

ECE385
Fall 2020
Experiment 7

SOC with NIOS-II in SystemVerlog

Name: Zhou Qinren
Zhang Yichi
Lab Section: LA3
TA's Name: Yu Yuqi

Nov. 17th 2020

1 Introduction

In this lab, we build a system on chip (SoC) with NIOS-II processor using the platform designer. We work on the eclipse for the first time to write, compile and execute C codes on the FPGA connecting software and hardware and implementing LED blink and accumulator.

2 Written Description and Diagrams of NIOS-II System

The hardware component of this lab includes the NIOS-II processor and its peripherals. The NIOS-II processor works as a CPU that stores temporary data, do the basic arithmetics and execute instructions. It connects to the SDRAM controller to manage the data transfer with the SDRAM and the SDRAM PLL to phase shift the external clock to provide a precise clock signal for the SDRAM. Also, it connects to the PIOs to control hardware inputs and outputs such as LEDs, switches and Keys.

3 Top Level Block Diagram

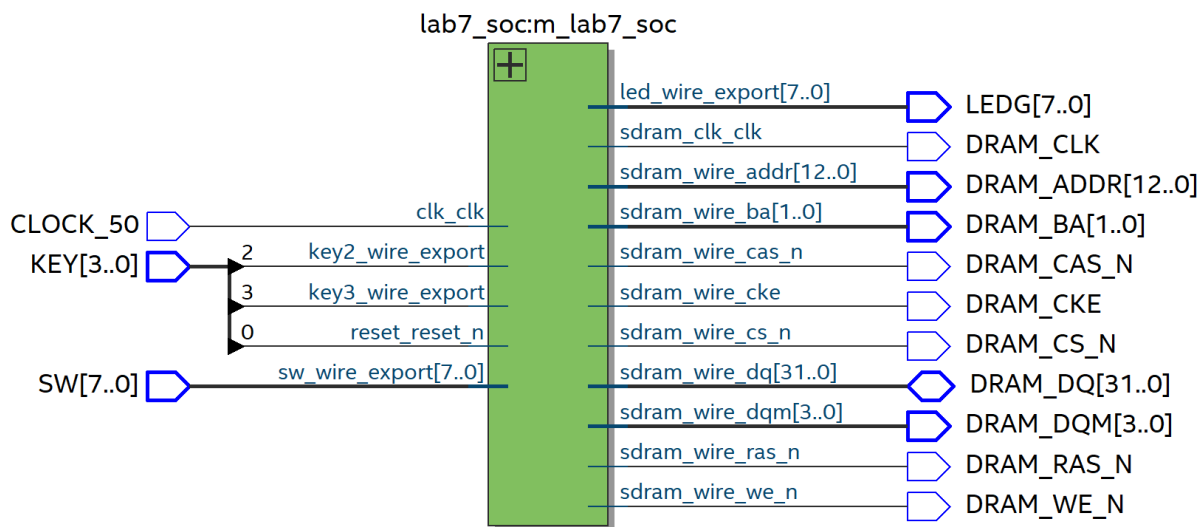


Figure 1: Top level block diagram.

4 Written Description of all .sv Modules

Module: lab7_soc

Inputs: clk_clk, key2_wire_export, key3_wire_export, reset_reset_n, sw_wire_export

Outputs: [7:0] led_wire_export, sdram_clk_clk, [12:0] sdram_wire_addr, [1:0] sdram_wire_ba, sdram_wire_cas_n, sdram_wire_cke, sdram_wire_cs_n, [3:0] sdram_wire_dqm, sdram_wire_ras_n, sdram_wire_we_n

Inouts: [31:0] sdram_wire_dq

Description: This is the whole system on chip module generated by platform designer that connects all the hardware together.

Purpose: This SoC module provide us a hardware environment to build software on. It takes in the inputs from the clock, switches and keys and outputs the result through the wires to the SDRAM and LEDs.

Module: lab7

Inputs: CLOCK_50, [3:0] KEY, [7:0] SW

Outputs: [7:0] LEDG, [12:0] DRAM_ADDR, [1:0] DRAM_BA, DRAM_CAS_N, DRAM_CKE, DRAM_CS_N, [3:0] DRAM_DQM, DRAM_RAS_N, DRAM_WE_N, DRAM_CLK

Inouts: [31:0] DRAM_DQ

Description: This is the top level module.

Purpose: This module connects the real LEDs, switches, keys and DRAM to the SoC module.

