library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity SWITCH_DEBOUNCER_CORE is
  generic
  (NUM_SWITCHES : integer := 8; -- Width of input & output
   DELAY : integer := 8);
  port
  (RESET : in std_logic;
   CLOCK : in std_logic;
   INPUT_SWITCH_ARRAY : in std_logic_vector((NUM_SWITCHES-1) downto 0);
   OUTPUT_SWITCH_ARRAY : out std_logic_vector((NUM_SWITCHES-1) downto 0);
   NEW_SWITCH_STATE : out std_logic;
   ACKNOWLEDGE : in std_logic);
end SWITCH_DEBOUNCER_CORE;

architecture Behavioral of SWITCH_DEBOUNCER_CORE is
  -- Determine the minimum counter size
  constant SIZE : integer := integer(ceil(log2(real(DELAY))));

  constant DELAY_U : unsigned((SIZE-1) downto 0) := to_unsigned((DELAY-1),SIZE);

  -- The counter for the debouncer
  signal DEBOUNCE_COUNTER : unsigned((SIZE-1) downto 0);
  signal INPUT_REGISTER : std_logic_vector((NUM_SWITCHES-1) downto 0);
  signal OUTPUT_REGISTER : std_logic_vector((NUM_SWITCHES-1) downto 0);
  signal INT_NEW_SWITCH_STATE : std_logic;

  -- The debouncer state machine
  type DEBOUNCE_SM is (IDLE, DEBOUNCE_IN_PROGRESS);
  signal DEBOUNCE_STATE : DEBOUNCE_SM;

begin

  DEBOUNCE_INPUT_SWITCH_ARRAY : process(CLOCK) is
    begin
      if rising_edge(CLOCK) then
        if(RESET = '1') then
          DEBOUNCE_COUNTER <= (others => '0');
          INPUT_REGISTER <= (others => '0');
          OUTPUT_REGISTER <= (others => '0');
          INT_NEW_SWITCH_STATE <= '0';
          DEBOUNCE_STATE <= IDLE;
        else
          case DEBOUNCE_STATE is
            when IDLE =>
              DEBOUNCE_COUNTER <= DEBOUNCE_COUNTER - 1;
              when DEBOUNCE_IN_PROGRESS =>

              if(INPUT_SWITCH_ARRAY /= INPUT_REGISTER) then
                INPUT_REGISTER <= INPUT_SWITCH_ARRAY;
                DEBOUNCE_STATE <= DEBOUNCE_IN_PROGRESS;
                DEBOUNCE_COUNTER <= DELAY_U;
              end if;
          end case;
        end if;
      end if;
    end process;

end Behavioral;
if(DEBOUNCE_COUNTER = 0) then
  OUTPUT_REGISTER <= INPUT_REGISTER;
  INT_NEW_SWITCH_STATE <= '1';
  DEBOUNCE_STATE <= IDLE;
end if;
else
  DEBOUNCE_STATE <= IDLE;
end if;
end case;

-- Clear NEW_SWITCH_STATE when ACKNOWLEDGE is asserted
if((INT_NEW_SWITCH_STATE = '1') and (ACKNOWLEDGE = '1')) then
  INT_NEW_SWITCH_STATE <= '0';
end if; -- if(RESET = '1')
end if; -- if(rising_edge(CLOCK))
end process DEBOUNCE_INPUT_SWITCH_ARRAY;

OUTPUT_SWITCH_ARRAY <= OUTPUT_REGISTER;
NEW_SWITCH_STATE <= INT_NEW_SWITCH_STATE;
end architecture;