```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;

entity BARRELSHIFT is
end entity BARRELSHIFT;

architecture TEST_BENCH of BARRELSHIFT is

signal INPUT_VAL  : STD_LOGIC_VECTOR ( 23 downto 0 ) := X"888888";
signal OUTPUT_VAL   : STD_LOGIC_VECTOR ( 23 downto 0 );
signal SHIFT_AMT    : STD_LOGIC_VECTOR (  4 downto 0 );

signal STAGE_ONE    : STD_LOGIC_VECTOR ( 23 downto 0 );
signal STAGE_TWO    : STD_LOGIC_VECTOR ( 23 downto 0 );
signal STAGE_THREE  : STD_LOGIC_VECTOR ( 23 downto 0 );
signal STAGE_FOUR   : STD_LOGIC_VECTOR ( 23 downto 0 );

signal ADDSUBOUT  : STD_LOGIC_VECTOR ( 27 downto 0 );
signal POST_N_ONE : STD_LOGIC_VECTOR ( 27 downto 0 );
signal POST_N_TWO : STD_LOGIC_VECTOR ( 27 downto 0 );
signal POST_N_THR : STD_LOGIC_VECTOR ( 27 downto 0 );
signal POST_N_FOR : STD_LOGIC_VECTOR ( 27 downto 0 );
signal POST_N_VAL : STD_LOGIC_VECTOR ( 27 downto 0 );
signal POSTN      : STD_LOGIC_VECTOR (  4 downto 0 );

constant ZEROS    : STD_LOGIC_VECTOR ( 31 downto 0 ) := X"0000_0000";

begin

  process
  begin
    for I in 0 to 23 loop
      SHIFT_AMT <= CONV_STD_LOGIC_VECTOR ( I, 5 );
      wait for 10 ns;
    end loop;
  end process;

  STAGE_ONE <= INPUT_VAL when SHIFT_AMT(4) = '0' else X"0000" & INPUT_VAL ( 23 downto
16);

  STAGE_TWO <= STAGE_ONE when SHIFT_AMT(3) = '0' else X"00" & STAGE_ONE ( 23 downto 8
);

  STAGE_THREE <= STAGE_TWO when SHIFT_AMT(2) = '0' else X"0" & STAGE_TWO ( 23 downto 4
 );

  STAGE_FOUR <= STAGE_THREE when SHIFT_AMT(1) = '0' else "00" & STAGE_THREE ( 23 downt
o 2 );

  OUTPUT_VAL <= STAGE_FOUR when SHIFT_AMT(0) = '0' else '0' & STAGE_FOUR ( 23 downto 1
 );


----------------------------  Post Normalization  stuff
--
--  value being post normalized....
  process
  variable THE_VAL : STD_LOGIC_VECTOR ( 27 downto 0 );
  begin
    for I in 0 to 27 loop
      if I = 0 then
        THE_VAL := X"800_0000";
      else
        THE_VAL := '0' & THE_VAL ( 27 downto 1 );
```

```
      end if;
      ADDSUBOUT <= THE_VAL;
      wait for 10 ns;
    end loop;
  end process;

  POSTN(4) <= '1' when ADDSUBOUT(27 downto 12) = X"0000" else '0';
  POST_N_ONE <= ADDSUBOUT when POSTN(4) = '0' else
                ADDSUBOUT ( 11 downto 0 ) & ZEROS(15 downto 0 );

  POSTN(3) <= '1' when POST_N_ONE ( 27 downto 20 ) = X"00" else '0';
  POST_N_TWO <= POST_N_ONE when POSTN(3) = '0' else
                POST_N_ONE ( 19 downto 0 ) & ZEROS ( 7 downto 0 );

  POSTN(2) <= '1' when POST_N_TWO ( 27 downto 24 ) = "0000" else '0';
  POST_N_THR <= POST_N_TWO when POSTN(2) = '0' else
                POST_N_TWO ( 23 downto 0 ) & ZEROS ( 3 downto 0 );

  POSTN(1) <= '1' when POST_N_THR ( 27 downto 26 ) = "00" else '0';
  POST_N_FOR <= POST_N_THR when POSTN(1) = '0' else
                POST_N_THR ( 25 downto 0 ) & ZEROS ( 1 downto 0 );

  POSTN(0) <= '1' when POST_N_FOR ( 27 ) = '0' else '0';
  POST_N_VAL <= POST_N_FOR when POSTN(0) = '0' else
                POST_N_FOR ( 26 downto 0 ) & ZEROS ( 0 );

end architecture TEST_BENCH;
```