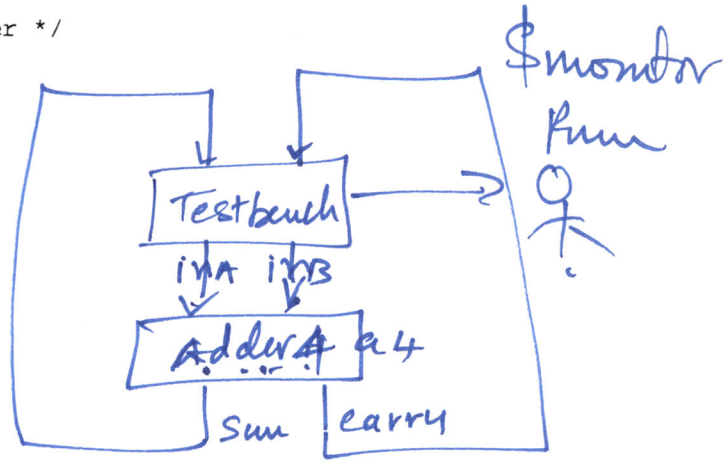
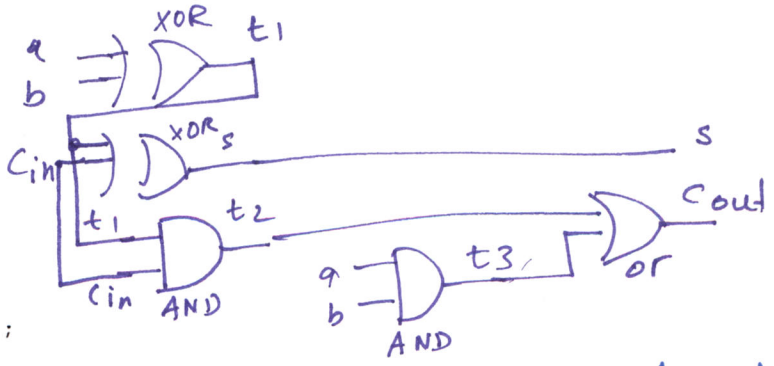
```
module ex2_1;
    /* gate level model to implement an adder */

    wire [3:0] sum, inA, inB;
    wire       carry;

    adder4 a4 (sum, carry, inA, inB);
    adderTest at (inA, inB, sum, carry);
endmodule
```



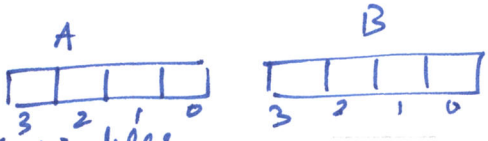

```
module adder4 (sum, carry, inA, inB);
    output [3:0] sum;
    output carry;
    input [3:0] inA, inB;

    adder1 a0 (sum[0], c0,    inA[0], inB[0], 1'b0);
    adder1 a1 (sum[1], c1,    inA[1], inB[1], c0);
    adder1 a2 (sum[2], c2,    inA[2], inB[2], c1);
    adder1 a3 (sum[3], carry, inA[3], inB[3], c2);
endmodule
```

1-bit full adder description

```
module adder1 (s, cout, a, b, cin);
    output s, cout;
    input a, b, cin;

    xor (t1, a, b);
    xor (s, t1, cin);
    and (t2, t1, cin),
        (t3, a, b);
    or  (cout, t2, t3);
endmodule
```




```
module adderTest (A, B, sum, carry);
    output [3:0] A, B;
    input [3:0] sum;
    input carry;
    reg [3:0] A, B;

    integer i, j; // loop variables
    initial begin
        $monitor ("A: %d  B: %d sum: %d  carry: %d", A, B, sum, carry);
        $dumpfile("adder_wv.vcd");
        $dumpvars(1, A,B, sum, carry);

        for (i=0; i<16; i=i+1) for (j=0; j<16; j=j+1)
        begin A=i; B=j; #10000 ;end
        $finish;
    end
endmodule
```



$ directive

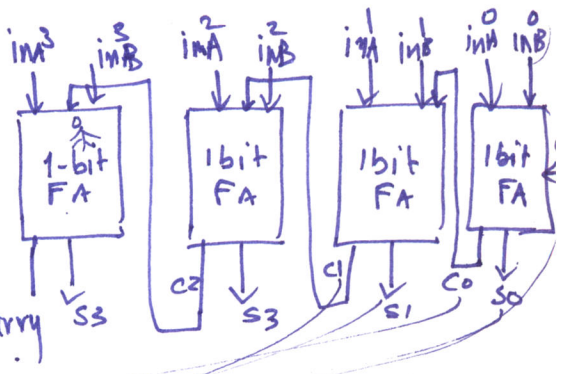
print format string

$dumpvars;

