RADD🡨0, WADD🡨0

S0

S1

RADD=2

W

S2

reset

1

RADD🡨RADD + 1, WADD🡨WADD +1

0

**Controller**

go

1

0

WADD=PWADD

DIN🡨PDOUT

+1

DIN🡨DOUT

+1

1

0

Delay=0

Delay🡨 Delay - 1

0

1

RADD🡨0

S3

SIGN\_i 🡨 DOUT(3 downto 0),  
RADD🡨RADD+1

S4

MOST\_i🡨"0000"&DOUT(3 downto 0),  
RADD🡨RADD+1

S5

LEAST\_i🡨 DOUT(3 downto 0), i\_a<="1010";  
RADD🡨RADD+1;Delay🡨50

S6

MOST\_i<= MOST\_i+P\_big;

(send to array multiplier)

S7

MOST\_i has full input in binary

S8

SIGN\_i==x"2D" (-)

0

1

Angle<= MOST\_i;

S9

S15

DIN<="0011"&D0; i\_WADD<=i\_WADD+"00001";

S14

DIN<="0011"&D1; i\_WADD<=i\_WADD+"00001";

S13

DIN<="0011"&D2; i\_WADD<=i\_WADD+"00001";

DIN<="0011"&D3; i\_WADD<=i\_WADD+"00001";

S12

S11

BV<=sin\_i ; i\_WADD<=i\_WADD+"0010";

S10

Cos\_i<=cos, sin\_i<=sin

**CONTROLLER**

library ieee;

use ieee.std\_logic\_1164.ALL;

use ieee.std\_logic\_arith.ALL;

use ieee.std\_logic\_signed.ALL;

use ieee.numeric\_std.all;

entity controller is

port( clk : in STD\_LOGIC;

DIN : out unsigned (7 downto 0);

W : out STD\_LOGIC;

WADD : out unsigned(4 downto 0);

DOUT : in unsigned(7 downto 0);

RADD : out unsigned(4 downto 0);

reset : in STD\_LOGIC;

PWADD : in unsigned(4 downto 0);

PDOUT : in unsigned(7 downto 0);

cout : in std\_logic;

sub : out std\_logic;

a,b : out unsigned(3 downto 0);

P\_big: in unsigned(7 downto 0);

go : in std\_logic;

led :out unsigned(4 downto 0));

end controller;

architecture controllerA of controller is

--signal counter: natural range 0 to 24;

signal i\_WADD,i\_RADD,state : unsigned( 4 downto 0 ):="00000";

signal i\_a,SIGN\_i,LEAST\_i : signed( 3 downto 0 ):="0000";

signal MOST\_i : signed(7 downto 0):="00000000"; --i\_r

signal Delay : unsigned( 5 downto 0 ):="000000";

--signal i\_W : std\_logic:='0';

begin

process

begin

led<=state;

--led<='0'&PD0;

wait until (clk'event and clk='1');

if (reset='1') then

state<="00000";

else

case state is

when "00000"=> i\_RADD<="00000"; i\_WADD<="00000"; state<="00001";

when "00001"=> if (i\_WADD=PWADD) then

DIN<=PDOUT; state<="00010";

else

DIN<=DOUT; state<="00010";

end if;

--when "00010"=> state<=state+1; --"00000"

when "00010"=> if (i\_RADD="0010") then

if(go='1') then

state<="00011";

else

state<="00000";

end if;

else

i\_RADD<=i\_RADD+"00001"; i\_WADD<=i\_WADD+"00001"; state<="00001"; --i\_WADD

end if;

when "00011"=> i\_RADD<="00000"; state<="00100"; --next state

when "00100"=> SIGN\_i<=DOUT(3 downto 0); i\_RADD<=i\_RADD+"00001"; state<="00101"; --DOUT(3 downto 0)

when "00101"=> MOST\_i<="0000"&DOUT(3 downto 0); i\_RADD<=i\_RADD+"00001"; state<="11001";

when "00110"=> LEAST\_i<=DOUT(3 downto 0); i\_RADD<=i\_RADD+"00001"; i\_a<="1010"; state<="00111";

when "00111"=> MOST\_i<= MOST\_i+P\_big; state<="01000";

when "01000"=>if (i\_RADD="0010") then

state<="10101";

else

state<="01001"; --i\_WADD

end if;

when "01001"=> angle<=MOST\_i; state<="01010";

when "01010"=> cos\_i<=cos; sin\_i<=sin; state<="01011";

when "01011"=> BV<=sin\_i; i\_WADD<=i\_WADD+"00100"; state<="01100";

when "01100"=> DIN<="0011"&D3; i\_WADD<=i\_WADD+"00001"; state<="01101";

when "01101"=> state<="01110"; --13

when "01110"=> DIN<="0011"&D2; i\_WADD<=i\_WADD+"00001"; state<="01111";

