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

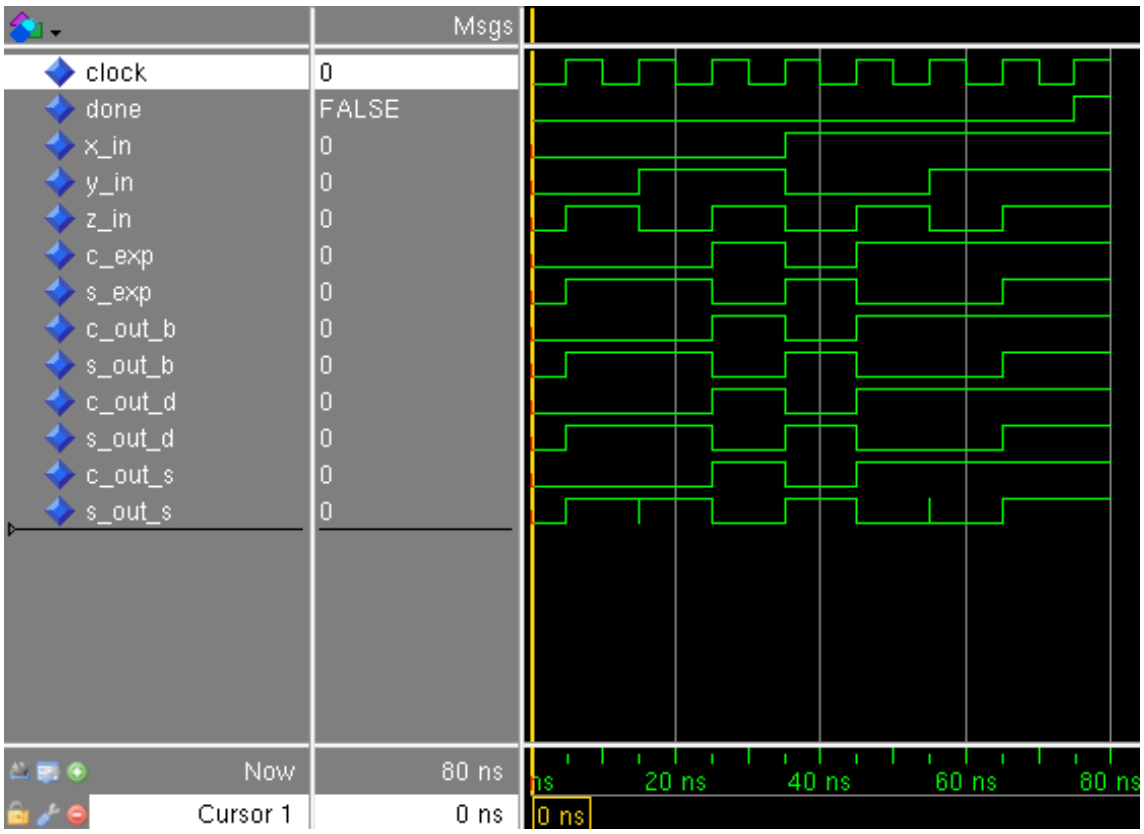
```
library IEEE;
use IEEE.std_logic_1164.all;
entity test_full_adder is
end entity test_full_adder;
```

```
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


Msgs
clock
done
x_in
y_in
z_in
c_exp
s_exp
c_out_b
s_out_b
c_out_d
s_out_d
c_out_s
s_out_s
0
FALSE
0
0
0
0
0
0
0
0
0
0
0
Now
80 ns
Cursor 1
0 ns
ns
20 ns
40 ns
60 ns
80 ns
0 ns