VERILOG PROJECT

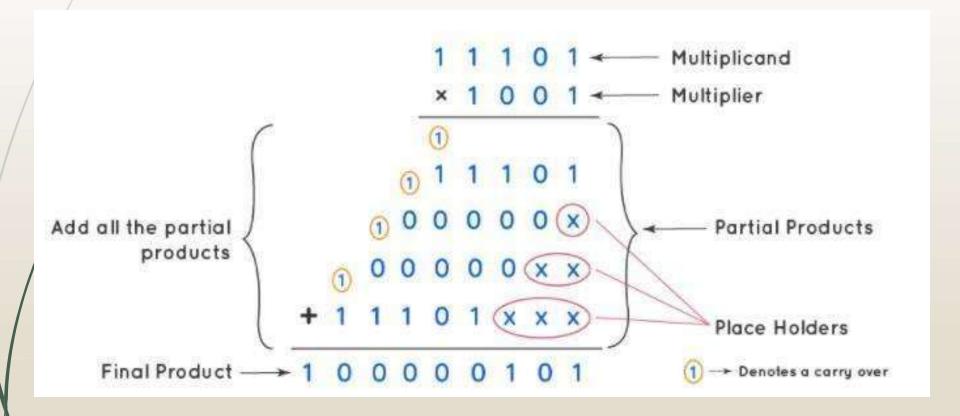
N-bit Multiplier with Binary to BCD Converter

INTRODUCTION

- ✓ This project is designed as an n-bit multiplier of two numbers. The
 multiplication is performed using the shift and add method of multiplying
 two numbers.
- ✓ Once the multiplication is completed, the output result is also converted to its binary-coded decimal (BCD) representation.
- ✓ This is done using the **double dabble method** of converting binary numbers to binary-coded decimal(BCD).
- ✓ In the output, both the binary and the binary-coded decimal (BCD) representation is shown.

SHIFT AND ADD ALGORITHM

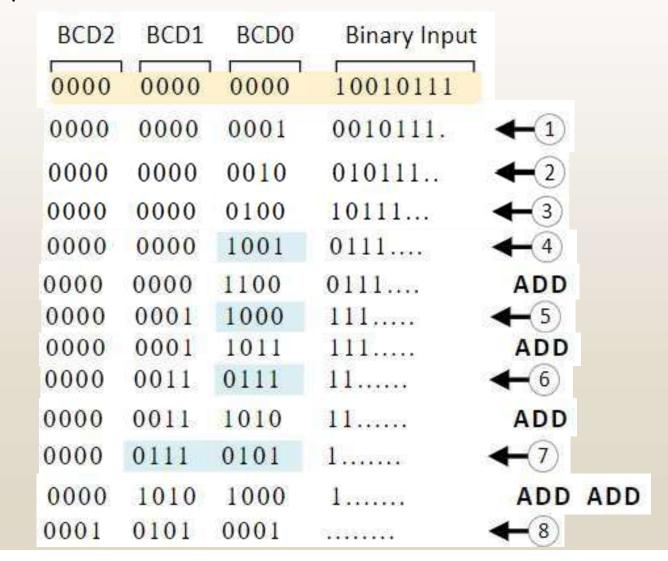
Consider two numbers **29 and 9**Their binary representation is **11101 and 1001**

