Embedded Systems

CS429 Lab1

Name: Dipean Dasgupta ID: 202151188

Q1.

1. Implement and analyze the circuit shown in Figure 1 at three levels of abstraction using Verilog.

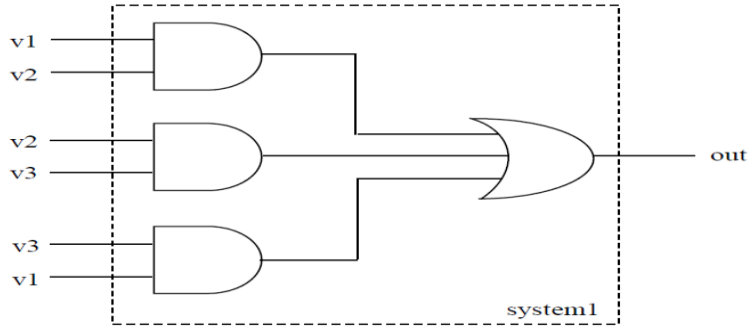


Figure 1

Structural Level:

```
module system1(v1,v2,v3,out);
input vi,v2,v3;
output out;
wire w1,w2,w3;

and g1(w1,v1,v2),
    g2(w2,v2,v3),
    g3(w3,v3,v1);
or  g4(out,w1,w2,w3);

endmodule
```

RTL Level:

Module code:

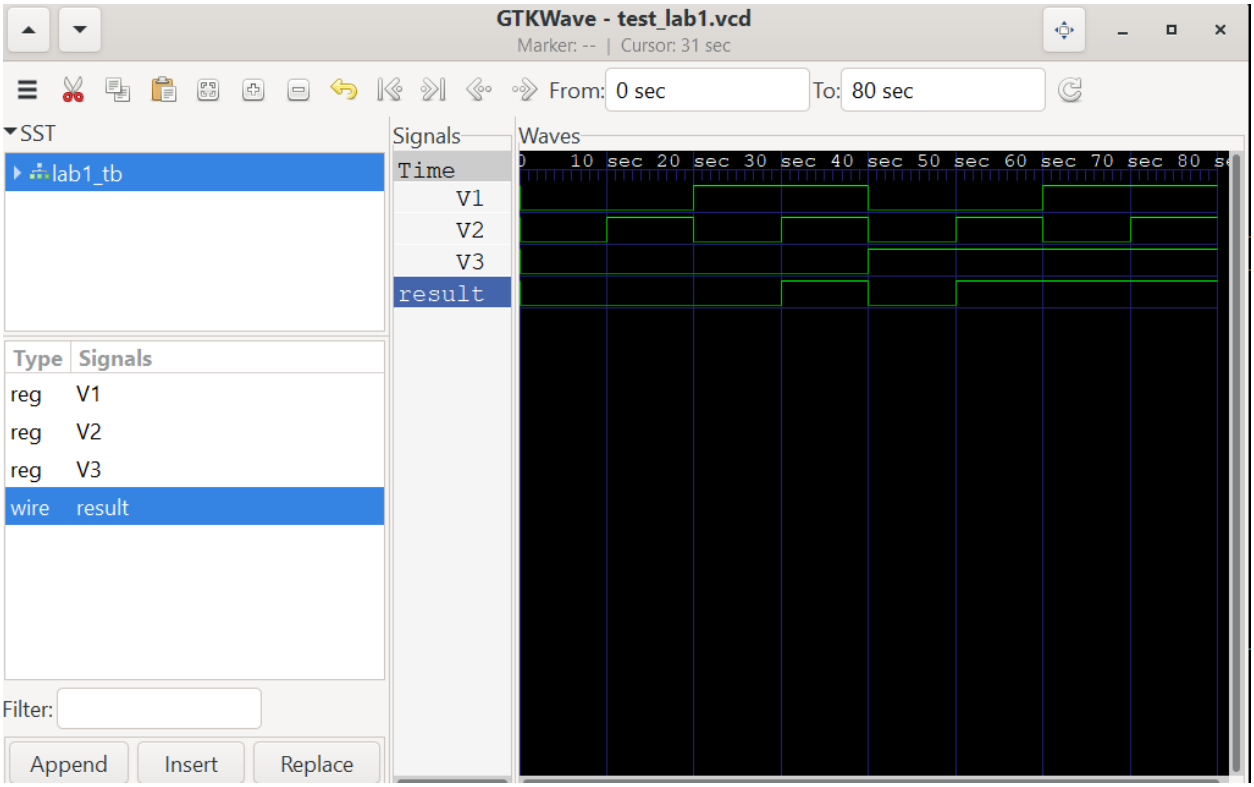
```
module lab1(
input v1,
input v2,
input v3,
output out
);
assign
out= (v1&v2)|(v2^v3)|(v1&v3);
endmodule
```

