A) FIFO MAIN CODE

```
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;
```

B) FIFO TESTBENCH

```
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 beh 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 beh;
```