VLSI Lab Report Assignment 4

BCSE 4th Year 2nd Semester

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

1. Half Adder

Description

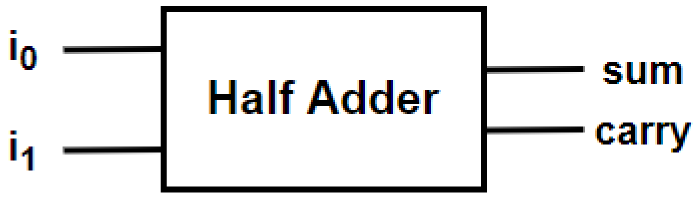
Implement a half adder using the package. Write a procedure implementing half adder and include the procedure in a package. Also, design a testbench of half adder.

Half adder is a combinational circuit that adds two bits and produces a sum bit(S) and carry bit(C) as the output. If A and B are the input bits, then sum bit(S) is the XOR of A and B and the carry bit will be the AND of A and B.

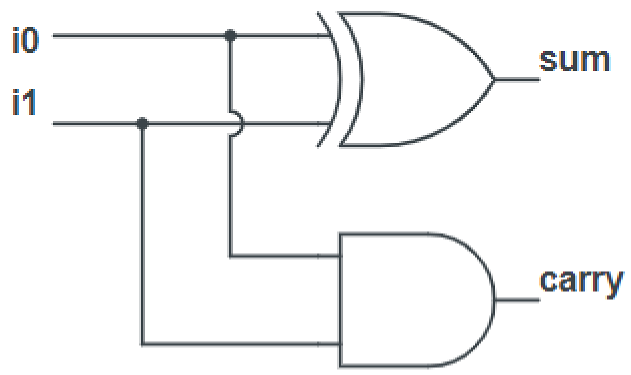
Truth Table

Input		Output	
i_1	i_0	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Block Diagram



Circuit Diagram



Entity

```
entity halfAdder is
    Port ( aa : in  STD_LOGIC;
           bb : in  STD_LOGIC;
           cc : out  STD_LOGIC;
           ss : out  STD_LOGIC);
end halfAdder;
```

Procedure for Half Adder in package

```
procedure halfAdderProcedure(a:in std_logic; b:in std_logic; c:out
std_logic; s:out std_logic) is
      begin
                  s:= a xor b;
                  c:= a and b;
end procedure;
```

Architecture

```
architecture Behavioral of halfAdder is
begin
      p1:process(aa, bb)
      variable carry,sum:std_logic;
      begin
            proca: halfAdderProcedure(aa,bb,carry,sum);
            cc<=carry;
            ss<=sum;
      end process;
end Behavioral;
```

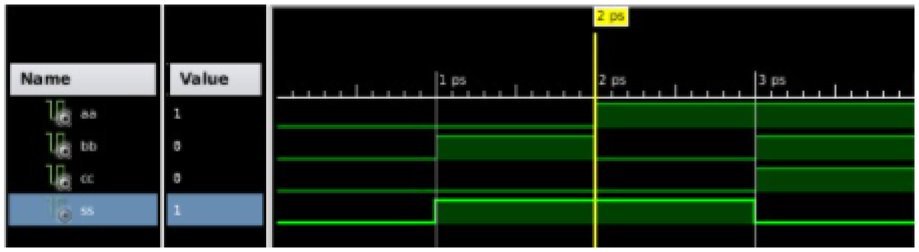
Procedure for Decimal to Binary in package

```
procedure decimalToBinaryProcedure(decimal:in integer; numberOfBits:in
integer; binary:out std_logic_vector) is
      variable deci:integer;
      variable bitPosition:integer;
      begin
                  deci:=decimal;
                  bitPosition:=0;
                  while(bitPosition<numberOfBits) loop
                        if (deci rem 2)=0 then
                              binary(bitPosition):='0';
                        elsif (deci rem 2)=1 then
                              binary(bitPosition):='1';
                        end if;
                        deci:=deci/2;
                        bitPosition:=bitPosition+1;
                  end loop;
      end procedure;
```

