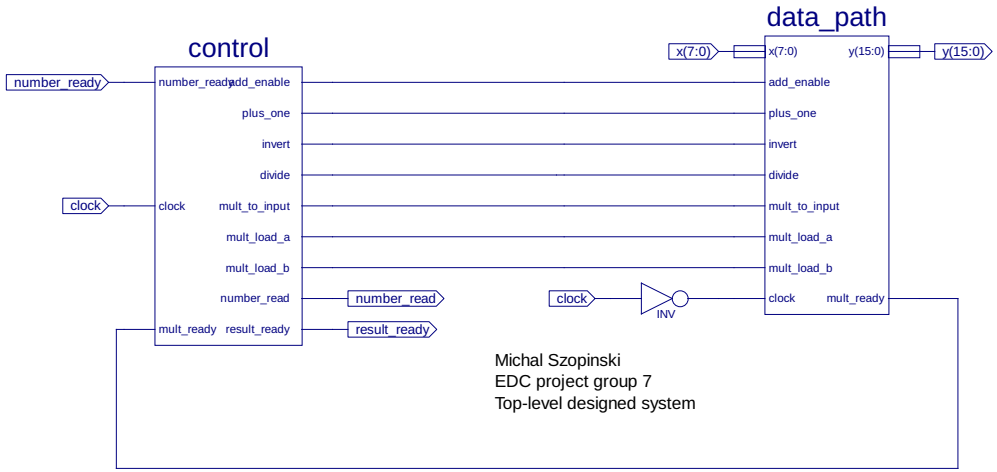
