William Melanson
John Dokoupil
ECE 711
Lab 1

Design

The 8-bit BCD Adder was designed using two 4-bit BCD Adders. The 4-bit adder is a simple module that adds the inputs A and B along with the carry in bit. If the SUM equals a value greater than 9 then the value 6 is added to the sum to reset the 4 bits to 0 and to send a value of 1 to the carry out bit. The 8-bit adder is implemented using the two 4-bit bcd adders by passing in the lower 4 bits of inputs A and B to one 4-bit adder and the upper 4 bits of inputs A and B to the other 4-bit adder. The carry out bit of the first 4-bit adder is then connected to the carry in bit of the second 4-bit adder.

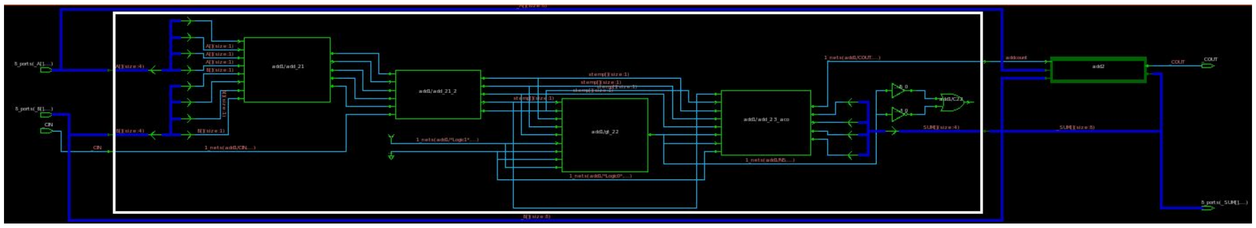


Figure 1. 4-bit BCD Adder

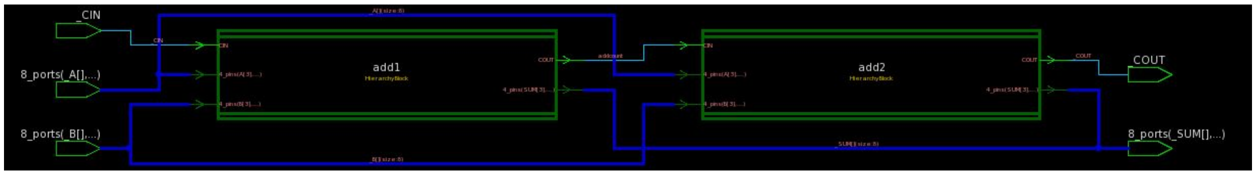


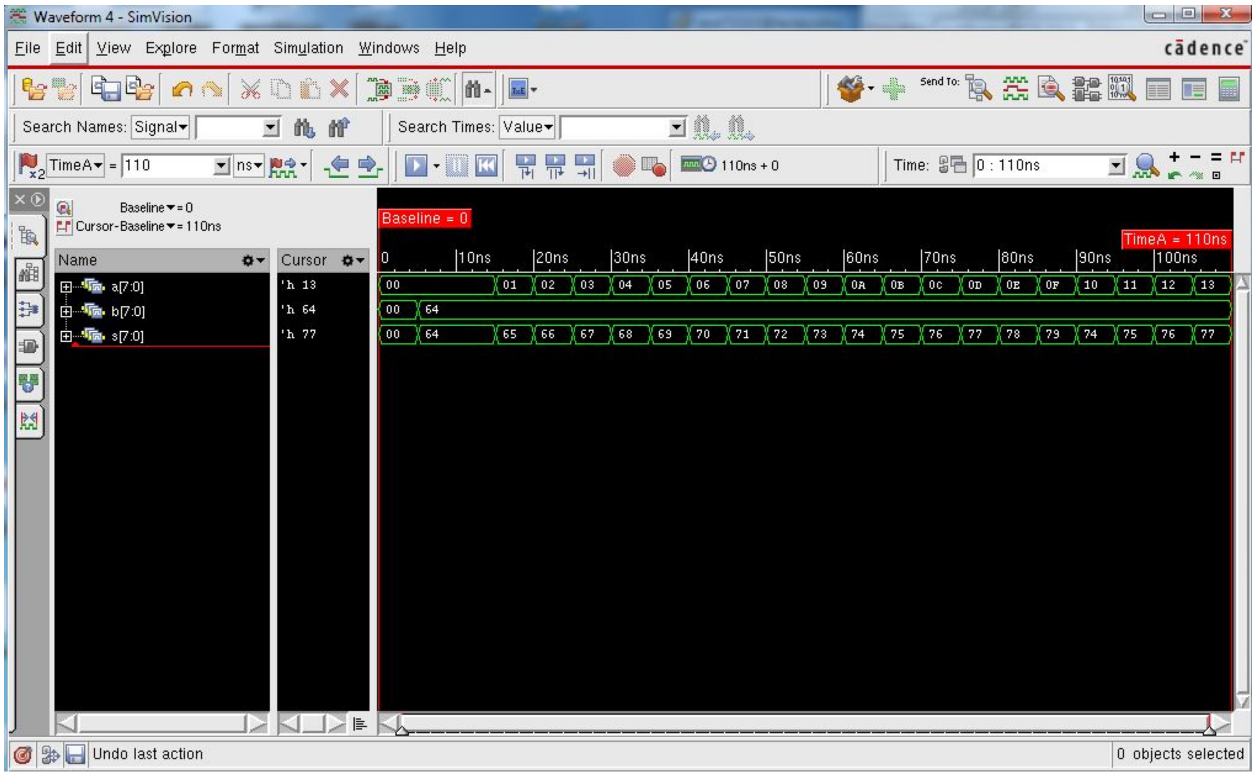
Figure 2. 8-bit BCD Adder

Design Justification

This design was chosen because BCD is represented in 4-bit chunks. Since BCD is represented in 4-bit chunks, it was best to create a 4-bit BCD adder module and then just pass in the COUT to the CIN of a second 4-bit BCD adder module. The adder design is correct due to the verification from the testbench. It is easy to check for correction when looking at the data in hexadecimal. We increment the A register every 5ns and when the value increments above 9, the value then increments the next hex bit instead of going to hex value A. i.e the SUM is currently 69, when A is incremented, instead of going to hex 6A, the value becomes hex 70.

