Prob 1

1.1

Verilog

```
module cp_4bit (
  input  [3:0] A     ,
  input  [3:0] B     ,
  output       A_lt_B,
  output       A_eq_B,
  output       A_gt_B
);

wire X3; // X3 = A[3] xnor B[3]
wire X2; // X2 = A[2] xnor B[2]
wire X1; // X1 = A[1] xnor B[1]
wire X0; // X0 = A[0] xnor B[0]
wire outl1; // outl1 = (~A[3]) & B[3]
wire outl2; // outl2 = X3 & (~A[2]) & B[2]
wire outl3; // outl3 = X3 & X2 & (~A[1]) & B[1]
wire outl4; // outl4 = X3 & X2 & X1 & (~A[0]) & B[0]
wire outg1; // outg1 = A[3] & (~B[3])
wire outg2; // outg2 = X3 & A[2] & (~B[2])
wire outg3; // outg3 = X3 & X2 & A[1] & (~B[1])
wire outg4; // outg4 = X3 & X2 & X1 & A[0] & (~B[0])

// X3 = A[3] xnor B[3]
xnor x0(X3, A[3], B[3]);

// X2 = A[2] xnor B[2]
xnor x1(X2, A[2], B[2]);
```

```
// X1 = A[1] xnor B[1]
xnor x2(X1, A[1], B[1]);

// X0 = A[0] xnor B[0]
xnor x3(X0, A[0], B[0]);

// outl1 = (~A[3]) & B[3]
and a0(outl1, ~A[3], B[3]);

// outl2 = X3 & (~A[2]) & B[2]
and a1(outl2, X3, ~A[2], B[2]);

// outl3 = X3 & X2 & (~A[1]) & B[1]
and a3(outl3, X3, X2, ~A[1], B[1]);

// outl4 = X3 & X2 & X1 & (~A[0]) & B[0]
and a4(outl4, X3, X2, X1, ~A[0], B[0]);

// A_lt_B = outl1 | outl2 | outl3 | outl4
or o0(A_lt_B, outl1, outl2, outl3, outl4);

// A_eq_B = X3 & X2 & X1 & X0
and a5(A_eq_B, X3, X2, X1, X0);

// outg1 = A[3] & (~B[3])
and a6(outg1, A[3], ~B[3]);

// outg2 = X3 & A[2] & (~B[2])
and a7(outg2, X3, A[2], ~B[2]);
```

// outg3 = X3 & X2 & A[1] & (~B[1])

and a8(outg3, X3, X2, A[1], ~B[1]);

// outg4 = X3 & X2 & X1 & A[0] & (~B[0])

and a9(outg4, X3, X2, X1, A[0], ~B[0]);

// A_gt_B = outg1 | outg2 | outg3 | outg4

or o1(A_gt_B, outg1, outg2, outg3, outg4);

endmodule : cp_4bit

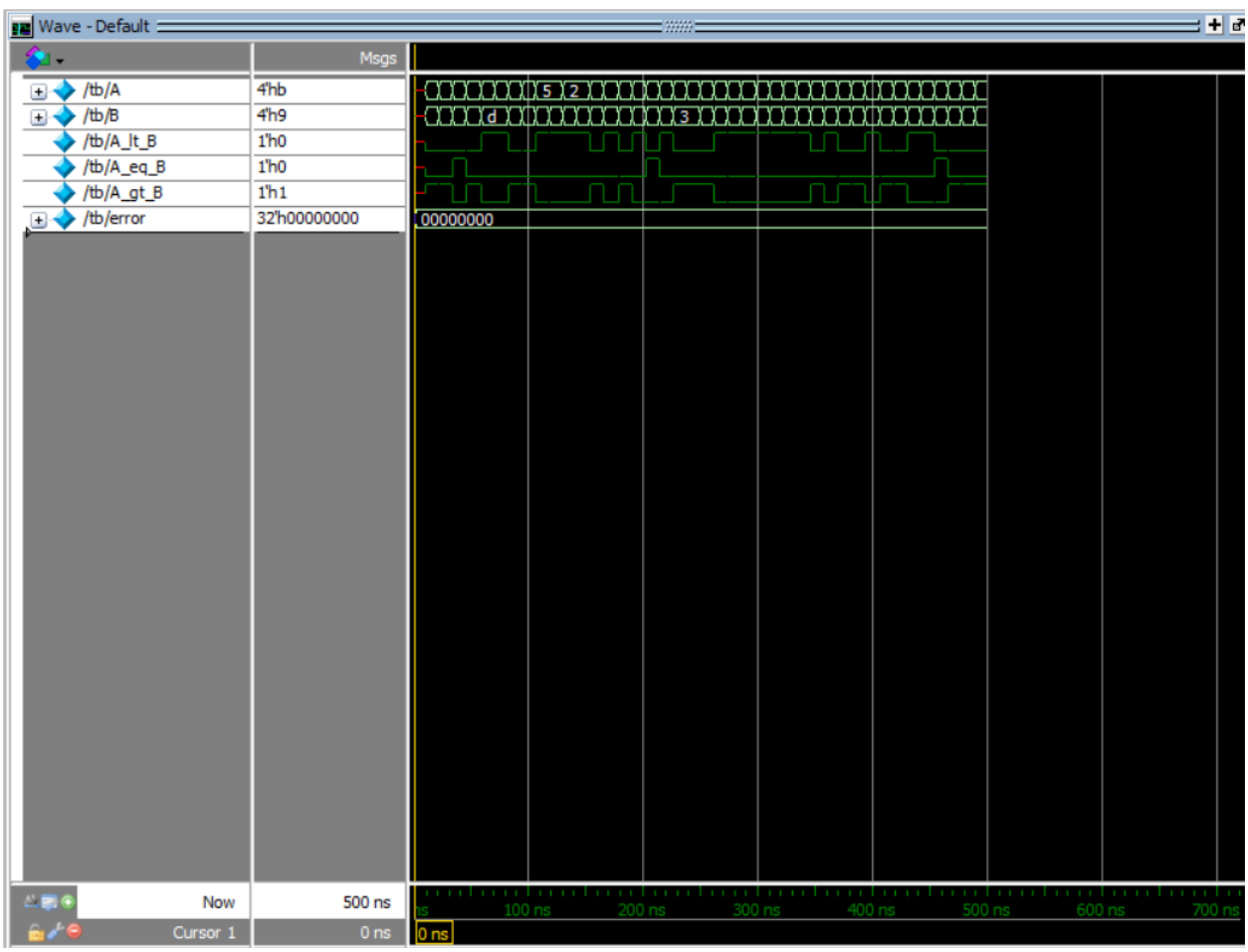
Testbench

```
module tb ();

reg [3:0] A     ;
reg [3:0] B     ;
wire      A_lt_B;
wire      A_eq_B;
wire      A_gt_B;
integer   error ;

cp_4bit dut(
  .A     (A     ),
  .B     (B     ),
  .A_lt_B(A_lt_B),
  .A_eq_B(A_eq_B),
  .A_gt_B(A_gt_B)
);

initial begin
  error = 0;
  repeat(100) begin
    #10;
    A = $random();
    B = $random();
    #2;
    if (A_lt_B != (A < B) || A_eq_B != (A == B) || A_gt_B != (A > B)) begin
      error = error + 1;
    end
    $display($stime, ":\t\tA = ", A, "\t\tB = ", B, "\t\tA_lt_B = ", A_lt_B, "\t\tA_eq_B = ", A_eq_B, "\t\tA_gt_B = ", A_gt_B, "\t\terror = ", error);
  end
  $stop;
end

endmodule : tb
```

