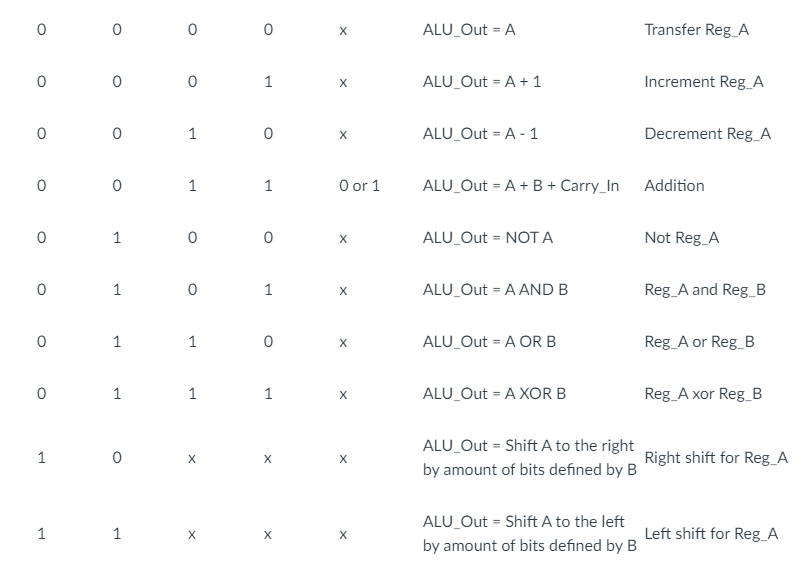
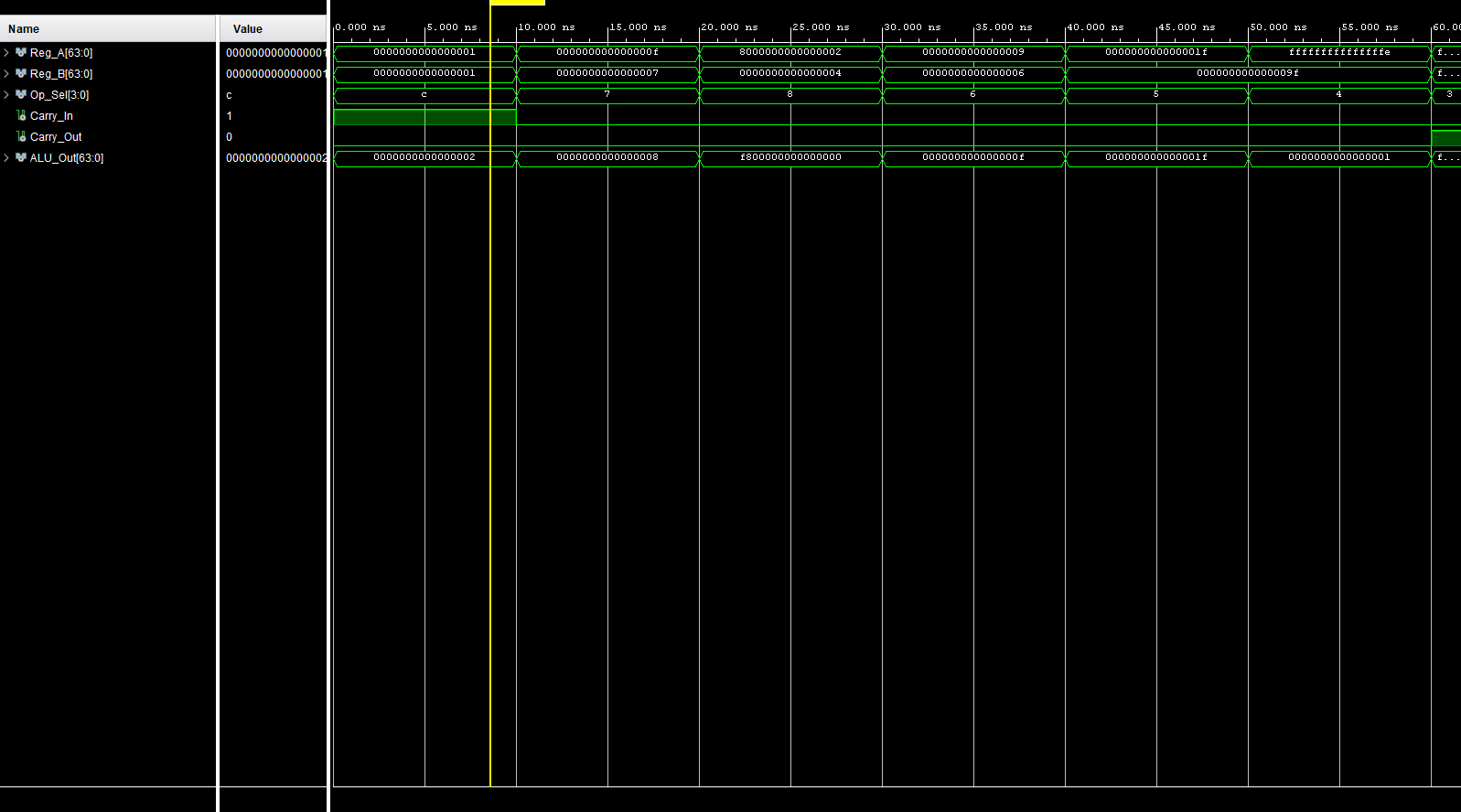