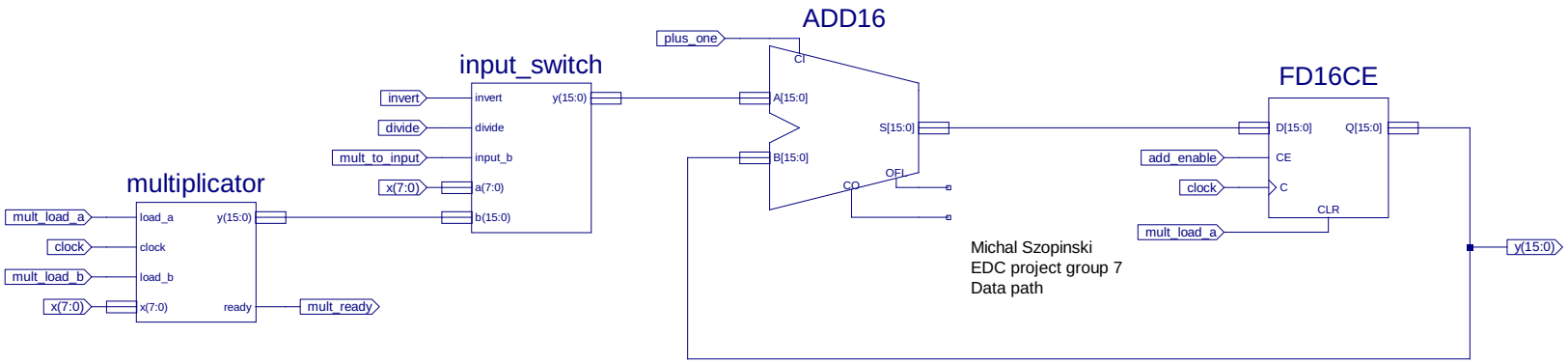
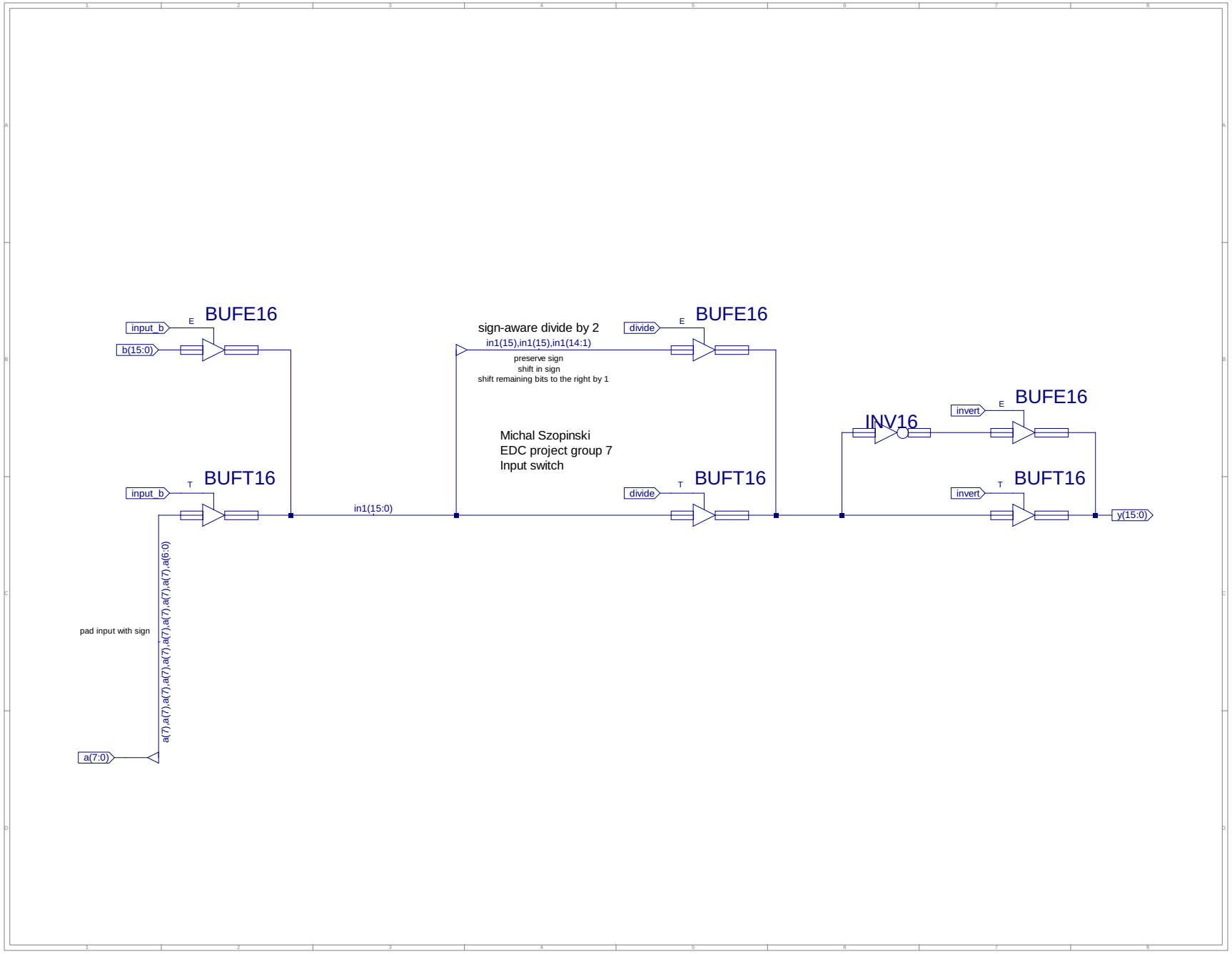