TestBench

```
BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: halfAdder PORT MAP (
          aa => aa,
          bb => bb,
          cc => cc,
          ss => ss
        );
   -- Stimulus process
   stim_proc: process
      variable k:integer;
      variable bin:std_logic_vector(1 downto 0);
      begin
                  for k in 0 to 3 loop
                        procb:decimalToBinaryProcedure(k,2,bin);
                        aa<=bin(1);
                        bb<=bin(0);
                        wait for 1ps;
                  end loop;
        end process;

END;
```

Timing Diagram



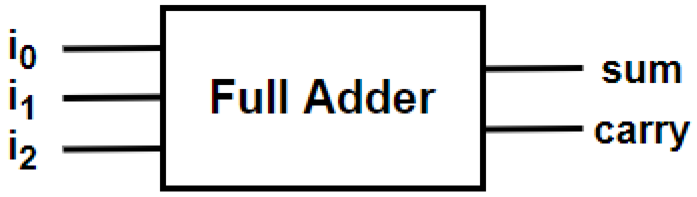
2. Full Adder

Description

Implement a full adder using the package. Write a procedure implementing full adder by using procedure for half adder and add the procedure into the same package. Also, design a testbench of full adder.

Full Adder is a combinational circuit which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as Cin. The outputs are sum bit and output carry bit after addition.

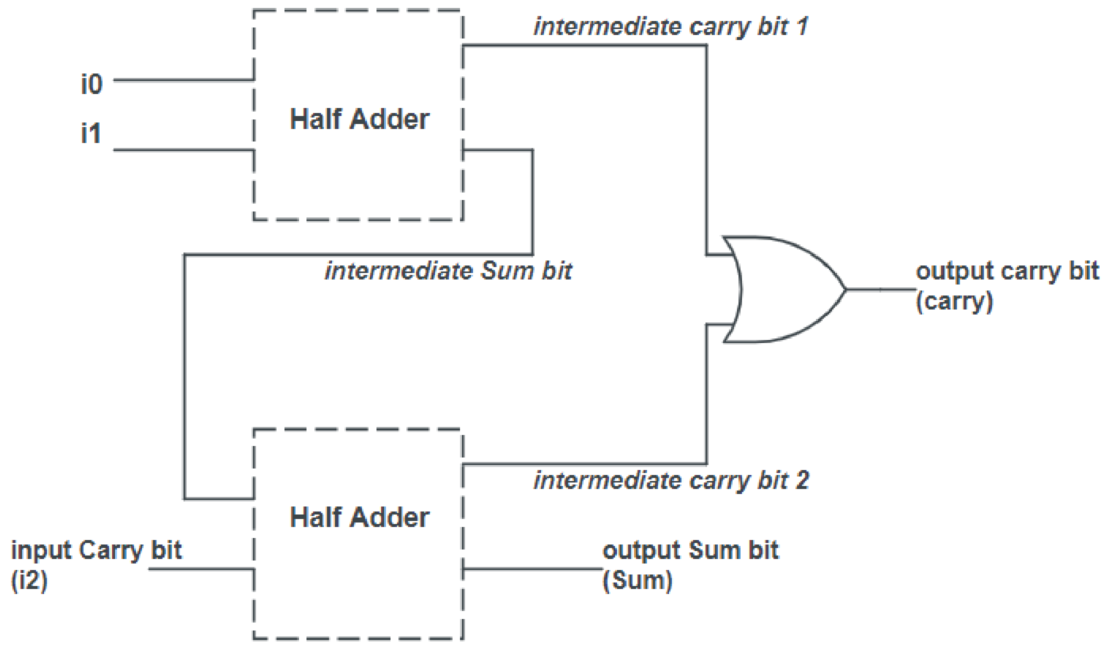
Block Diagram



Truth Table

Input			Output	
i_2	i_1	i_0	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Circuit Diagram



Entity

```vhdl
entity fullAdder is
    Port ( ain : in  STD_LOGIC;
           bin : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           cout : out  STD_LOGIC;
           sout : out  STD_LOGIC);
end fullAdder;
```

