ディジタル回路　演習1

担当教員 : 今井 正治

提出日 : 2015/11/16

提出者 : 中村 真也

所属 : 基礎工学部 情報科学科 計算機コース 2 回生

連絡先 : u110864b@ecs.osaka-u.ac.jp

1.半加算器

library IEEE;

use IEEE.std\_logic\_1164.all;

entity half\_adder is

port( x\_in: in std\_logic;

y\_in: in std\_logic;

c\_out:out std\_logic;

s\_out:out std\_logic );

end entity half\_adder;

---------------------------------------------

-- Behavior Model

---------------------------------------------

architecture behavior of half\_adder is

begin

CARRY: process (x\_in,y\_in)

begin

if( x\_in='1' and y\_in='1') then

c\_out <= '1';

else

c\_out <= '0';

end if;

end process;

SUM: process (x\_in,y\_in)

begin

if(( x\_in='0' and y\_in='1')or

( x\_in='1' and y\_in='0')) then

s\_out <= '1';

else

s\_out <= '0';

end if;

end process;

end architecture behavior;

----------------------------------------------

-- Dataflow Model

-----------------------------------------------

architecture dataflow of half\_adder is

begin

c\_out <= x\_in and y\_in;

s\_out <= x\_in xor y\_in;

end architecture dataflow;

2.全加算器

library IEEE;

use IEEE.std\_logic\_1164.all;

entity full\_adder is

port( x\_in: in std\_logic;

y\_in: in std\_logic;

z\_in: in std\_logic;

c\_out: out std\_logic;

s\_out: out std\_logic );

end entity full\_adder;

------------------------

-- Behavior Model

-----------------------

architecture behavior of full\_adder is

begin

CARRY: process(x\_in, y\_in, z\_in)

begin

if (( x\_in = '0' and y\_in='1' and z\_in = '1') or

( x\_in = '1' and y\_in='0' and z\_in = '1') or

( x\_in = '1' and y\_in='1' and z\_in = '0') or

(x\_in = '1' and y\_in='1' and z\_in = '1') ) then

c\_out <= '1';

else

c\_out <= '0';

end if ;

end process;

SUM: process(x\_in, y\_in, z\_in)

begin

if( ( x\_in = '0' and y\_in = '0' and z\_in = '1') or

( x\_in = '0' and y\_in = '1' and z\_in = '0') or

( x\_in = '1' and y\_in = '0' and z\_in = '0') or

( x\_in = '1' and y\_in = '1' and z\_in = '1') ) then

s\_out <= '1';

else

s\_out <= '0';

end if;

end process;

end architecture;

-------------------------------------

--Dataflow Model

---------------------------------------

architecture dataflow of full\_adder is

begin

c\_out <= ( x\_in and y\_in ) or

( x\_in and z\_in ) or

( y\_in and z\_in );

s\_out <= x\_in xor y\_in xor z\_in;

end architecture dataflow;

----------------------------------------

-- Structure Model

----------------------------------------

architecture structure of full\_adder is

component half\_adder is

port( x\_in: in std\_logic;

y\_in: in std\_logic;

c\_out: out std\_logic;

s\_out: out std\_logic );

end component half\_adder;

signal sum\_1: std\_logic;

signal carry\_1: std\_logic;

signal carry\_2: std\_logic;

begin

HA\_1: half\_adder

port map(

x\_in => x\_in,

y\_in => y\_in,

c\_out => carry\_1,

s\_out => sum\_1 );

HA\_2: half\_adder

port map(

x\_in => sum\_1,

y\_in => z\_in,

c\_out => carry\_2,

s\_out => s\_out );

c\_out <= carry\_1 or carry\_2;

end architecture structure;

3.全加算器テストベンチ

library IEEE;

use IEEE.std\_logic\_1164.all;

entity test\_full\_adder is

end entity test\_full\_adder;

architecture test\_bench of test\_full\_adder is

component full\_adder is

port (

x\_in: in std\_logic;

y\_in: in std\_logic;

z\_in: in std\_logic;

c\_out: out std\_logic;

s\_out: out std\_logic );

end component full\_adder;

constant PERIOD: time := 10 ns;

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

signal done: boolean := FALSE;

signal valid: boolean := TRUE;

signal x\_in: std\_logic;

signal y\_in: std\_logic;

signal z\_in: std\_logic;

signal c\_exp: std\_logic;

signal s\_exp: std\_logic;

signal c\_out\_b: std\_logic;

signal s\_out\_b: std\_logic;

signal c\_out\_d: std\_logic;

signal s\_out\_d: std\_logic;

signal c\_out\_s: std\_logic;

signal s\_out\_s: std\_logic;

for FADD\_B: full\_adder use entity work.full\_adder( behavior );

for FADD\_D: full\_adder use entity work.full\_adder( dataflow );

for FADD\_S: full\_adder use entity work.full\_adder( structure );

begin

FADD\_B: full\_adder port map (

x\_in => x\_in,

y\_in => y\_in,

z\_in => z\_in,

c\_out => c\_out\_b,

s\_out => s\_out\_b );

FADD\_D: full\_adder port map (

x\_in => x\_in,

y\_in => y\_in,

z\_in => z\_in,

c\_out => c\_out\_d,

s\_out => s\_out\_d );

FADD\_S: full\_adder port map (

x\_in => x\_in,

y\_in => y\_in,

z\_in => z\_in,

c\_out => c\_out\_s,

s\_out => s\_out\_s );

CLK\_GEN: process

variable clktmp: std\_logic := '0';

begin

clock <= clktmp;

clktmp := not clktmp;

wait for PERIOD/2;

if ( done ) then

wait;

end if;

end process CLK\_GEN;

STIMULUS1: process

type test\_vec\_t is record

x\_in: std\_logic;

y\_in: std\_logic;

z\_in: std\_logic;

c\_exp: std\_logic;

s\_exp: std\_logic;

end record;

type tt\_type is array( natural range <> ) of test\_vec\_t;

constant truth\_table: tt\_type :=

--x\_in y\_in z\_in c\_exp s\_exp

( ( '0', '0', '0', '0', '0' ),

( '0', '0', '1', '0', '1' ),

( '0', '1', '0', '0', '1' ),

( '0', '1', '1', '1', '0' ),

( '1', '0', '0', '0', '1' ),

( '1', '0', '1', '1', '0' ),

( '1', '1', '0', '1', '0' ),

( '1', '1', '1', '1', '1' ) );

begin

for i in truth\_table'range loop

x\_in <= truth\_table( i ).x\_in;

y\_in <= truth\_table( i ).y\_in;

z\_in <= truth\_table( i ).z\_in;

c\_exp <= truth\_table( i ).c\_exp;

s\_exp <= truth\_table( i ).s\_exp;

wait until rising\_edge( clock );

valid <= ( c\_out\_b = c\_exp ) and ( s\_out\_b = s\_exp ) and

( c\_out\_d = c\_exp ) and ( s\_out\_d = s\_exp ) and

( c\_out\_s = c\_exp ) and ( s\_out\_s = s\_exp );

assert( ( c\_out\_b = c\_exp ) and ( s\_out\_b = s\_exp ) )

report "Error at full\_adder ( behavior )";

assert( ( c\_out\_d = c\_exp ) and ( s\_out\_d = s\_exp ) )

report "Error at full\_adder ( dataflow )";

assert( ( c\_out\_s = c\_exp ) and ( s\_out\_s = s\_exp ) )

report "Error at full\_adder ( structure )";

end loop;

done <= TRUE;

wait;

end process STIMULUS1;

end architecture test\_bench;

4.実行結果

