VLSI System Design

ELE301P

LAB - 2 - Report

Praveen B R

COE19B007

Submission Date: 01/09/2021

INDEX

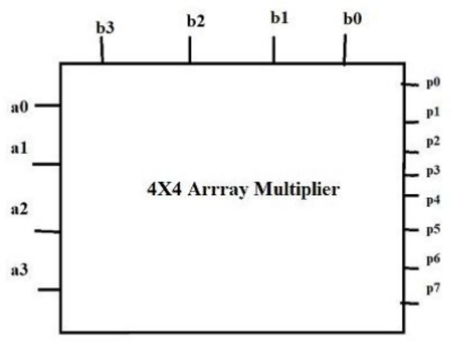
Implement 4*4 array multiplier

Objective:
To implement 4*4 array multiplier in Verilog using,
- Structural modelling
- Behavioural modelling

Theory:
Array multiplier is used to perform the arithmetic operations.The design structure of the array Multiplier is regular, it is based on the add shift algorithm principle.

*Partial product = the multiplicand * multiplier bit*

where AND gates are used for the product, the summation is done using Full Adders and Half Adders where the partial product is shifted according to their bit orders. The 4×4 array multiplier shown below has two 4 inputs and 8 output,



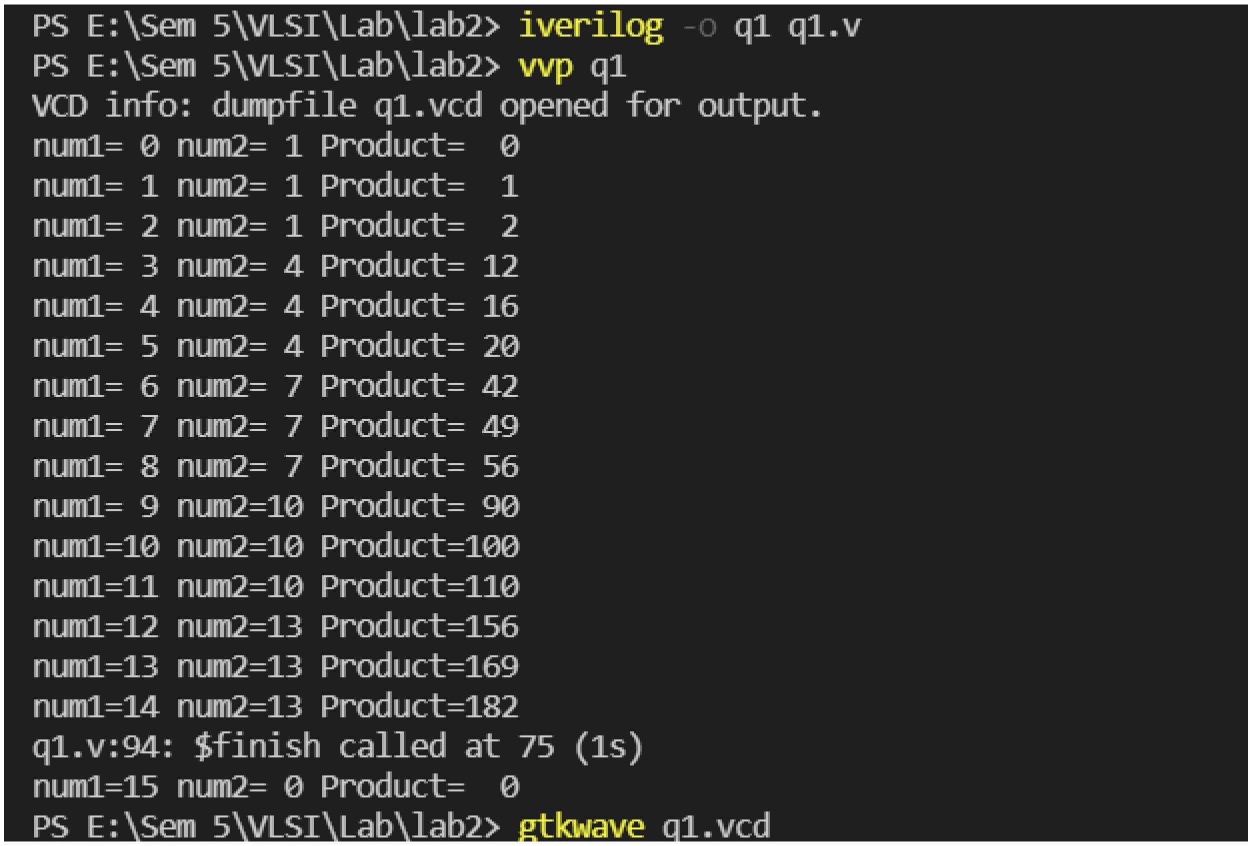
Structural Modelling:

Code

```
module fa(a,b,cin,sum,cout);  //module for full adder
  input a,b,cin;
  output cout,sum;

  wire s1,s2,s3;
  xor(s1,a,b);
  and(s2,a,b);
  xor(sum,s1,cin);
  and(s3,s1,cin);
  or(cout,s3,s2);
endmodule
```

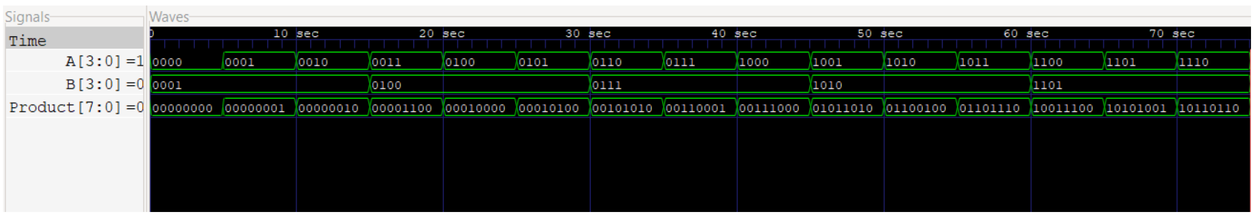
```
module rca(A,B,Cin,Cout,Sum); //module for product carry adder
  input [3:0]A,B;
  input Cin;
  output [3:0]Sum;
  output Cout;
  wire t[3:0];
  fa f1(A[0],B[0],Cin,Sum[0],t[0]);
  fa f2(A[1],B[1],t[0],Sum[1],t[1]);
  fa f3(A[2],B[2],t[1],Sum[2],t[2]);
  fa f4(A[3],B[3],t[2],Sum[3],Cout);
endmodule

module mul(A,B,Product);  //module for 4*1 multiplier
  input [3:0]A;
  input B;
  output [3:0]Product;

  and a1(Product[3],A[3],B);
  and a2(Product[2],A[2],B);
  and a3(Product[1],A[1],B);
  and a4(Product[0],A[0],B);
endmodule

module am_structural(A,B,Sum);  //main module
  input [3:0]A,B;
  output [7:0]Sum;
  wire [3:0]t1,t2,s1,t3,t4,t5,s2,t6,s3;
  wire [3:0]Cout;
  wire cin=1'b0;

  and a1(Sum[0],A[0],B[0]);
  and a2(t1[0],A[1],B[0]);
  and a3(t1[1],A[2],B[0]);
  and a4(t1[2],A[3],B[0]);
  assign t1[3]=0;

  mul m1(A,B[1],t2);  //first rca program
  rca f1(t1,t2,cin,Cout[0],s1);
  assign Sum[1] = s1[0];
  assign t5 = {Cout[0],s1[3:1]};
  mul m2(A,B[2],t3);  //second rca program
  rca f2(t5,t3,cin,Cout[1],s2);
  assign Sum[2] = s2[0];
  assign t6 = {Cout[1],s2[3:1]};

  mul m3(A,B[3],t4);  //third rca program
  rca f3(t6,t4,cin,Cout[2],s3);
  assign Sum[6:3] = s3;
  assign Sum[7] = Cout[2];
  endmodule
module am_structural_tb;  //test_bench
  reg [3:0]A,B;
  wire [7:0]Product;

  am_structural product(A,B,Product);
  initial
  begin
    #0 A=4'b0000;B=4'b0001;
    #75 $finish;
  end
  initial
   begin
   $dumpfile ("q1.vcd");
   $dumpvars (0,am_structural_tb);
   end
  initial
  begin
   $monitor("num1=%d num2=%d Product=%d",A,B,Product);
  end
  always
  begin
   #5 A = A + 1;
  end
   always
  begin
   #15 B = B + 3;
  end
endmodule
```