Testbench code:

```
`include "lab1.v"
module lab1_tb;
 reg V1;
 reg V2;
 reg V3;
 wire result;

 lab1 u0_DUT(
 .v1(V1),
 .v2(V2),
 .v3(V3),
 .out(result)
 );
 initial begin
 $dumpfile("test_lab1.vcd");
 $dumpvars(0,lab1_tb);
V1=0; V2=0; V3=0; #10;
V1=0; V2=1; V3=0; #10;
V1=1; V2=0; V3=0; #10;
V1=1; V2=1; V3=0; #10;
V1=0; V2=0; V3=1; #10;
V1=0; V2=1; V3=1; #10;
V1=1; V2=0; V3=1; #10;
V1=1; V2=1; V3=1; #10;
 end
 endmodule
```

Output:



Behavioral Level:

```
module system1(v1,v2,v3,out);
input v1,v2,v3;
output out;
wire w1,w2,w3;

if(v1&&v2)||(v2&&v3)||(v3&&v1)
out=1;
else
out=0;

endmodule
```

Q2:

2. Using system 1 of Task 1 as instance model, implement the model shown in Figure 2.

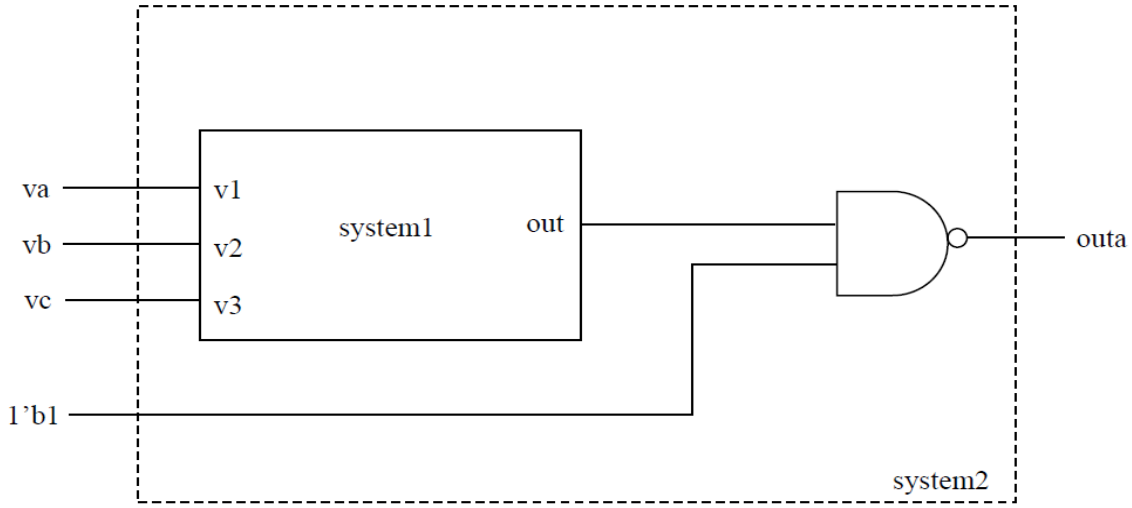


Figure 2

Structural Level:

```
module system1(v1,v2,v3,out);
input vi,v2,v3;
output out;
wire w1,w2,w3;
and g1(w1,v1,v2),
    g2(w2,v2,v3),
    g3(w3,v3,v1);
or  g4(out,w1,w2,w3);
endmodule
module system2(Va,Vb,Vc,outa);
input Va,Vb,Vc;
output outa;
```

wire=out;

system1 x1(Va,Vb,Vc,out);

nand g4(outa,out,1'b1);

endmodule

EXERCISE SOLUTIONS:

Q1. Implement half adder at three levels of abstraction.