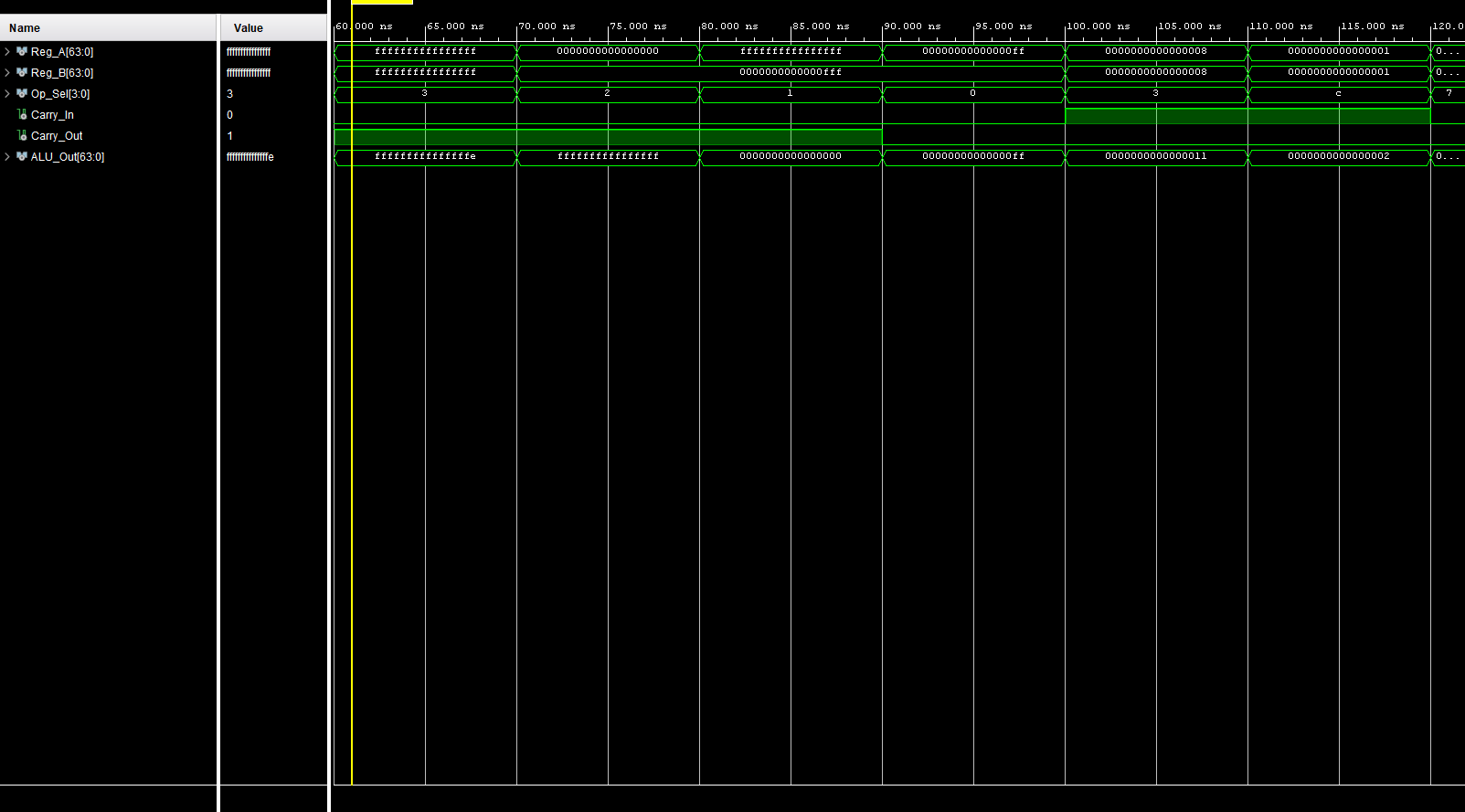
OP 3 OP2 OP1 OP0 Cin



