Report BTP

Team Size :2

Jashwanth Reddy

Kurra Lokesh

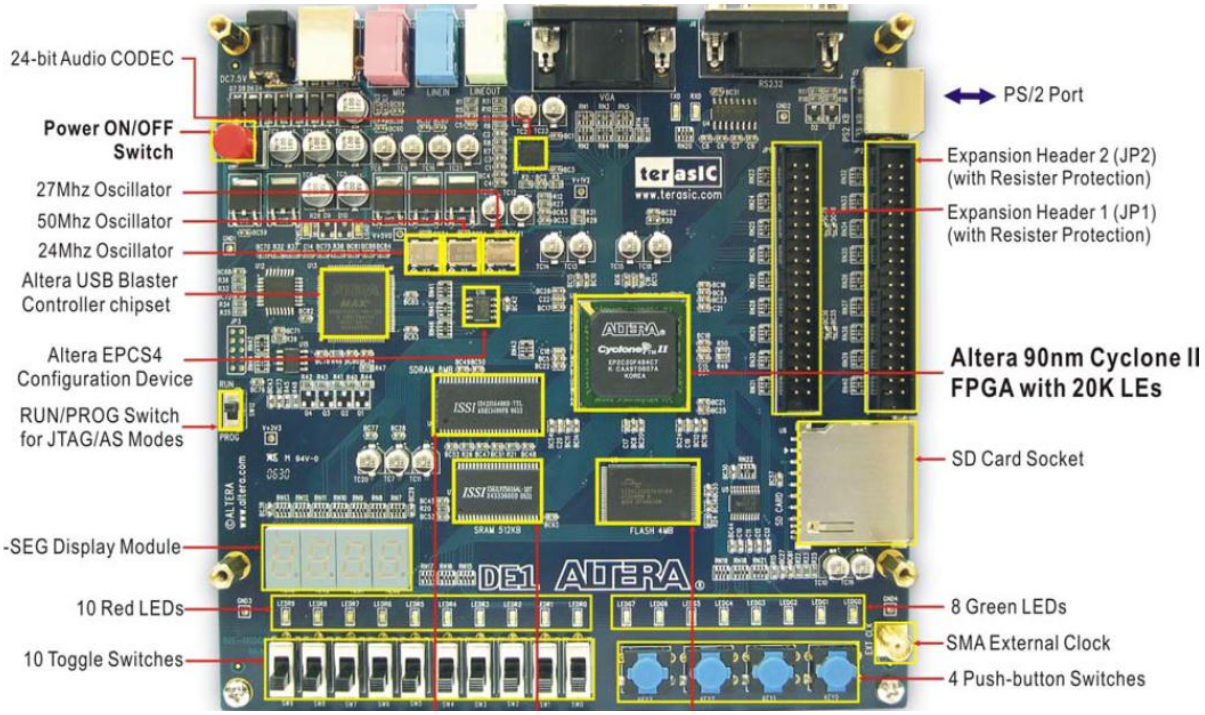
Guide : shubhajit roy chowdhury

Project : Computer on FPGA

Aim : The aim of the project is to use FPGA as a mother board to interface with I/O devices and memory management to develop some multimedia applications if possible.

Start to BTP eval-1:

During this part we are busy in understanding state of art of Altera Cyclone II 90nm FPGA.



The following hardware is provided on the DE1 board:

- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 512-Kbyte SRAM
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory
- SD Card socket
- 4 pushbutton switches
- 10 toggle switches
- 10 red user LEDs
- 8 green user LEDs
- 50-MHz oscillator, 27-MHz oscillator and 24-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (4-bit resistor network) with VGA-out connector
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- Two 40-pin Expansion Headers with resistor protection
- Powered by either a 7.5V DC adapter or a USB cable .

In order to use the DE1 board , the user has to be familiar with quartus software.

First we tried with the installation of altera USB blaster driver for communication between host computer and DE1 board

The default file loaded from flash memory helps to blink LED's , 7 Segment Displays start counting and outputs in Hexadecimal format.

We can also connect a micro phone and also VGA monitor to display default images.

