library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity lab6 is

generic(WIDTH : integer := 32);

port( A : in unsigned(WIDTH-1 downto 0);

X : in unsigned(WIDTH-1 downto 0);

clk:in std\_logic;

prod : out unsigned(2\*WIDTH-1 downto 0)

);

end lab6;

architecture Behavioral of lab6 is

--MUX signals

signal sel : std\_logic\_vector(2 downto 0);

signal shift\_cnt : unsigned(3 downto 0);

--MUX I/O will be 35 bits when applying Radix 16

signal zeroA, oneA : unsigned((WIDTH+3)-1 downto 0);

signal twoA, threeA, fourA : unsigned((WIDTH+3)-1 downto 0);

signal fiveA, sixA, sevenA : unsigned((WIDTH+3)-1 downto 0);

signal mux\_out : unsigned((WIDTH+3)-1 downto 0);

signal carry : unsigned(WIDTH-30 downto 0):=(others=>'0');

--Shift and add

signal sum : unsigned((WIDTH+3)-1 downto 0);

signal sum\_32,X\_temp,A\_temp,in\_sense,X\_temp1,A\_temp1 : unsigned(WIDTH-1 downto 0):=(others=>'0');

signal prod\_concat : unsigned(2\*WIDTH-1 downto 0);

signal carry\_32 : unsigned(WIDTH-1 downto 0):=(others=>'0');

begin

zeroA <= (others => '0');

oneA <= resize(A, (WIDTH+3));

twoA <= oneA sll 1;

threeA <= twoA + oneA;

fourA <= oneA sll 2;

fiveA <= fourA + oneA;

sixA <= fiveA + oneA;

sevenA <= sixA + oneA;

carry<=(sum(WIDTH-30 downto WIDTH-WIDTH));

sum<=mux\_out+sum\_32;

prod\_concat <= sum\_32(WIDTH-1 downto 0) & carry\_32(WIDTH-1 downto 0) ;

sum\_shift: process(clk)

begin

if clk'event and clk='1' then

if (shift\_cnt/=0) then

sum\_32<=sum(WIDTH+2 downto 3);

else

sum\_32<=(others=>'0');

end if;

end if;

end process;

cnt:process(clk)

begin

if clk'event and clk='1' then

-- carry\_32(WIDTH-1 downto WIDTH-3)<=carry;

if(shift\_cnt /= 11) then

shift\_cnt <= shift\_cnt + 1; --shift\_cnt=0 at t=0

else

shift\_cnt <= (others => '0');

end if;

end if;

end process;

out\_data:process(clk)

begin

if falling\_edge (clk) then

if(shift\_cnt=0) then

prod <= prod\_concat sll 1;

end if;

end if;

end process;

carry\_mem:process(clk)

begin

if (clk='1') then

if (shift\_cnt/=0) then

carry\_32(WIDTH-1 downto WIDTH-3)<=carry;

else

carry\_32<=(others=>'0');

end if;

else

carry\_32<=carry\_32 srl 3;

end if;

end process;

mux: process(sel)

begin

-- carry31<=carry\_32 srl 3;

case sel is

--Pass P0

when "000" =>

mux\_out <= zeroA;

--Pass A

when "001" =>

mux\_out <= oneA;

--Pass 2A (shift by 1)

when "010" =>

mux\_out <= twoA;

---Pass 3A (2A + A)

when "011" =>

mux\_out <= threeA;

---Pass 4A (shift by 2)

when "100" =>

mux\_out <= fourA;

---Pass 5A (4A + A)

when "101" =>

mux\_out <= fiveA;

--Pass 6A (5A + A)

when "110" =>

mux\_out <= sixA;

---Pass 7A (6A + A)

when "111" =>

mux\_out <= sevenA;

when others =>

mux\_out <= mux\_out;

end case;

end process;

sel\_generate: process(shift\_cnt)

begin

--Observe LSBs of X input using Radix 8

case shift\_cnt is

when "0000" =>

sel <= (others=>'0');

when "0001" =>

sel <= std\_logic\_vector(X(2 downto 0));

--Pass A

when "0010" =>

sel <= std\_logic\_vector(X(5 downto 3));

--Pass 2A (shift by 1)

when "0011" =>

sel <= std\_logic\_vector(X(8 downto 6));

---Pass 3A (2A + A)

when "0100" =>

sel <= std\_logic\_vector(X(11 downto 9));

---Pass 4A (shift by 2)

when "0101" =>

sel <= std\_logic\_vector(X(14 downto 12));

---Pass 5A (4A + A)

when "0110" =>

sel <= std\_logic\_vector(X(17 downto 15));

when "0111" =>

sel <= std\_logic\_vector(X(20 downto 18));

--Pass 6A (5A + A)

when "1000" =>

sel <= std\_logic\_vector(X(23 downto 21));

---Pass 3A (2A + A)

when "1001" =>

sel <= std\_logic\_vector(X(26 downto 24));

---Pass 4A (shift by 2)

when "1010" =>

sel <= std\_logic\_vector(X(29 downto 27));

---Pass 5A (4A + A)

when "1011" =>

sel <= '0' & std\_logic\_vector(X(31 downto 30));

when others =>

sel <=sel;

end case;

end process;

end Behavioral;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity tb\_lab6 is

-- Port ( );

end tb\_lab6;

architecture Behavioral of tb\_lab6 is

component lab6 is

generic(WIDTH : integer := 32);

port( A : in unsigned(WIDTH-1 downto 0);

X : in unsigned(WIDTH-1 downto 0);

clk:in std\_logic;

prod : out unsigned(2\*WIDTH-1 downto 0)

);

end component;

constant WIDTH : integer := 32;

constant PERIOD:time:=10 ns;

signal A,X : unsigned(WIDTH-1 downto 0);

signal prod : unsigned(2\*WIDTH-1 downto 0);

signal clk:std\_logic;

--signal temp:unsigned(1 downto 0):="10";

--signal temp1:unsigned(2 downto 0):="101";

--signal temp2:unsigned(2 downto 0);

begin

UUT: lab6

port map(A => A, X => X, prod => prod,clk=>clk);

--temp2<=temp+temp1;

--A <= to\_unsigned(16#02F170A6#, A'length);

process

begin

X <= to\_unsigned(16#50000000#, A'length);

A <= to\_unsigned(16#A6#, A'length);

wait for 120ns;

X <= to\_unsigned(16#2#, A'length);

wait for 120 ns;

X <= to\_unsigned(16#10#, A'length);

wait for 120 ns;

X <= to\_unsigned(16#100#, A'length);

wait for 120 ns;

X <= to\_unsigned(16#1000#, A'length);

wait for 120 ns;

X <= to\_unsigned(16#10000#, A'length);

wait for 120 ns;

X <= to\_unsigned(16#100000#, A'length);

wait for 120 ns;

X <= to\_unsigned(16#1000000#, A'length);

wait for 120 ns;

X <= to\_unsigned(16#1000000#, A'length);

wait for 120 ns;

X <= to\_unsigned(16#10000000#, A'length);

end process;

--X <= to\_unsigned(16#AD#, A'length);

process

begin

clk<='1';

wait for PERIOD/2;

clk<= not clk;

wait for PERIOD/2;

end process;

end Behavioral;