ImplementationofSegment_LCDdisplaybasedonSoCdesign

Ma Ling, Kab-Il Kim, Young I. Son
Dept. of Electrical Engineering, Myongji University
E-mail: maling0428@hotmail.com

Abstract : The purpose of this paper is to present how to implement Segment_LCD display using SoC design. The SoC design is achieved by using an ARM_based Excalibur device. The Excalibur device offers an outstanding embedded development platform with ARM922T and FPGA. The design in the Excalibur device uses the embedded ARM processor core and the AMBA high-performance bus (AHB) to write to a memory-mapped slave peripheral in the FPGA portion of the device. Here, Segment_LCD is one kind of memory-mapped slave peripherals. In order to implement the Segment_LCD display based on SoC design, four steps are followed. At first, IP modules are made by using Verilog HDL. Secondly, the ARM processor of the Excalibur is programmed using C in ADS (ARM Developer Suite). And in the third step, the whole system is simulated and verified. At last, modules are downloaded to SoCMaster kit. Both Quartus II software and ModelSim5.5e software are the key software tools during the design.

Keywords : SoC, Excalibur, IP, AHB, Verilog HDL, ARM922T, FPGA, SoCMaster

Ⅰ. Introduction

System-on-Chip(SoC)design based on Intellectual Property (IP) cores has become a growing trend in Integrated Circuit (IC) design [1]. It not only integrates processor cores and memory, but calls for a much higher level of integration where specialized IP cores, communication peripherals, and bus interfaces interact in order to function as a complete system [2].

Recently, the platform-based design approach to SoC designs has emerged [3]. A notable industrial example of platform is reconfigurable platform [3] from Altera Excalibur. This paper represents a novel approach that implements the Segment_LCD display using Excalibur device. The Excalibur device offers an outstanding embedded system development platform, providing a cost-efficient access to leading-edge embedded processors and PLD(processor programmable logic device) performance[4]. The SoCMaster, developed by Huins Corp, is an embedded SoC development Kit that provides a hardware platform for 'System on Chip' design [10]. It's a very useful device to implement Segment_LCD display by using the Excalibur (EPXA10F1020C2) SoC processor.

HDL (hardware description language) such as VHDL or Verilog HDL is used to design the digital circuits. Verilog HDL is used to describe the behavior of digital circuits in this paper.

This paper is structured as follows. The next section represents the basic principle. Section III is about a full design flow of Segment_LCD display based on SoC design. Section IV gets the results and Section V summarizes and concludes the paper.

Ⅱ. Basic principle

1. SoC design and IP module

SoC is a technology that integrates CPU, ROM, RAM, controller, and peripheral logic into one chip as well as hardware logic [10]. The key of SoC lies in the design. The traditional design bases on function, but SoC bases on the IP reuse or the function assemble]. That is to build a large system by a number of large macro modules. These macro modules are called IP (intellectual property) Cores [5].

2. Excalibur device

Excalibur embedded processor programmable logic devices (PLDs) combines an unparalleled degree of integration and programmability. The Excalibur family offers a variety of PLD densities and memory sizes to fita wide range of applications and requirements. The high-performance embedded architectures are ideal for compute as well as high data-bandwidth applications [6].

Figure 1 shows the structure of the Excalibur. The embedded stripe contains the processor core, peripherals, and memory subsystem.

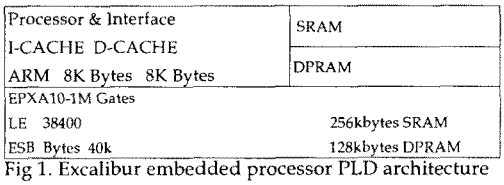

Fig 1. Excalibur embedded processor PLD architecture

Figure 2 shows the system architecture of the embedded stripe and the interfaces to the PLD portion of the devices. This architecture promotes maximum integration with minimal system cost and allows the embedded stripe and PLD to be independently optimized for maximum performance [6].

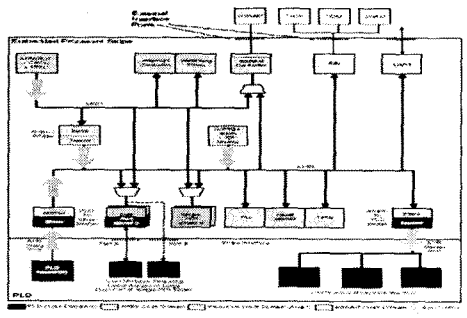

Fig 2. Excalibur embedded processor PLD system architecture

3. ARM922T processor module

The ARM922T is a member of the ARM9 family of processor cores and industry-standard processor core operating at up to 200 MHz. It support both the 32-bit ARM and 16-bit Thumb instruction sets, allowing users to achieve a balance high code-density and performance. Its Harvard architecture, implemented using a five-stage pipeline, allows single clock-cycle instruction operation through simultaneous fetch, decode, execute, memory, and write stages. The ARM922T implements an enhanced ARM Architecture V4 MMU, to provide translation and access permission checks for instruction and data addresses [6].

