VLSI Lab Report Assignment 5

BCSE 4th Year 2nd Semester

Name: ***Priti Shaw***
Roll No. : ***001710501076***
Batch: ***A3***

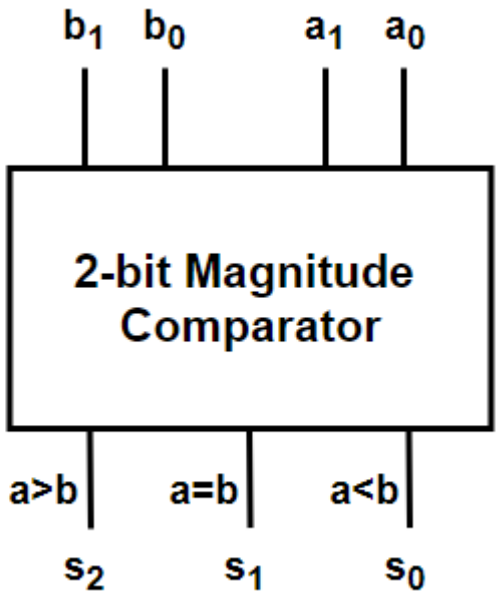
1. 2-Bit Magnitude Comparator

Description

Implement a 2-bit magnitude comparator using the package. Write a procedure implementing a 2-bit magnitude comparator and include the procedure in a package. Also, design a testbench for 2-bit magnitude comparator.

Two Bit Magnitude Comparator takes two 2-bit binary numbers as input, compares their magnitude and gives which one is greater or both equal as output.

Block Diagram



Entity

```
entity twoBitComparator is
    Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
           b : in  STD_LOGIC_VECTOR (1 downto 0);
           s : out  STD_LOGIC_VECTOR (2 downto 0));
end twoBitComparator;
```

Truth Table

Input				Output		
b_1	b_0	a_1	a_0	s_2	s_1	s_0
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Architecture

```
architecture Behavioral of twoBitComparator is
begin
      process(a,b)
      variable ss:std_logic_vector(2 downto 0);
      begin
           twoBitComparatorProcedure(a(1 downto 0),b(1 downto 0),ss(2
downto 0));
           s(2 downto 0)<=ss(2 downto 0);
      end process;
end Behavioral;
```

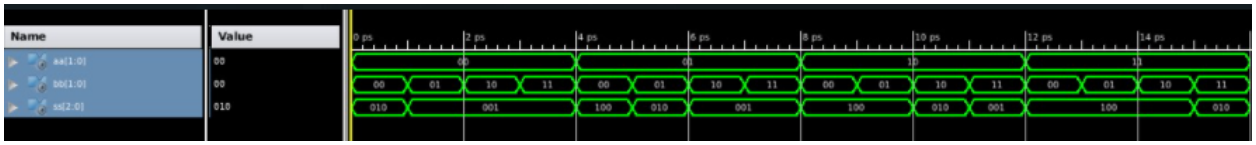
Procedure of 2-bit comparator in Package

```
procedure twoBitComparatorProcedure(a:in std_logic_vector;b:in
std_logic_vector;s:out std_logic_vector) is
begin
s(2):=(a(1) and not(b(1))) or ((a(1) xnor b(1)) and (a(0) and not(b(0))));
s(1):=(a(1) xnor b(1)) and (a(0) xnor b(0));
s(0):=(not(a(1)) and b(1)) or ((a(1) xnor b(1)) and (not(a(0)) and b(0)));
end procedure;
```

TestBench

```
stim_proc: process
   variable j,k:integer;
   variable binA,binB:std_logic_vector(1 downto 0);
   begin
            for k in 0 to 3 loop
                  proca:decimalToBinaryProcedure(k,2,binA);
                  a<=binA;
                  for j in 0 to 3 loop
                        procb:decimalToBinaryProcedure(j,2,binB);
                        b<=binB;
                        wait for 1 ps;
                  end loop;
            end loop;
   end process;
END;
```