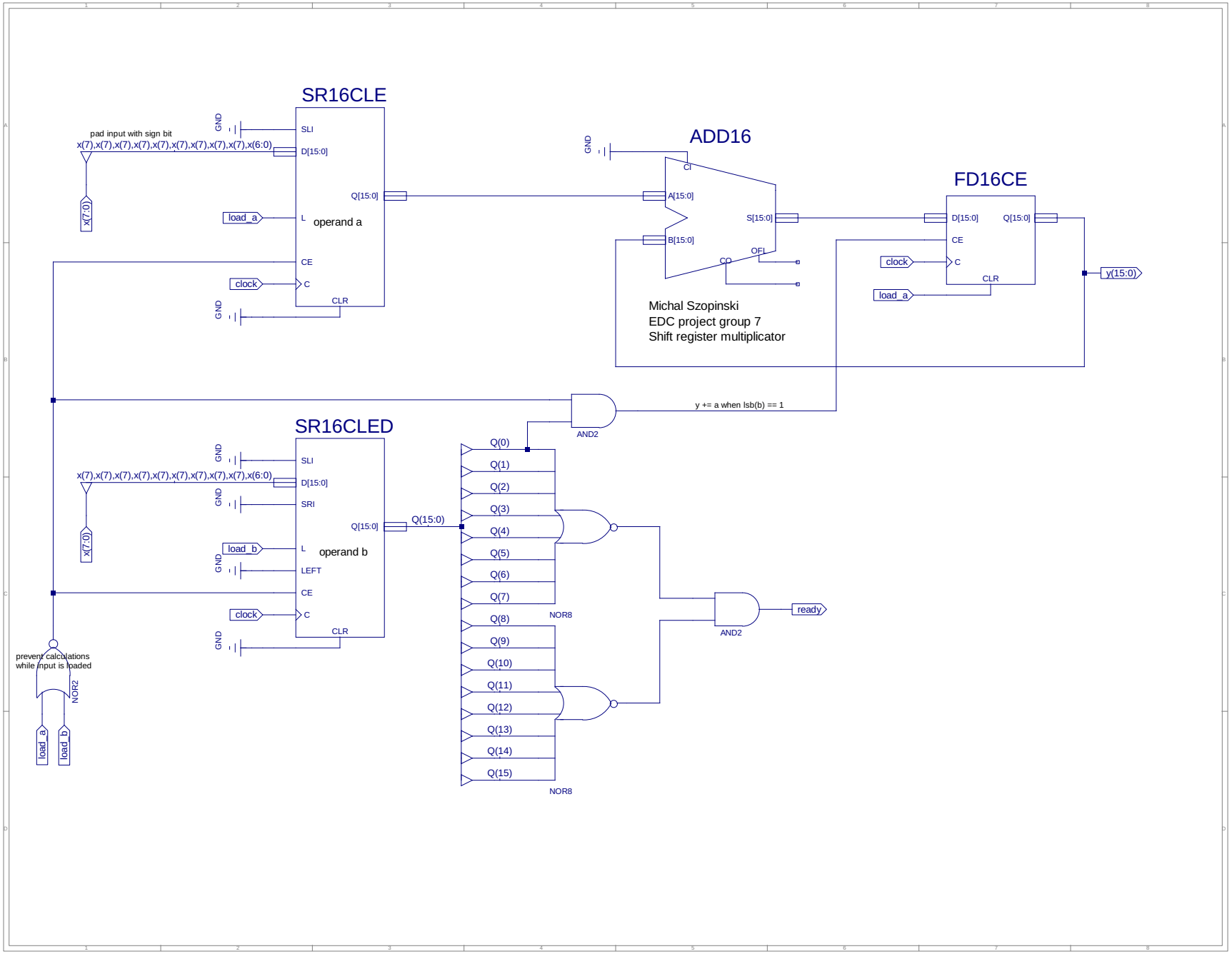