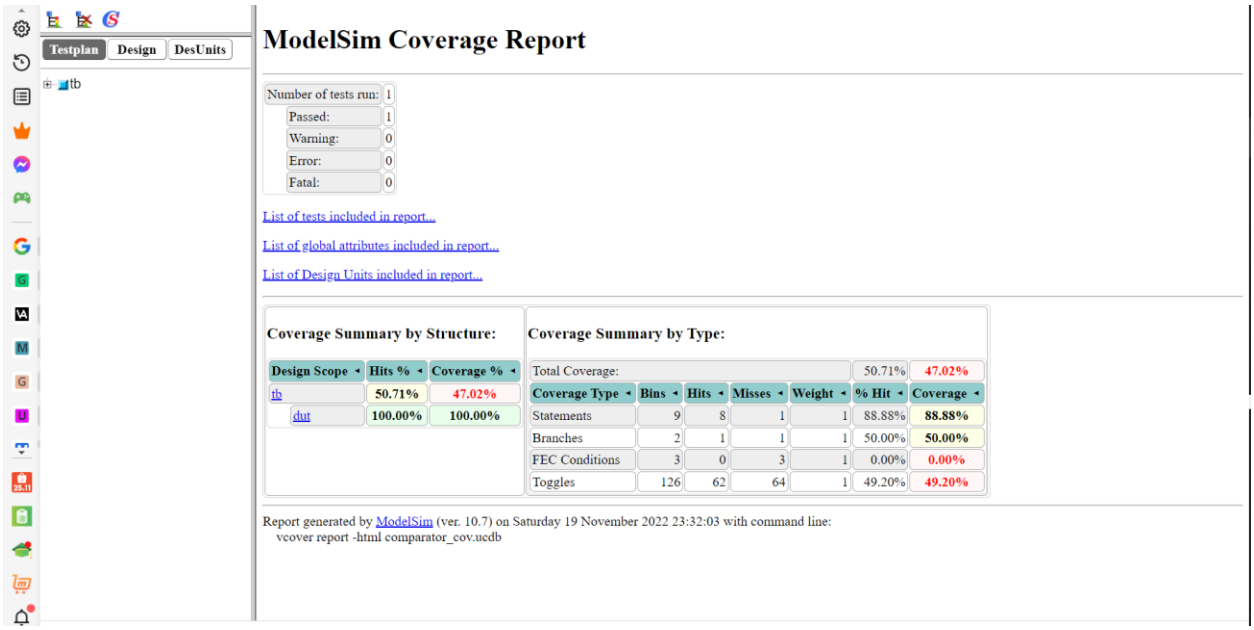
Script

```
# vsim -voptargs="+acc" work.tb
# Start time: 23:20:12 on Nov 19,2022
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.tb(fast)
# Loading work.cp_4bit(fast)
add wave -position insertpoint sim:/tb/*
VSIM 11> run
#          12:   A =  4          B =  1          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#          24:   A =  9          B =  3          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#          36:   A = 13          B = 13          A_lt_B = 0      A_eq_B = 1      A_gt_B = 0      error =          0
#          48:   A =  5          B =  2          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#          60:   A =  1          B = 13          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#          72:   A =  6          B = 13          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#          84:   A = 13          B = 12          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#          96:   A =  9          B =  6          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         108:   A =  5          B = 10          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         120:   A =  5          B =  7          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         132:   A =  2          B = 15          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         144:   A =  2          B = 14          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         156:   A =  8          B =  5          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         168:   A = 12          B = 13          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         180:   A = 13          B =  5          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         192:   A =  3          B = 10          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         204:   A =  0          B =  0          A_lt_B = 0      A_eq_B = 1      A_gt_B = 0      error =          0
#         216:   A = 10          B = 13          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         228:   A =  6          B =  3          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         240:   A = 13          B =  3          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         252:   A = 11          B =  5          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         264:   A =  2          B = 14          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         276:   A = 13          B = 15          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         288:   A =  3          B = 10          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         300:   A = 10          B = 12          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         312:   A =  2          B = 10          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         324:   A =  1          B =  8          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         336:   A =  8          B =  9          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         348:   A = 11          B =  6          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         360:   A =  6          B = 14          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         372:   A = 12          B = 10          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         384:   A = 11          B =  1          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         396:   A =  5          B = 15          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         408:   A = 11          B = 10          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         420:   A = 14          B =  5          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         432:   A =  1          B =  9          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         444:   A =  2          B = 12          A_lt_B = 1      A_eq_B = 0      A_gt_B = 0      error =          0
#         456:   A = 15          B = 15          A_lt_B = 0      A_eq_B = 1      A_gt_B = 0      error =          0
#         468:   A =  8          B =  7          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         480:   A = 15          B = 12          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0
#         492:   A = 11          B =  9          A_lt_B = 0      A_eq_B = 0      A_gt_B = 1      error =          0

VSIM 12>
```

