Selected Methodology:

My methodology is little different from the one described in manual. It is because the FPGA is parallel in nature whereas micro-controller execute instruction line by line. So, in our methodology, FPGA is independently running in its parallel nature. All of the control is in hand of micro-controller as shown in given figure no.1.

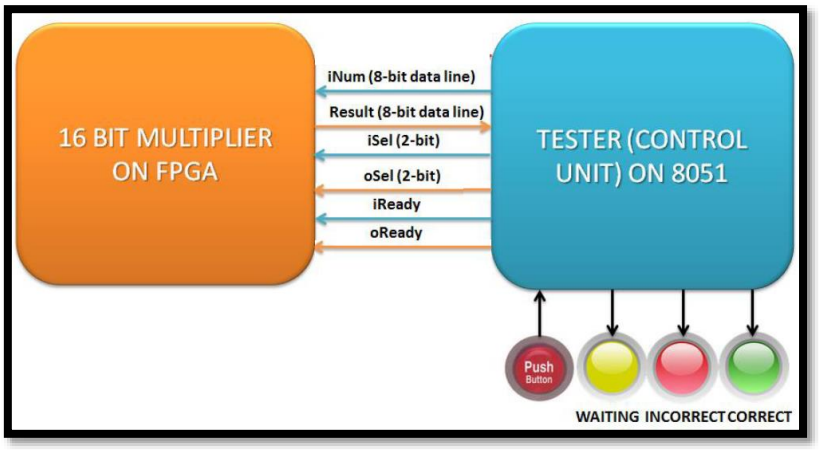

Figure no. 1

Verilog Code:

Mul16.v

```
`timescale 1ns / 1ps

module mul16( clk,
              bus_in,
              i_sel,
              i_enable,
              bus_out,
              o_sel,
              o_enable
              );
    parameter     AH= 2'b00, AL=2'b01, BH= 2'b10, BL=2'b11;
    parameter     C1= 2'b00, C2=2'b01, C3= 2'b10, C4=2'b11;

    input                clk;
    input                i_enable;
    input        [1:0]   i_sel;
    input        [7:0]   bus_in;
    input                o_enable;
    input        [1:0]   o_sel;
```

```
output reg      [7:0]    bus_out = 8'd0;

reg             [15:0]   a,b;
reg             [31:0]   c;

always @(posedge clk)begin
if(o_enable)begin
        case (i_sel)
        AH:begin
        a[15:8] <= bus_in;
        a[7:0]  <= a[7:0];
        b[15:8] <= b[15:8];
        b[7:0]  <= b[7:0];
        end
        AL:begin
        a[15:8] <= a[15:8];
        a[7:0]  <= bus_in;
        b[15:8] <= b[15:8];
        b[7:0]  <= b[7:0];
        end
        BH:begin
        a[15:8] <= a[15:8];
        a[7:0]  <= a[7:0];
        b[15:8] <= bus_in;
        b[7:0]  <= b[7:0];
        end
        BL:begin
        a[15:8] <= a[15:8];
        a[7:0]  <= a[7:0];
        b[15:8] <= b[15:8];
        b[7:0]  <= bus_in;
        end
        default:begin
        a[15:8] <= a[15:8];
        a[7:0]  <= a[7:0];
        b[15:8] <= b[15:8];
        b[7:0]  <= b[7:0];
        end
        endcase
end
else begin
        a[15:8] <= a[15:8];
        a[7:0]  <= a[7:0];
        b[15:8] <= b[15:8];
        b[7:0]  <= b[7:0];
end
end
```

```
	always @(a,b)begin
		c= a*b;
	end

	always @(posedge clk)begin
	if (i_enable)
		case (o_sel)
		C1:	bus_out <= c[31:24];
		C2:	bus_out <= c[23:16];
		C3:	bus_out <= c[15:8];
		C4:	bus_out <= c[7:0];
		default: bus_out <= 8'd0;
		endcase
	end
endmodule
```

mul16.ucf

```
# PlanAhead Generated physical constraints

NET "bus_in[0]" LOC = H33;
NET "bus_in[1]" LOC = F34;
NET "bus_in[2]" LOC = H34;
NET "bus_in[3]" LOC = G33;
NET "bus_in[4]" LOC = G32;
NET "bus_in[5]" LOC = H32;
NET "bus_in[6]" LOC = J32;
NET "bus_in[7]" LOC = J34;

NET "bus_out[0]" LOC = L33;
NET "bus_out[1]" LOC = M32;
NET "bus_out[2]" LOC = P34;
NET "bus_out[3]" LOC = N34;
NET "bus_out[4]" LOC = AA34;
NET "bus_out[5]" LOC = AD32;
NET "bus_out[6]" LOC = Y34;
NET "bus_out[7]" LOC = Y32;

NET "clk" LOC = AH15;
NET "i_enable" LOC = AH32;
NET "i_sel[0]" LOC = W32;
NET "i_sel[1]" LOC = AH34;
NET "o_enable" LOC = AK34;
NET "o_sel[0]" LOC = AE32;
NET "o_sel[1]" LOC = AG32;
```