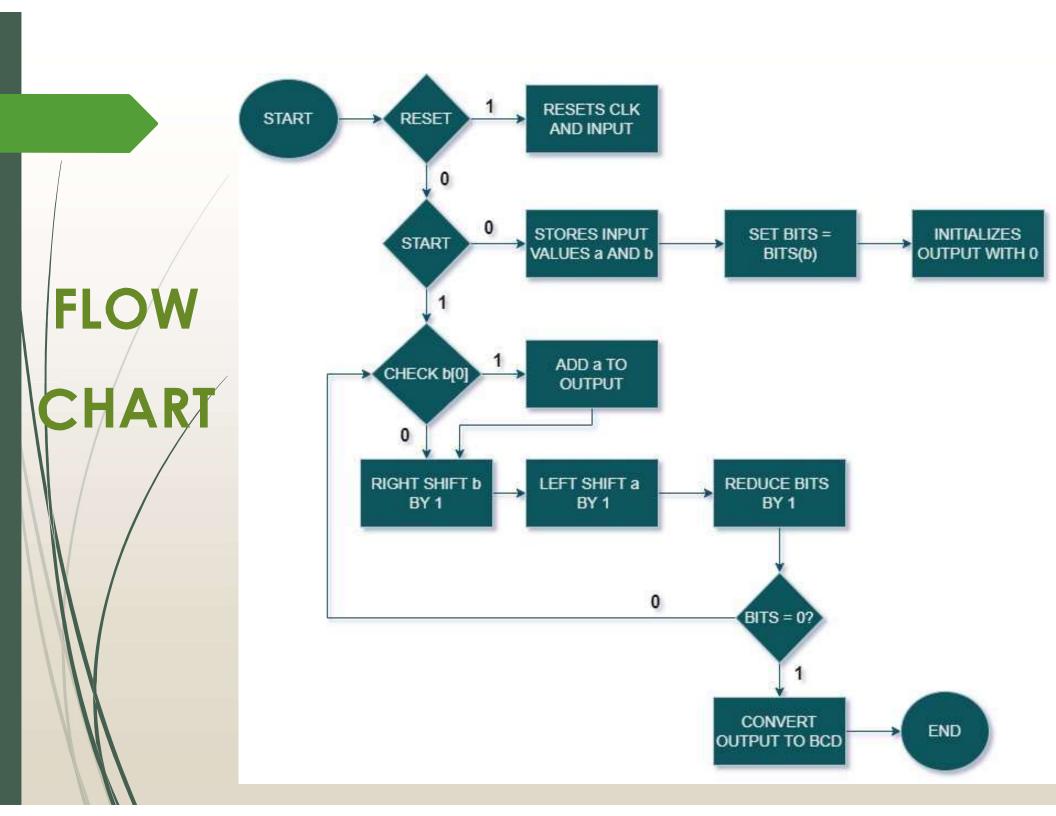


The result is **261**Their binary representation is **100000101**

DOUBLE DABBLE ALGORITHM

Consider the number **151**. Its binary representation is **10010111**





VERILOG CODE

```
module multiplier (
      out, a in, b in, clk, start, reset, finish, bcd
   );
parameter N = 8;
// Outputs
output[(((N*2)/3)+1)*4-1:0] bcd;
                                       Declaring Outputs
output[(N*2)-1:0] out;
output finish;
// Inputs
input start;
input clk;
                                       Declaring Inputs
input reset;
input [N-1:0] a in;
input [N-1:0] b in;
// Reference registers
reg[(((N*2)/3)+1)*4-1:0] bcd reg = 0;
reg[(N*2)-1:0] out reg;
                                                   Creating Reference Registers
```

reg finish reg = 0;

reg [8:0] bits;

reg [(N*2)-1:0] a in reg; reg [(N*2)-1:0] b in reg;

VERILOG CODE ...continued

```
// Continuous assignment
assign bcd = bcd reg;
assign out = out reg;
                                            Continuous Assignments
assign finish = finish reg;
integer i;
// Reset clk and inputs
always @(negedge reset)
begin
                                                Reset clk and inputs
   out reg = 0;
  a in reg = 0;
   b in reg = 0;
end
always @ (posedge clk)
begin
   if (!reset)
   begin
      case (start)
         1'b0: begin
            a in reg = a in;
            b in reg = b in;
                                                                        Load Input
            bits = N:
                                                                          Values
            finish reg = 0;
            out reg = 0;
            bcd reg = 0;
             $display("Values loaded into the input register!");
         end
```

VERILOG CODE ...continued

```
l'bl: begin
            if(b in reg[0] == 1)
                                                               Shift and Add
               out reg = out reg + a in reg;
                                                               Multiplication
           bits = bits - 1;
            a in reg = a in reg<<l;
           b in reg = b in reg>>1;
      endcase
     if(bits==0)
                                                    Multiplication Complete
         $display("Multiplication completed!");
        finish reg = 1'bl;
         // conversion of binary to bod
        for (i=0; i<(N*2); i=i+1)
        begin
           if (3 \le (((N*2)/3)+1)*4-1 & bcd reg[3:0] >= 5) bcd reg[3:0] = bcd reg[3:0] + 3;
           if (7 \le (((N^2)/3)+1)*4-1 & bcd reg[7:4] >= 5) bcd reg[7:4] = bcd reg[7:4] + 3;
           if (11 \le (((N*2)/3)+1)*4-1 & bcd reg[11:8] >= 5) bcd reg[11:8] = bcd reg[11:8] + 3;
           if (15 \le (((N*2)/3)+1)*4-1 && bcd reg[15:12] >= 5) bcd reg[15:12] = bcd reg[15:12] + 3;
           bcd reg = {bcd reg[(((N*2)/3)+1)*4-2:0],out reg[(N*2)-1-i]};
         end
        $display("Conversion of binary to BCD completed!");
     end
                                                                                 Double Dabble
  end
                                                                                   Algorithm
endmodule
```