Wave



Report



Testplan Design DesUnits

tb

ModelSim Coverage Report

Number of tests run:	1
Passed:	1
Warning:	0
Error:	0
Fatal:	0

List of tests included in report...

List of global attributes included in report...

List of Design Units included in report...

Coverage Summary by Structure:

Design Scope	Hits %	Coverage %
tb	50.71%	47.02%
dut	100.00%	100.00%

Coverage Summary by Type:

Total Coverage:					50.71%	47.02%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	9	8	1	1	88.88%	88.88%
Branches	2	1	1	1	50.00%	50.00%
FEC Conditions	3	0	3	1	0.00%	0.00%
Toggles	126	62	64	1	49.20%	49.20%

Report generated by ModelSim (ver. 10.7) on Saturday 19 November 2022 23:32:03 with command line:
vcover report -html comparator_cov.ucdb

1.2

Full adder 4 bit

Verilog

F:/VHDL/ET5080E - 20221/HW6/Prob 1/4bit_fulladder/FA.v (/test_adder/dut/FA3) - Default

```
module full_adder_4bit(a,b,cin,sum,cout);
input [3:0]a,b;
input cin;
output [3:0] sum;
output cout;
wire c1,c2,c3;

// Instantiate the 1-bit adders
full_adder FA0(a[0],b[0],cin,sum[0],c1);
full_adder FA1(a[1],b[1],c1,sum[1],c2);
full_adder FA2(a[2],b[2],c2,sum[2],c3);
full_adder FA3(a[3],b[3],c3,sum[3],cout);
endmodule

module full_adder(a,b,cin,sum,cout);
input a,b,cin;
output sum,cout;
reg sum,cout;
always@(a or b or cin)
begin
sum = a^b^cin;
cout = (a&b)|( a& cin)|( b & cin);
end
endmodule
```

