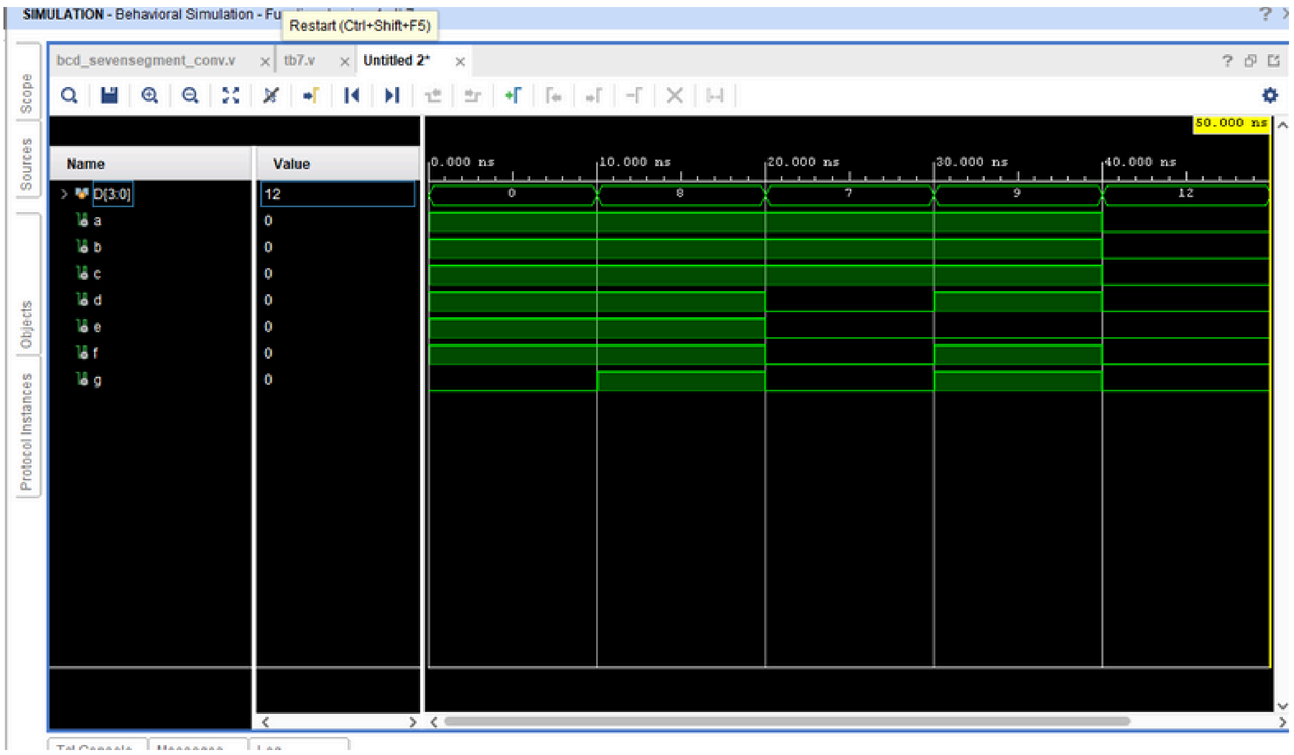
26.Binary-Seven Segment Display Converter

Verilog Code:

```
module bcd_sevensegment_conv(D,a,b,c,d,e,f,g );
input[3:0]D;
output reg a,b,c,d,e,f,g;
always @(*)

  begin
 case(D)
           4'b0000: {a,b,c,d,e,f,g} = 7'b1111110;
           4'b0001: {a,b,c,d,e,f,g} = 7'b0110000;
           4'b0010: {a,b,c,d,e,f,g} = 7'b1101101;
           4'b0011: {a,b,c,d,e,f,g} = 7'b1111001;
           4'b0100: {a,b,c,d,e,f,g} = 7'b0110011;
           4'b0101: {a,b,c,d,e,f,g} = 7'b1011011;
           4'b0110: {a,b,c,d,e,f,g} = 7'b1011111;
           4'b0111: {a,b,c,d,e,f,g} = 7'b1110000;
           4'b1000: {a,b,c,d,e,f,g} = 7'b1111111;
           4'b1001: {a,b,c,d,e,f,g} = 7'b1111011;
           default: {a,b,c,d,e,f,g} = 7'b0000000;
        endcase
    end
endmodule
```

