Main:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity fibonacci is

Port ( clk : in STD\_LOGIC;

--data\_out : out STD\_LOGIC\_VECTOR (7 downto 0));

dec\_out : out INTEGER);

end fibonacci;

architecture Behavioral of fibonacci is

signal data\_in : STD\_LOGIC\_VECTOR (15 downto 0) := "0000000000000011"; -- Aloitussisältö

signal xor3 : STD\_LOGIC;

begin

-- palautusbitti

xor3 <= data\_in(0) xor data\_in(3) xor data\_in(5) xor data\_in(7);

process(clk)

begin

if rising\_edge(clk) then

data\_in <= xor3 & data\_in(15 downto 1); -- Siirrä ja lisää palautusbitti

end if;

end process;

dec\_out <= to\_integer(unsigned(data\_in(15 downto 8))); -- tallennetaan kahdeksan bittiä alkaen MSB desimaaleina

end Behavioral;

TB:

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_textio.ALL;

USE std.textio.ALL;

ENTITY fibonacci\_vhd\_tst IS

END fibonacci\_vhd\_tst;

ARCHITECTURE fibonacci\_arch OF fibonacci\_vhd\_tst IS

-- Signals

SIGNAL clk : STD\_LOGIC := '0';

SIGNAL dec\_out : INTEGER;

-- Component declaration

COMPONENT fibonacci

PORT (

clk : IN STD\_LOGIC;

dec\_out : OUT INTEGER

);

END COMPONENT;

-- File for results

FILE file\_RESULTS : text;

BEGIN

i1 : fibonacci

PORT MAP (

clk => clk,

dec\_out => dec\_out

);

init : PROCESS

VARIABLE line\_buffer : line; -- A line buffer for textio

BEGIN

-- Open the file in write mode

file\_open(file\_RESULTS, "output\_results.txt", WRITE\_MODE);

WAIT FOR 20 ns; -- Wait for initial setup

WHILE TRUE LOOP

WAIT FOR 20 ns; -- Wait for each clock cycle

-- Write the data\_out values to the line buffer

WRITE(line\_buffer, dec\_out);

-- Write the line buffer to the file

WRITELINE(file\_RESULTS, line\_buffer);

END LOOP;

-- Close the file when done

file\_close(file\_RESULTS);

WAIT;

END PROCESS init;

always : PROCESS

BEGIN

LOOP

clk <= '1';

WAIT FOR 10 ns;

clk <= '0';

WAIT FOR 10 ns;

END LOOP;

WAIT;

END PROCESS always;

END fibonacci\_arch;