Testbench

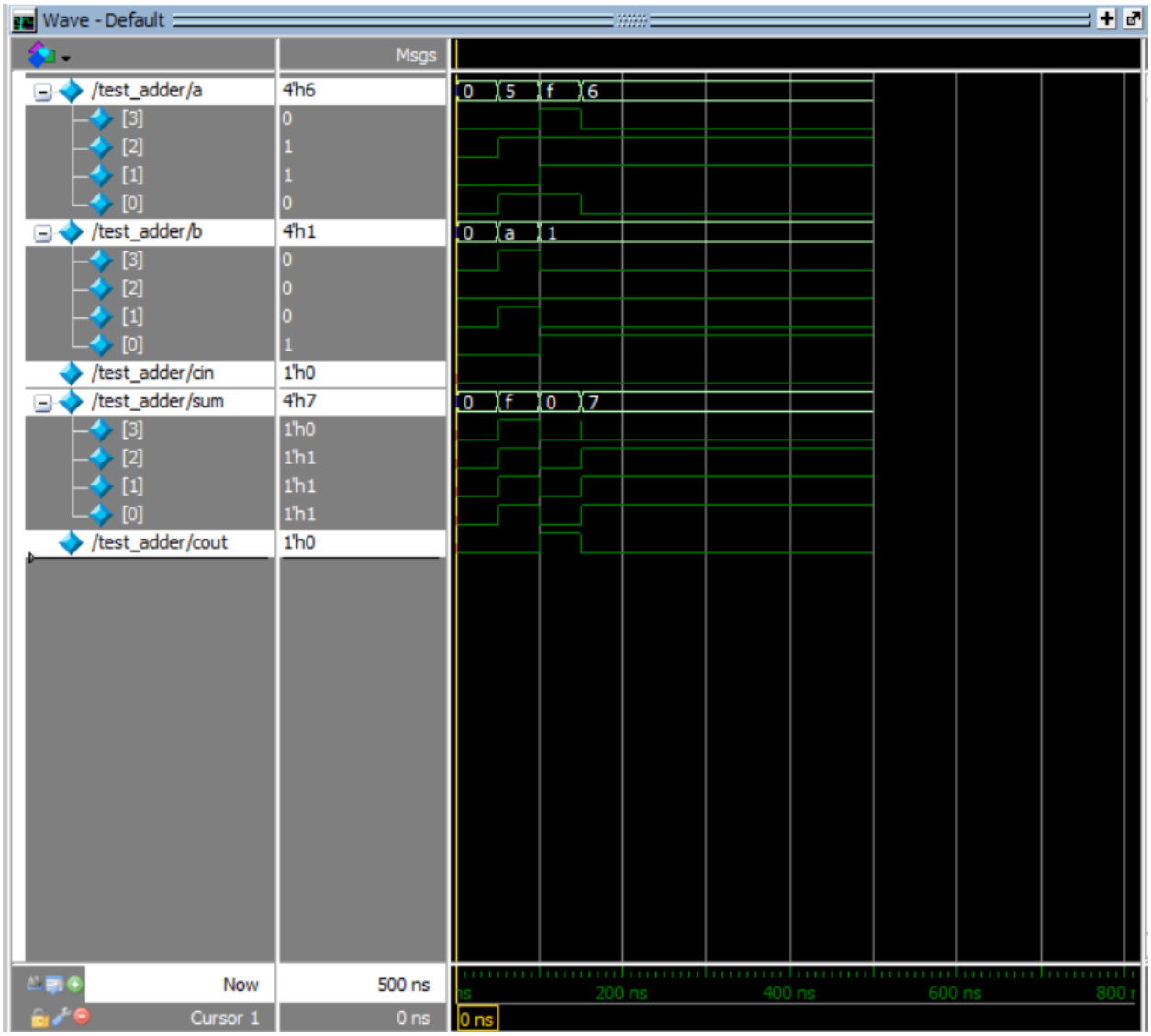
F:/VHDL/ET5080E - 20221/HW6/Prob 1/4bit_fulladder/FA_tb.v (/test_adder) - Default

```
module test_adder;
reg[3:0] a,b;
reg cin;
wire[3:0] sum;
wire cout;
full_adder_4bit dut(a,b,cin,sum,cout);
initial
begin
a = 4'b0000;
b = 4'b0000;
cin = 1'b0;
#50;
a = 4'b0101;
b = 4'b1010;
#50;
a = 4'b1111;
b = 4'b0001;
#50;
a = 4'b0110;
b = 4'b0001;
$display(":\t\t a =",a,"\t\t b =",b, "\t\t cin= ",cin,"\t\t sum = ", sum, "\t\t cout= ",
end
endmodule
```

Script

```
VSIM 8> run
# :                a = 6            b = 1
 cin= 0            sum =  0         cout= 1
VSIM 9> run -all
VSIM 10> run -all
```

Wave


Wave - Default
Msgs
/test_adder/a
4'h6
[3]
0
[2]
1
[1]
1
[0]
0
/test_adder/b
4'h1
[3]
0
[2]
0
[1]
0
[0]
1
/test_adder/cin
1'h0
/test_adder/sum
4'h7
[3]
1'h0
[2]
1'h1
[1]
1'h1
[0]
1'h1
/test_adder/cout
1'h0
Now
500 ns
Cursor 1
0 ns
200 ns
400 ns
600 ns
0 ns

Report

Testplan | Design | DesUnits

fa_tb

ModelSim Coverage Report

Number of tests run:	1
Passed:	1
Warning:	0
Error:	0
Fatal:	0

List of tests included in report...

List of global attributes included in report...

List of Design Units included in report...

Coverage Summary by Structure:

Design Scope	Hits %	Coverage %
fa_tb	73.33%	81.81%
dut	62.50%	62.50%

Coverage Summary by Type:

Total Coverage:					73.33%	81.81%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	8	8	0	1	100.00%	100.00%
Toggles	22	14	8	1	63.63%	63.63%

Report generated by ModelSim (ver. 10.7) on Sunday 13 November 2022 19:59:21 with command line:
vcover report -html fa_cov.ucdb

javascript: d.o(1);

Carry look ahead adder 4 bit

Verilog

F:/VHDL/ET5080E - 20221/HW6/Prob 1/4bit_carrylookahead/CA.v (/Test_CLA_4bit/uut) - Default

```verilog
module CLA_4bit(
    output [3:0] S,
    output Cout,PG,GG,
    input [3:0] A,B,
    input Cin
    );
    wire [3:0] G,P,C;

    assign G = A & B; //Generate
    assign P = A ^ B; //Propagate
    assign C[0] = Cin;
    assign C[1] = G[0] | (P[0] & C[0]);
    assign C[2] = G[1] | (P[1] & G[0]) | (P[1] & P[0] & C[0]);
    assign C[3] = G[2] | (P[2] & G[1]) | (P[2] & P[1] & G[0]) |             (P[2] & P[1] & P[0] & C[0]);
    assign Cout = G[3] | (P[3] & G[2]) | (P[3] & P[2] & G[1]) | (P[3] & P[2] & P[1] & G[0]) |(P[3] & P[2] & P[1] & P[0] & C[0]);
    assign S = P ^ C;

    assign PG = P[3] & P[2] & P[1] & P[0];
    assign GG = G[3] | (P[3] & G[2]) | (P[3] & P[2] & G[1]) | (P[3] & P[2] & P[1] & G[0]);
endmodule
```