Procedure for Full Adder in package

```vhdl
procedure fullAdderProcedure(a:in std_logic; b:in std_logic; c:in
std_logic; cout:out std_logic; sum:out std_logic) is
      variable cc,ss,ccc,sss:std_logic;
      begin
                  proc1:halfAdderProcedure(b,c,cc,ss);
                  proc2:halfAdderProcedure(a,ss,ccc,sss);
                  cout:=cc or ccc;
                  sum:=sss;
end procedure;
```

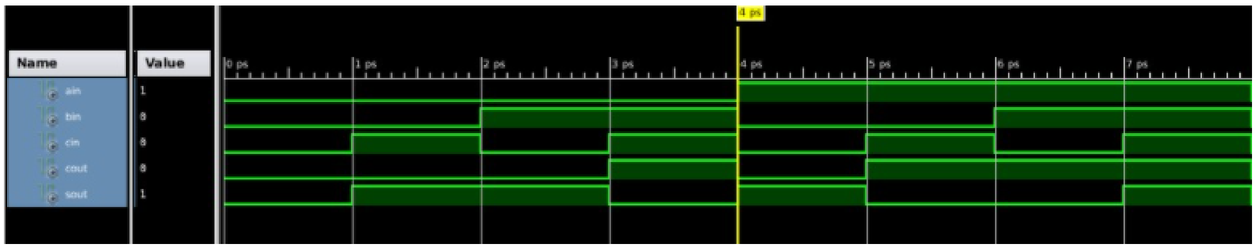
Architecture

```vhdl
architecture Behavioral of fullAdder is

begin
            p1:process(ain,bin,cin)
            variable ccout,ssum:std_logic;
            begin
                        proc1:fullAdderProcedure(ain,bin,cin,ccout,ssum);
                        cout<=ccout;
                        sout<=ssum;
            end process;
end Behavioral;
```

TestBench

```
BEGIN

      -- Instantiate the Unit Under Test (UUT)
   uut: fullAdder PORT MAP (
          ain => ain,
          bin => bin,
          cin => cin,
          cout => cout,
          sout => sout
        );

   -- Stimulus process
   stim_proc: process
   variable k:integer;
      variable bitPos:integer;
      variable binVal:std_logic_vector(2 downto 0);
      begin
            for k in 0 to 7 loop
                  proc2:decimalToBinaryProcedure(k,3,binVal);
                  ain<=binVal(2);
                  bin<=binVal(1);
                  cin<=binVal(0);
                  wait for 1ps;
            end loop;
      end process;

END;
```

Timing Diagram



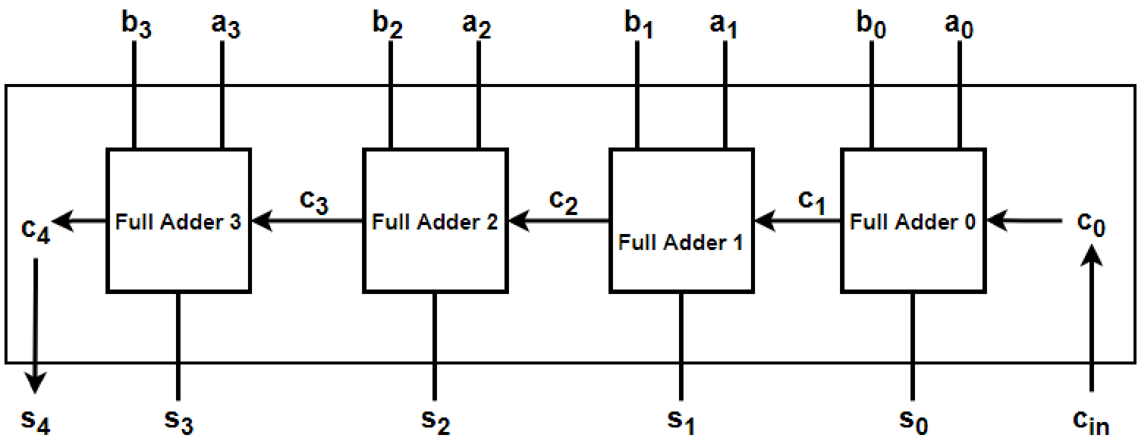
3. 4-Bit Ripple Carry Adder

Description

Implement a 4-bit ripple carry adder using the package. Write a procedure for 4-bit ripple carry adder implemented using full adder and include the procedure into the same package. Also, design a testbench for 4-bit ripple carry adder.