Michal Szopinski
EDC project group 7
Top-level designed system



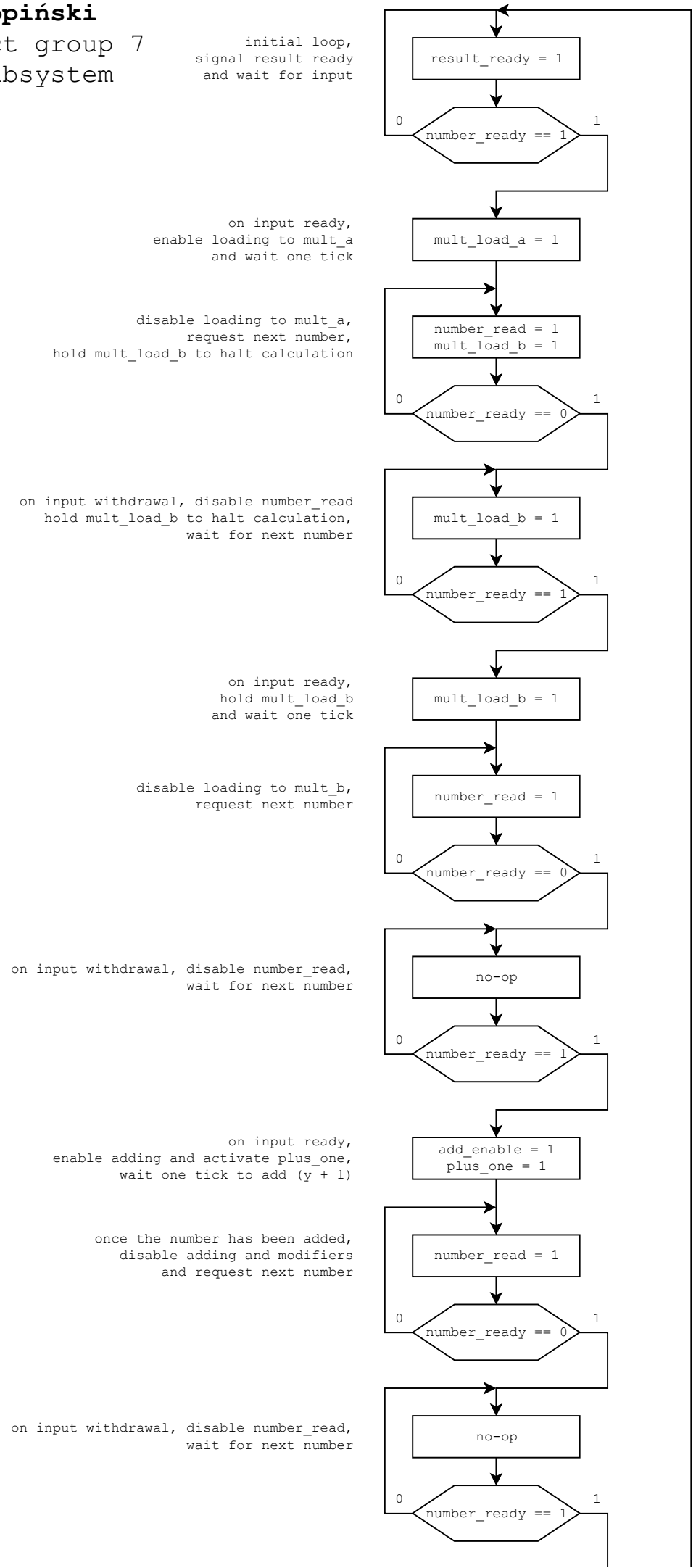
Michal Szopinski
EDC project group 7
Data path