5 System Level Block Diagram

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>		clk_0	Clock Source					
		clk_in	Clock Input	clk	exported			
		clk_in_reset	Reset Input	reset				
		clk	Clock Output	Double-click to	clk_0			
		clk_reset	Reset Output	Double-click to				
<input checked="" type="checkbox"/>		nios2_gen2_0	Nios II Processor					
		clk	Clock Input	Double-click to	clk_0			
		reset	Reset Input	Double-click to	[clk]			
		data_master	Avalon Memory Mapped Master	Double-click to	[clk]			
		instruction_master	Avalon Memory Mapped Master	Double-click to	[clk]			
		irq	Interrupt Receiver	Double-click to	[clk]			IRQ 0
		debug_reset_request	Reset Output	Double-click to	[clk]			IRQ 31
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to	[clk]			
		custom_instruction_master	Custom Instruction Master	Double-click to	[clk]			
				Double-click to		0x0000_1000	0x0000_17ff	
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or RO...					
		clk1	Clock Input	Double-click to	clk_0			
		s1	Avalon Memory Mapped Slave	Double-click to	[clk]			
		reset1	Reset Input	Double-click to	[clk]			
<input checked="" type="checkbox"/>		led	PIO (Parallel I/O) Intel ...					
		clk	Clock Input	Double-click to	clk_0			
		reset	Reset Input	Double-click to	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to	[clk]			
		external_connection	Conduit	Double-click to	led_wire	0x0000_0060	0x0000_006f	
<input checked="" type="checkbox"/>		sdram	SDRAM Controller Intel FP...					
		clk	Clock Input	Double-click to	sdram_pll_c0			
		reset	Reset Input	Double-click to	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to	[clk]			
		wire	Conduit	Double-click to	sdram_wire	0x1000_0000	0x17ff_ffff	
<input checked="" type="checkbox"/>		sdram_pll	ALTPLL Intel FPGA IP					
		inclk_interface	Clock Input	Double-click to	clk_0			
		inclk_interface_reset	Reset Input	Double-click to	[inclk_interface]			
		pll_slave	Avalon Memory Mapped Slave	Double-click to	[inclk_interface]			
		c0	Clock Output	Double-click to	sdram_pll_c0	0x0000_0070	0x0000_007f	
		c1	Clock Output	Double-click to	sdram_pll_c1			
<input checked="" type="checkbox"/>		sysid_qsys_0	System ID Peripheral Inte...					
		clk	Clock Input	Double-click to	clk_0			
		reset	Reset Input	Double-click to	[clk]			
		control_slave	Avalon Memory Mapped Slave	Double-click to	[clk]	0x0000_0088	0x0000_008f	
<input checked="" type="checkbox"/>		sw	PIO (Parallel I/O) Intel ...					
		clk	Clock Input	Double-click to	clk_0			
		reset	Reset Input	Double-click to	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to	[clk]			
		external_connection	Conduit	Double-click to	sw_wire	0x0000_0050	0x0000_005f	
<input checked="" type="checkbox"/>		key2	PIO (Parallel I/O) Intel ...					
		clk	Clock Input	Double-click to	clk_0			
		reset	Reset Input	Double-click to	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to	[clk]			
		external_connection	Conduit	Double-click to	key2_wire	0x0000_0040	0x0000_004f	
<input checked="" type="checkbox"/>		key3	PIO (Parallel I/O) Intel ...					
		clk	Clock Input	Double-click to	clk_0			
		reset	Reset Input	Double-click to	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to	[clk]			
		external_connection	Conduit	Double-click to	key3_wire	0x0000_0030	0x0000_003f	

Figure 2: Platform Designer view.

block: clk_0

Inputs: clk_in, clk_in_reset

Outputs: clk, clk_reset

functionality: The clock of all the components of SoC except for the SDRAM controller.

block: nios2_gen2_0

Inputs: clk, reset, debug_mem_slave

Outputs: data_master, instruction_master, debug_reset_request

functionality: NIOS-II processor, the CPU.

block: onchip_memory2_0

Inputs: clk1, s1, reset1

functionality: An onchip memory storing temporary data like a Regfile in lc3.

block: led

Inputs: clk, reset, s1

functionality: LED.

block: sdram

Inputs: clk, reset, s1

functionality: SDRAM controller. It interfaces and transfers data with the SDRAM chip.

block: sdram_pll

Inputs: inclk_interface, inclk_interface_reset, pll_slave

Outputs: c0, c1

functionality: The clock for SDRAM controller and the clock for SDRAM.

block: sysid_qsys_0

Inputs: clk, reset, control_slave

functionality: System ID. It ensures the compatibility between hardware and software.

block: sw

Inputs: clk, reset, s1

functionality: Switch inputs.

block: key2

Inputs: clk, reset, s1

functionality: Key[2] input for reset.

block: key3

Inputs: clk, reset, s1

functionality: Key[3] input for accumulating.