HEX CODES

|  |  |
| --- | --- |
| 0 | Transfer A |
| 1 | Increment A |
| 2 | Decrement A |
| 3 | ADDer |
| 4 | NOT |
| 5 | AND |
| 6 | OR |
| 7 | XOR |
| 8 + X | Shifter right |
| C + X | Shifter left |





--------ALU\_SIM---------------

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

-- Company:

-- Engineer:

--

-- Create Date: 07/02/2024 06:31:22 PM

-- Design Name:

-- Module Name: ALU64Bit\_Sim - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ALU64Bit\_Sim is

-- Port ( );

end ALU64Bit\_Sim;

architecture Behavioral of ALU64Bit\_Sim is

component ALU64Bit is

Port (

Reg\_A : in std\_logic\_vector(63 downto 0); -- register 1 UNSIGNED binary

Reg\_B : in std\_logic\_vector(63 downto 0); -- register 2 UNSIGNED binary

Op\_Sel : in std\_logic\_vector(3 downto 0); -- operator selector, only 10 operations exist

Carry\_In : in std\_logic; -- in bit

Carry\_Out : out std\_logic; -- out bit

ALU\_Out : out std\_logic\_vector(63 downto 0)

);

end component;

signal Reg\_A : std\_logic\_vector(63 downto 0);

signal Reg\_B : std\_logic\_vector(63 downto 0);

signal Op\_Sel : std\_logic\_vector(3 downto 0);

signal Carry\_In : std\_logic;

signal Carry\_Out : std\_logic;

signal ALU\_Out : std\_logic\_vector(63 downto 0);

begin

UUT : ALU64Bit port map( reg\_a, reg\_b, op\_sel, Carry\_in, Carry\_out, ALU\_OUT );

process begin

-- regs are 16 hex size

op\_sel <= x"C"; reg\_a <= x"0000000000000001" ; reg\_b <= x"0000000000000001"; carry\_IN <= '1'; -- shifter op logical expected 2

wait for 10 ns;

op\_sel <= x"7"; reg\_a <= x"000000000000000F" ; reg\_b <= x"0000000000000007"; carry\_IN <= '0'; -- XOR op arithmatic expected unsigned number with 1 bit high 1 bit low

wait for 10 ns;

op\_sel <= x"8"; reg\_a <= x"8000000000000002" ; reg\_b <= x"0000000000000004"; carry\_IN <= '0'; -- shifter op arithmatic expected unsigned number with 1 bit high 1 bit low

wait for 10 ns;

op\_sel <= x"6"; reg\_a <= x"0000000000000009" ; reg\_b <= x"0000000000000006"; carry\_IN <= '0'; -- OR op expected

wait for 10 ns;

op\_sel <= x"5"; reg\_a <= x"000000000000001f" ; reg\_b <= x"000000000000009f"; carry\_IN <= '0'; -- AND op expected f expect 1F

wait for 10 ns;