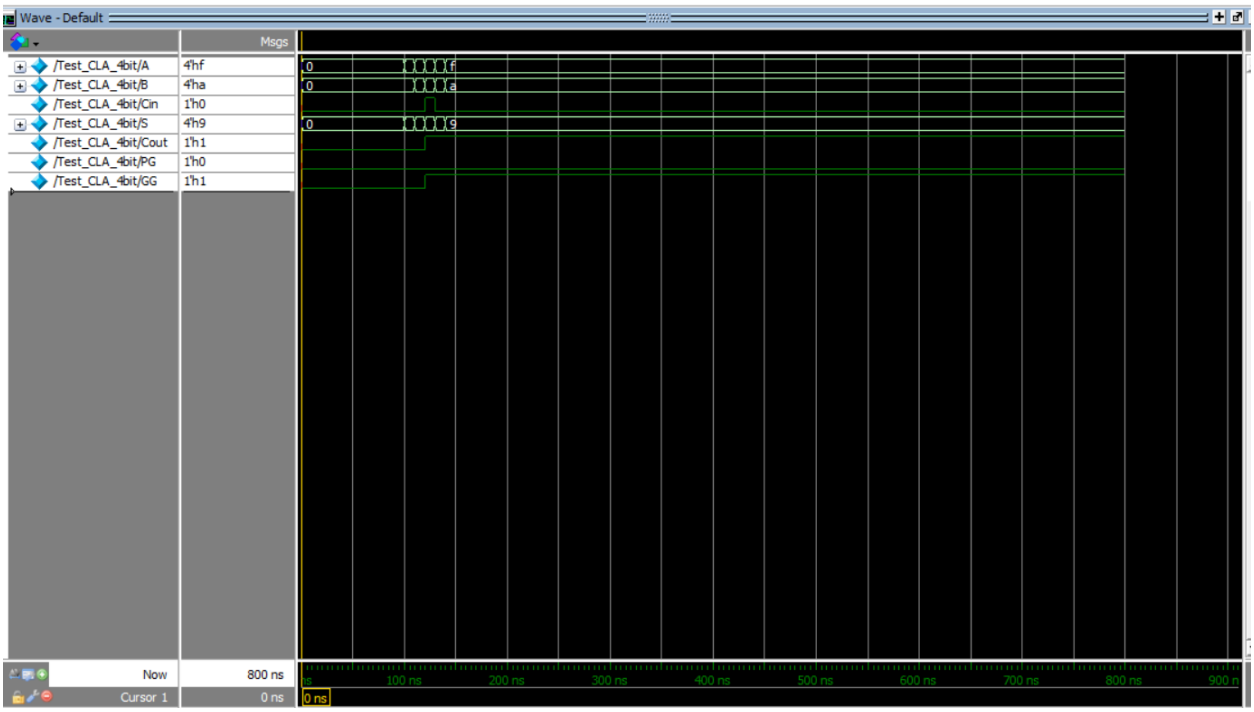
Testbench

```
module Test_CLA_4bit;
    // Inputs
    reg [3:0] A;
    reg [3:0] B;
    reg Cin;

    // Outputs
    wire [3:0] S;
    wire Cout;
    wire PG;
    wire GG;

    // Instantiate the Unit Under Test (UUT)
    CLA_4bit uut (
    .S(S),
    .Cout(Cout),
    .PG(PG),
    .GG(GG),
    .A(A),
    .B(B),
    .Cin(Cin)
    );

    initial begin
    // Initialize Inputs
    A = 0;  B = 0;  Cin = 0;
    // Wait 100 ns for global reset to finish
    #100;

    // Add stimulus here
    A=4'b0001;B=4'b0000;Cin=1'b0;
    #10 A=4'b100;B=4'b0011;Cin=1'b0;
    #10 A=4'b1101;B=4'b1010;Cin=1'b1;
    #10 A=4'b1110;B=4'b1001;Cin=1'b0;
    #10 A=4'b1111;B=4'b1010;Cin=1'b0;
    end

    initial begin
  $monitor("time=",$time,, "A=%b B=%b Cin=%b : Sum=%b Cout=%b PG=%b GG=%b",A,B,Cin,S,Cout,PG,GG);
    end
endmodule
```