Output Terminal



```
PS E:\Sem 5\VLSI\Lab\lab2> iverilog -o q1 q1.v
PS E:\Sem 5\VLSI\Lab\lab2> vvp q1
VCD info: dumpfile q1.vcd opened for output.
num1= 0 num2= 1 Product=  0
num1= 1 num2= 1 Product=  1
num1= 2 num2= 1 Product=  2
num1= 3 num2= 4 Product= 12
num1= 4 num2= 4 Product= 16
num1= 5 num2= 4 Product= 20
num1= 6 num2= 7 Product= 42
num1= 7 num2= 7 Product= 49
num1= 8 num2= 7 Product= 56
num1= 9 num2=10 Product= 90
num1=10 num2=10 Product=100
num1=11 num2=10 Product=110
num1=12 num2=13 Product=156
num1=13 num2=13 Product=169
num1=14 num2=13 Product=182
q1.v:94: $finish called at 75 (1s)
num1=15 num2= 0 Product=  0
PS E:\Sem 5\VLSI\Lab\lab2> gtkwave q1.vcd
```

Waveform



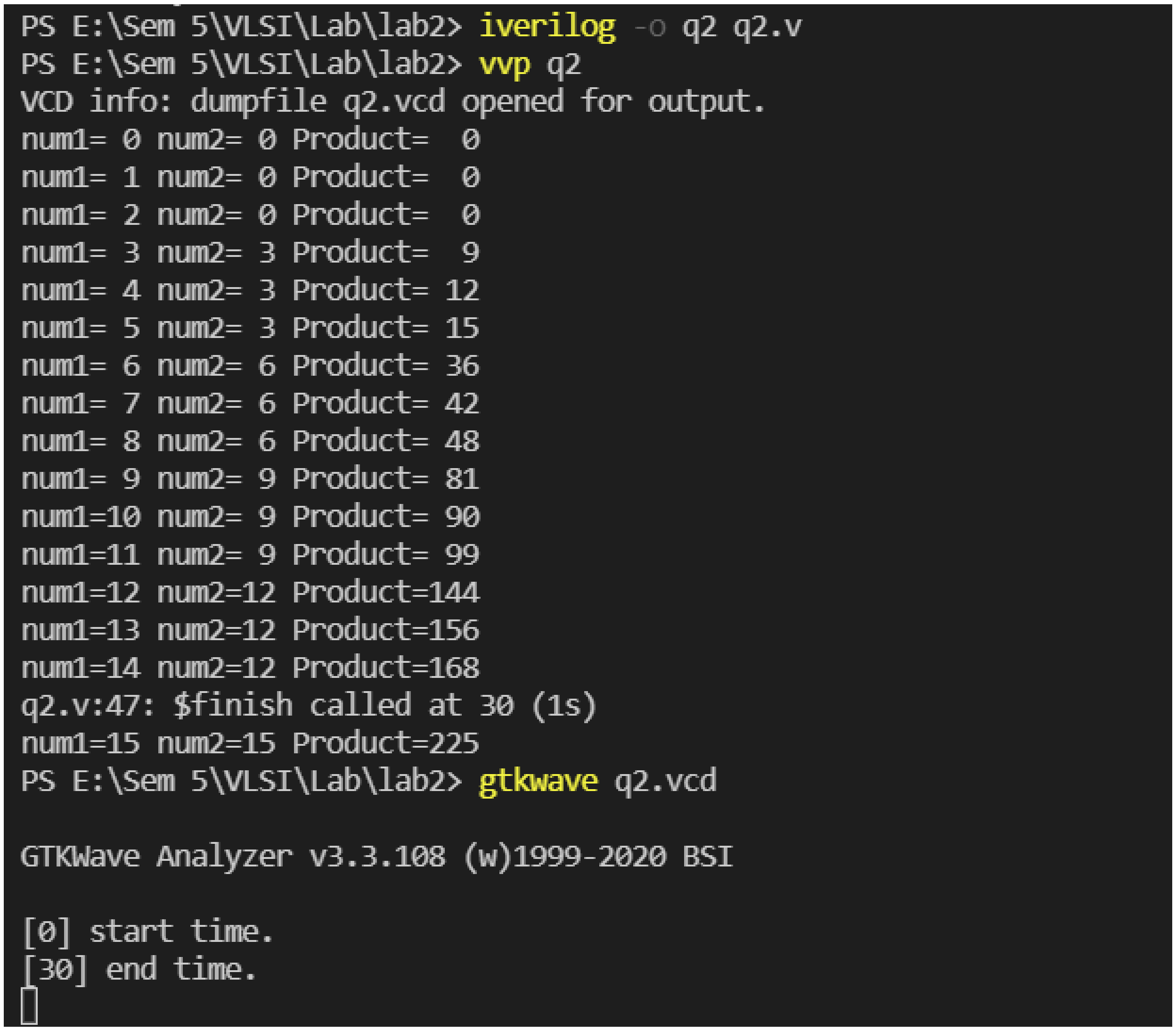
Behavioural Modelling:

Code

```
module am_behavioural(A,B,Sum); //main module
  input [3:0]A,B;
  output reg[7:0] Sum;

  reg[5:0][3:0] t;
  reg[3:0][4:0] s;
  reg[2:0] cout;

  always@(A,B)
  begin
    t[0] = Product(A,B[0]);
    t[1] = Product(A,B[1]);
    Sum[0] = t[0][0];
    s[0] = t[1] + {1'b0,t[0][3:1]};
    t[2] = Product(A,B[2]);
    t[3] = Product(A,B[3]);
    s[1] = t[2] + s[0][4:1];
    s[2] = t[3] + s[1][4:1];
    Sum[1] = s[0][0];
    Sum[2] = s[1][0];
    Sum[7:3] = s[2];
  end


  function [3:0] Product(input [3:0]C,D);
    begin
      Product[3] = C[3]&D;
      Product[2] = C[2]&D;
      Product[1] = C[1]&D;
      Product[0] = C[0]&D;
    end
  endfunction
```

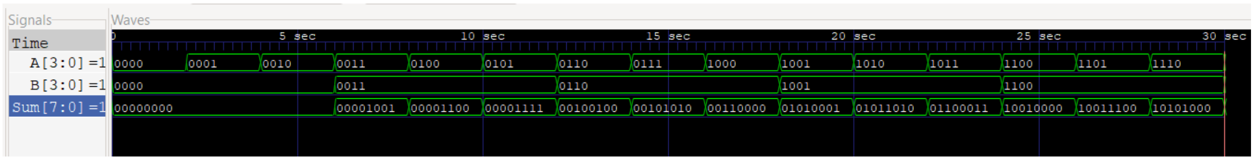
```
endmodule

module am_behavioural_tb;   //test_bench
  reg [3:0]A,B;
  wire [7:0]Product;

  am_behavioural product(A,B,Product);
  initial
  begin
    #0 A=4'b0000;B=4'b0000;
    #30 $finish;
  end
  initial
   begin
   $dumpfile ("q2.vcd");
   $dumpvars (0,am_behavioural_tb);
   end
  initial
  begin
   $monitor("num1=%d num2=%d Product=%d",A,B,Product);
  end
  always
  begin
   #2 A = A + 1;
  end
   always
  begin
   #6 B = B + 3;
  end
endmodule
```

Output Terminal



```
PS E:\Sem 5\VLSI\Lab\lab2> iverilog -o q2 q2.v
PS E:\Sem 5\VLSI\Lab\lab2> vvp q2
VCD info: dumpfile q2.vcd opened for output.
num1= 0 num2= 0 Product=  0
num1= 1 num2= 0 Product=  0
num1= 2 num2= 0 Product=  0
num1= 3 num2= 3 Product=  9
num1= 4 num2= 3 Product= 12
num1= 5 num2= 3 Product= 15
num1= 6 num2= 6 Product= 36
num1= 7 num2= 6 Product= 42
num1= 8 num2= 6 Product= 48
num1= 9 num2= 9 Product= 81
num1=10 num2= 9 Product= 90
num1=11 num2= 9 Product= 99
num1=12 num2=12 Product=144
num1=13 num2=12 Product=156
num1=14 num2=12 Product=168
q2.v:47: $finish called at 30 (1s)
num1=15 num2=15 Product=225
PS E:\Sem 5\VLSI\Lab\lab2> gtkwave q2.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[30] end time.
```

