VHDL Code for FIFO (8x3)

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_logic\_arith.ALL;

use IEEE.STD\_logic\_unsigned.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity fifo is

generic (depth : integer := 8); --depth of fifo

port ( clk : in 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.

data\_in : in std\_logic\_vector (2 downto 0); --input data

data\_out : out std\_logic\_vector(2 downto 0); --output data

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

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

);

end fifo;

architecture Behavioral of fifo is

type memory\_type is array (0 to depth-1) of std\_logic\_vector(2 downto 0);

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

signal readptr,writeptr : integer := 0; --read and write pointers.

signal empty,full : std\_logic := '0';

signal dataout:std\_logic\_vector(2 downto 0);

begin

fifo\_empty <= empty;

fifo\_full <= full;

p12:process(clk,reset)

--this is the number of elements stored in fifo at a time.

--this variable is used to decide whether the fifo is empty or full.

variable num\_elem : integer := 0;

begin

if(reset = '1') then

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

empty <= '0';

full <= '0';

readptr <= 0;

writeptr <= 0;

num\_elem := 0;

elsif(rising\_edge(clk)) then

if(enr = '1' and empty = '0') then --read

dataout <= memory(readptr);

readptr <= readptr + 1;

num\_elem := num\_elem-1;

end if;

if(enw ='1' and full = '0') then --write

memory(writeptr) <= data\_in;

writeptr <= writeptr + 1;

num\_elem := num\_elem+1;

end if;

--rolling over of the indices.

if(readptr = depth-1) then --resetting read pointer.

readptr <= 0;

end if;

if(writeptr = depth-1) then --resetting write pointer.

writeptr <= 0;

end if;

--setting empty and full flags.

if(num\_elem = 0) then

empty <= '1';

else

empty <= '0';

end if;

if(num\_elem = depth) then

full <= '1';

else

full <= '0';

end if;

end if;

end process;

data\_out<=dataout;

end Behavioral;

**VHDL Test Bench**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_arith.ALL;

ENTITY fifo\_tb IS

END fifo\_tb;

ARCHITECTURE behavior OF fifo\_tb IS

--Inputs and outputs

signal Clk,reset,enr,enw,empty,full : std\_logic := '0';

signal data\_in,data\_out : std\_logic\_vector(2 downto 0) := (others => '0');

--temporary signals

signal i : integer := 0;

-- Clock period definitions

constant Clk\_period : time := 10 ns;

constant depth : integer := 8; --specify depth of fifo here.

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: entity work.fifo generic map(depth => depth) PORT MAP (clk,reset,enr,enw,data\_in,data\_out,empty,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

reset <= '1'; --apply reset for one clock cycle.

wait for clk\_period;

reset <= '0';

wait for clk\_period\*3; --wait for 3 clock periods(simply)

enw <= '1'; enr <= '0'; --write 8 values to fifo.

for i in 1 to depth loop

Data\_In <= conv\_std\_logic\_vector(i,3);

wait for clk\_period;

end loop;

enw <= '0'; enr <= '1'; --read 8 values from fifo.

wait for clk\_period\*depth;

wait;

end process;

end behavior;

**Implementation code with slow clock on Spartan 6**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_logic\_arith.ALL;

use IEEE.STD\_logic\_unsigned.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity fifo is

generic (depth : integer := 8); --depth of fifo

port ( clk : in 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.

data\_in : in std\_logic\_vector (2 downto 0); --input data

data\_out : out std\_logic\_vector(2 downto 0); --output data

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

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

);

end fifo;

architecture Behavioral of fifo is

type memory\_type is array (0 to depth-1) of std\_logic\_vector(2 downto 0);

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

signal readptr,writeptr : integer := 0; --read and write pointers.

signal empty,full : std\_logic := '0';

signal temp:std\_logic\_vector(27 downto 0);

signal sclk:std\_logic;

signal dataout:std\_logic\_vector(2 downto 0);

begin

fifo\_empty <= empty;

fifo\_full <= full;

p11: process(clk,reset)

begin

if(reset='1')then temp<=(others=>'0');

elsif(clk'event and clk='1') then

temp<= temp + 1;

end if;

end process;

sclk<=temp(27);

p12:process(sclk,reset)

--this is the number of elements stored in fifo at a time.

--this variable is used to decide whether the fifo is empty or full.

variable num\_elem : integer := 0;

begin

if(reset = '1') then

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

empty <= '0';

full <= '0';

readptr <= 0;

writeptr <= 0;

num\_elem := 0;

elsif(rising\_edge(sclk)) then

if(enr = '1' and empty = '0') then --read

dataout <= memory(readptr);

readptr <= readptr + 1;

num\_elem := num\_elem-1;

end if;

if(enw ='1' and full = '0') then --write

memory(writeptr) <= data\_in;

writeptr <= writeptr + 1;

num\_elem := num\_elem+1;

end if;

--rolling over of the indices.

if(readptr = depth-1) then --resetting read pointer.

readptr <= 0;

end if;

if(writeptr = depth-1) then --resetting write pointer.

writeptr <= 0;

end if;

--setting empty and full flags.

if(num\_elem = 0) then

empty <= '1';

else

empty <= '0';

end if;

if(num\_elem = depth) then

full <= '1';

else

full <= '0';

end if;

end if;

end process;

data\_out<=dataout;

end Behavioral;

**Implementation Constraint File**

NET "reset" LOC = "E4";

NET "CLK" LOC ="L15";

NET "enr" LOC ="T5";

NET "enw" LOC ="R5";

NET "data\_in<0>" LOC ="A10";

NET "data\_in<1>" LOC ="D14";

NET "data\_in<2>" LOC ="C14";

NET "data\_out<0>" LOC = "U18"; # Bank = 1, Pin name = IO\_L52N\_M1DQ15, Sch name = LD0

NET "data\_out<1>" LOC = "M14"; # Bank = 1, Pin name = IO\_L53P, Sch name = LD1

NET "data\_out<2>" LOC = "N14"; # Bank = 1, Pin name = IO\_L53N\_VREF, Sch name = LD2

NET "fifo\_empty" LOC = "P16"; # Bank = 1, Pin name = IO\_L61P, Sch name = LD3

NET "fifo\_full" LOC = "D4";