TEST BENCH for the designed system:

Test_bench.v

```
`timescale 1ns / 1ps

module test_bench;

	// Inputs
	reg clk;
	reg [7:0] bus_in;
	reg [1:0] i_sel;
	reg i_enable;
	reg [1:0] o_sel;
	reg o_enable;

	// Outputs
	wire [7:0] bus_out;

	// Instantiate the Unit Under Test (UUT)
	mul16 uut (
		.clk(clk),
		.bus_in(bus_in),
		.i_sel(i_sel),
		.i_enable(i_enable),
		.bus_out(bus_out),
		.o_sel(o_sel),
		.o_enable(o_enable)
	);

	reg	[15:0]	a= 16'd12;
	reg	[15:0]	b= 16'd24;
	reg	[31:0]	c = 32'd288;
	reg	[31:0]	out_result = 32'd0;


	initial begin
		// Initialize Inputs
		clk = 0;  bus_in = 0; i_sel = 0; i_enable = 0; o_sel = 0; o_enable = 0;		#6;
		o_enable = 1;	#6;
		i_sel = 2'b00;	bus_in = a[15:8];		#6;
		i_sel = 2'b01;	bus_in = a[7:0];		#6;
		i_sel = 2'b10;	bus_in = b[15:8];		#6;
		i_sel = 2'b11;	bus_in = b[7:0];		#6;
		o_enable =	0;				#200;
		i_enable = 1;					#6;
		o_sel = 2'b00;					#6;
		out_result[31:24] = bus_out;			#6;
```

```
                o_sel = 2'b01;                  #6;
                out_result[23:16] = bus_out;    #6;
                o_sel = 2'b10;                  #6;
                out_result[15:8] = bus_out;     #6;
                o_sel = 2'b11;                  #6;
                out_result[7:0] = bus_out;      #6;
                i_enable = 0;                   #6

                // Wait 100 ns for global reset to finish
                #100;
        end
  always #1 clk=~clk;
endmodule
```

89C51 micro-controller code:

```
#include<reg51.h> //Header file of generic 80C51
sbit A1=P1^0;
sbit A2=P1^1;
sbit A3=P1^2;
sbit A4=P1^3;
sbit A5=P1^4;
sbit A6=P1^5;
sbit A7=P1^6;
sbit A8=P1^7;

sbit O1=P2^0;
sbit O2=P2^1;
sbit O3=P2^2;
sbit O4=P2^3;
sbit O5=P2^4;
sbit O6=P2^5;
sbit O7=P2^6;
sbit O8=P2^7;

sbit incorrect=P3^0;
sbit correct=P3^1;
sbit i_enable=P3^2;
sbit o_enable=P3^3;
sbit sel_i1=P3^4;
sbit sel_i2=P3^5;
sbit sel_o1=P3^6;
sbit sel_o2=P3^7;

sbit waitting=P0^0;
sbit clk_01=P0^1;

void Delay_msec(unsigned int); //Delay Function declaration using onboard timer
```

```
void main() // Start of program
{
unsigned char A1, B1, C1, D1;
TMOD=0x10; //Setting mode: timer,software based, 16 bit register
Delay_msec(100); //calling delay function
while(1) // infinite loop
{
        o_enable=1;
        sel_i1=0;
        sel_i2=0;
        P1=0x00;
        clk_01=1;
        clk_01=0;
        Delay_msec(100); //calling delay function
        sel_i1=1;
        sel_i2=0;
        P1=0x03;
        clk_01=1;
        clk_01=0;
        Delay_msec(100); //calling delay function
        sel_i1=1;
        sel_i2=1;
        P1=0x04;
        clk_01=1;
        clk_01=0;
        Delay_msec(100); //calling delay function
        sel_i1=0;
        sel_i2=1;
        P1=0x00;
        clk_01=1;
        clk_01=0;
        Delay_msec(100); //calling delay function
        o_enable=0;

        Delay_msec(1000); //calling delay function

        i_enable=1;
        sel_o1=0;
        sel_o2=0;
        A1=P2;
        Delay_msec(100); //calling delay function
        sel_o1=1;
        sel_o2=0;
        B1=P2;
        Delay_msec(100); //calling delay function
        sel_o1=1;
        sel_o2=1;
        C1=P2;
```

```
        Delay_msec(100); //calling delay function
        sel_o1=0;
        sel_o2=1;
        D1=P2;
        Delay_msec(100); //calling delay function
        o_enable=0;
        waitting=0;
        if (A1==0x00 & B1==0x00 & C1==0x0C & D1==0x00){
                correct=1;
                incorrect=0;
        }
        else{
                correct=0;
                incorrect=1;
        }
        Delay_msec(1000); //calling delay function
        waitting = 1;
}
}

void Delay_msec(unsigned int num_msec) //Delay function with input in milliseconds
{
int i;
TH1=0x00;
TL1=0x00; //Clear the timer 1 register
TR1=0x40; //Start timer
/*
Timer register is 16 bit
Clock speed =occilator speed/12
Occilator Speed is 11.5 MHz
Timer register overflow in 71 milliseconds
*/
for(i=0; i<num_msec/71;i++){ //Set delay
TF1=0; //Clear overflow flag
while(TF1==0){ //while overflow flag is clear
}
}
}
```