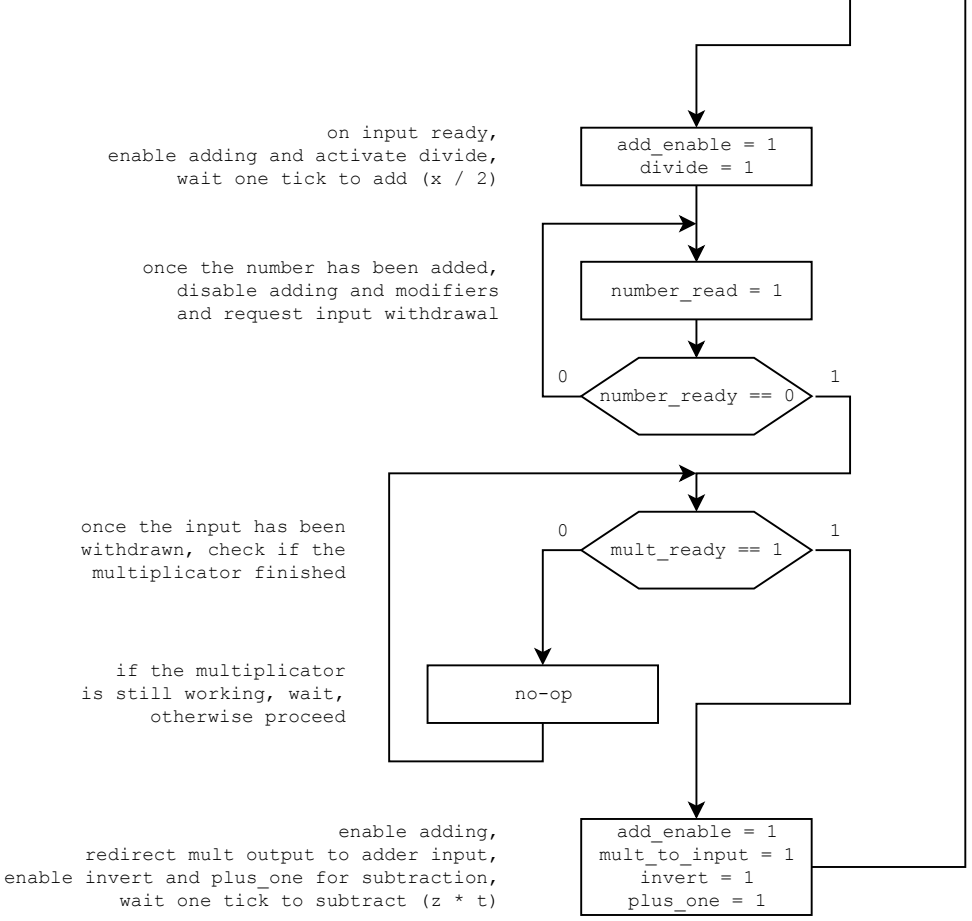

BUFE16
input_b
E
b(15:0)
sign-aware divide by 2
in1(15),in1(15),in1(14:1)
preserve sign
shift in sign
shift remaining bits to the right by 1
divide
E
BUFE16
Michal Szopinski
EDC project group 7
Input switch
INV16
invert
E
BUFE16
BUFT16
input_b
T
in1(15:0)
divide
T
BUFT16
invert
T
BUFT16
y(15:0)
pad input with sign
a(7),a(7),a(7),a(7),a(7),a(7),a(7),a(7),a(7),a(6:0)
a(7:0)


SR16CLE
pad input with sign bit
x(7),x(7),x(7),x(7),x(7),x(7),x(7),x(7),x(7),x(6:0)
GND
SLI
D[15:0]
Q[15:0]
load_a
L
operand a
CE
clock
C
CLR
GND
x(7:0)
ADD16
GND
CI
A[15:0]
B[15:0]
S[15:0]
OFL
CO
FD16CE
D[15:0]
Q[15:0]
CE
clock
C
CLR
load_a
y(15:0)
Michal Szopinski
EDC project group 7
Shift register multiplicator
y += a when lsb(b) == 1
AND2
SR16CLED
x(7),x(7),x(7),x(7),x(7),x(7),x(7),x(7),x(7),x(6:0)
SRI
load_b
operand b
LEFT
Q(15:0)
Q(0)
Q(1)
Q(2)
Q(3)
Q(4)
Q(5)
Q(6)
Q(7)
Q(8)
Q(9)
Q(10)
Q(11)
Q(12)
Q(13)
Q(14)
Q(15)
NOR8
NOR8
AND2
ready
prevent calculations
while input is loaded
NOR2
load_a
load_b