Timing Diagram



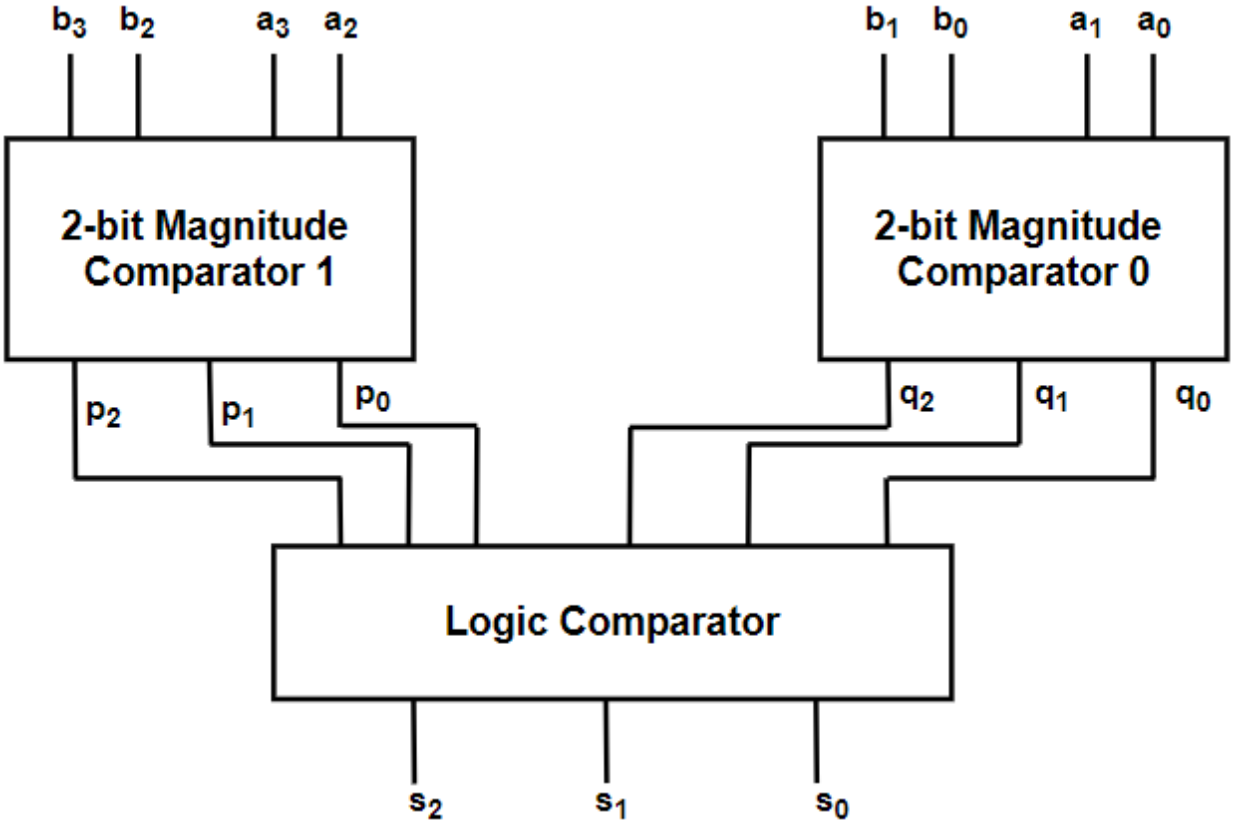
2. 4-Bit Magnitude Comparator using 2-bit comparator

Description

Implement a 4-bit magnitude comparator using a 2-bit magnitude comparator. Write a procedure implementing a 4-bit magnitude comparator and include the procedure in a package. Also, design a testbench for a 4-bit magnitude comparator.