Verification of Circuit on ModelSim:

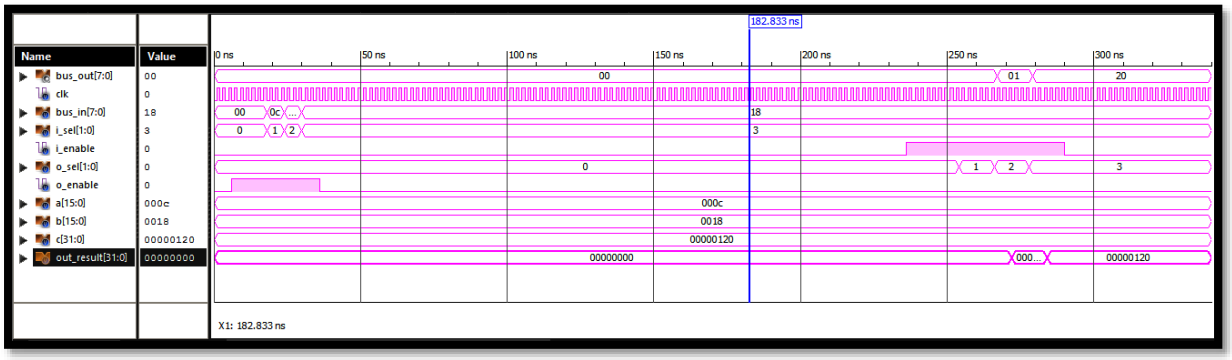

Figure no. 2 Verification of Circuit on ModelSim

RTL Circuit diagram:

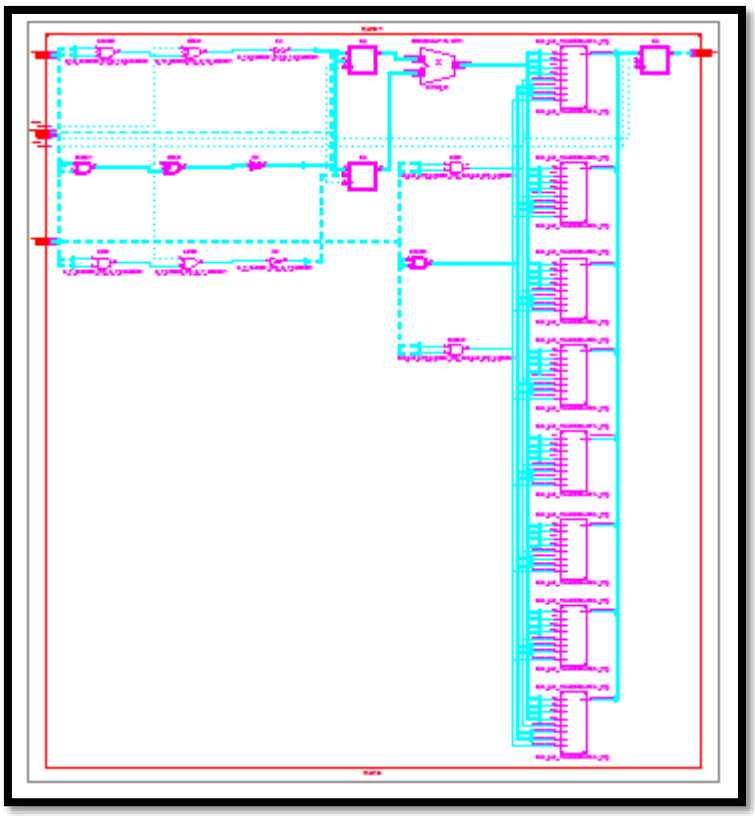

Figure no. 3 RTL Circuit Diagram

Proteus simulation of 89c51 Code:

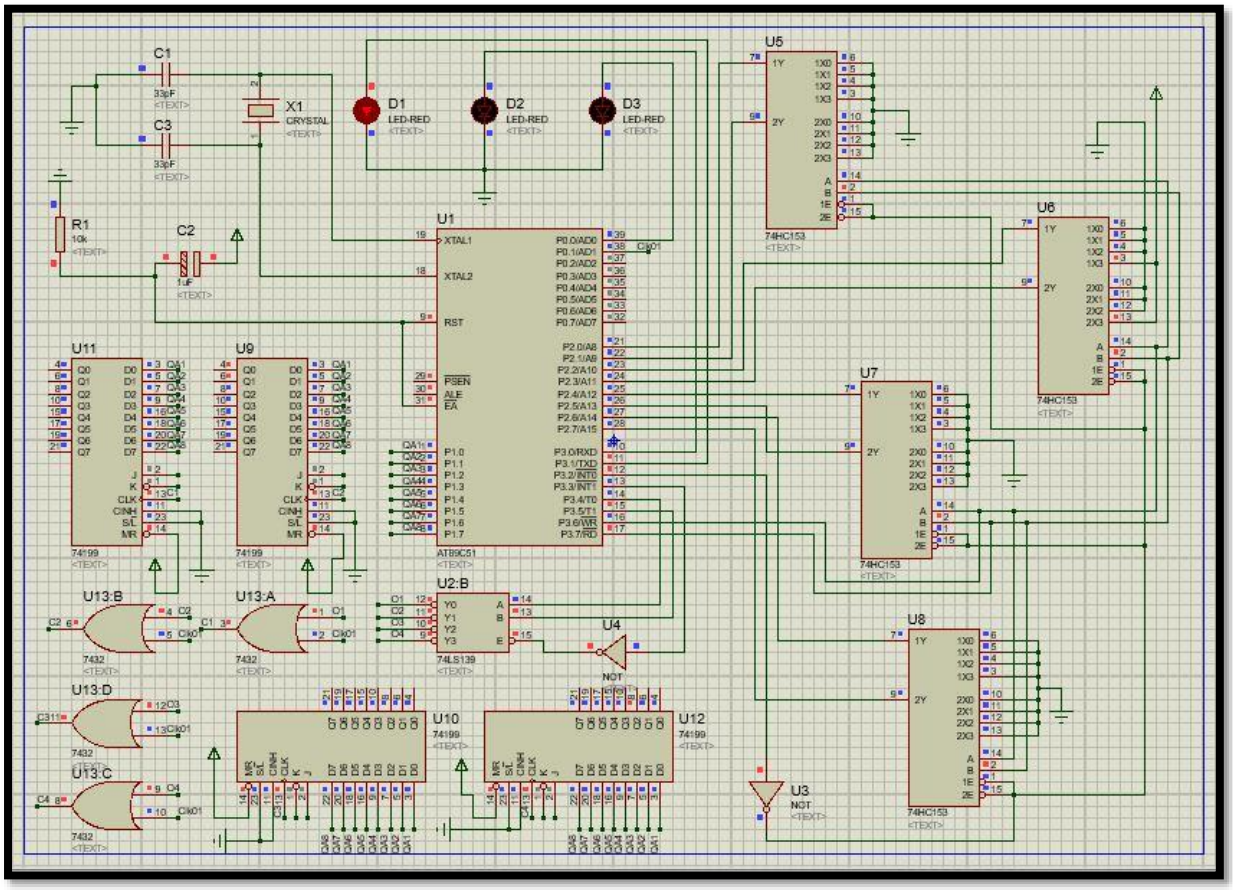

Figure no. 4 Proteus Simulation

NOTE: Methodology given in the manual to attempt this lab is not suitable. In my opinion, my approach should be reviewed for making lab manual more practical.