Michał Szopiński

EDC project group 7 Control subsystem flowchart



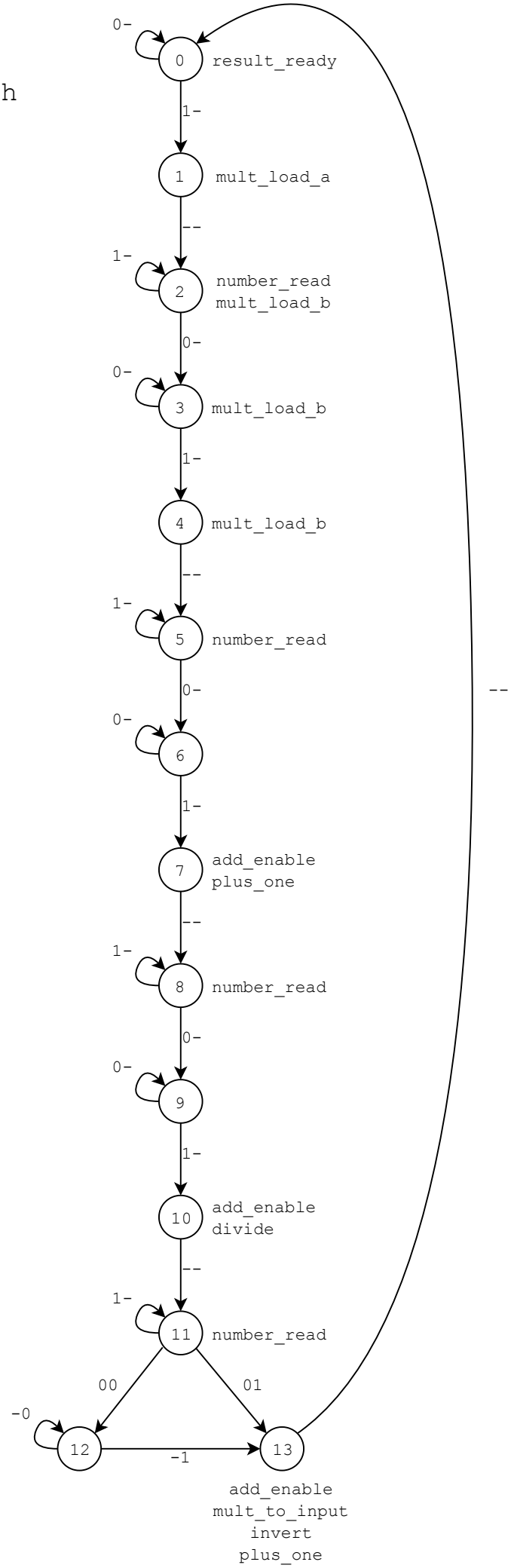

on input ready,
enable adding and activate divide,
wait one tick to add (x / 2)
add_enable = 1
divide = 1
once the number has been added,
disable adding and modifiers
and request input withdrawal
number_read = 1
0
number_ready == 0
1
once the input has been
withdrawn, check if the
multiplicator finished
0
mult_ready == 1
1
if the multiplicator
is still working, wait,
otherwise proceed
no-op
enable adding,
redirect mult output to adder input,
enable invert and plus_one for subtraction,
wait one tick to subtract (z * t)
add_enable = 1
mult_to_input = 1
invert = 1
plus_one = 1

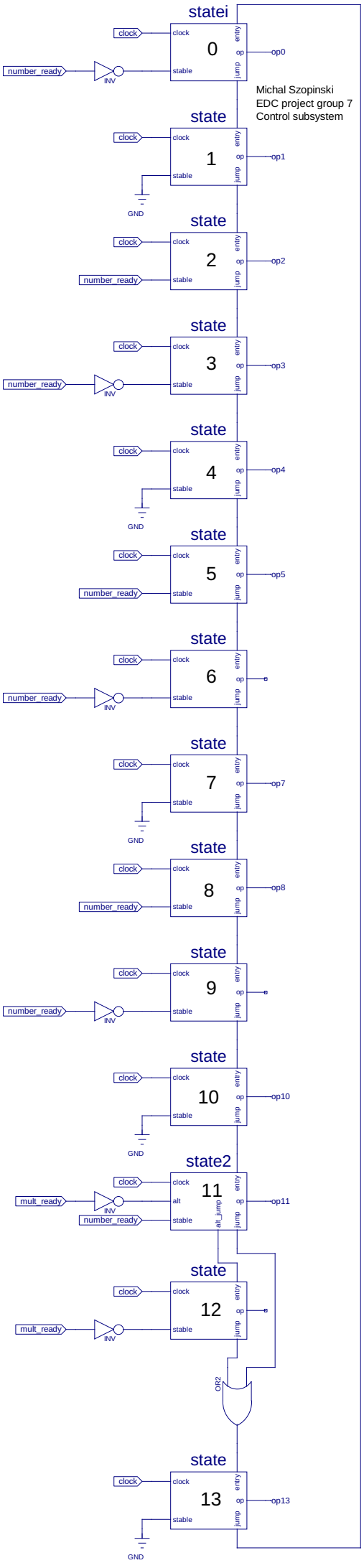
Michał Szopiński

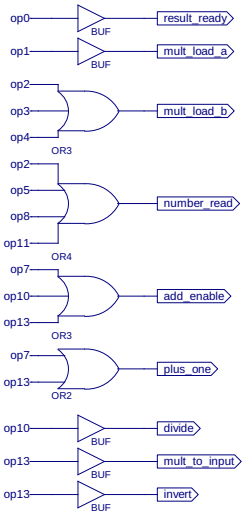
EDC project group 7
Control subsystem
state transition graph

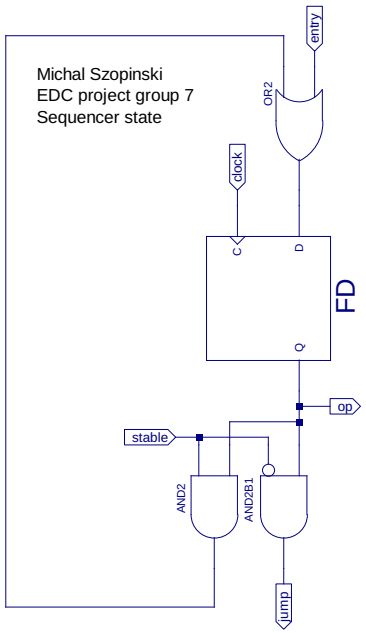
Inputs are:
number_ready
mult_ready

Text next to node denotes
active outputs.