4. Bus Architecture

The bus architecture of the Excalibur family conforms to AHB specifications. The following figure 3 illustrates the Excalibur embedded processor stripe.

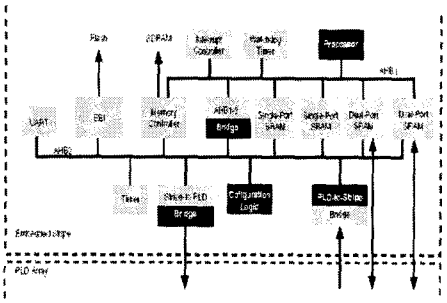

Fig 3. Excalibur embedded processor stripe

The Excalibur embedded processor PLD uses two AHBs as its communication medium, AHB1 and AHB2. Each bus has 32-bit address, read, and write data buses, which maximize access to the on-chip and off-chip memory resources, as well as to the shared peripherals [6].

5. Programmable Logic Architecture

The embedded stripe interfaces with a programmable logic architecture similar to that of an APEX20KE device. APEX20KE devices are designed with MultiCore architecture, which combines the strengths of LUT-based and product term-based devices with an enhanced memory structure. LUT-based and product term-based logic, make the APEX20KE-like device architecture suited for system-on-a-programmable-chip (SOPC) designs [6]. In this paper, the combination of ARM922T processor and 1 million gates FPGAs can design 7-Segment, Text_LCD, USB, UART etc.

III. A full design flow of Segment_LCD display using SoC design

In order to implement the Segment _LCD display based on SoC design, four steps are followed. The first step is an important step in the whole flow. That is, IP modules are made by using Verilog HDL. The Segment_LCD display mainly consists of three partsarm922t processor, AHB and Segment_LCD. Here, the ARM922T processor module is achieved by the Excalibur MegaWizard. The key is to set the parameters correctly on the basis of the structure of Excalibur. At the same time, the .sbd component is made by the Excalibur MegaWizard.The other IP modules are made by using Verilog HDL in the Quartus II software, such as seven_seg module in the Figure 4.

```
module seven_seg(
                 clk,
                 data,
                 reset_n,
                 enable_n,
                 address,
                 seg_data,
                 seg_out1,
                 seg_out2,
                 seg_gnd1,
                 seg_gnd2,
                 cnt3
               );
//Input
input                clk;          // Clock
input       [31:0]   data;         // Data Input
input                reset_n;      // Active Low reset
input                enable_n;     // Active Low enable
input       [ 9:2]   address;      //

//Output
output      [31:0]   seg_data;
output      [7:0]    seg_out1;     // Triple 7-segment Data_out 1
output      [7:0]    seg_out2;     // Triple 7-segment Data_out 2
output      [2:0]    seg_gnd1;     // 7-segment GND control 1
output      [1:0]    cnt3;

//Register
reg         [31:0]   D_Reg;        // Data register
reg         [2:0]    seg_gnd;
reg         [15:0]   cnt64k;
reg         [7:0]    seg_out1;     // Triple 7-segment Data_out 1
reg         [7:0]    seg_out2;     // Triple 7-segment Data_out 2
reg         [1:0]    cnt3;


//wire
wire        [7:0]    seg1;
wire        [7:0]    seg2;
wire        [7:0]    seg3;
wire        [7:0]    seg4;
wire        [7:0]    seg5;
wire        [7:0]    seg6;
wire        [2:0]    seg_gnd1 = seg_gnd;
wire        [2:0]    seg_gnd2 = seg_gnd;

//Instantiate the binary to 7-segment converter
bin2seg BIN2SEG_1( clk, D_Reg[23:20], seg1);
bin2seg BIN2SEG_2( clk, D_Reg[19:16], seg2);
bin2seg BIN2SEG_3( clk, D_Reg[15:12], seg3);
bin2seg BIN2SEG_4( clk, D_Reg[11: 8], seg4);
bin2seg BIN2SEG_5( clk, D_Reg[ 7: 4], seg5);
bin2seg BIN2SEG_6( clk, D_Reg[ 3: 0], seg6);

assign seg_data[31:3] = 26'b0;
assign seg_data[ 2:0] = seg_gnd;

//Data Latch
always @(negedge reset_n or posedge clk)
begin
   if (!reset_n)
   begin
      D_Reg <= 32'h0000;
   end
   else
   begin
      if (address[9:2]==1)
      begin
         D_Reg <= data;
      end
   end
end

//Segment display time out
```