6 Software Component

```
int main()
{
    int i = 0;
    volatile unsigned int *LED_PIO = (unsigned int*)0x60; //make a pointer to access the PIO block
    volatile unsigned int *KEY2 = (unsigned int*)0x40;
    volatile unsigned int *KEY3 = (unsigned int*)0x30;
    volatile unsigned int *SW = (unsigned int*)0x50;
    int flag = 0;

    *LED_PIO = 0; //clear all LEDs
    while ( (1+1) != 3) //infinite loop
    {
        for (i = 0; i < 100000; i++); //software delay
        *LED_PIO |= 0x1; //set LSB
        for (i = 0; i < 100000; i++); //software delay
        *LED_PIO &= ~0x1; //clear LSB
    }
    return 1; //never gets here
}
```

Figure 3: Codes for LED blink.

This is the code for LED blink. We use for loop to simulate the delay, so every 100000 loops we turn on or turn off the LED.

```
int main()
{
    int i = 0;
    volatile unsigned int *LED_PIO = (unsigned int*)0x60; //make a pointer to access the PIO block
    volatile unsigned int *KEY2 = (unsigned int*)0x40;
    volatile unsigned int *KEY3 = (unsigned int*)0x30;
    volatile unsigned int *SW = (unsigned int*)0x50;
    int flag = 0;

    *LED_PIO = 0; //clear all LEDs
    while ( (1+1) != 3) //infinite loop
    {
        if (*KEY3 == 0 && flag == 0)
        {
            *LED_PIO += *SW;
            flag = 1;
        }
        if (*KEY3 == 1)
        {
            flag = 0;
        }
        if (*KEY2 == 0)
        {
            *LED_PIO = 0;
        }
    }
    return 1; //never gets here
}
```

Figure 4: Codes for accumulator.

This is the code for accumulator. Every time the Key3 is pressed, we add the value represented by switches to the LED so the corresponding LEDs light up and set the flag to 1 to prevent unwanted accumulation when we press the key for a long time. The flag will be reset to 0 until we release the key3. When we press key2, we reset the LEDs.

7 Answers to all 11 INQ Questions

Question: What are the differences between the Nios II/e and Nios II/f CPUs?

Answer: Nios II/e is the economy version of the processor, it has less functionalities than Nios II/f, but it takes up less places and resources.

Question: What advantage might on-chip memory have for program execution?

Answer: The transmission time may be much shorter, i.e., the write and read operations are more efficient, since the memory is allocated on chip, rather than somewhere far away such as sdram.

Question: Note the bus connections coming from the NIOS II; is it a Von Neumann, “pure Harvard”, or “modified Harvard” machine and why?

Answer: It is a modified Harvard machine. Because the instruction memory may be accessed as data.

Question: Note that while the on-chip memory needs access to both the data and program bus, the led peripheral only needs access to the data bus. Why might this be the case?

Answer: The reason is that the led only needs data so that it can output the corresponding information. LED does not run the program.

Question: Why does SDRAM require constant refreshing?

Answer: Because SDRAM has to preserve the contents, otherwise the charges in the capacitor may leak out, and the information may be lost.

Question: make sure this is consistent with your above numbers; you will need to justify how you came up with 1 Gbit to your TA.

Answer: $32 \text{ bits} * 2^{13} * 2^{10} * 4 * 2 = 1 \text{ Gbit}$

Question: What is the maximum theoretical transfer rate to the SDRAM according to the timings given?

Answer: $1 / 5.5\text{ns} * 32\text{bit} = 5.42\text{Gbit /s}$

Question: The SDRAM also cannot be run too slowly (below 50 MHz). Why might this be the case?

Answer: The reason might be that the refresh rate is too slow, and the charge leaks out, resulting in data corruption.

Question: This puts the clock going out to the SDRAM chip (clk c1) 3ns behind of the controller

clock (clk c0). Why do we need to do this?

Answer: SDRAM requires a precise clock. We have to compensate for the delay due to transmission or other reasons.

Question: What address does the NIOS II start execution from? Why do we do this step after assigning the addresses?

Answer: The program starts from address 0x10000000, which is the reset vector.

Question: Look at the various segment (.bss, .heap, .rodata, .rwdata, .stack, .text), what does each section mean? Give an example of C code which places data into each segment.

Answer:

bss: a global variable not initialized to a specific value

Code: int i;

.heap: memory which is allocated on heap

Code: int *ptr = (int*)malloc(sizeof(int));

.rodata: read only data, const

Code: const int i = 3;

.rwdata: read and write data

Code: int i = 3;

.stack: stack which implements function call

Code: i = foo(parameter);

.text: string segment

Code: char[] str = "hello world";

8 Postlab Question

SDRAM Parameter	Value
Data width	32
# of Rows	13
# of Columns	10
# of Chip selects	1
# of Banks	4

Table 1: SDRAM parameter.

LUT	2222
DSP	0
BRAM	36864
Flip-Flop	1960
Frequency	79.21MHz
Static Power	102.03mW
Dynamic Power	39.97mW
Total Power	195.41mW

Table 2: Design statistics table for the multiplier.

9 Conclusion

This lab is easy to conduct but it takes much time to digest the information. We had little trouble during the lab.