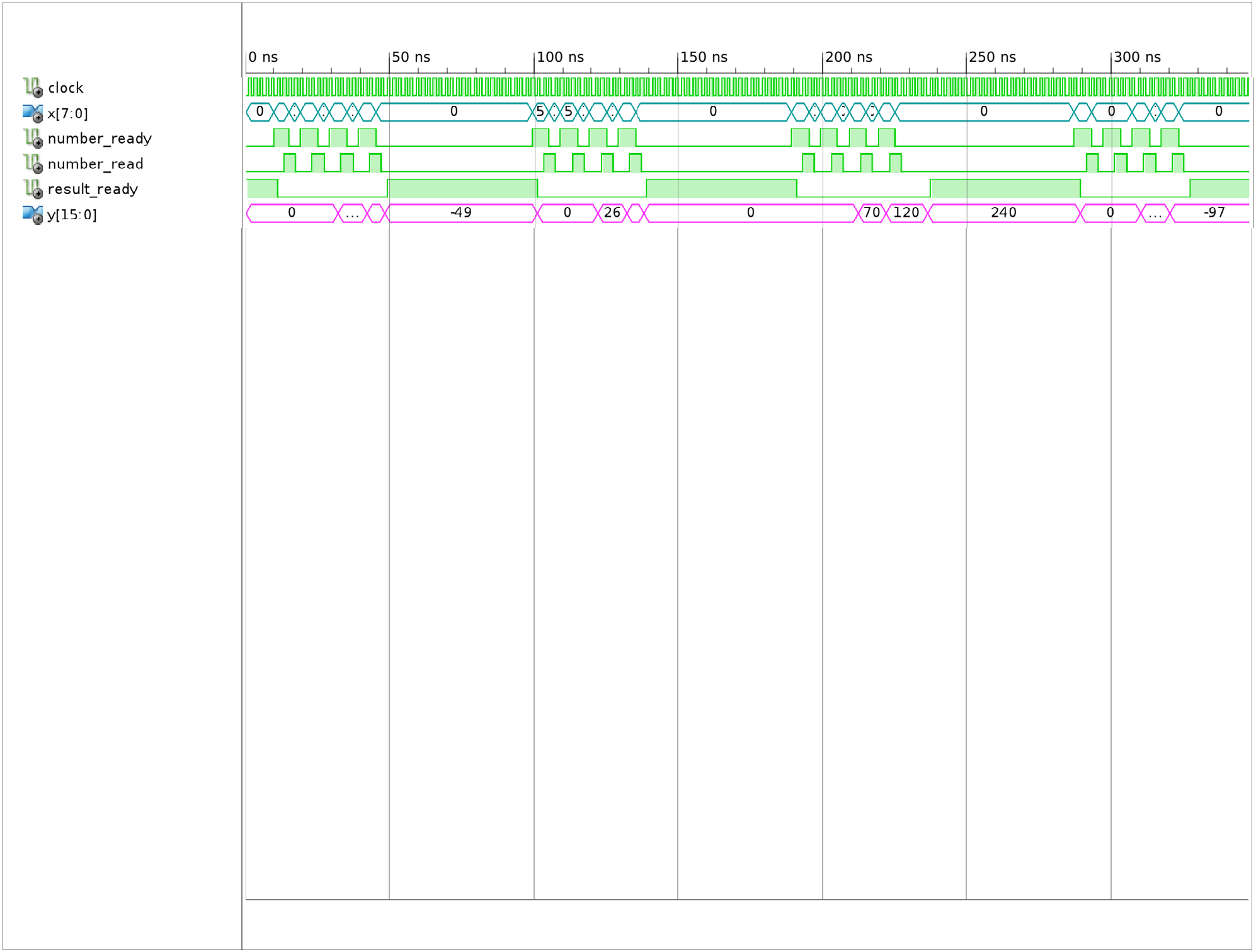



statei
clock
0
entry
op
jump
stable
op0
number_ready
INV
Michal Szopinski
EDC project group 7
Control subsystem
state
1
op1
GND
2
op2
3
op3
4
op4
5
op5
6
7
op7
8
op8
9
10
op10
state2
11
alt
alt_jump
op11
mult_ready
12
13
op13
OR2