A 4-bit magnitude comparator takes two four-bit binary numbers as inputs and gives their magnitude comparison result as output. It uses two 2-bit magnitude comparators and one logic comparator as components. Logic comparator takes the result of two magnitude comparators as input and tells which input to the magnitude comparators is greater assuming the inputs were partitioned to the two magnitude comparators in same order.

Block Diagram



Truth Table

p_2	p_1	p_0	q_2	q_1	q_0	s_2	s_1	s_0
1	0	0	1	0	0	1	0	0
			0	1	0			
			0	0	1			
0	1	0	1	0	0	1	0	0
			0	1	0	0	1	0
			0	0	1	0	0	1
0	0	1	1	0	0	0	0	1
			0	1	0			
			0	0	1			

Entity

```
entity fourBitComparator is
    Port ( aa : in  STD_LOGIC_VECTOR (3 downto 0);
           bb : in  STD_LOGIC_VECTOR (3 downto 0);
           ss : out  STD_LOGIC_VECTOR (2 downto 0));
end fourBitComparator;
```

Architecture

```
architecture Behavioral of fourBitComparator is

begin
p1:process(aa,bb)
variable sss:std_logic_vector(2 downto 0);
begin
            fourBitComparatorProcedure(aa,bb,sss);
            ss<=sss;
end process;
end Behavioral;
```

Procedure of 4-bit comparator in Package

```
procedure fourBitComparatorProcedure(a:in std_logic_vector;b:in
std_logic_vector;s:out std_logic_vector) is
      variable p,q,c,d:std_logic_vector(1 downto 0);
      variable t,r,ss:std_logic_vector(2 downto 0);
      begin
            p(1 downto 0):= a(3 downto 2);
            q(1 downto 0):= b(3 downto 2);

            proc1:twoBitComparatorProcedure(p(1 downto 0),q(1 downto 0),t(2
downto 0));
            c(1 downto 0):=a(1 downto 0);
            d(1 downto 0):=b(1 downto 0);

            proc2:twoBitComparatorProcedure(c(1 downto 0),d(1 downto 0),r(2
downto 0));

            proc3:logicComparatorProcedure(t(2 downto 0),r(2 downto 0),ss(2
downto 0));
            s(2 downto 0):=ss(2 downto 0);
      end procedure;
```

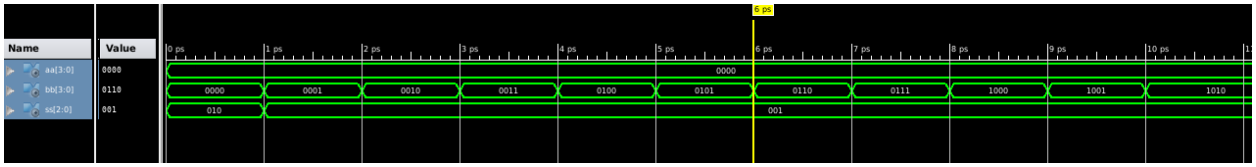
Procedure of Logic comparator in Package

```
procedure logicComparatorProcedure(t:in std_logic_vector; r:in
std_logic_vector; s:out std_logic_vector) is
      begin
            s(2):=t(2) or (t(1) and r(2));
            s(1):=t(1) and r(1);
            s(0):=t(0) or (t(1) and r(0));
      end procedure;
```

TestBench

```
stim_proc: process
            variable j,k:integer;
            variable binA,binB:std_logic_vector(3 downto 0);
            begin
                  for k in 0 to 15 loop
                        procA:decimalToBinaryProcedure(k,4,binA);
                        aa<=binA;
                        for j in 0 to 15 loop
                              procB:decimalToBinaryProcedure(j,4,binB);
                              bb<=binB;
                              wait for 1 ps;
                        end loop;
                  end loop;
            end process;
END;
```

