3,6,10,13,15,23,26,28

3)

Entity is

Port(D : in slv(7 downto 0);

OE\_Bar : in sl;

LE : in sl;

Q : out slv(7 downto 0));

End entity

Arch behavioral is

Begin

Process(OE\_Bar, LE, D)

Begin

If OE\_Bar = ‘0’ then

If LE = ‘1’ then

Q <= D;

End if;

Elsif OE\_Bar = ‘1’ then

Q <= “ZZZZZZZZ”;

End if;

End process;

End arch;

6) For both a and b, moving the statement qout <= q would put it inside the process statement, which means that qout would be updated with the value that q entered the process with rather than the value it was updated with. This would put qout one clock-cycle behind the original design. A synthesizer would generate the same logic for the two new descriptions, which would be different from the original.

10)

Entity shift8 is

Port(ser\_in, clk, load, clear, shift\_r: in sl;

Ser\_out: out sl;

Din : in slv(7 downto 0));

End shift8;

Architecture behavioral of shift8

Signal r : slv(7 downto 0);

Begin

Process(clk)

Variable r\_out : sl := ’0’;

Begin

If rising\_edge(clk) then

If load = ‘1’ then

R = D;

Elsif clear = ‘1’ then

R = “00000000”;

Elsif shift\_r = ‘1’ then

r\_out <= r(7);

r(7 downto 1) <= r(6 downto 0);

r(0) <= ser\_in);

end if;

end if;

ser\_out <= r\_out;

end process;

end behavioral;

13)

Architecture behavioral of ring\_cntr

signal qint : slv(3 downto 0) := “1110”;

Process(clk)

begin

If rising\_edge(clk) then

If rst\_bar = ‘0’ then

Qint <= “1110”;

Elsif enable = ‘1’ then

qint(3 downto 1) <= qint(2 downto 0);

qint(0) <= qint(3);

end if;

end if;

end process;

qout <= qint;

15)

Entity udcnt is

Port(up,clk : in sl;

count : buffer slv(3downto0));

End udcnt;

Architecture behavioral of udcnt

Begin

Process(clk)

If rising\_edge(clk) then

If up = ‘1’ then

If count = “1111” then

Count = count;

else

Count = slv(unsigned(count + 1));

End if

Else

If count = “0000” then

Count = count;

Else

Count = slv(unsigned(count – 1));

End if;

End if;

End if;

End process;

End behavioral;

23)

Entity delay is

Port(fire, clk : in sl;

delayed\_out : out sl);

end delay;

Architecture behavioral of delay

Signal count : integer() := 0;

Begin

Process(clk)

Variable prev\_fire : sl := ‘0’;

Begin

If rising\_edge(clk) then

If count = 499999 then

Delayed\_out <= ‘1’;

Count <= 0;

elsif count > 0 then

Count <= count + 1;

Delayed\_out <= ‘0’;

Elsif fire = ‘1’ then

If prev\_fire = ‘0’ then

Count <= 1;

Delayed\_out <= ‘0’;

End if;

End if;

Prev\_fire <= fire;

End if;

End process;

End behavioral;

26) It would only write when the write enable transitioned from high to low, instead of every time that there was a change in address or data while the write\_enable was low.

28)

entity memory is

port( we\_bar : in std\_logic;-- write enable

oe\_bar : in std\_logic;-- output enable

data : inout std\_logic\_vector(7 downto 0);

address : in std\_logic\_vector(2 downto 0));

end memory;

architecture behavioral of memory is

type mem\_array is array (0 to 2\*\*(address'length) - 1) of std\_logic\_vector (7 downto 0);

begin

write\_read: process(we\_bar, oe\_bar, data\_in, address)

variable mem\_v : mem\_array;

begin

if we\_bar = '0' then

mem\_v(to\_integer(unsigned(address))) := data;

elsif oe\_bar = '0' then

data <= mem\_v(to\_integer(unsigned(address)));

else

data <= (others => 'Z');

end if;

end process;

end behavioral;