Based on C lang

adder4 adderTester

Ripple carry adder




```
module ex2_1;
    /* gate level model to implement an adder */

    wire [3:0] sum, inA, inB;
    wire       carry;

    adder4 a4 (sum, carry, inA, inB);
    adderTest at (inA, inB, sum, carry);
endmodule
```

output input

```
module adder4 (sum, carry, inA, inB);
    output [3:0] sum;
    output carry;
    input [3:0] inA, inB;

   .adder1 a0 (sum[0], c0,    inA[0], inB[0], 1'b0);
   . adder1 a1 (sum[1], c1,   inA[1], inB[1], c0);
   . adder1 a2 (sum[2], c2,   inA[2], inB[2], c1);
   . adder1 a3 (sum[3], carry, inA[3], inB[3], c2);
endmodule
```

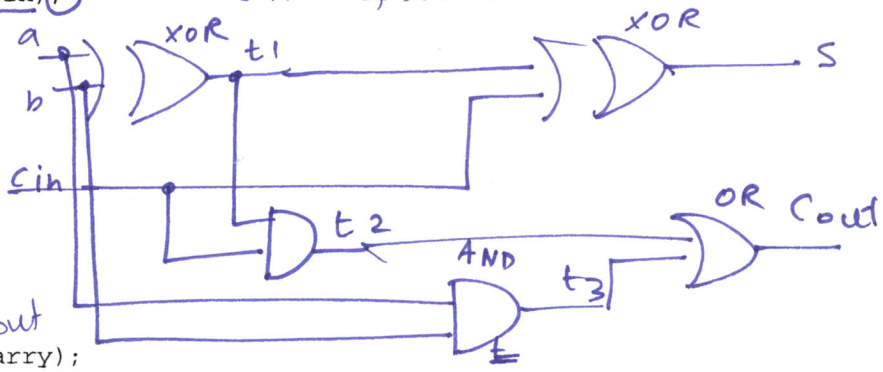
"pattern"

1 bit full-adder description

outputs inputs

```
module adder1 (s, cout, a, b, cin);
    output s, cout;
    input a, b, cin;

    xor (t1, a, b);
    xor (s, t1, cin);
    and (t2, t1, cin),
        (t3, a, b);
    or  (cout, t2, t3);
endmodule
```

"call up" structural verilog"



output input

```
module adderTest (A, B, sum, carry);
    output [3:0] A, B;
    input [3:0] sum;
    input carry;
    reg [3:0] A, B;

    integer i, j;
    initial begin
        $monitor ("A: %d  B: %d sum: %d  carry: %d", A, B, sum, carry);
        $dumpfile("adder_wv.vcd");
        $dumpvars(1, A,B, sum, carry);

        for (i=0; i<16; i=i+1) for (j=0; j<16; j=j+1)
          begin A=i; B=j; #10000 ;end
        $finish;
    end
endmodule
```

// place holder for the test patterns

// loop variables

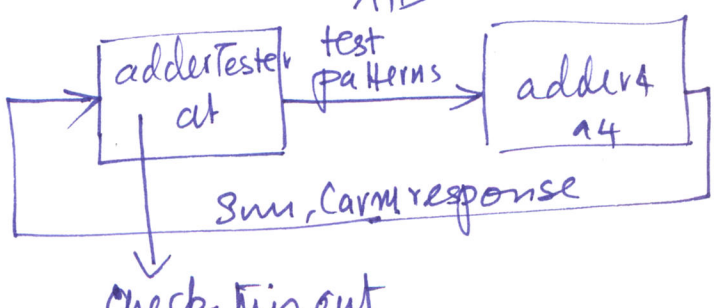
$ directive

directive

directive

simulation

A,B



%d decimal format

Check this out

```
module ex2_1;
    /* gate level model to implement an adder */

    wire [3:0] sum, inA, inB;
    wire       carry;

    adder4 a4 (sum, carry, inA, inB);
    adderTest at (inA, inB, sum, carry);
endmodule


module adder4 (sum, carry, inA, inB);
    output [3:0] sum;
    output carry;
    input [3:0] inA, inB;

    adder1 a0 (sum[0], c0,    inA[0], inB[0], 1'b0);
    adder1 a1 (sum[1], c1,    inA[1], inB[1], c0);
    adder1 a2 (sum[2], c2,    inA[2], inB[2], c1);
    adder1 a3 (sum[3], carry, inA[3], inB[3], c2);
endmodule


module adder1 (s, cout, a, b, cin);
    output s, cout;
    input a, b, cin;

    xor (t1, a, b);
    xor (s, t1, cin);
    and (t2, t1, cin),
        (t3, a, b);
    or  (cout, t2, t3);
endmodule


module adderTest (A, B, sum, carry);
    output [3:0] A, B;
    input [3:0] sum;
    input carry;
    reg [3:0] A, B;

    integer i, j;
    initial begin
       $monitor ("A: %d  B: %d sum: %d  carry: %d", A, B, sum, carry);
       $dumpfile("adder_wv.vcd");
       $dumpvars(1, A,B, sum, carry);

       for (i=0; i<16; i=i+1) for (j=0; j<16; j=j+1)
        begin A=i; B=j; #10000 ;end
       $finish;
    end
endmodule
```