Timing Diagram



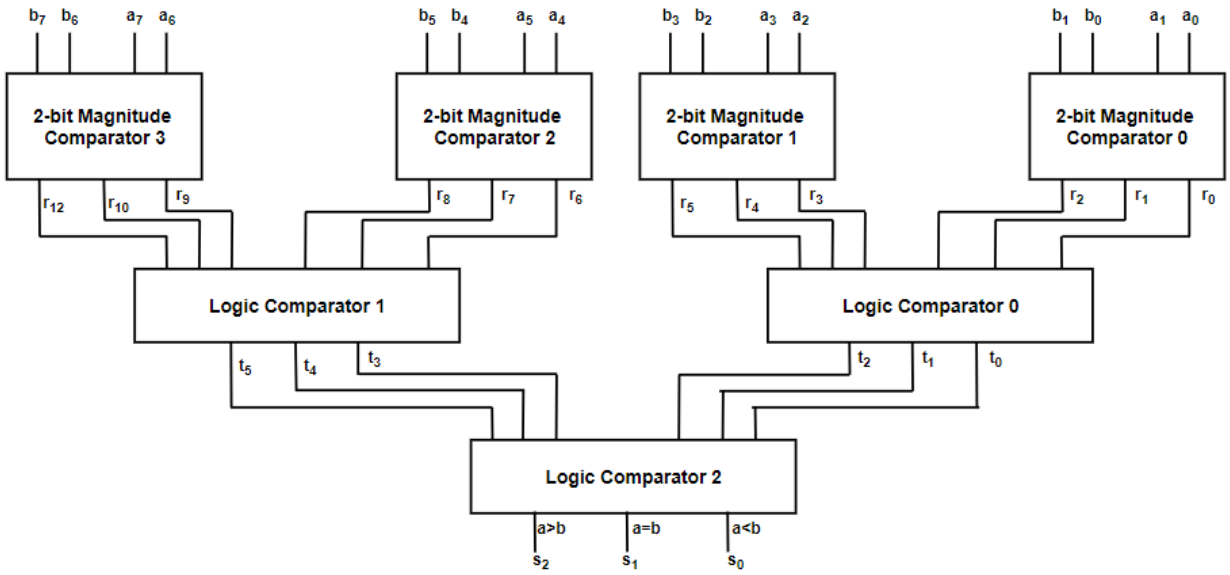
3. 8-Bit Magnitude Comparator using 2-bit magnitude comparator Only

Description

Implement a 8-bit magnitude comparator using a 2-bit magnitude comparator only. Write a procedure implementing a 8-bit magnitude comparator and include the procedure in a package. Also, design a testbench for a 8-bit magnitude comparator.

A 8-bit magnitude comparator takes two eight-bit binary numbers as inputs and gives their magnitude comparison result as output. It uses four 2-bit magnitude comparators and three logic comparators as components.

Block Diagram



Entity

```
entity eightBitComparatorUsing2Bit is
    Port ( aaa : in  STD_LOGIC_VECTOR (7 downto 0);
           bbb : in  STD_LOGIC_VECTOR (7 downto 0);
           sss : out  STD_LOGIC_VECTOR (2 downto 0));
end eightBitComparatorUsing2Bit;
```

Architecture

```
architecture Behavioral of eightBitComparatorUsing2Bit is

begin
      p1:process(aaa,bbb)
      variable s:std_logic_vector(2 downto 0);
      begin
            eightBitComparatorProcedure(aaa(7 downto 0),bbb(7 downto 0),s);
            sss<=s;
      end process;
end Behavioral;
```