Multiple full adder circuits can be cascaded in parallel to add an N-bit number. For an N-bit parallel adder, there must be N number of full adder circuits. A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry-in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage.In a 4-bit ripple carry adder, it takes two 4-bit binary numbers and one input carry bit as input and gives their sum(maximum five bits) as output. Each full adder gives one bit of final sum and the output carry bit of the last full adder is passed to MSB of output sum.

Block Diagram



Entity

```
entity fourBitRippleCarryAdder is
    Port ( ain : in  STD_LOGIC_VECTOR (3 downto 0);
           bin : in  STD_LOGIC_VECTOR (3 downto 0);
           cin : in  STD_LOGIC;
           sout : out  STD_LOGIC_VECTOR (4 downto 0));
end fourBitRippleCarryAdder;
```

Procedure for 4-bit Ripple Carry Adder in package

```
procedure fourBitRippleCarryAdderProcedure(a:in std_logic_vector;b:in
std_logic_vector;cin:in std_logic;s:out std_logic_vector) is
      variable c:std_logic_vector(4 downto 0);
      variable k:integer;
      begin
                  c(0):=cin;
                  for k in 0 to 3 loop
                      prock:fullAdderProcedure(a(k),b(k),c(k),c(k+1),s(k));
                  end loop;
                  s(4):=c(4);
end procedure;
```

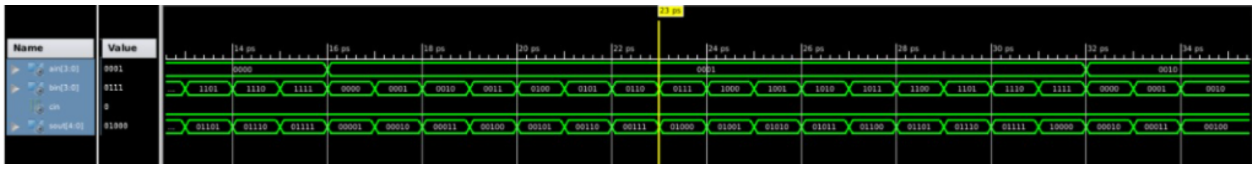
Architecture

```
architecture Behavioral of fourBitRippleCarryAdder is

begin
      p1:process(ain,bin,cin)
      variable ss:std_logic_vector(4 downto 0);
      begin
           proc1: fourBitRippleCarryAdderProcedure(ain(3 downto 0),bin(3
downto 0),cin,ss);
           sout<=ss;
      end process;
end Behavioral;
```

TestBench

```vhdl
BEGIN

      -- Instantiate the Unit Under Test (UUT)
   uut: fourBitRippleCarryAdder PORT MAP (
          ain => ain,
          bin => bin,
          cin => cin,
          sout => sout
        );

   -- Stimulus process
   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);
                  ain<=binA;
                  for j in 0 to 15 loop
                        procB:decimalToBinaryProcedure(j,4,binB);
                        bin<=binB;
                        cin<='0';
                        wait for 1 ps;
                  end loop;
            end loop;
   end process;
END;
```