op0
BUF
result_ready
op1
mult_load_a
op2
op3
op4
OR3
mult_load_b
op5
op8
op11
OR4
number_read
op7
op10
op13
add_enable
plus_one
OR2
divide
mult_to_input
invert


Michal Szopinski
EDC project group 7
Sequencer state
entry
OR2
clock
C
D
FD
Q
op
stable
AND2
AND2B1
jump

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY system_system_sch_tb IS
END system_system_sch_tb;
ARCHITECTURE behavioral OF system_system_sch_tb IS

   COMPONENT system
   PORT( clock  :   IN  STD_LOGIC;
         number_read   :   OUT STD_LOGIC;
         result_ready  :   OUT STD_LOGIC;
         number_ready  :   IN  STD_LOGIC;
         x :   IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
         y :   OUT STD_LOGIC_VECTOR (15 DOWNTO 0));
   END COMPONENT;

   SIGNAL clock :   STD_LOGIC := '0';
   SIGNAL number_read   :   STD_LOGIC;
   SIGNAL result_ready  :   STD_LOGIC;
   SIGNAL number_ready  :   STD_LOGIC := '0';
   SIGNAL x :   STD_LOGIC_VECTOR (7 DOWNTO 0) := "00000000";
   SIGNAL y :   STD_LOGIC_VECTOR (15 DOWNTO 0);

BEGIN

   UUT: system PORT MAP(
        clock => clock,
        number_read => number_read,
        result_ready => result_ready,
        number_ready => number_ready,
        x => x,
        y => y
   );

    clock <= not clock after 1 ns;

-- *** Test Bench - User Defined Section ***
   tb : PROCESS

    procedure Feed_number(
        constant num : in std_logic_vector) is
    begin
        x <= num;
        number_ready <= '1';
        wait until number_read = '1';
        wait for 2 ns; -- comment out wait for faster operation
        number_ready <= '0';
        x <= "00000000";
        wait until number_read = '0';
        wait for 2 ns;
    end procedure;

    procedure Calculate_f(
        constant num_x : in std_logic_vector;
        constant num_y : in std_logic_vector;
        constant num_z : in std_logic_vector;
        constant num_t : in std_logic_vector) is
    begin
        Feed_number(num_t);
        Feed_number(num_z);
        Feed_number(num_y);
        Feed_number(num_x);

        if result_ready = '0' then
            wait until result_ready = '1';
        end if;
    end procedure;

   BEGIN
      wait for 10 ns;

        -- TEST CASE 1: 1 + 97 / 2 + 123 - 13 * 17 = -49 (11001111)
        -- (General test)
        Calculate_f("01100001", "01111011", "00001101", "00010001");
        wait for 50 ns;

        -- TEST CASE 2: 1 + (-1) / 2 + 25 - 5 * 5 = 0 (00000000)
        -- (Dividing -1 by 2 returns -1)
        Calculate_f("11111111", "00011001", "00000101", "00000101");
        wait for 50 ns;

        -- TEST CASE 3: 1 + 101 / 2 + 69 - (-1) * 120 = 240 (00000000)
        -- (Multiplying by -1 takes a long time)
        Calculate_f("01100101", "01000101", "11111111", "01111000");
        wait for 50 ns;

        -- TEST CASE 4: 1 + (-53) / 2 + (-71) - 0 * 42 = -97 (10011111)
        -- (Multiplying by 0 is instant)
        Calculate_f("11001011", "10111001", "00000000", "00101010");
        wait for 50 ns;

        WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```


0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
clock
x[7:0]
number_ready
number_read
result_ready
y[15:0]
0
0
5
5
0
0
0
0
0
-49
0
26
0
70
120
240
0
-97