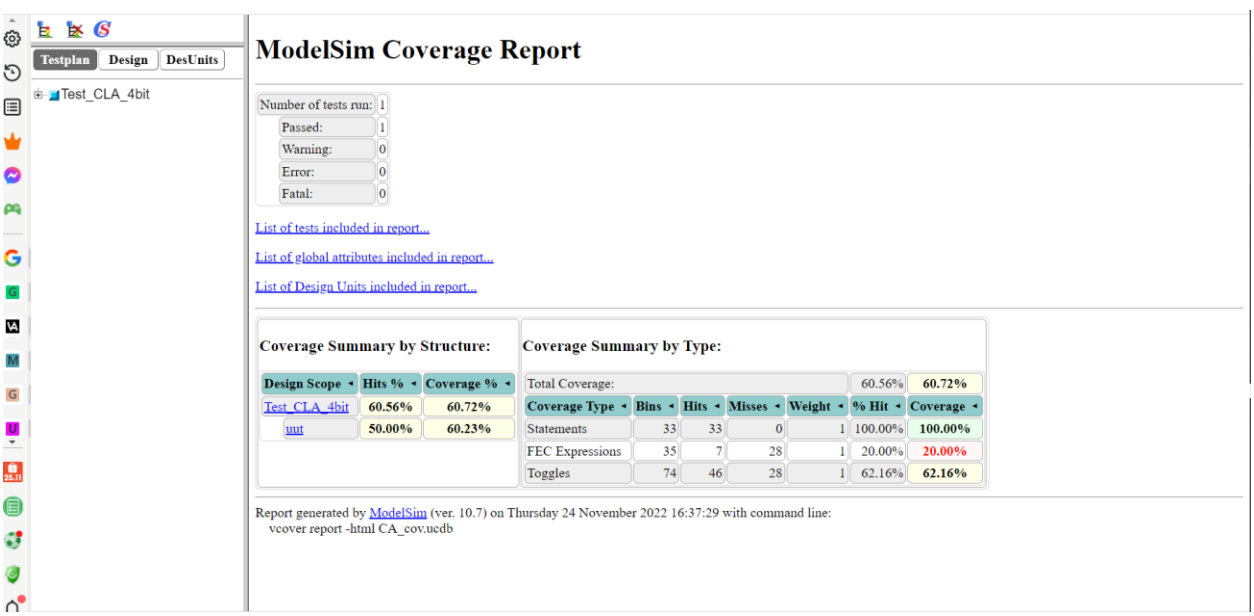
Script

```
# Loading work.CLA_4bit(fast)
add wave -position insertpoint sim:/Test_CLA_4bit/*
VSIM 14> run
# time=                    0 A=0000 B=0000 Cin=0 : Sum=0000 Cout=0 PG=0 GG=0
# time=                  100 A=0001 B=0000 Cin=0 : Sum=0001 Cout=0 PG=0 GG=0
# time=                  110 A=0100 B=0011 Cin=0 : Sum=0111 Cout=0 PG=0 GG=0
# time=                  120 A=1101 B=1010 Cin=1 : Sum=1000 Cout=1 PG=0 GG=1
# time=                  130 A=1110 B=1001 Cin=0 : Sum=0111 Cout=1 PG=0 GG=1
# time=                  140 A=1111 B=1010 Cin=0 : Sum=1001 Cout=1 PG=0 GG=1
```

Wave

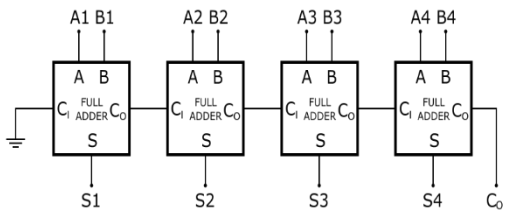
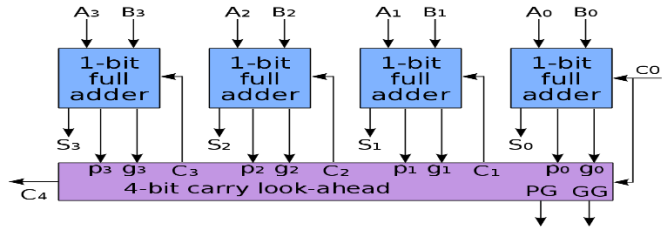


Report



Structure, delay and speed

	4-bit Full Adder	4-bit Carry Look Ahead Adder

Circuit		
Structure	+ Including n full adders connecting in series + Each full adder has to wait for its carry-in from its prev stage full adder + Thus, n th full adder has to wait until all (n – 1) full adders have completed their operations	+ The carry-in of any full adder is independent of the carry bits generated during intermediate stages + Known carry-in provided at the beginning and bits being added in the prev stages. Which enables the ability to evaluate the carry-in of any stages at the instant of time + No need for waiting the carry-in generated by its prev stage full adder
Delay	High delay as n increases	Low delay
Speed	Extremely slow as n increases	Faster than 4-bit Full adder

Prob 2

Prob 2:

1. **Statement assign**
   A continuous assignment is the most basic statement in dataflow modelling, used to drive a value onto a net. This assignment replaces gates in the description of the circuit and describes the circuit at a higher level of abstraction. The assignment statement starts with the keyword assign

```
continuous_assign ::= assign [ drive_strength ] [ delay3 ]
                      list_of_net_assignments ;
list_of_net_assignments ::= net_assignment { , net_assignment }
net_assignment ::= net_lvalue = expression
```

   Some further notations for the assignment:
   - The left hand side of an assignment must always be a scalar or vector net or a concatenation of scalar and vector nets. It cannot be a scalar or vector register

- Continuous assignments are always active. The assignment expression is evaluated as soon as one of the right-hand-side operands changes and the value is assigned to the left hand side net
- The operands on the right-hand side can be registers or nets or function calls. Registers or nets can be scalars or vectors.
- Delay values can be specified for assignments in terms of time units. Delay value are used to control the time when a net is assigned the evaluated value. This feature is similar to specifying delays for gates. It is very useful in modeling timing behavior in real circuits.

1.1 Implicit Continuous Assignment

Instead of declaring a net and then writing a continuous assignment on the net, Verilog provides a shortcut by which a continuous assignment can be placed on a net when it is declared. There can be only 1 implicit declaration assignment per net because a net is declared only once.

```
//Regular continuous assignment
wire out;
assign out = in1 & in2;

//Same effect is achieved by an implicit continuous assignment
wire out = in1 & in2;
```

1.2 Implicit Net Declaration

If a signal name is used to the left of the continuous assignment, an implicit net declaration will be inferred for that signal name. If the net is connected to a modul port the width of the inferred net is equal to the width of the module port

```
// Continuous assign. out is a net.
wire i1, i2;
assign out = i1 & i2; //Note that out was not declared as a wire
                      //but an implicit wire declaration for out
                      //is done by the simulator
```

2. Define, examples of expressions, operands, operator in Dataflow Modelling

2.1 Expressions

- Are constructs that combine operators and operands to produce a result

```
// Examples of expressions. Combines operands and operators
a ^ b
addr1[20:17] + addr2[20:17]
in1 | in2
```