Timing Diagram



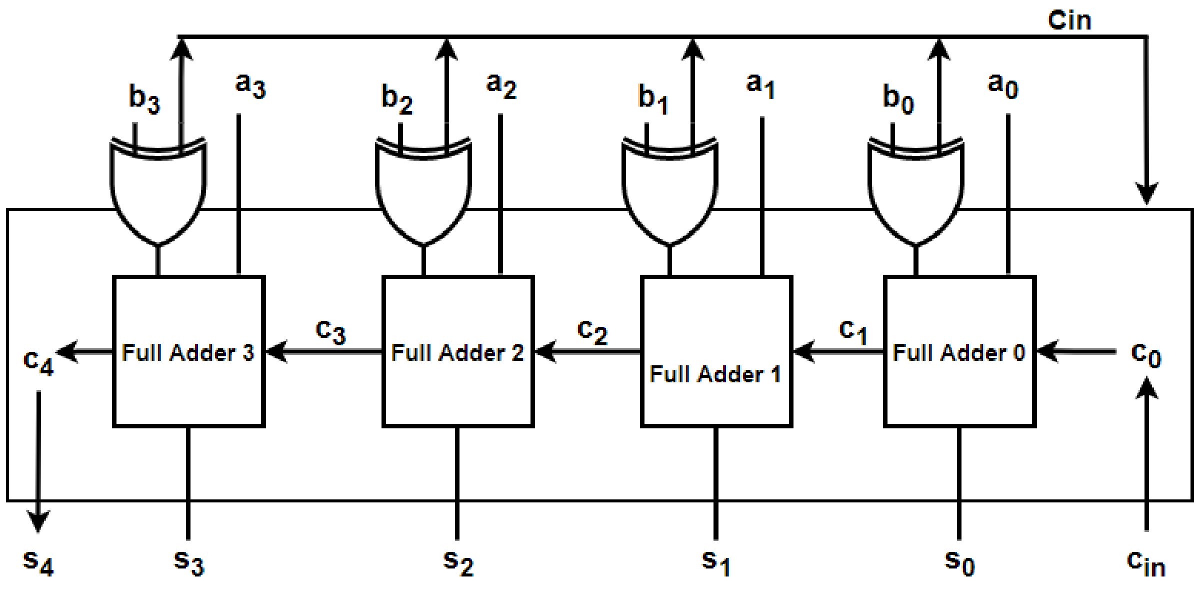
4. 4-bit Adder/Subtractor

Description

Implement a 4-bit Adder/Subtracter circuit using the package. Write a procedure for a 4-bit adder/subtractor circuit implemented using 4-bit ripple carry adder and include the procedure into the same package. Also, design a testbench for 4-bit adder/subtractor.

A 4 bit ripple carry adder is used to make a 4 bit Adder/Subtractor. It takes two input numbers (4 bit long) and one input carry bit and depending on the input carry bit value it gives addition or subtraction value as output(5 bit long). If input carry bit is 0 then it acts as adder and if input carry bit is 1 then it acts as a subtractor. For negative number output (MSB is 1), it gives result in 2's complement format.

Block Diagram



Entity

```
entity adderSubtractor is
    Port ( aa : in  STD_LOGIC_VECTOR (3 downto 0);
           bb : in  STD_LOGIC_VECTOR (3 downto 0);
           cin : in  STD_LOGIC;
           s : out  STD_LOGIC_VECTOR (4 downto 0));
end adderSubtractor;
```

Procedure for 4-bit Adder/Subtractor in package

```
procedure adderSubtractorProcedure(a:in std_logic_vector;b:in
std_logic_vector;c:in std_logic;s:out std_logic_vector) is
      variable p:std_logic_vector(3 downto 0);
      variable ss:std_logic_vector(4 downto 0);
      begin
                  p(3 downto 0):=b(3 downto 0) xor (c & c & c & c);
                  proc3:fourBitRippleCarryAdderProcedure(a(3 downto 0),p(3
downto 0),c,ss(4 downto 0));
                  if c='1' then
                              if ss(4)='1' then
                                          ss(4):='0';
                              elsif ss(4)='0' then
                                          ss(4):='1';
                              end if;
                  end if;
                  s:=ss;
      end procedure;
```

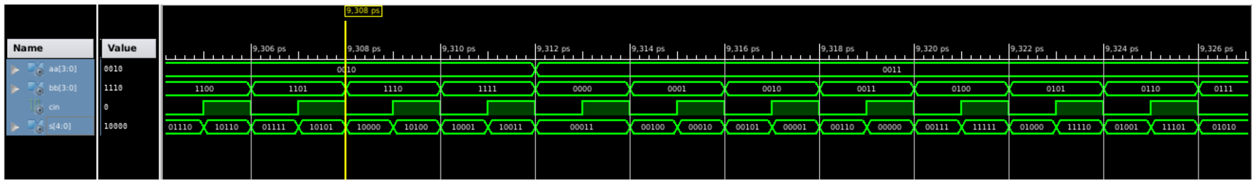
Architecture

```
architecture Behavioral of adderSubtractor is

begin
     p1:process(aa,bb,cin)
     variable ss:std_logic_vector(4 downto 0);
     begin
            proc0:adderSubtractorProcedure(aa(3 downto 0),bb(3 downto
0),cin,ss(4 downto 0));
            s<=ss;
     end process;
end Behavioral;
```