Structural Level:

`timescale 1ns / 1ps

module system1(a,b,s,c);

input a,b;

output s,c;

xor x1(s,a,b),

and a1(c,a,b)

endmodule

RTL Level:

Module code:

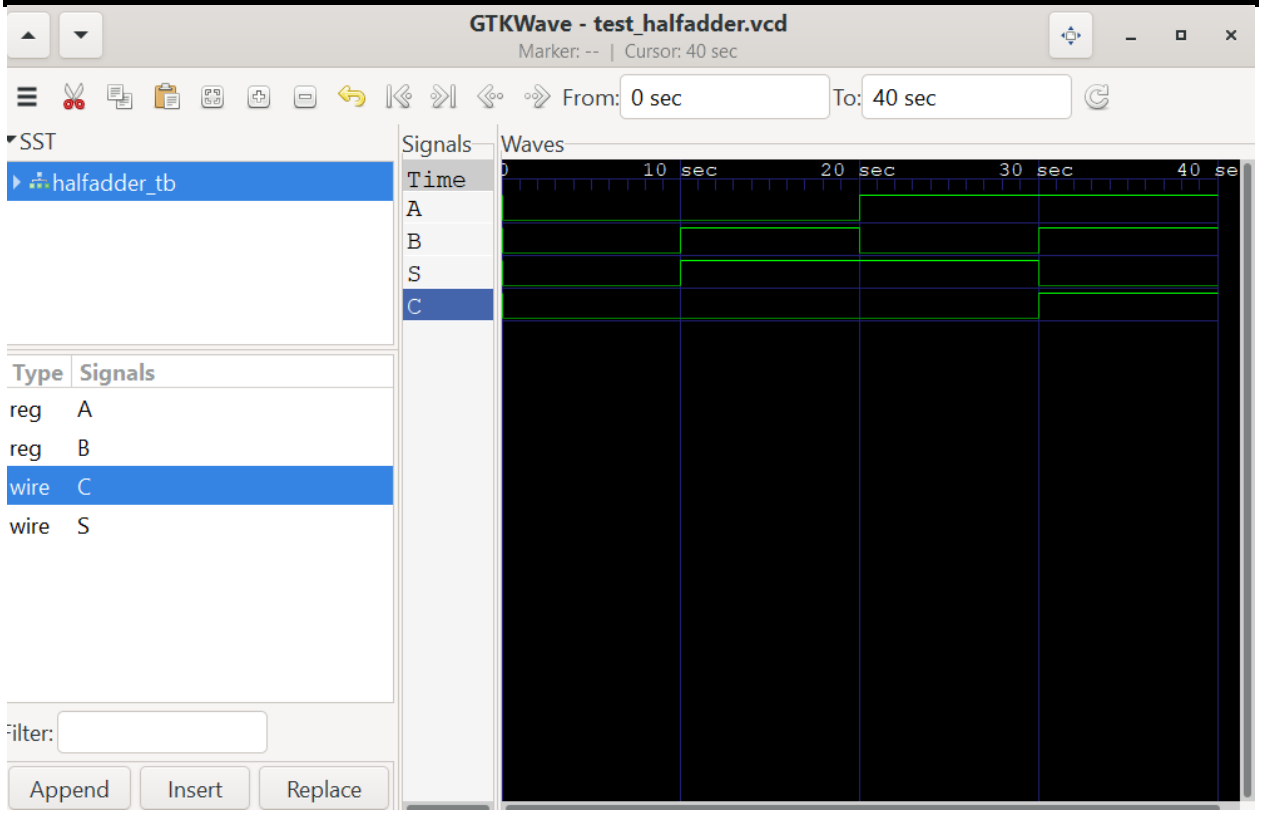
```
module halfadder(
    input a,
    input b,
    output s,
    output c
    );
    assign s = a ^ b;
    assign c = a & b;
endmodule
```