2.2 Operands

There are varied data types. Some constructs will take only certain types of operands. Operands can be constants, integers, real numbers, nets, registers, times, bit select, part select, memories or function calls

```
integer count, final_count;
final_count = count + 1;//count is an integer operand

real a, b, c;
c = a - b; //a and b are real operands

reg [15:0] reg1, reg2;
reg [3:0] reg_out;
reg_out = reg1[3:0] ^ reg2[3:0];//reg1[3:0] and reg2[3:0] are
                                //part-select register operands

reg ret_value;
ret_value = calculate_parity(A, B);//calculate_parity is a
                                //function type operand
```

2.3 Operators

Act on the operands to produce desired results as various types of operators are provided in Verilog

```
d1 && d2 // && is an operator on operands d1 and d2
!a[0] // ! is an operator on operand a[0]
B >> 1 // >> is an operator on operands B and 1
```

3. Dataflow Modelling operators

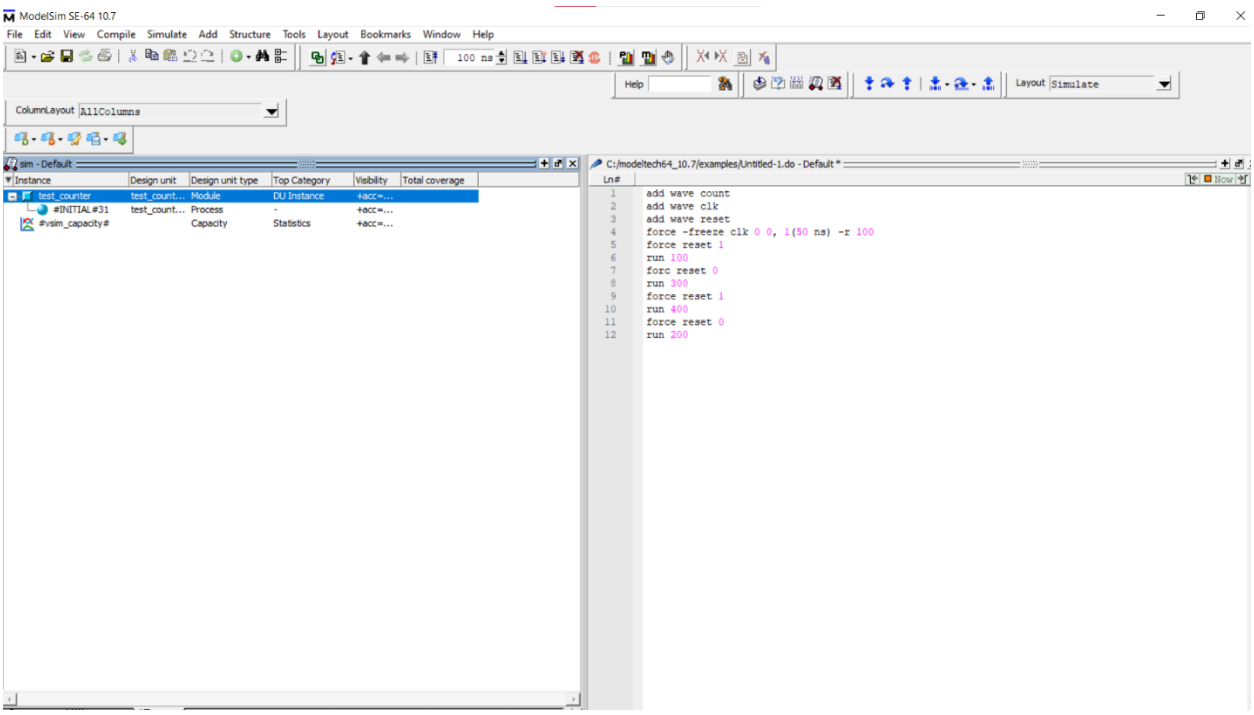
This provides many different operator types. Operators can be arithmetic, logical, relational, equality, bitwise, reduction, shift, concatenation or conditional.