Procedure of 8-bit comparator in Package

```
procedure eightBitComparatorProcedure(a:in std_logic_vector;b:in
std_logic_vector;s:out std_logic_vector) is
      variable j,k:integer;
      variable aa,bb:std_logic_vector(1 downto 0);
      variable ss,gg,ee,ff,t,e,f:std_logic_vector(2 downto 0);
      variable g:std_logic_vector(5 downto 0);
      variable tt:std_logic_vector(11 downto 0);
      begin
            for k in 0 to 3 loop
                  aa(1 downto 0):=a(((2*k)+1) downto (2*k));
                  bb(1 downto 0):=b(((2*k)+1) downto (2*k));
                  prock:twoBitComparatorProcedure(aa(1 downto 0),bb(1
downto 0),t(2 downto 0));
                  tt(((3*k)+2) downto (3*k)):=t(2 downto 0);
            end loop;
            for j in 0 to 1 loop
                  e(2 downto 0):=tt(((6*j)+5) downto ((6*j)+3));
                  f(2 downto 0):=tt(((6*j)+2) downto (6*j));
                  procj:logicComparatorProcedure(e(2 downto 0),f(2 downto
0),gg(2 downto 0));
                  g(((3*j)+2) downto (3*j)):=gg(2 downto 0);
```

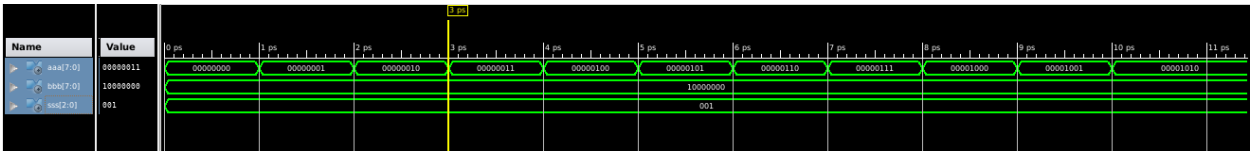
```
            end loop;
            ee(2 downto 0):=g(5 downto 3);
            ff(2 downto 0):=g(2 downto 0);
            proco:logicComparatorProcedure(ee(2 downto 0),ff(2 downto
0),ss(2 downto 0));
            s:=ss;
      end procedure;
```

TestBench

```
stim_proc: process
   variable j,k:integer;
      variable binA,binB:std_logic_vector(7 downto 0);
   begin
            j:=128;
            for k in 0 to 255 loop
                  procA:decimalToBinaryProcedure(k,8,binA);
                  aaa<=binA;
                  procB:decimalToBinaryProcedure(j,8,binB);
                  bbb<=binB;
                  wait for 1 ps;
            end loop;
   end process;

END;
```

Timing Diagram



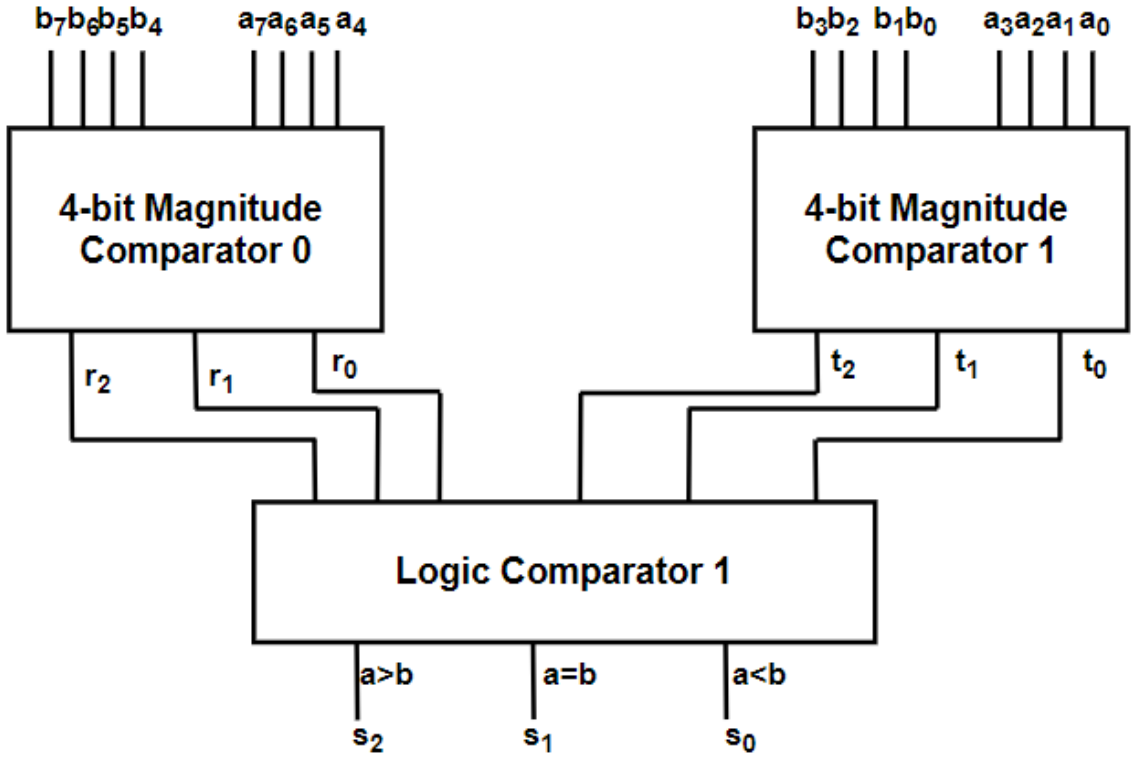
4. 8-Bit Magnitude Comparator using 4-bit magnitude comparator Only