op\_sel <= x"4"; reg\_a <= x"fffffffffffffffe" ; reg\_b <= x"000000000000009f"; carry\_IN <= '0'; -- NOT op expected f expect 1

wait for 10 ns;

op\_sel <= x"3"; reg\_a <= x"fffffffffffffffF" ; reg\_b <= x"fffffffffffffffF"; carry\_IN <= '0'; -- adder op expected f expect

wait for 10 ns;

op\_sel <= x"2"; reg\_a <= x"0000000000000000" ; reg\_b <= x"0000000000000fff"; carry\_IN <= '0'; -- decrement op expected overflow

wait for 10 ns;

op\_sel <= x"1"; reg\_a <= x"ffffffffffffffff" ; reg\_b <= x"0000000000000fff"; carry\_IN <= '0'; -- increment op expected overflow

wait for 10 ns;

op\_sel <= x"0"; reg\_a <= x"00000000000000ff" ; reg\_b <= x"0000000000000fff"; carry\_IN <= '0'; -- transfer op expected reg\_A

wait for 10 ns;

op\_sel <= x"3"; reg\_a <= x"0000000000000008" ; reg\_b <= x"0000000000000008"; carry\_IN <= '1'; -- adder op expected f expect

wait for 10 ns;

end process;

end Behavioral;

---------------------ALU\_DESIGN-------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 07/02/2024 06:02:54 PM

-- Design Name:

-- Module Name: ALU64Bit - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ALU64Bit is

Port (

Reg\_A : in std\_logic\_vector(63 downto 0); -- register 1 UNSIGNED binary

Reg\_B : in std\_logic\_vector(63 downto 0); -- register 2 UNSIGNED binary

Op\_Sel : in std\_logic\_vector(3 downto 0); -- operator selector, only 10 operations exist

Carry\_In : in std\_logic; -- in bit

Carry\_Out : out std\_logic; -- out bit

ALU\_Out : out std\_logic\_vector(63 downto 0)

);

end ALU64Bit;

architecture Behavioral of ALU64Bit is

----------------------------- component map for shifter

component Shifter64Bit is --- shifter component for ALU does

Port ( selector : in STD\_LOGIC; -- OP(3) -- selects the shifter

shift\_mode : in STD\_LOGIC; -- the Cin value, 1 is logical mode

shift\_dir : in STD\_LOGIC; -- the OP(2) bit, 1 is left

shift\_COUNT : in STD\_LOGIC\_VECTOR(5 downto 0) ; -- B register but reads lower 6 bits

DATA\_IN : in STD\_LOGIC\_VECTOR(63 downto 0); -- register a UNSIGNED

DATA\_OUT: out STD\_LOGIC\_VECTOR(63 downto 0); -- ALU OUT

constant\_0: out std\_logic -- a constant 0 out to carry\_OUT

);

end component;

-------------------------- component map for adder

component Adder16x4bit is --- shifter component for ALU does

Port ( a : in STD\_LOGIC\_vector(63 downto 0); -- reg\_a

b : in std\_logic\_vector(63 downto 0); --- reg\_B

cin : in std\_logic; --- carryIN

addOUT : out std\_logic\_vector(63 downto 0); --- ALUout

cout : out std\_logic ------

);

end component;

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

signal shifterOut : std\_logic\_vector(63 downto 0);

signal shifterCarry : std\_logic;

signal incrementHold : std\_logic\_vector(63 downto 0);

signal incrementCarry : std\_logic;

signal decrementHold : std\_logic\_vector(63 downto 0);

signal decrementCarry: std\_logic;

signal adderHold : std\_logic\_vector(63 downto 0);

signal adderCarry : std\_logic;

begin

------------------------------------------ alu output when statements

alu\_OUT <= REG\_A when (op\_SEL = x"0")

else (incrementHold) when (op\_SEL = x"1")

else (decrementHold) when (op\_SEL = x"2")

else (adderHold) when (op\_SEL = x"3")

else (NOT REG\_A) when (op\_SEL = x"4")

else (REG\_A AND REG\_B) when (op\_SEL = x"5")

else (REG\_A OR REG\_B) when (op\_SEL = x"6")

else (REG\_A XOR REG\_B) when (OP\_SEL = x"7")

else shifterOUT when (OP\_SEL(3) = '1');