TestBench:

```
module tb7();
reg[3:0]D;
wire a,b,c,d,e,f,g;
bcd_sevensegment_conv uut(D,a,b,c,d,e,f,g);
initial
begin
D=4'b0000; #10
D=4'b1000; #10
D=4'b0111; #10
D=4'b1001; #10
D=4'b1100; #10
$finish;
end
endmodule
```



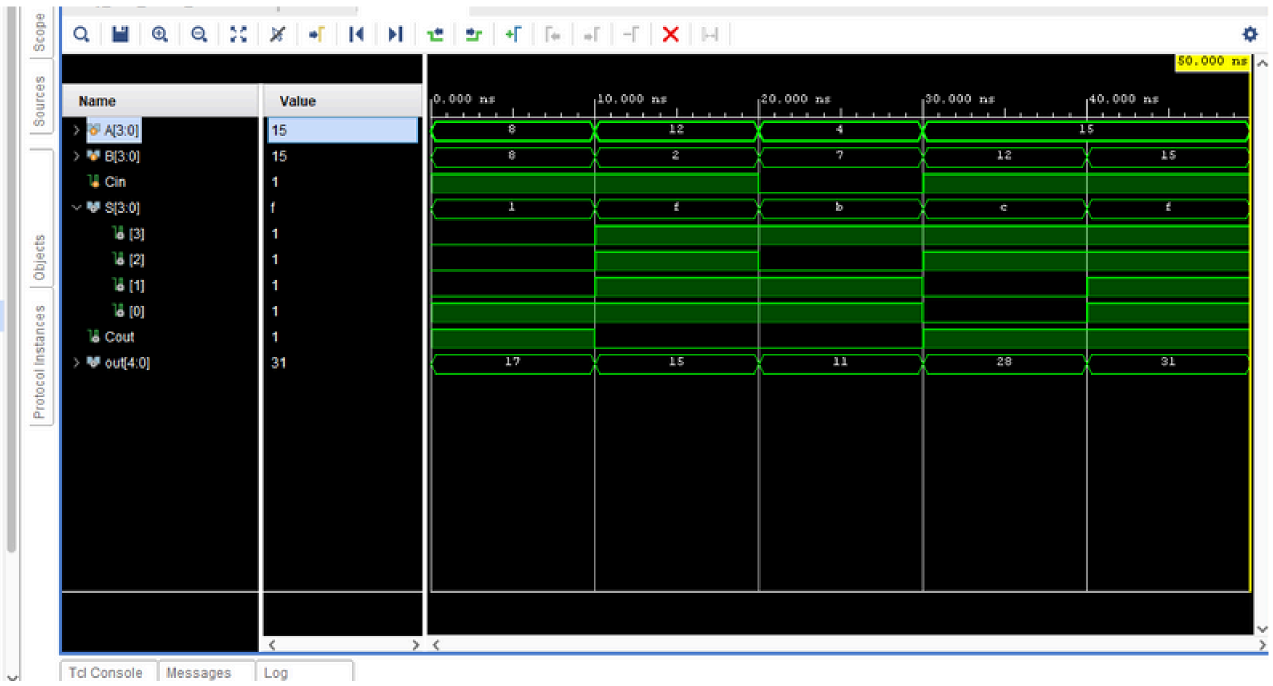
27.Carry Look Ahead Adder

Verilog Code:

```
module carry_look_ahead_adder(

   input [3:0] A, B,
   input Cin,
   output [3:0] S,
   output Cout,
   output [4:0]out);
   wire G[3:0], P[3:0];
   wire C[4:0];

 assign G[0] = A[0] & B[0];
 assign G[1] = A[1] & B[1];
 assign G[2] = A[2] & B[2];
 assign G[3] = A[3] & B[3];

   assign P[0] = A[0] | B[0];
   assign P[1] = A[1] | B[1];
   assign P[2] = A[2] | B[2];
   assign P[3] = A[3] | B[3];
   assign C[0] = Cin;

 assign C[1] = G[0] | (P[0] & Cin);
 assign C[2] = G[1] | (P[1] & (G[0] | (P[0] & Cin)));
 assign C[3] = G[2] | (P[2] & (G[1] | (P[1] & (G[0] | (P[0] & Cin)))));
assign Cout = G[3] | (P[3] & (G[2] | (P[2] & (G[1] | (P[1] & (G[0] | (P[0] & Cin)))))));

   assign S[0] = A[0]  ^B[0]  ^C[0];
   assign S[1] = A[1]  ^B[1]  ^C[1];
   assign S[2] = A[2]  ^B[2]  ^C[2];
   assign S[3] = A[3]  ^B[3]  ^C[3];
    assign out={Cout,S};

endmodule
```

