**fifo.main**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity fifo\_correct is

GENERIC

(

ADDRESS\_WIDTH : integer:=2;---8 bit

DATA\_WIDTH : integer:=4 ---32 bit

);

port ( clk : in std\_logic;

clk\_div : inout std\_logic;

reset : in std\_logic;

enr : in std\_logic; --enable read,should be '0' when not in use.

enw : in std\_logic; --enable write,should be '0' when not in use.

dataout : out std\_logic\_vector(DATA\_WIDTH-1 downto 0); --output data

datain : in std\_logic\_vector (DATA\_WIDTH-1 downto 0); --input data

empty : out std\_logic; --set as '1' when the queue is empty

err : out std\_logic;

full : out std\_logic --set as '1' when the queue is full

);

end fifo\_correct;

architecture Behavioral of fifo\_correct is

type memory\_type is array (0 to ((2\*\*ADDRESS\_WIDTH)-1)) of std\_logic\_vector(DATA\_WIDTH-1 downto 0);

-----distributed-------

signal memory : memory\_type ;-- :=(others => (others => '0')); --memory for queue.-----

signal readptr,writeptr : std\_logic\_vector(ADDRESS\_WIDTH-1 downto 0); --read and write pointers.

signal full0 : std\_logic;

signal empty0 : std\_logic;

signal counter: std\_logic\_vector(28 downto 0):=( others=>'0');

begin

full <= full0;

empty <= empty0;

fifo0: process(clk\_div,reset,datain,enw,enr)

begin

if reset='1' then

readptr <= (others => '0');

writeptr <= (others => '0');

empty0 <='1';

full0<='0';

err<='0';

elsif clk\_div'event and clk\_div = '1' then

if enw='1' and full0='0' then

memory (conv\_integer(writeptr)) <= datain ;

writeptr <= writeptr + '1' ;

if (writeptr + '1' = readptr) then

full0<='1';

empty0<= '0';

else

full0<='0';

empty0<= '1';

end if ;

end if ;

if enr='1' and empty0='0' then

dataout <= memory (conv\_integer(readptr));

readptr <= readptr + '1' ;

if (readptr + '1' = writeptr ) then

empty0<='1';

full0<='0';

else

empty0<='0';

full0<='1';

end if ;

end if ;

if (empty0='1' and enr='1') or (full0='1' and enw='1') then

err<='1';

else

err<= '0';

end if ;

end if;

end process;

process(clk)

begin

if clk'event and clk= '1' then

counter<= counter + '1';

end if;

end process;

clk\_div<= counter(0);

end Behavioral;

**fifo\_tb**

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 fifo\_test IS

END fifo\_test;

ARCHITECTURE behavior OF fifo\_test IS

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

COMPONENT fifo\_correct

PORT(

clk : in std\_logic;

clk\_div : inout std\_logic;

reset : in std\_logic;

enr : in std\_logic; --enable read,should be '0' when not in use.

enw : in std\_logic; --enable write,should be '0' when not in use.

dataout : out std\_logic\_vector(3 downto 0); --output data

datain : in std\_logic\_vector (3 downto 0); --input data

empty : out std\_logic; --set as '1' when the queue is empty

err : out std\_logic;

full : out std\_logic --set as '1' when the queue is full

);

END COMPONENT;

--Inputs

signal clk : std\_logic := '0';

signal reset,clk\_div: std\_logic;

signal enr : std\_logic := '0';

signal enw : std\_logic := '0';

signal datain : std\_logic\_vector(3 downto 0) := (others => '0');

--Outputs

signal dataout : std\_logic\_vector(3 downto 0);

signal empty : std\_logic;

signal err : std\_logic;

signal full : std\_logic;

-- Clock period definitions

constant clk\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: fifo\_correct PORT MAP (

clk => clk,

clk\_div=>clk\_div,

reset=>reset,

enr => enr,

enw => enw,

dataout => dataout,

datain => datain,

empty => empty,

err => err,

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;

reset<='1','0' after 50ns;

enw<= '1', '0' after 200 ns ;

enr<= '0','1' after 200 ns ;

--Stimulus process

stim\_proc: process

begin

datain<="1010";

wait for 10 ns;

datain<="1111";

wait for 10 ns;

datain<="1001";

wait for 10 ns;

datain<="0001";

wait for 10 ns;

end process;

END;

**OUTPUT:**

