic; |
|  | party1 : in std\_logic; |
|  | party2 : in std\_logic; |
|  | party3 : in std\_logic; |
|  | select\_party : in std\_logic; |
|  |  |
|  | count1\_op : out std\_logic\_vector(5 downto 0); |
|  | count2\_op : out std\_logic\_vector(5 downto 0); |
|  | count3\_op : out std\_logic\_vector(5 downto 0) |
|  | ); |
|  | end VotingMachine; |
|  |  |
|  | architecture Behavioral of VotingMachine is |
|  | signal count1,count2,count3: std\_logic\_vector(5 downto 0); --for performing arithmetic operation and displaying giving to output |
|  | signal state: std\_logic\_vector(5 downto 0); --6 states |
|  | constant initial: std\_logic\_vector(5 downto 0):= "000001"; --state1 |
|  | constant check: std\_logic\_vector(5 downto 0):= "000010"; --state2 |
|  | constant party1\_state: std\_logic\_vector(5 downto 0):= "000100"; --state3 |
|  | constant party2\_state: std\_logic\_vector(5 downto 0):= "001000"; --state4 |
|  | constant party3\_state: std\_logic\_vector(5 downto 0):= "010000"; --state5 |
|  | constant done: std\_logic\_vector(5 downto 0):= "100000"; --state6 |
|  |  |
|  | begin |
|  |  |
|  | process (clk, reset, party1, party2, party3) |
|  | begin |
|  | if (reset='1')then |
|  | count1 <= (others=>'0'); |
|  | count2 <= (others=>'0'); |
|  | count3 <= (others=>'0'); |
|  | state <= initial; |
|  | else |
|  | if (rising\_edge(clk) and reset='0') then |
|  | -- NSL : Next State Logic |
|  | case state is |
|  | when initial => |
|  | --NSL |
|  | if (party1='1' or party2='1' or party3='1') then |
|  | state <=check; |
|  | else |
|  | state <= initial; |
|  | --OFL |
|  | end if; |
|  |  |
|  | when check=> |
|  | --NSL |
|  | if (party1='1') then |
|  | state <= party1\_state; |
|  | elsif (party2='1') then |
|  | state <= party2\_state; |
|  | elsif (party3='1') then |
|  | state <= party3\_state; |
|  | else |
|  | state<=check; |
|  | end if; |
|  | --OFL |
|  |  |
|  | when party1\_state=> |
|  | --NSL |
|  | if (select\_party='1') then |
|  | state <= done;  count1 <= count1 + 1; |
|  | else |
|  | state <= party1\_state; |
|  | end if; |
|  |  |
|  |  |
|  |  |
|  | when party2\_state=> |
|  | --NSL |
|  | if (select\_party='1') then |
|  | state <= done;  count2 <= count2 + 1; |
|  | else |
|  | state <= party2\_state; |
|  | end if; |
|  |  |
|  |  |
|  |  |
|  | when party3\_state=> |
|  | --NSL |
|  | if (select\_party='1') then |
|  | state <= done;  count3 <= count3 + 1; |
|  | else |
|  | state <= party3\_state; |
|  | end if; |
|  |  |
|  |  |
|  |  |
|  | when done => |
|  | --NSL |
|  | state <= initial; |
|  |  |
|  |  |
|  | when others=> |
|  | state <= initial; |
|  |  |
|  | end case; |
|  |  |
|  | end if; |
|  | end if; |
|  | end process; |
|  |  |
|  | count1\_op <= count1; |
|  | count2\_op <= count2; |
|  | count3\_op <= count3; |
|  |  |
|  | end Behavioral; |