TestBench:

```
module tb8( );
reg[3:0] A, B;
reg Cin;
wire[3:0] S;
wire Cout;
wire [4:0]out;
carry_look_ahead_adder
 uut(A,B,Cin,S,Cout,out);
initial
begin
A=4'b1000; B=4'b1000; Cin=1'b1; #10
A=4'b1100; B=4'b0010; Cin=1'b1; #10
A=4'b0100; B=4'b0111; Cin=1'b0; #10
A=4'b1111; B=4'b1100; Cin=1'b1; #10
A=4'b1111; B=4'b1111; Cin=1'b1; #10
$finish;
end
endmodule
```



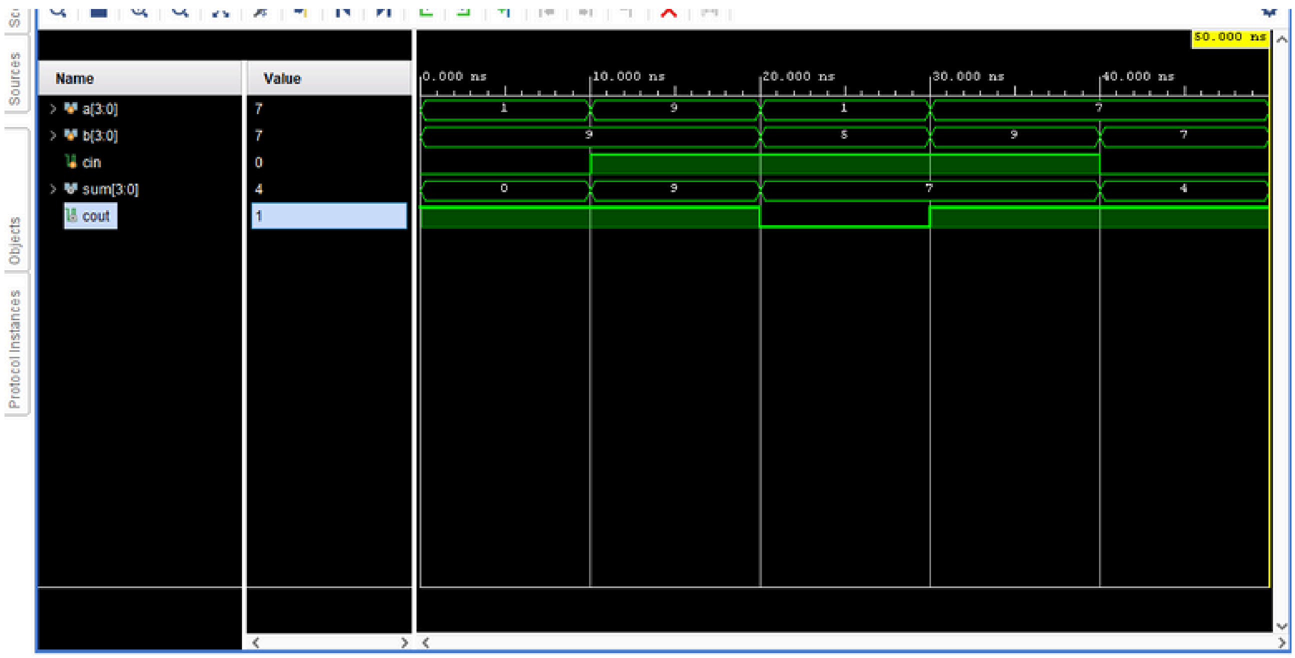
28.BCD Adder

Verilog Code:

```
module bcd_addition(a,b,cin,sum,cout );
input[3:0]a,b;
input cin;
output[3:0]sum;
output cout;
reg cout_temp;reg[4:0]sum_temp;
always @(*)
begin
sum_temp=a+b+cin;
if(sum_temp>9)
begin
sum_temp=sum_temp+6;
cout_temp=1;
end
else
sum_temp=sum_temp[3:0];
cout_temp=sum_temp[4];
end
assign sum=sum_temp;
assign cout=cout_temp;
endmodule
```

TestBench:

```
module tb9( );
reg [3:0]a,b;
reg cin;wire[3:0]sum;wire cout;
bcd_addition uut(a,b,cin,sum,cout);
initial
begin
a=4'b0001; b=4'b1001; cin=0; #10;
a=4'b1001; b=4'b1001; cin=1; #10;
a=4'b0001; b=4'b0101; cin=1; #10;
a=4'b0111; b=4'b1001; cin=1; #10;
a=4'b0111; b=4'b0111; cin=0; #10;
$finish;
end
endmodule
```