Waveform



Conclusion:

Thus, 4*4 array multipliers have been implemented successfully using Verilog in Structural style and Behavioural style.

Other Questions

Q1) There is a conditional operator in verilog which can be used as follows: if the given condition is true then a given value will be assigned to the variable and if false other user defined value will be assigned. There is a similar operator in C. Find the verilog operator and its syntax

Ans:

The described operator is called **Conditional Operator** in verilog also.

Syntax : ***Condition?Expression1:Expression2;***

eg : `assign a = (0)?8:9;` // 9 will be assigned to a

If the condition is true, expression 1 will be executed else expression 2 will be executed.

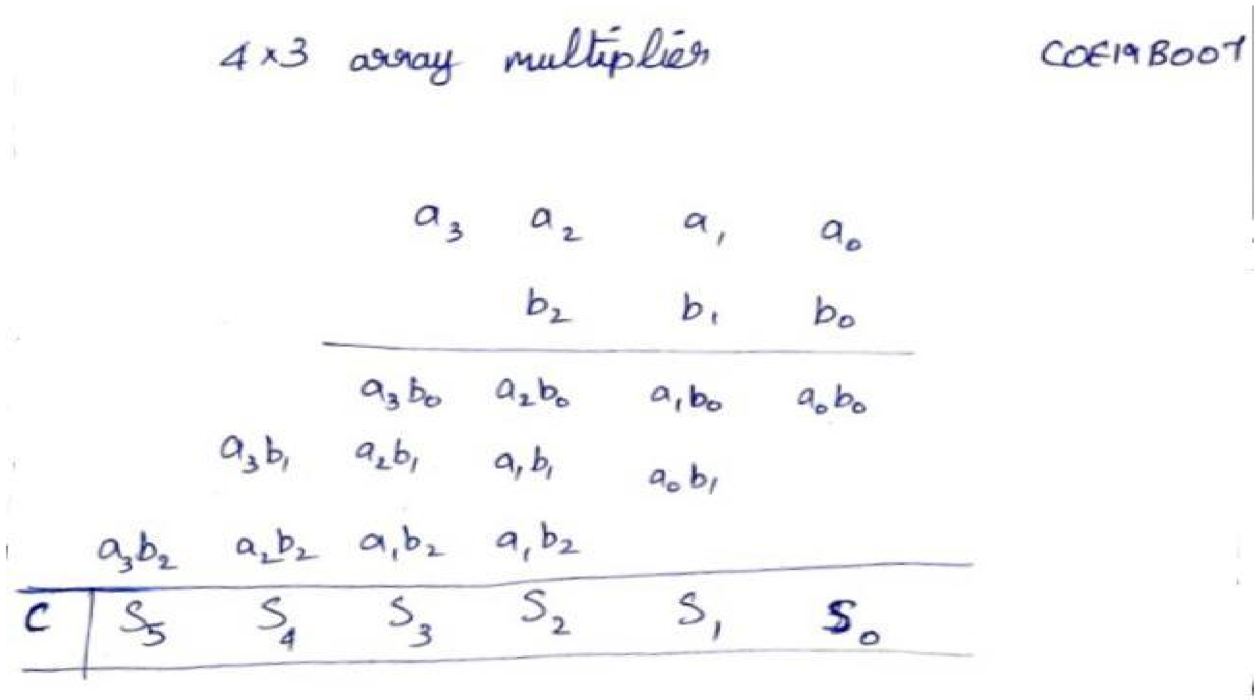
Q2) Find the functionality and use of verilog keyword "parameter"

Ans:

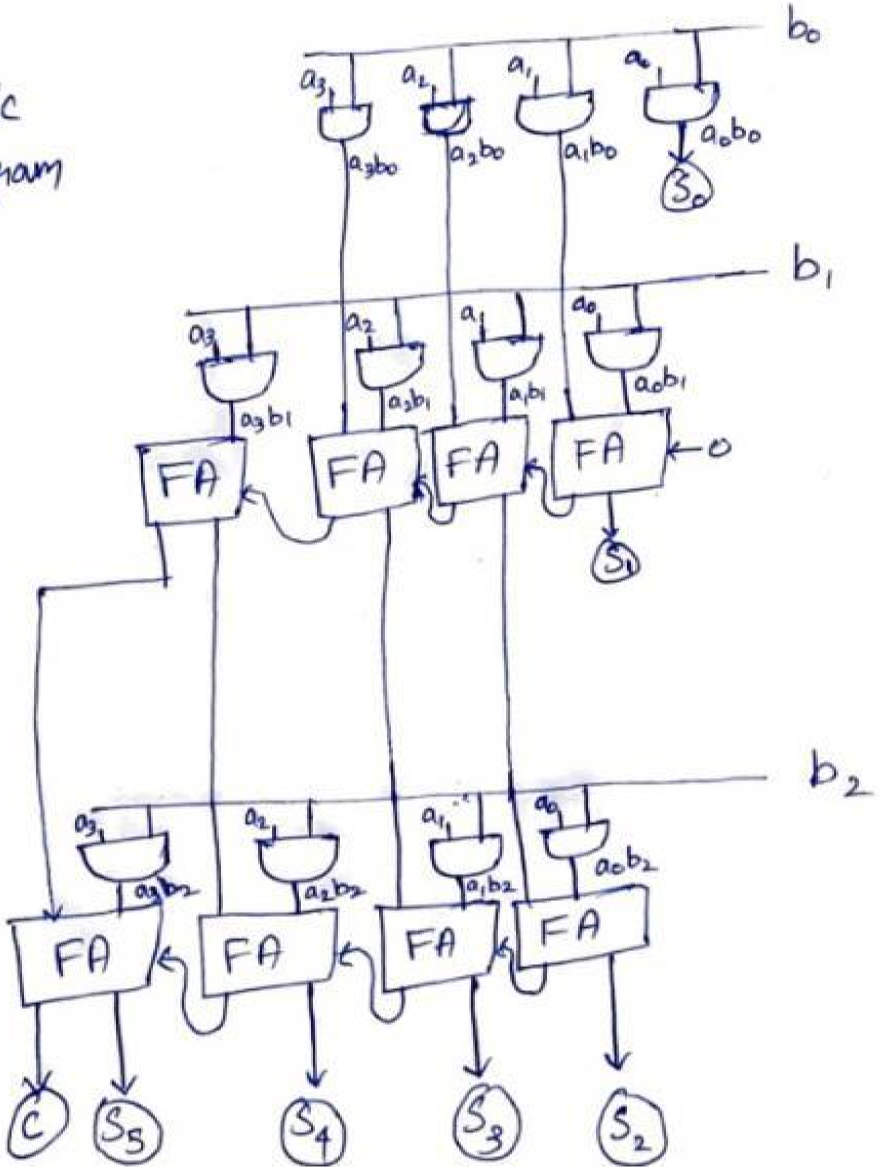
Parameter keyword in verilog is used to define constants, they do not belong to any other data type such as register or net data types. Value of a parameter can be changed using defparam.

Q3) In this experiment you have implemented 4*4 array multiplier. Now draw the logic diagram representation for 4*3 array multiplier.

Ans:



Logic Diagram


b_0
a_3
a_2
a_1
a_0
a_3b_0
a_2b_0
a_1b_0
a_0b_0
S_0
b_1
a_3
a_2
a_1
a_0
a_3b_1
a_2b_1
a_1b_1
a_0b_1
FA
FA
FA
FA
0
S_1
b_2
a_3
a_2
a_1
a_0
a_3b_2
a_2b_2
a_1b_2
a_0b_2
FA
FA
FA
FA
C
S_5
S_4
S_3
S_2