The supporting software are

Quartus II web Edition

Nios II Software Build Tools

SOPC builder

Nios II Embedded Design Suite

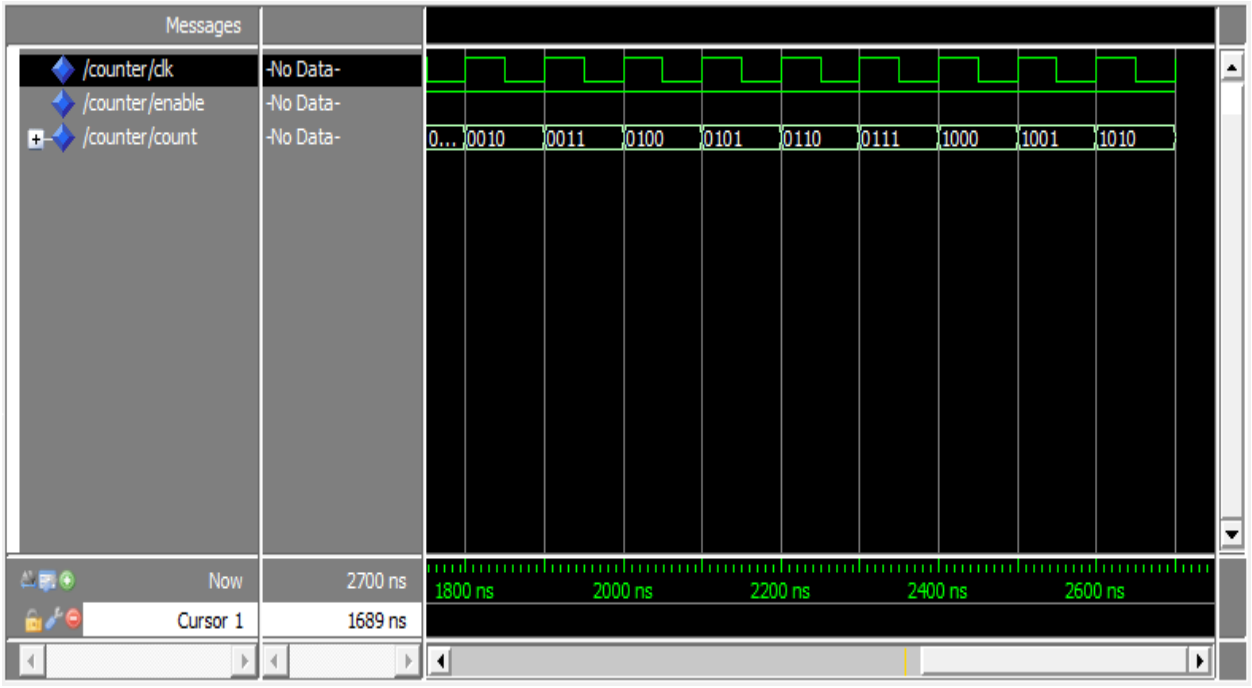
Modelsim Altera Starter Edition 6.5b

Coming to our work we developed some simple processor of 4 bit which can perform adder, substractor , comparator etc… first in ModelSim

And then in Quartus II web edition. The code and simulation result of a simple binary counter code in Modelsim looks like

```
library ieee;
Use ieee.std_logic_1164.all;
entity counter is
 port(  clk:  in std_logic; reset:  in std_logic;
        enable:  in std_logic;
count: out std_logic_vector (3 downto 0));
 end counter;
architecture behav of counter is
signal pre_count:
                std logic_vector (3 downto 0);
   begin
```

```
    process( clk , enable, reset)
      begin
       if reset = '1' then
        pre_count <= "0000";
Elsif  (clk='1' and clk'event) then
        if enable = '1' then
          pre_count <= pre_count + "1";
        end if;
      end if;
end process;
    count <= pre_count;
end behav;
```



There are pins availability for mapping these functions on FPGA .