Testbench code:

```
`include "halfadder.v"
module halfadder_tb;
    reg A;
    reg B;
    wire S;
    wire C;
```

```
halfadder u0_DUT(
.a(A),
.b(B),
.s(S),
.c(C)
);
initial begin
$dumpfile("test_halfadder.vcd");
$dumpvars(0,halfadder_tb);
A=0; B=0; #10;
A=0; B=1; #10;
A=1; B=0; #10;
A=1; B=1; #10;

end
endmodule
```



Behavioural level:

```
module half_adder(a,b,sum,cout);
input a,b;
output reg sum,cout;


always @(*)
  begin
  case ({a,b})
  2'b00: sum = 0;
  2'b01: sum = 1;
  2'b10: sum = 1;
  2'b11: sum = 0;
  default : sum = 0;
  endcase

  case ({a,b})
  2'b00: cout = 0;
  2'b01: cout = 0;
  2'b10: cout = 0;
  2'b11: cout = 1;
  default : cout = 0;
  endcase
  end
endmodule
```

Q2. Implement full adder at three levels of abstraction.

Structural level:

```
module full_adder(
input a,b,c,
output sum,cout);
  wire w1, c1, c2, c3,out1;
  xor x1(w1, a, b);
  xor x2(sum, w1, c);

  and a1(c1, a, b);
  and a2(c2, b, c);
  and a3(c3, a, c);

  or o1(out1, c1, c2);
```

```
  or o2(cout, out1, c3);
endmodule
```

RTL Level:

Module Code:

```
module fulladder(
input a_i,
input b_i,
input c_i,
output sum_o,
output carry_o
);
assign
sum_o= a_i^b_i^c_i;
assign
carry_o=(a_i&b_i)|(b_i&c_i)|(c_i&a_i);

endmodule
```

Testbench Code:

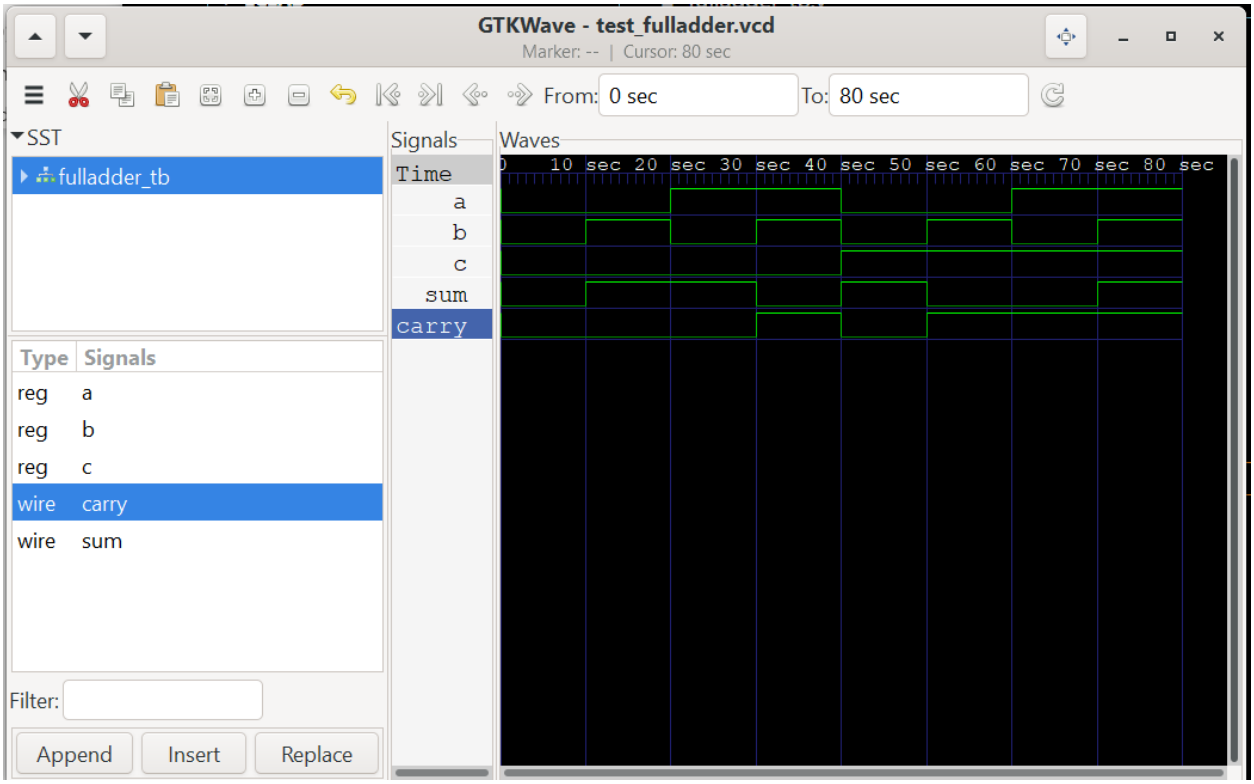
```
`include "fulladder.v"
module fulladder_tb;
 reg a;
 reg b;
 reg c;
 wire sum;
 wire carry;

 fulladder u0_DUT(
 .a_i(a),
 .b_i(b),
 .c_i(c),
 .sum_o(sum),
 .carry_o(carry)
 );
 initial begin
 $dumpfile("test_fulladder.vcd");
 $dumpvars(0,fulladder_tb);
a=0; b=0; c=0; #10;
a=0; b=1; c=0; #10;
a=1; b=0; c=0; #10;
a=1; b=1; c=0; #10;
a=0; b=0; c=1; #10;
a=0; b=1; c=1; #10;
```

```
a=1; b=0; c=1; #10;
a=1; b=1; c=1; #10;

 end
 endmodule
```