Fig 4.1 The section Verilog HDL program of the seven_seg module

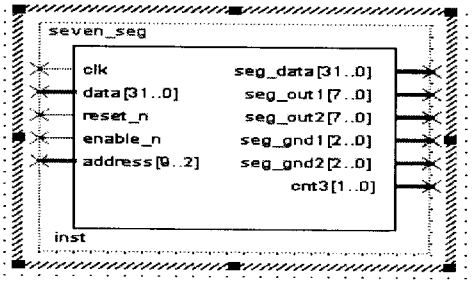

Fig 4.2 The symbol file of seven_seg module

After IP modules generated, we connected each module according to the AHB. We can obtain the segment_lcd display top-level file. The top-level files are shown as Figure 5 and 6.

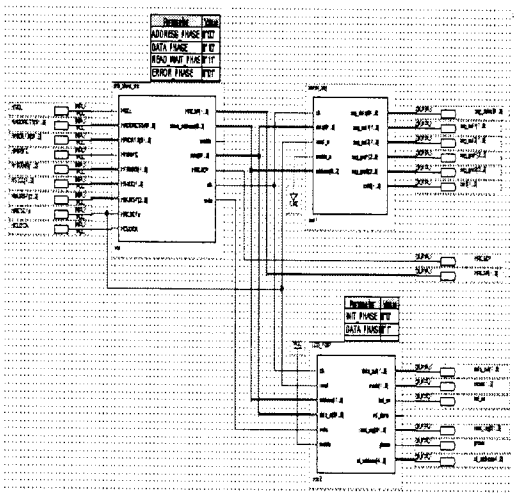

Fig 5. AHB state machine and Segment_LCD interface

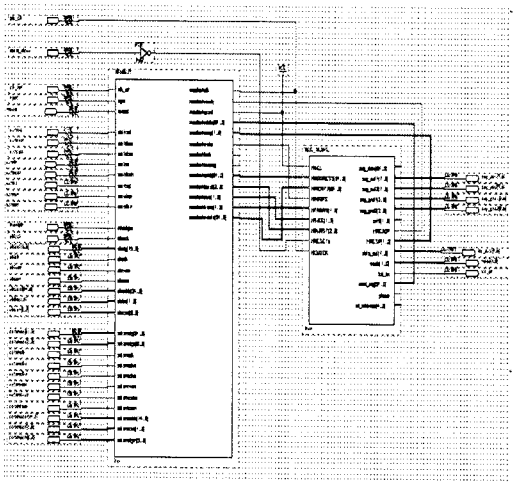

Fig 6. The whole arm_top file

Secondly, the ARM processor of the Excalibur is programmed by using C in ADS (ARM Developer Suite) to control the 7-Segment and text_lcd display. Figure 7 shows the Segment_LCD program in the ADS.

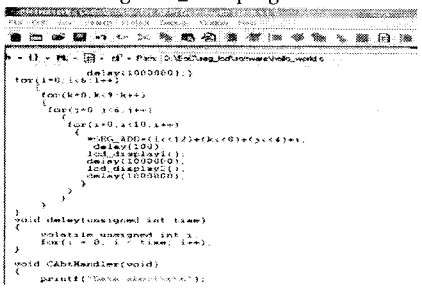

Fig 7. The section program of 7 segment and text_lcd

For software compile setting, we use software build settings in QuartusII software. And in the software build settings, we need to set debug and release according to the ADS respectively. Thus, the .HEX component is made by Quartus II in software mode. And in the third step, the whole system is simulated and verified. The slave peripheral is an arithmetic unit that performs a function based on data received from the embedded processor core. The Excalibur bus transactor, the bus functional model provided in the Quartus II software, is used along with the ModelSim software to simulate the design and verify the AHB compatibility of the slave peripheral [4]. Figure 7 shows a section of the ModelSim wave window to view the simulation results.