This is done using IO-Assignment block editor in the software.The pin mappings are as follows:

27 MHz Clock	PIN_D12 and PIN_E12
50 MHz Clock	PIN_L1
24 MHz Clock	PIN_A12 and PIN_B12
Green LEDs	
Green_LED_0	PIN_U22
Green_LED_1	PIN_U21
Green_LED_2	PIN_V22
Green_LED_3	PIN_V21
Green_LED_4	PIN_W22
Green_LED_5	PIN_W21
Green_LED_6	PIN_Y22
Green_LED_7	PIN_Y21
Seven-segment Displays	
Hex_0[0]	PIN_J2
Hex_0[1]	PIN_J1
Hex_0[2]	PIN_H2
Hex_0[3]	PIN_H1
Hex_0[4]	PIN_F2
Hex_0[5]	PIN_F1
Hex_0[6]	PIN_E2
Hex_0, Decimal Point	No Connection
Hex_1[0]	PIN_E1
Hex_1[1]	PIN_H6
Hex_1[2]	PIN_H5
Hex_1[3]	PIN_H4
Hex_1[4]	PIN_G3
Hex_1[5]	PIN_D2
Hex_1[6]	PIN_D1
Hex_1, Decimal Point	No Connection
Hex_2[0]	PIN_G5
Hex_2[1]	PIN_G6
Hex_2[2]	PIN_C2

Hex_2[3]	PIN_C1
Hex_2[4]	PIN_E3
Hex_2[5]	PIN_E4
Hex_2[6]	PIN_D3
Hex_2, Decimal Point	No Connection
Hex_3[0]	PIN_F4
Hex_3[1]	PIN_D5
Hex_3[2]	PIN_D6
Hex_3[3]	PIN_J4
Hex_3[4]	PIN_L8
Hex_3[5]	PIN_F3
Hex_3[6]	PIN_D4
Hex_3, Decimal Point	No Connection
Push Buttons	
Key_0	PIN_R22
Key_1	PIN_R21
Key_2	PIN_T22
Key_3	PIN_T21
Red LEDs	
Red_LED_0	PIN_R20
Red_LED_1	PIN_R19
Red_LED_2	PIN_U19
Red_LED_3	PIN_Y19
Red_LED_4	PIN_T18
Red_LED_5	PIN_V19
Red_LED_6	PIN_Y18
Red_LED_7	PIN_U18
Red_LED_8	PIN_R18
Red_LED_9	PIN_R17
Slide Switches	
Switch_0	PIN_L22
Switch_1	PIN_L21
Switch_2	PIN_M22
Switch_3	PIN_V12
Switch_4	PIN_W12
Switch_5	PIN_U12
Switch_6	PIN_U11

Switch_7	PIN_M2
Switch_8	PIN_M1
Switch_9	PIN_L2

We achieved control of FPGA using the DE1 control panel user interface developed through which we can control LED's , SRAM , SDRAM , Flash memory on the chip.

We also got acquainted with SOPC builder . One can use the SOPC Builder to implement a desired system simply by choosing the required components and specifying the parameters needed to make each component fit the overall requirements of the system.
the overall requirements of the system.

This ended the first part of evaluation.

SECOND PART OF BTP:

In this part of BTP we provided VGA interface to FPGA .The VGA connector looks like