Output:



Behavioural Level:

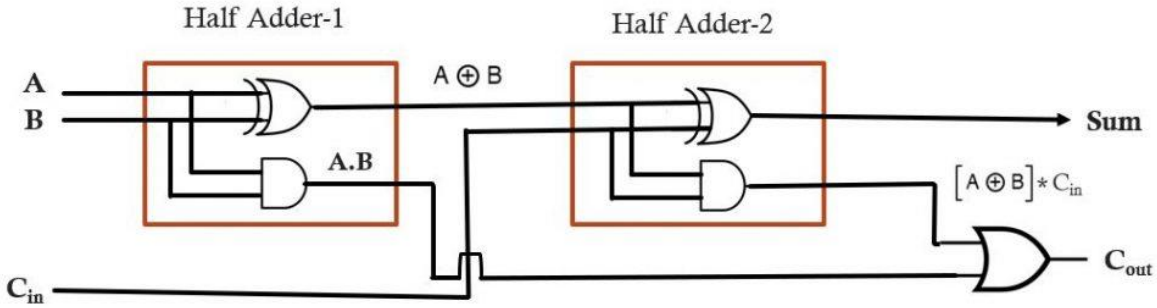
```
module full_adder(
input a,b,c,
output reg sum,cout);

  always @(*)
  begin
  case ({a,b,c})
  3'b000: sum = 0;
  3'b001: sum = 1;
  3'b010: sum = 1;
  3'b011: sum = 0;
  3'b100: sum = 1;
  3'b101: sum = 0;
  3'b110: sum = 0;
  3'b111: sum = 1;
```

```
  default : sum = 0;
  endcase

  case ({a,b,c})
  3'b000: cout = 0;
  3'b001: cout = 0;
  3'b010: cout = 0;
  3'b011: cout = 1;
  3'b100: cout = 0;
  3'b101: cout = 1;
  3'b110: cout = 1;
  3'b111: cout = 1;
  default : cout = 0;
  endcase
  end
endmodule
```

Q3. Implement full adder using half adder as instance model.



Module Code:

```
module full_adder(
    input a,b,cin,
    output sum,carry
);

wire c,c1,s;

half_adder ha0(a,b,s,c);
```

```
half_adder ha1(cin,s,sum,c1);

assign carry = c | c1 ;

endmodule
```

Testbench Code:

```
module full_adder_tb;
reg a,b,cin;
wire sum,carry;

full_adder uut(a,b,cin,sum,carry);

initial begin
a = 0; b = 0; cin = 0;
#10
a = 0; b = 0; cin = 1;
#10
a = 0; b = 1; cin = 0;
#10
a = 0; b = 1; cin = 1;
#10
a = 1; b = 0; cin = 0;
#10
a = 1; b = 0; cin = 1;
#10
a = 1; b = 1; cin = 0;
#10
a = 1; b = 1; cin = 1;
#10
$finish();
end

endmodule
```