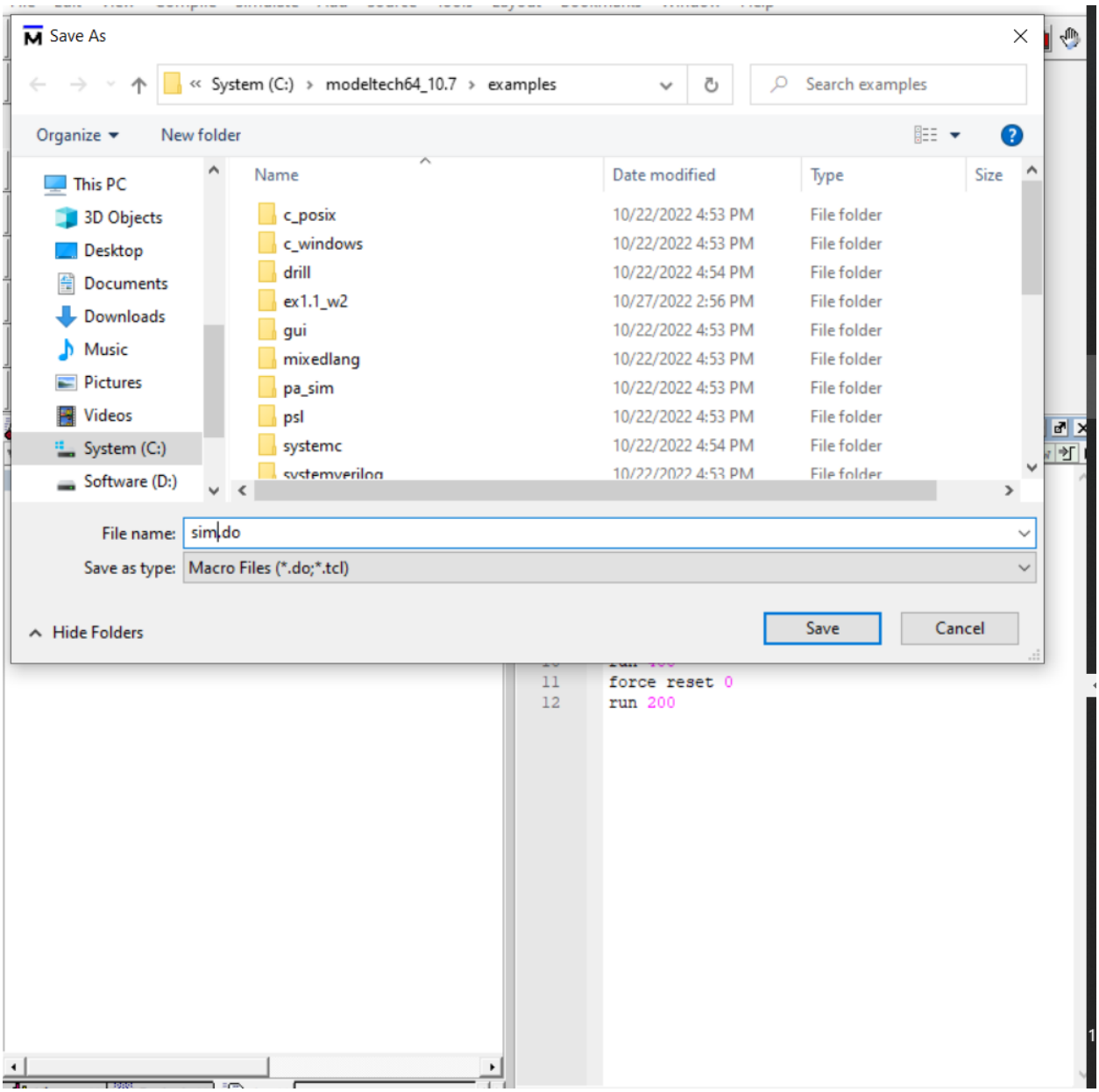
Table 6-1. Operator Types and Symbols

Operator Type	Operator Symbol	Operation Performed	Number of Operands
Arithmetic	*	multiply	two
	/	divide	two
	+	add	two
	-	subtract	two
	%	modulus	two
	**	power (exponent)	two
Logical	!	logical negation	one
	&&	logical and	two
	||	logical or	two
Relational	>	greater than	two
	<	less than	two
	>=	greater than or equal	two
	<=	less than or equal	two
Equality	==	equality	two
	!=	inequality	two
	===	case equality	two
	!==	case inequality	two

Bitwise	~	bitwise negation	one
	&	bitwise and	two
	|	bitwise or	two
	^	bitwise xor	two
	^~ or ~^	bitwise xnor	two
Reduction	&	reduction and	one
	~&	reduction nand	one
	|	reduction or	one
	~|	reduction nor	one
	^	reduction xor	one
	^~ or ~^	reduction xnor	one
Shift	>>	Right shift	Two
	<<	Left shift	Two
	>>>	Arithmetic right shift	Two
	<<<	Arithmetic left shift	Two
Concatenation	{ }	Concatenation	Any number
Replication	{ { } }	Replication	Any number
Conditional	?:	Conditional	Three

Prob 3


ModelSim SE-64 10.7
File Edit View Compile Simulate Add Structure Tools Layout Bookmarks Window Help
100 ns
Help
Layout Simulate
ColumnLayout AllColumns
sim - Default
Instance Design unit Design unit type Top Category Visibility Total coverage
test_counter test_count... Module DU Instance +acc=...
#INITIAL#31 test_count... Process - +acc=...
#vsim_capacity# Capacity Statistics +acc=...
C:/modeltech64_10.7/examples/Untitled-1.do - Default *
Ln#
1 add wave count
2 add wave clk
3 add wave reset
4 force -freeze clk 0 0, 1(50 ns) -r 100
5 force reset 1
6 run 100
7 forc reset 0
8 run 300
9 force reset 1
10 run 400
11 force reset 0
12 run 200


Save As
System (C:) › modeltech64_10.7 › examples
Search examples
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
System (C:)
Software (D:)
Name
Date modified
Type
Size
c_posix 10/22/2022 4:53 PM File folder
c_windows 10/22/2022 4:53 PM File folder
drill 10/22/2022 4:54 PM File folder
ex1.1_w2 10/27/2022 2:56 PM File folder
gui 10/22/2022 4:53 PM File folder
mixedlang 10/22/2022 4:53 PM File folder
pa_sim 10/22/2022 4:53 PM File folder
psl 10/22/2022 4:53 PM File folder
systemc 10/22/2022 4:54 PM File folder
File name: sim.do
Save as type: Macro Files (*.do;*.tcl)
Hide Folders
Save
Cancel
force reset 0
run 200

test_counter	test_count...	Module	DU Instance	+acc=...
#INITIAL#31	test_count...	Process	-	+acc=...
#vsim_capacity#		Capacity	Statistics	+acc=...