29.BCD-Excess_3 Converter

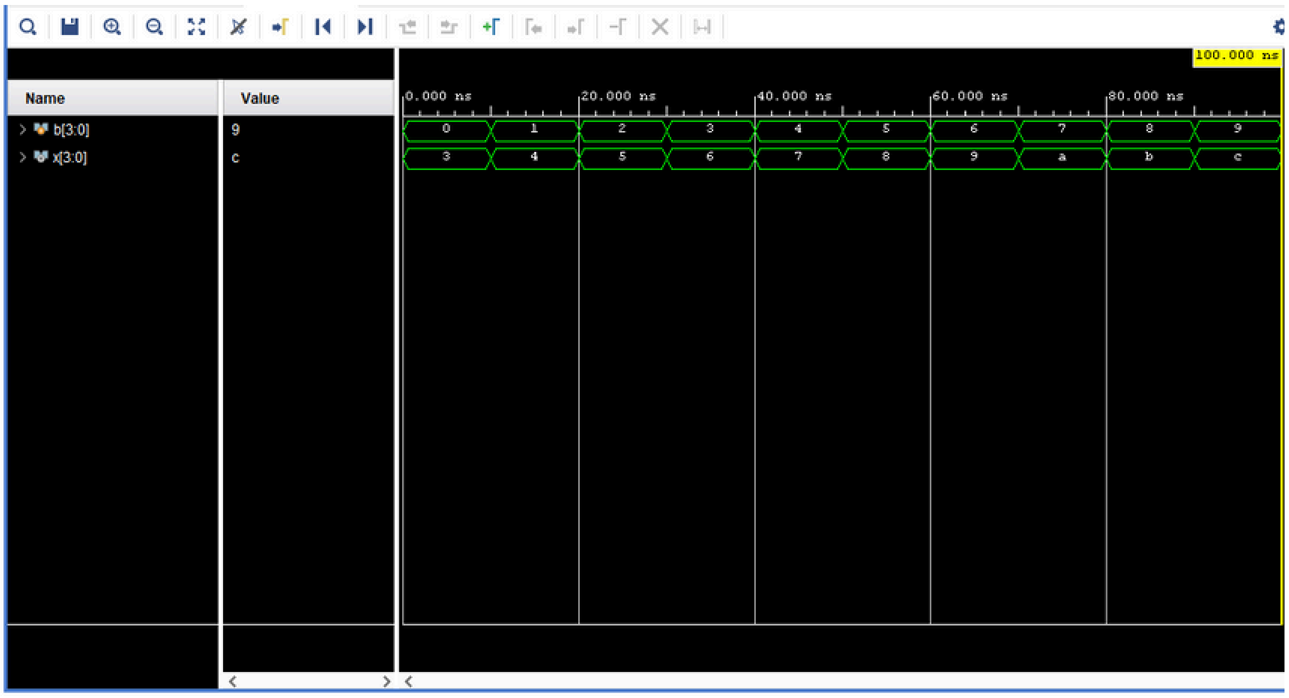
Verilog Code:

```
module bcd_x3_conv(b,x);
input[3:0]b;
output[3:0]x;
assign x[3]=b[3]|(b[2]&b[1])|(b[2]&b[0]);
assign x[2]=(~b[2]&b[0])|(~b[2]&b[1])|(b[2]&~b[1]&~b[0]);
assign x[1]=(b[1]&b[0])|(~b[1]&~b[0]);
assign x[0]=~b[0];
endmodule
```

TestBench:

```
module tb1( );
reg[3:0]b;
wire[3:0]x;
bcd_x3_conv uut(b,x);
initial
begin
b=4'b0000; #10
b=4'b0001; #10
b=4'b0010; #10
b=4'b0011; #10
b=4'b0100; #10
b=4'b0101; #10
b=4'b0110; #10
b=4'b0111; #10
b=4'b1000; #10
```

```
b=4'b1001; #10
$finish;
end
endmodule
```