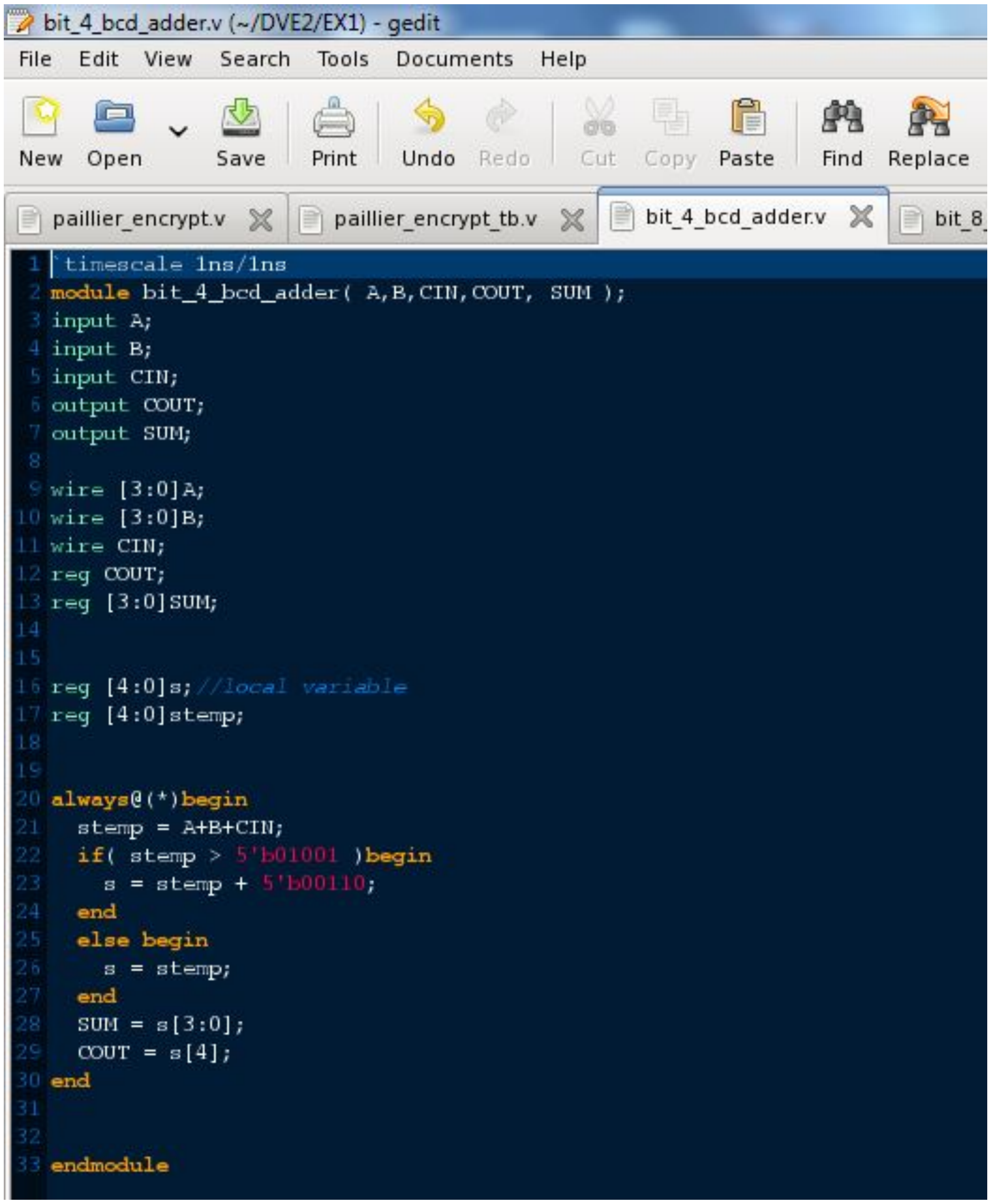
Testbench Justification

The test bench was written to test the functionality of the designed 8 bit BCD full adder. Our test bench increments the value retained in the four bit register A, while four bit register B holds a value of 64 (hexadecimal). This lab bench was chosen to as a simple, elegant test to prove that two, two-hexadecimal-digit values could be consistently, reliably, and precisely be added to provide a mathematically accurate result. Rather than incrementing both A and B, this simplistic testing design begins B at the