----------------------------------------------- carryout output when statements

carry\_out <= ('0') when (op\_sel = x"0")

else (incrementCarry) when (op\_sel = x"1")

else (decrementCarry) when (op\_sel = x"2")

else (adderCarry) when (op\_sel = x"3")

else '0';

-------------- start of increment process ------------- OUTPUT incrementHOLD , incrementCARRY

process(op\_SEL)

begin

if (op\_sel = x"1") then -- selected

if( REG\_A = x"ffffffffffffffff" ) then -- if reg\_a MAX CAPACITY -> about to overflow

incrementHold <= x"0000000000000000"; -- reset reg\_a

incrementCarry <= '1'; -- reset carry and set to 1 ---- assumes reg\_a is 64 bits NOT 65 bits, carryin is 0

else

incrementHold <= std\_logic\_vector( unsigned(REG\_A) + 1 ) ; -- perform normal increment math

incrementCarry <= '0'; -- we did not overflow set to 0

end if;

end if; -- end of selector check

end process;

--end of increment process

-------------- start of decrement process ------------- OUTPUT decrementHOLD , decrementCARRY

process(op\_SEL)

begin

if (op\_sel = x"2") then -- selected

if( REG\_A = x"0000000000000000" ) then -- if reg\_a MAX CAPACITY -> about to underflow

decrementHold <= x"ffffffffffffffff"; -- reset reg\_a

decrementCarry <= '1'; -- reset carry and set to 1 ---- assumes reg\_a is 64 bits NOT 65 bits, carryin is 0

else

decrementHold <= std\_logic\_vector( unsigned(REG\_A) - 1 ) ; -- perform normal increment math

decrementCarry <= '0'; -- we did not overflow set to 0

end if;

end if; -- end of selector check

end process;

--end of increment process

----------------------------- start of adder process ----------------- OUTPUT adderHOLD , adderCarry

FA3 : Adder16x4bit port map ( reg\_a , reg\_B, CARRY\_IN, adderHold, adderCarry );

----- end of adder process

----------------------- start of shifter process ----------------------- OUTPUT shifterHold , shifterCarry

FA9 : Shifter64Bit port map( OP\_SEL(3), Carry\_in, OP\_SEL(2), Reg\_B(5 downto 0), Reg\_A, shifterOut, shifterCarry ); -- shifter process

-- end of shifter process

end Behavioral;

-----SHIFTER COMPONENT DESIGN—

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

-- Company:

-- Engineer:

--

-- Create Date: 07/02/2024 06:21:06 PM

-- Design Name:

-- Module Name: Shifter64Bit - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity Shifter64Bit is --- shifter component for ALU does \_\_\_ shifting to

Port (

selector : in STD\_LOGIC; -- OP(3) -- selects the shifter

shift\_mode : in STD\_LOGIC; -- the Cin value, 1 is logical mode

shift\_dir : in STD\_LOGIC; -- the OP(2) bit, 1 is left

shift\_COUNT : in STD\_LOGIC\_VECTOR(5 downto 0) ; -- B register but reads lower 6 bits

DATA\_IN : in STD\_LOGIC\_VECTOR(63 downto 0); -- register a UNSIGNED

DATA\_OUT: out STD\_LOGIC\_VECTOR(63 downto 0); -- ALU OUT

constant\_0: out std\_logic -- a constant 0 out to carry\_OUT

);

end Shifter64Bit;

architecture Behavioral of Shifter64Bit is

begin

process(selector) begin

if (selector = '1') then

constant\_0 <= '0';

if(shift\_mode = '1') then -- if the carry in value is 1

if(shift\_dir = '1') then -- if if OP(2) is 1

DATA\_OUT <= std\_logic\_vector(shift\_left(unsigned( DATA\_IN ), to\_integer(unsigned(shift\_COUNT) ) ) );

else

DATA\_OUT <= std\_logic\_vector(shift\_right(unsigned( DATA\_IN ),to\_integer( unsigned(shift\_COUNT) ) ) );

end if;

elsif(shift\_mode = '0' ) then

if(shift\_dir = '1') then

DATA\_OUT <= std\_logic\_vector(shift\_left(signed( DATA\_IN ),to\_integer(unsigned(shift\_COUNT) ) ) );

else

DATA\_OUT <= std\_logic\_vector( shift\_right(signed( DATA\_IN ),to\_integer( unsigned(shift\_COUNT) ) ) );

end if;

end if;

end if;

end process;

end Behavioral;