TestBench

```vhdl
BEGIN

      -- Instantiate the Unit Under Test (UUT)
   uut: adderSubtractor PORT MAP (
          aa => aa,
          bb => bb,
          cin => cin,
          s => s
        );
   -- Stimulus process
   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;
                        cin<='0';
                        wait for 1 ps;
                        cin<='1';
                        wait for 1 ps;
                  end loop;
            end loop;
   end process;
END;
```

Timing Diagram



5. BCD Adder

Description

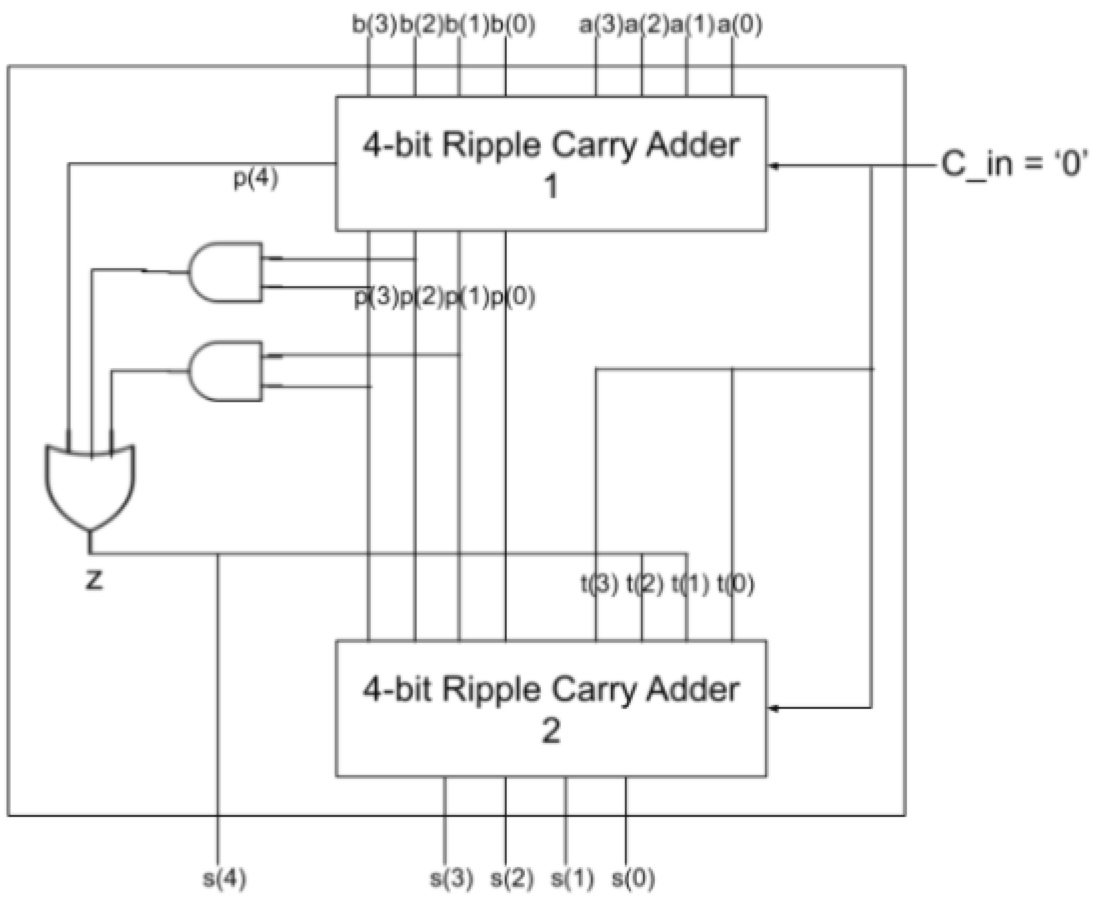
Implement a BCD adder using the package. Write a procedure for a BCD adder implemented using 4-bit ripple carry adder and include the procedure into the same package. Also, design a testbench for BCD adder.

