FIFO simple code

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;
entity fifo is
Generic (
constant DATA_WIDTH : positive := 8;
constant FIFO_DEPTH : positive := 8
);
Port(
CLK : in STD_LOGIC;
RST : in STD_LOGIC;
WriteEn : in STD_LOGIC;
DataIn : in STD_LOGIC_VECTOR (DATA_WIDTH - 1 downto 0);
ReadEn : in STD_LOGIC;
DataOut : out STD_LOGIC_VECTOR (DATA_WIDTH - 1 downto 0);
Empty : out STD_LOGIC;
Full : out STD_LOGIC);
end fifo;
architecture Behavioral of fifo is
begin
fifo_proc : process(CLK)
type FIFO_Memory is array (0 to FIFO_DEPTH -1) of
STD_LOGIC_VECTOR(DATA_WIDTH - 1 downto 0);
variable Memory : FIFO_Memory;

variable Head : natural range 0 to FIFO_DEPTH - 1;
variable Tail : natural range 0 to FIFO_DEPTH - 1;
variable Looped : boolean;
begin
if rising_edge(CLK) then
```

```
if RST = '1' then
Head := 0;
Tail := 0;
Looped := false;
Full <= '0';
Empty <= '1';
else
if (ReadEn = '1') then
if ((Looped = true) or (Head /= Tail)) then
DataOut <= Memory(Tail);
if(Tail = FIFO_DEPTH - 1) then
Tail := 0;
Looped := false;
else
Tail := Tail + 1;
end if;
end if;
end if;
if (WriteEn = '1') then
if ((Looped = false) or (Head /= Tail)) then

Memory(Head) := DataIn;
if (Head = FIFO_DEPTH - 1) then
Head := 0;
Looped := true;
else
Head := Head + 1;
end if;
end if;
end if;
if (Head = Tail) then
```

```
if Looped then
Full <= '1';
else
Empty <= '1';
end if;
else
Empty <= '0';
Full <='0';
end if;
end if;
end if;
end process;
end Behavioral;
```

FIFO test bench code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY fifotb IS
END fifotb;

ARCHITECTURE behavior OF fifotb IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT fifo
   PORT(
```

```
        CLK : IN  std_logic;
        RST : IN  std_logic;
        WriteEn : IN  std_logic;
        DataIn : IN  std_logic_vector(7 downto 0);
        ReadEn : IN  std_logic;
        DataOut : OUT  std_logic_vector(7 downto 0);
        Empty : OUT  std_logic;
        Full : OUT  std_logic
       );
    END COMPONENT;


   --Inputs
   signal CLK : std_logic := '0';
   signal RST : std_logic := '0';
   signal WriteEn : std_logic := '0';
   signal DataIn : std_logic_vector(7 downto 0) := (others => '0');
   signal ReadEn : std_logic := '0';

            --Outputs
   signal DataOut : std_logic_vector(7 downto 0);
   signal Empty : std_logic;
   signal Full : std_logic;

   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

            -- Instantiate the Unit Under Test (UUT)
   uut: fifo PORT MAP (
```

```
      CLK => CLK,
      RST => RST,
      WriteEn => WriteEn,
      DataIn => DataIn,
      ReadEn => ReadEn,
      DataOut => DataOut,
      Empty => Empty,
      Full => Full
    );

-- Clock process definitions
CLK_process :process
begin
                CLK <= '0';
                wait for CLK_period/2;
                CLK <= '1';
                wait for CLK_period/2;
end process;


-- Stimulus process
stim_proc: process
begin
rst <='1';
  wait for 100 ns;
rst <='0';
  WriteEn <='1';
    DataIn<="11110000";
                        wait for 100 ns;
                         DataIn<="10101010";
                        wait for 100 ns;
```

```
                ReadEn <='1';
                wait for 100 ns;
                assert DataOut<="11110000"report"Error:incorrect data read out";
                wait for 100 ns;
                assert DataOut<="10101010"report"Error:incorrect data read out";
                wait for 100 ns;

                assert Empty ='1'report"Error:fifo is not empty";

                wait for CLK_period*10;


    wait;
  end process;
END;
```