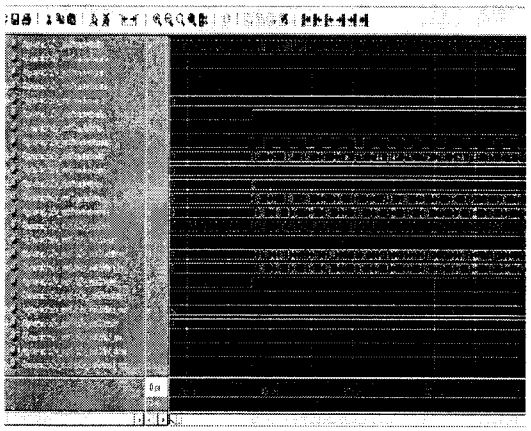

Fig 8. Simulation results

After finishing top-level file and simulation, we need to do some settings, such as device setting, pin assign and option setting. Then, we compile the hardware design. We can get the .sbi file. At last, modules are downloaded to the flash memory of SoCMaster kit by byteblaster. The intel. HEX programming file is generated by a combined effort of Quartus II and external softwares [9]. The configuration file in Flash

memory is as follows:

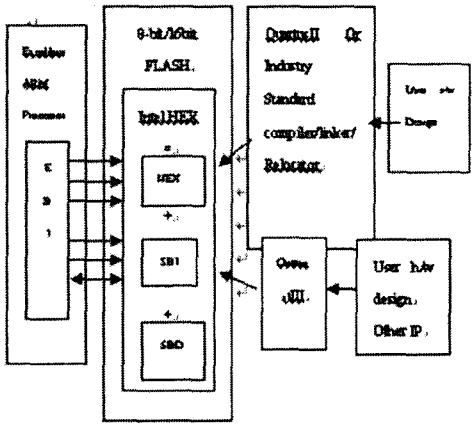

Fig 8. Configuration file in Flash memory

Fig 9. Download by Flash memory

Ⅳ. Results

We can show the results on 7-segment and Text_LCD.

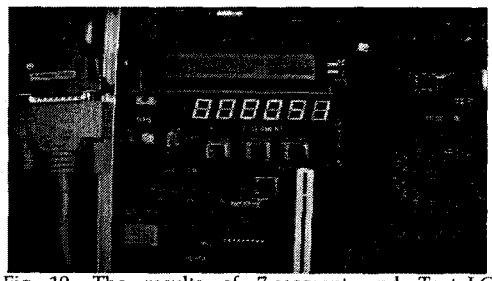

Fig 10. The results of 7-segment and Text_LCD display

By programming, we can make use of 7-segment as a watch showing the current time or counter calculating the numbers. Meanwhile, Text_LCD can display the information whatever we want to know.

Ⅴ. Summary and conclusion

In this paper, we have presented a good technique based on SoC design. ARM_based Excalibur device helps the implementation of Segment_LCD display by hardware/software co-development. At the same time, the combination of the ARM and FPGA offers to fit a wide range of applications and requirements. These improvements reflect the ability of the SoC design.

According to the full design flow, we can draw the conclusion. Although the whole process is complex, it is not difficult but useful. Once you master this method, it will help you develop more logic designs easily. There are two important and difficult parts while applying for SoC design. One is to get the top-level file, the other is to perform simulation using bus functional module.

With the method mentioned in this paper, more complex applications can be achieved, such as controlling robot. The field in using SoC design will be evolved quickly in the future.

Reference

[1] Wong, M.W.T. Ko, K.Y., Lee, Y.S., "Study of Test Approach for IP Cores Applicable to SOC Designs", *ASIC, 2001. Proceedings. 4th International Conference on* , 23-25 Oct. 2001

[2] Santanu DuTTA, "Architecure, Design, Verification and Validation of Multi-Processor SoCs for DTV, ASTB, and Media Processing Applications", *IEEE,2002*

[3] Grant Martin and Henry Cheng, "System-on-chip Design", *ASIC, 2001. Proceedings. 4th International Conference on* 23-25 Oct. 2001.

[4] *Excalibur Hardware Design Tutorial*, August 2002 version1.5

[5] Zhang Guiqing, Feng Tao, Wang Jianhua,A "The SoC Design and Implementation of Digital Protective Relay Based on IP Cores", *Power System Technology, 2002. Proceedings. Power Con 2002. International Conference on*, Volume: 4 , 13-17 Oct. 2002.

[6] *Excalibur Devices, hardware reference manual*, July 2002 version 3

[7] Steven J.E, Wiltion and Resve Saleh, "Programmable Logic IP Cores in SoC Design: Opportunities and Challenges", *custom integrated circuits conference, IEEE*, 2002.

[8] (주)휴인스, *임베디드 리눅스 시스*, 2003.

[9] (주)휴인스, *ARM, SoC 설계 교육*, 2003

[10] on line documents available at www.huins.com