large decimal value of 100, so that testing may be limited to the incrementing of A to prove the validity of this design. The design has been tested sufficiently enough to provide evidence of valid results.



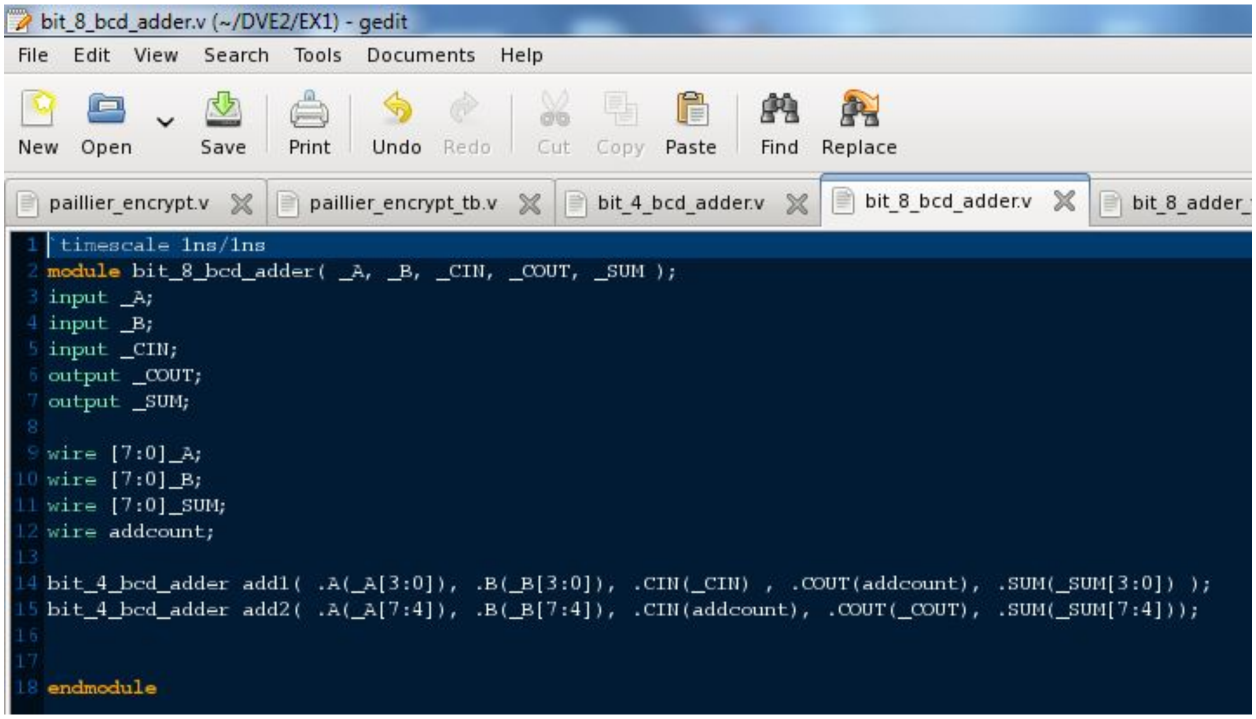
```
Name      Cursor   0        |10ns |20ns |30ns |40ns |50ns |60ns |70ns |80ns |90ns |100ns
a[7:0]    'h 13    00          01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F  10  11  12  13
b[7:0]    'h 64    00  64
s[7:0]    'h 77    00  64      65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  74  75  76  77
```