when "01111"=> state<="10000"; --15

when "10000"=> DIN<="0011"&D1; i\_WADD<=i\_WADD+"00001"; state<="10001";

when "10001"=> state<="10010"; --17

when "10010"=> DIN<="0011"&D0; i\_WADD<=i\_WADD+"00001"; state<="10011";

when "10011"=> state<="10100"; --19

when "10100"=> BV<=sin\_i; i\_WADD<=i\_WADD+"00100"; state<="10101";

when "10101"=> DIN<="0011"&D3; i\_WADD<=i\_WADD+"00001"; state<="10110";

when "10110"=> state<="10111"; --22

when "10111"=> DIN<="0011"&D2; i\_WADD<=i\_WADD+"00001"; state<="11000";

when "11000"=> state<="11001"; --24

when "11001"=> DIN<="0011"&D1; i\_WADD<=i\_WADD+"00001"; state<="11010";

when "11010"=> state<="11011"; --26

when "11011"=> DIN<="0011"&D0; i\_WADD<=i\_WADD+"00001"; state<="11100";

when "11100"=> state<="00000"; --28

when others => state<="00000";

end case;

end if;

end process;

W<='1' when (--state="00001" --1

state="00010" --2

--or state="00011" --3

--or state="01001" --9

--or state="01101" --11

or state="01101" --13

or state="01111" --15

or state="10001" --17

or state="10011" --19

or state="10110" --22

or state="11000"--23

or state="11010" --26

or state="11100")--28

--or (state>="01001" and state<="11001"))--9

--or state="01011" --11

--or state="01110" --14

--or state="10010" --18

--or state="10100" --20

--or state="10110" --22

--or state<="11001")--24

else '0';

WADD<=i\_WADD; RADD<=i\_RADD; a<=i\_a; b<=MOST\_i;

end controllerA;

**B2BCD**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

--use ieee.std\_logic\_arith.ALL;

use ieee.std\_logic\_signed.ALL;

use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity B2BCD is

port (

go,reset, clk:in std\_logic;

done:out std\_logic;

BV: in signed(12 downto 0);

D0,D1,D2,D3: out signed(3 downto 0));

end B2BCD;

architecture Behavioral of B2BCD is

signal state: unsigned (2 downto 0):="000";

signal R: signed (12 downto 0):="0000000000000";

signal TD: signed (3 downto 0):="0000";

begin

process

begin

wait until (clk'event and clk='1');

if (reset='1') then

state<="000";

else

case state is

when "000"=> if(go='1') then

R<=BV; TD<="0000"; state<="001";

else

state<="000";

end if;

when "001"=> if(R<"0001111100111") then --R<999

TD<="0000";

D3<=TD;

state<="010";

else

R<=R-"0001111101000";

TD<=TD+"0001";

state<="001";

end if;

when "010"=> if(R<"0000001100011") then --R<99

TD<="0000";

D2<=TD;

state<="011";

else

R<=R-"0000001100100";

TD<=TD+"0001";

state<="010";

end if;

when "011"=> if(R<"0000000001001") then --R<9

D0<=R(3 downto 0);

D1<=TD;

state<="100";

else

R<=R-"0000000001010";

TD<=TD+"0001";

state<="011";

end if;

when "100"=> done<='1'; state<="000";

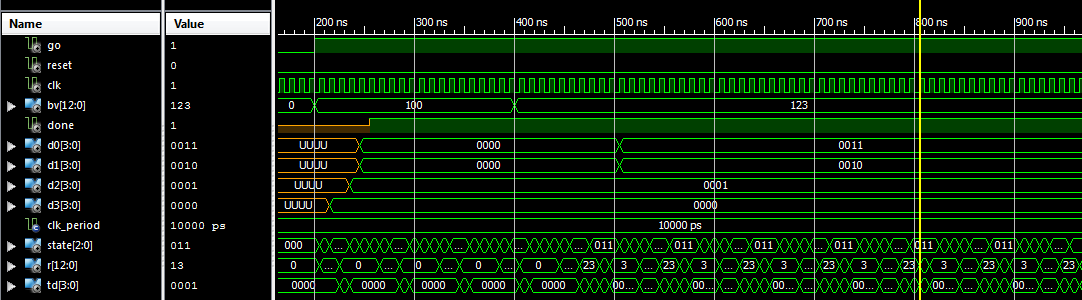
when others=> state<="000";

end case;

end if;

end process;

end Behavioral;



MOST\_i<=signed(temp+1);

S22

S21

Tmp<= not MOST\_i;

BV<=cos\_i ; i\_WADD<=i\_WADD+"00001";

S16

DIN<="0011"&D3; i\_WADD<=i\_WADD+"00001";

S17

DIN<="0011"&D2; i\_WADD<=i\_WADD+"00001";

S18

DIN<="0011"&D1; i\_WADD<=i\_WADD+"00001";

S19

DIN<="0011"&D0; i\_WADD<=i\_WADD+"00001";

S20