The main signals are VGA horizontal sync ,Vertical Sync

The pin connections and code are given below. VGA RGB data which were controlled in the following code using clock of 25 MHZ generated by FPGA.

```
module monitor{
                CLOCK_24  , CLOCK_27,
                CLOCK_50, EXT_CLOCK,
                KEY,  SW,   VGA_HS,
                VGA_VS,    VGA_R,      VGA_G,
                VGA_B,
        );
input [1:0] CLOCK_24;
input [1:0] CLOCK_27;
input  CLOCK_50;
input  EXT_CLOCK;
input [3:0] KEY;
input [9:0] SW;
output      VGA_HS;    VGA_VS;VGA V_SYNC
output      [3:0]  VGA_R;
output      [3:0]  VGA_G;
output      [3:0]  VGA_B;
 wire VGA_CTRL_CLK;
wire  AUD_CTRL_CLK;

wire  [9:0]  mVGA_X ; wire   [9:0]  mVGA_Y;wire     [9:0]  mVGA_R;
```

```
wire  [9:0] mVGA_G; wire   [9:0] mVGA_B; wire   [9:0] mPAR_R;
wire  [9:0] mPAR_G; wire   [9:0] mPAR_B; wire   [9:0] mPAR1_R;
wire  [9:0] mPAR1_G; wire  [9:0] mPAR1_B; wire  [9:0] mPAR2_R;
wire  [9:0] mPAR2_G; wire  [9:0] mPAR2_B; wire  [9:0] mPAR3_R;
wire  [9:0] mPAR3_G; wire  [9:0] mPAR3_B; wire  [9:0] mPAR4_R;
wire  [9:0] mPAR4_G; wire  [9:0] mPAR4_B; wire  [9:0] mPAR5_R;
wire  [9:0] mPAR5_G; wire  [9:0] mPAR5_B; wire  [9:0] mOSD_R;
wire  [9:0] mOSD_G; wire   [9:0] mOSD_B; wire   [9:0] oVGA_R;
wire  [9:0] oVGA_G; wire   [9:0] oVGA_B;
wire  [19:0] mVGA_ADDR;
reg   [27:0]       Cont; reg   ST;
always@(posedge CLOCK_50)            Cont <=    Cont+1'b1;
//     VGA Data 10-bit to 4-bit
assign      VGA_R            =      oVGA_R[9:6];
assign      VGA_G            =      oVGA_G[9:6];
assign      VGA_B            =      oVGA_B[9:6];
//     VGA Source pattern assignment of colors
  assign    mVGA_R           =      ( SW[0] & !SW[1] & !SW[2] ) ?
mPAR1_R : ( !SW[0] & SW[1] & !SW[2]  ) ? mPAR2_R : ( !SW[0] &
!SW[1] & SW[2]  ) ? mPAR3_R : mPAR4_R ;
      assign      mVGA_G           =      ( SW[0] & !SW[1] & !SW[2] ) ?
mPAR1_G : ( !SW[0] & SW[1] & !SW[2]  ) ? mPAR2_G : ( !SW[0] &
!SW[1] & SW[2]  ) ? mPAR3_G : mPAR4_G ;
```

```
assign mVGA_B = ( SW[0] & !SW[1] & !SW[2] ) ? mPAR1_B : ( !SW[0] & SW[1] & !SW[2] ) ? mPAR2_B : ( !SW[0] & !SW[1] & SW[2] ) ? mPAR3_B : mPAR4_B ;


VGA_Audio_PLL u1(.inclk0(CLOCK_27[0]),.c0(VGA_CTRL_CLK),.c1(AUD_CTRL_CLK) );

VGA_Pattern u6(.oRed(mPAR4_R), .oGreen(mPAR4_G) , .oBlue(mPAR4_B), .iVGA_X(mVGA_X), .iVGA_Y(mVGA_Y), .iVGA_CLK(VGA_CTRL_CLK), .iRST_N(KEY[0]) );

endmodule

/ *…….. The VGA  pattern code looks as follows …….. */

module

VGA_Pattern2 (  oRed , oGreen , oBlue , iVGA_X , iVGA_Y ,

                iVGA_CLK, iRST_N      );

output      reg   [9:0]  oRed;

output      reg   [9:0]  oGreen;

output      reg   [9:0]  oBlue;

input [9:0]       iVGA_X;

input [9:0]       iVGA_Y;

input                 iVGA_CLK;

//      Control Signals

input                 iRST_N;

always@(posedge iVGA_CLK or negedge iRST_N)
```

```
begin
	if(!iRST_N)
	begin
		oRed  <=    0;
		oGreen <= 0;
		oBlue	<=    0;
	end
	else
	begin
		oRed  <=    0;
		oGreen <=  512 ;
		oBlue	<=    0 ;
	end
end
endmodule
```

The results of VGA patterns were found satisfactory to the panel members and recommended further changes like character printing and also other interfaces to the board which we are looking forward to.