TEST BENCH

```
module multiplier tb;
   parameter n bits = 5;
   // Inputs
  reg [n bits-1:0] a in;
  reg [n bits-1:0] b in;
   red clk;
   reg start;
                                                Declaring Inputs
   reg reset;
                                                  and Outputs
   // Outputs
   wire [(n bits*2)-1:0] out;
  wire finish;
   wire [(((n bits*2)/3)+1)*4-1:0] bcd;
   // Instantiate the Unit Under Test (UUT)
   multiplier uut (
      .out (out) ,
     .a in(a in),
     .b in(b in),
```

Initializing the multiplier

```
initial
begin
   #50 clk= ~clk;
end
```

.clk(clk),

.bcd(bcd)

.start(start),

.reset (reset),

.finish (finish),

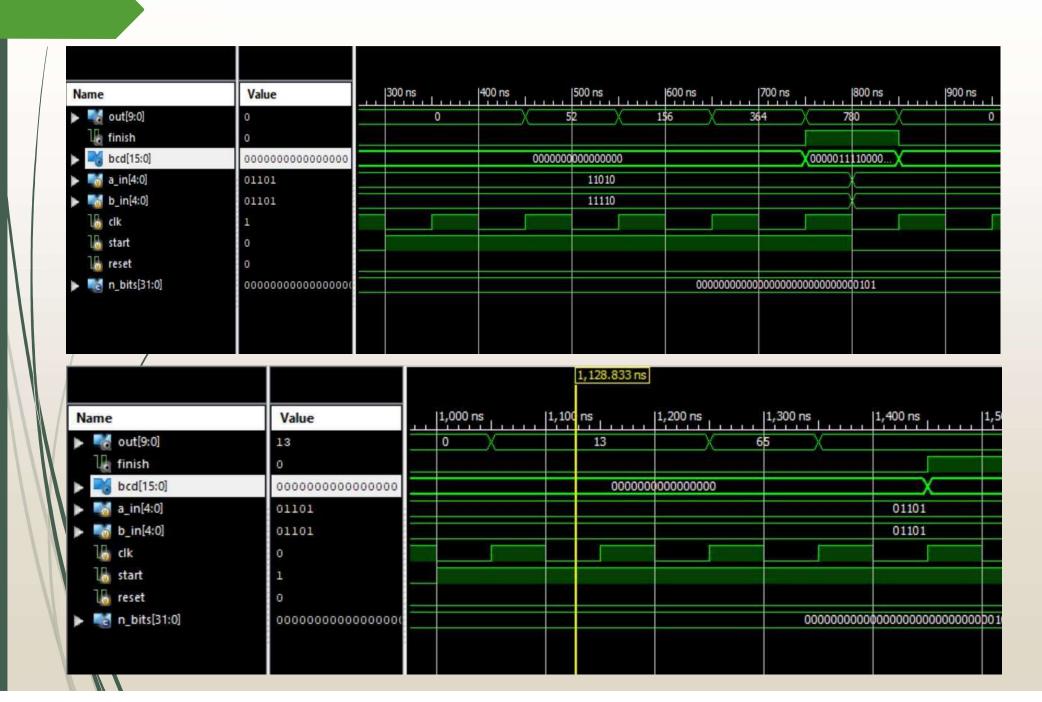
defparam uut.N = n bits;

Creating clock

```
initial begin
   // Initialize Inputs
   a in = 0;
  b in = 0;
  clk = 0:
  start = 0;
  reset = 0:
  // Wait 100 ns for global reset to finish
  #100;
   // Add stimulus here
   a in = 'd26;
  b in = 'd30;
   start = 0:
   #200
   start = 1;
   #1000
  a in = 'd13;
  b in = 'd13;
   start = 0;
  #200
   start = 1;
end
```

Test Inputs

SIMULATION OUTPUT



OBSERVATIONS

- The out represent the output of the multiplication of two numbers.
- A value of 1 in *finish* signal indicates that the multiplication is complete while a value of 0 indicates that the multiplication is ongoing.
- The *clk* and *reset* are input signals used to set the clock and reset the circuit to initial state respectively.
- The **bcd** represents the binary-coded decimal representation of the output.
- •/ The *a_in* represents the input multiplicand.
- The b_in represents the input multiplier.
- A value of 0 in *start* loads the initial value into the registers, while a value of 1 performs the multiplication.

THANK YOU