Description

Implement a 8-bit magnitude comparator using a 4-bit magnitude comparator only. Write a procedure implementing a 4-bit magnitude comparator and include the procedure in a package. Also, design a testbench for a 4-bit magnitude comparator.

A 8-bit magnitude comparator takes two eight-bit binary numbers as inputs and gives their magnitude comparison result as output. It uses two 4-bit magnitude comparators and one logic comparator as components.

Block Diagram



Entity

```
entity eightBitComparatorUsing4Bit is
    Port ( aaa : in  STD_LOGIC_VECTOR (7 downto 0);
           bbb : in  STD_LOGIC_VECTOR (7 downto 0);
           sss : out  STD_LOGIC_VECTOR (2 downto 0));
end eightBitComparatorUsing4Bit;
```

Architecture

```
architecture Behavioral of eightBitComparatorUsing4Bit is

begin
      p1:process(aaa,bbb)
      variable s:std_logic_vector(2 downto 0);
      begin
            eightBitComparatorUsing4BitProcedure(aaa(7 downto 0),bbb(7
downto 0),s);
            sss<=s;
      end process;

end Behavioral;
```

Procedure of 8-bit comparator in Package

```
procedure eightBitComparatorUsing4BitProcedure(a:in std_logic_vector;b:in
std_logic_vector;s: out std_logic_vector) is
      variable p,q,c,d:std_logic_vector(3 downto 0);
      variable t,r,ss:std_logic_vector(2 downto 0);
      begin
            p(3 downto 0):= a(3 downto 0);
            q(3 downto 0):= b(3 downto 0);
            proc1:fourBitComparatorProcedure(p(3 downto 0),q(3 downto
0),t(2 downto 0));
            c(3 downto 0):=a(7 downto 4);
            d(3 downto 0):=b(7 downto 4);
            proc2:fourBitComparatorProcedure(c(3 downto 0),d(3 downto
0),r(2 downto 0));
            proc3:logicComparatorProcedure(r(2 downto 0),t(2 downto 0),ss(2
downto 0));
            s(2 downto 0):=ss(2 downto 0);
      end procedure;
```

TestBench

```
stim_proc: process
   variable j,k:integer;
      variable binA,binB:std_logic_vector(7 downto 0);
   begin
            j:=128;
            for k in 0 to 255 loop
                  procA:decimalToBinaryProcedure(k,8,binA);
                  aaa<=binA;
                  procB:decimalToBinaryProcedure(j,8,binB);
                  bbb<=binB;
                  wait for 1 ps;
            end loop;
   end process;
```

Timing Diagram