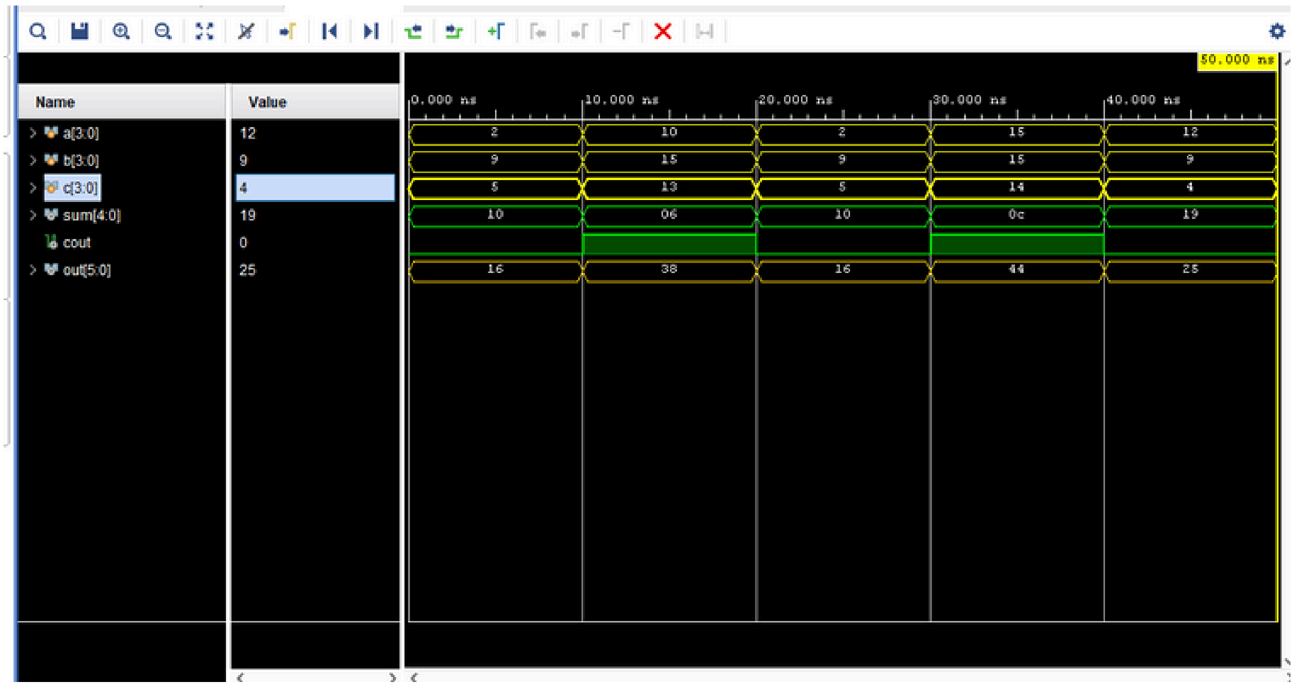
30.Carry Save Adder

Verilog Code:

```
module full_addr(a,b,c,sum,carry);
input a,b,c; output sum,carry;
assign sum=abc^^;
assign carry=(a&b)|(b&c)|(a&c);
 endmodule
modulecarry_save_adder(a,b,c,sum,cout,out);
input[3:0]a,b,c;
 output [4:0] sum;
output court;
output[5:0]out;
wire[3:0]sum_temp,cout_temp,co;
full_addr         fa1(a[0],b[0],c[0],sum_temp[0],cout_temp[0]);
full_addr         fa2(a[1],b[1],c[1],sum_temp[1],cout_temp[1]);
full_addr         fa3(a[2],b[2],c[2],sum_temp[2],cout_temp[2]);
full_addr         fa4(a[3],b[3],c[3],sum_temp[3],cout_temp[3]);
full_addr   fa5(sum_temp[1],cout_temp[0],1'b0,sum[1],co[0]);
full_addr fa6(sum_temp[2],cout_temp[1],co[0],sum[2],co[1]);
full_addr fa7(sum_temp[3],cout_temp[2],co[1],sum[3],co[2]);
full_addr fa8(1'b0,cout_temp[3],co[2],sum[4],cout);
assign sum[0]=sum_temp[0];
assign out={cout,sum};
endmodule
```

TestBench:

```
module tb2();
reg[3:0]a,b,c;
wire[4:0]sum; wire cout; wire[5:0]out;
carry_save_adder uut(a,b,c,sum,cout,out);
initial
begin
a=4'b0010; b=4'b1001; c=4'b0101; #10
a=4'b1010; b=4'b1111; c=4'b1101; #10
a=4'b0010; b=4'b1001; c=4'b0101; #10
a=4'b1111; b=4'b1111; c=4'b1110; #10
a=4'b1100; b=4'b1001; c=4'b0100; #10
$finish;
end
endmodule
```



Name	Value
a[3:0]	12
b[3:0]	9
c[3:0]	4
sum[4:0]	19
cout	0
out[5:0]	25