--------------------ADDER COMPONENT DESIGN--------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 07/03/2024 10:08:05 AM

-- Design Name:

-- Module Name: Adder16x4bit - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity Adder16x4bit is

Port (

a : in STD\_LOGIC\_vector(63 downto 0); -- reg\_a

b : in std\_logic\_vector(63 downto 0); --- reg\_B

cin : in std\_logic; --- carryIN

addOUT : out std\_logic\_vector(63 downto 0); --- ALUout

cout : out std\_logic ------

);

end Adder16x4bit;

architecture Behavioral of Adder16x4bit is

component bit4Adder is

port(

X : in STD\_LOGIC\_VECTOR (3 downto 0); -- vector input because we have three of these as inputs

Y : in STD\_LOGIC\_VECTOR (3 downto 0);

Carryin : in STD\_LOGIC;

S : out STD\_LOGIC\_VECTOR (3 downto 0);

Carryout : out STD\_LOGIC

);

end component;

signal temp : std\_logic\_vector(15 downto 0);

begin

FA0 : bit4Adder PORT MAP(a(3 downto 0), b(3 downto 0), cin, addOUT(3 downto 0), temp(0)); -- the first hexadecimal

FA1 : bit4Adder PORT MAP(a(7 downto 4), b(7 downto 4), temp(0), addOUT(7 downto 4), temp(1) );

FA2 : bit4Adder PORT MAP(a(11 downto 8), b(11 downto 8), temp(1), addOUT(11 downto 8), temp(2) );

FA3 : bit4Adder PORT MAP(a(15 downto 12), b(15 downto 12), temp(2), addOUT(15 downto 12), temp(3) );

FA4 : bit4Adder PORT MAP(a(19 downto 16), b(19 downto 16), temp(3), addOUT(19 downto 16), temp(4) );

FA5 : bit4Adder PORT MAP(a(23 downto 20), b(23 downto 20), temp(4), addOUT(23 downto 20), temp(5) );

FA6 : bit4Adder PORT MAP(a(27 downto 24), b(27 downto 24), temp(5), addOUT(27 downto 24), temp(6) );

FA7 : bit4Adder PORT MAP(a(31 downto 28), b(31 downto 28), temp(6), addOUT(31 downto 28), temp(7) );

FA8 : bit4Adder PORT MAP(a(35 downto 32), b(35 downto 32), temp(7), addOUT(35 downto 32), temp(8) );

FA9 : bit4Adder PORT MAP(a(39 downto 36), b(39 downto 36), temp(8), addOUT(39 downto 36), temp(9) );

FA10 : bit4Adder PORT MAP(a(43 downto 40), b(43 downto 40), temp(9), addOUT(43 downto 40), temp(10) );

FA11 : bit4Adder PORT MAP(a(47 downto 44), b(47 downto 44), temp(10), addOUT(47 downto 44), temp(11) );

FA12 : bit4Adder PORT MAP(a(51 downto 48), b(51 downto 48), temp(11), addOUT(51 downto 48), temp(12) );

FA13 : bit4Adder PORT MAP(a(55 downto 52), b(55 downto 52), temp(12), addOUT(55 downto 52), temp(13) );

FA14 : bit4Adder PORT MAP(a(59 downto 56), b(59 downto 56), temp(13), addOUT(59 downto 56), temp(14) );

FA15 : bit4Adder PORT MAP(a(63 downto 60), b(63 downto 60), temp( 14 ), addOUT(63 downto 60), cout); -- the last hexadecimal

end Behavioral;