Figure 3. 8-bit BCD Adder Test Bench Simulation


bit_4_bcd_adder.v (~/DVE2/EX1) - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
paillier_encrypt.v
paillier_encrypt_tb.v
bit_4_bcd_adder.v
bit_8

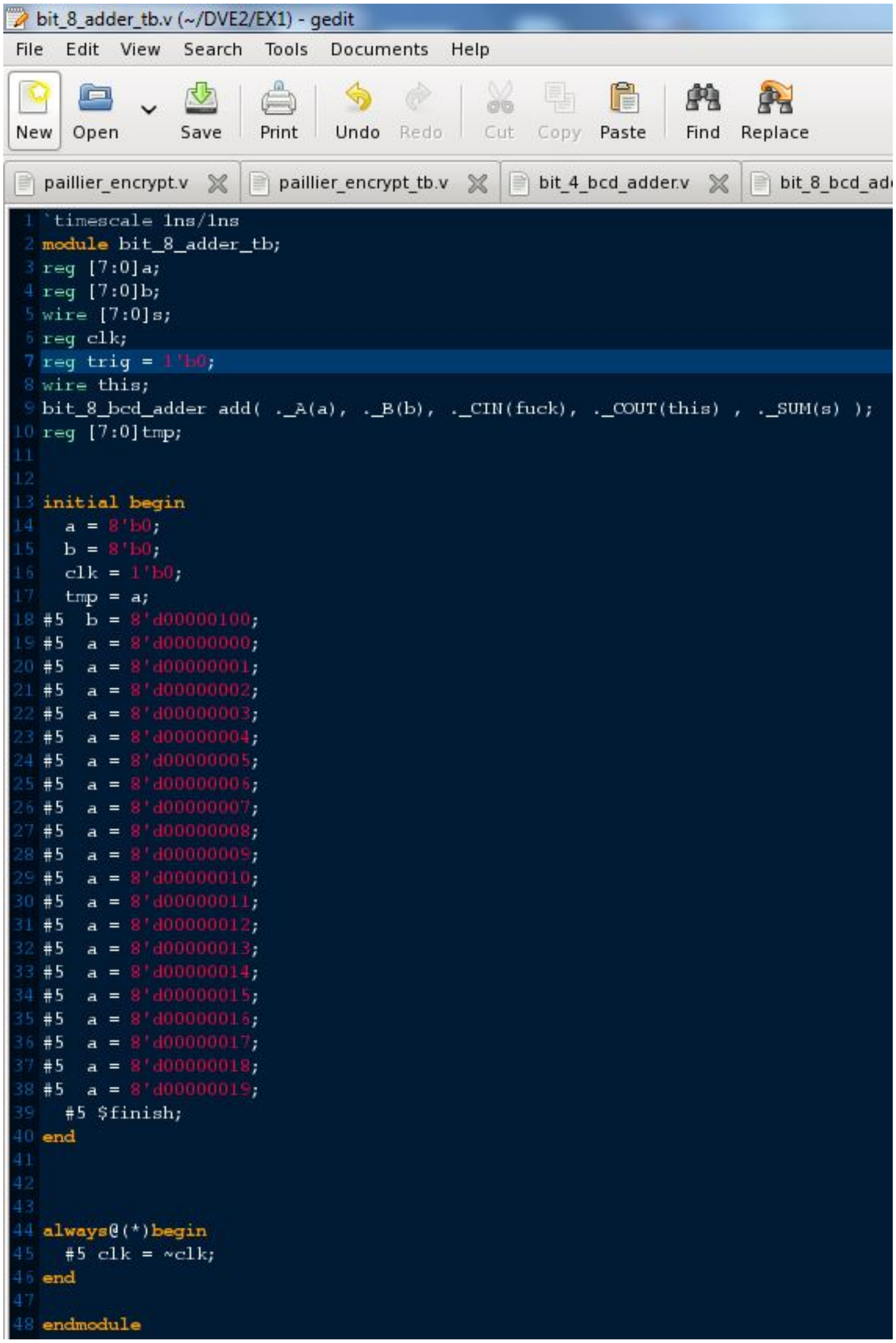
```verilog
`timescale 1ns/1ns
module bit_4_bcd_adder( A,B,CIN,COUT, SUM );
input A;
input B;
input CIN;
output COUT;
output SUM;

wire [3:0]A;
wire [3:0]B;
wire CIN;
reg COUT;
reg [3:0]SUM;


reg [4:0]s;//local variable
reg [4:0]stemp;


always@(*)begin
  stemp = A+B+CIN;
  if( stemp > 5'b01001 )begin
    s = stemp + 5'b00110;
  end
  else begin
    s = stemp;
  end
  SUM = s[3:0];
  COUT = s[4];
end


endmodule
```