In BCD or Binary Coded Decimal representation, every digit of a decimal number is converted to a four bit binary number. So, if the decimal number is 16 then the corresponding BCD number will be "00010110". Rest of the things are the same as a four bit ripple carry adder. It takes two four bit binary numbers (maximum 9 or 1001) as inputs and gives their sum as output (5 bit long) in BCD format.

Truth Table

Dec	Binary Sum					BCD Sum				
	p4	**p3**	**p2**	**p1**	**p0**	**q4(z)**	**q3**	**q2**	**q1**	**q0**
0-9	b4	b3	b2	b1	b0	b4	b3	b2	b1	b0
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
...	...	...	...	...	...	...	...	...	...	...
31	...	...	...	...	...	...	...	...	…	...

Block Diagram



Entity

```
entity bcdAdder is
    Port ( aa : in  STD_LOGIC_VECTOR (3 downto 0);
           bb : in  STD_LOGIC_VECTOR (3 downto 0);
           ss : out  STD_LOGIC_VECTOR (4 downto 0));
end bcdAdder;
```

Procedure BCD Adder in package

```
procedure bcdAdderProcedure(a:in std_logic_vector;b:in
std_logic_vector;s:out std_logic_vector) is
      variable p:std_logic_vector(4 downto 0);
      variable q:std_logic_vector(3 downto 0);
      variable c,z:std_logic;
      begin
                  c:='0';
                  proc4:fourBitRippleCarryAdderProcedure(a(3 downto 0),b(3
downto 0),c,p(4 downto 0));
                  z:=p(4) or (p(3) and p(2)) or (p(3) and p(1));
                  q:=c & z & z & c;
                  proc5:fourBitRippleCarryAdderProcedure(p(3 downto 0),q(3
downto 0),c,s(4 downto 0));
                  s(4):=z;
      end procedure;
```

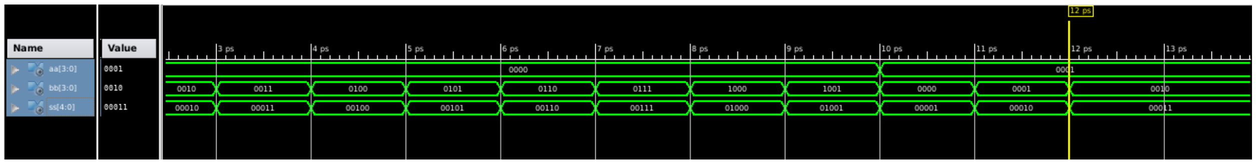
Architecture

```
architecture Behavioral of bcdAdder is

begin

p1:process(aa,bb)
   variable s:std_logic_vector(4 downto 0);
   begin
      proc0:bcdAdderProcedure(aa(3 downto 0),bb(3 downto 0),s(4 downto 0));
      ss<=s;
   end process;

end Behavioral;
```

TestBench

```vhdl
BEGIN

      -- Instantiate the Unit Under Test (UUT)
   uut: bcdAdder PORT MAP (
          aa => aa,
          bb => bb,
          ss => ss
        );

   -- Stimulus process
   stim_proc: process
   variable j,k:integer;
      variable binA,binB:std_logic_vector(3 downto 0);
      begin
            for k in 0 to 9 loop
                  procA:decimalToBinaryProcedure(k,4,binA);
                  aa<=binA;
                  for j in 0 to 9 loop
                        procB:decimalToBinaryProcedure(j,4,binB);
                        bb<=binB;
                        wait for 1 ps;
                  end loop;
            end loop;
   end process;
END;
```

Timing Diagram



12 ps
Name
Value
aa[3:0]
bb[3:0]
ss[4:0]
0001
0010
00011
3 ps
4 ps
5 ps
6 ps
7 ps
8 ps
9 ps
10 ps
11 ps
12 ps
13 ps
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
0000
0001
0010
00010
00011
00100
00101
00110
00111
01000
01001
00001
00010
00011