Figure 4. 4-bit BCD Adder Verilog Source Code


bit_8_bcd_adder.v (~/DVE2/EX1) - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
paillier_encrypt.v
paillier_encrypt_tb.v
bit_4_bcd_adder.v
bit_8_bcd_adder.v
bit_8_adder_

```verilog
`timescale 1ns/1ns
module bit_8_bcd_adder( _A, _B, _CIN, _COUT, _SUM );
input _A;
input _B;
input _CIN;
output _COUT;
output _SUM;

wire [7:0]_A;
wire [7:0]_B;
wire [7:0]_SUM;
wire addcount;

bit_4_bcd_adder add1( .A(_A[3:0]), .B(_B[3:0]), .CIN(_CIN) , .COUT(addcount), .SUM(_SUM[3:0]) );
bit_4_bcd_adder add2( .A(_A[7:4]), .B(_B[7:4]), .CIN(addcount), .COUT(_COUT), .SUM(_SUM[7:4]));


endmodule
```

Figure 5. 8-bit BCD Adder Verilog Source Code



```
`timescale 1ns/1ns
module bit_8_adder_tb;
reg [7:0]a;
reg [7:0]b;
wire [7:0]s;
reg clk;
reg trig = 1'b0;
wire this;
bit_8_bcd_adder add( ._A(a), ._B(b), ._CIN(fuck), ._COUT(this) , ._SUM(s) );
reg [7:0]tmp;


initial begin
  a = 8'b0;
  b = 8'b0;
  clk = 1'b0;
  tmp = a;
#5  b = 8'd00000100;
#5  a = 8'd00000000;
#5  a = 8'd00000001;
#5  a = 8'd00000002;
#5  a = 8'd00000003;
#5  a = 8'd00000004;
#5  a = 8'd00000005;
#5  a = 8'd00000006;
#5  a = 8'd00000007;
#5  a = 8'd00000008;
#5  a = 8'd00000009;
#5  a = 8'd00000010;
#5  a = 8'd00000011;
#5  a = 8'd00000012;
#5  a = 8'd00000013;
#5  a = 8'd00000014;
#5  a = 8'd00000015;
#5  a = 8'd00000016;
#5  a = 8'd00000017;
#5  a = 8'd00000018;
#5  a = 8'd00000019;
  #5 $finish;
end



always@(*)begin
  #5 clk = ~clk;
end

endmodule
```

Figure 5. 8-bit BCD Adder Test Bench Source Code