2011 International Conference on Computer Science and Information Technology (ICCSIT 2011)
IPCSIT vol. 51 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V51.133

Exploration on Affecting Performance Results of Nios II

Peng Xiao-hong, Ma Meng-ya[+], Hou Li-gang, Wang Jin-hui, Wu Wu-chen

VLSI & System Laboratory, Beijing University of Technology, Beijing, China

Abstract. This paper introduces Nios II and Dhrystone 2.1 benchmark. Since DhrystoneV2.1 Benchmark is widely used by various embedded processor vendors, this paper adopts Dhrystone 2.1 benchmark to evaluate performance of Nios II processor in different architectures and conditions. The first condition is whether the system has on-chip memory. The second is whether the REG option is turned on in source files. The third is whether the source code has been optimized. The fourth is whether the off-chip memory SRAM is used.

Keywords: Nios II; Dhrystone 2.1; performance; factors;

1. Introduction

Nios II is a 32-bit embedded-processor architecture designed specifically for the Altera family of FPGAs. Nios II incorporates many enhancements over the original Nios architecture, making it more suitable for a wider range of embedded computing applications, from DSP to system-control [1].

But when we use Dhrystone 2.1 benchmark to evaluate its performance, we can get different results under the influence of various factors. These factors may be from system architectures, Nios II structure, and software configuration and so on. This paper will explore these factors that may affect the performance results.

2. Nios II

Like the original Nios, the Nios II architecture is a RISC soft-core architecture which is implemented entirely in the programmable logic and memory blocks of Altera FPGAs. The soft-core nature of the Nios II processor lets the system designer specify and generate a custom Nios II core, tailored for his or her specific application requirements. System designers can extend the Nios II's basic functionality by adding a predefined memory management unit, or defining custom instructions and custom peripherals.

Using the Nios II processor, users can adjust the characteristics, performance and costs of the embedded system, quickly pushing the products into market, extending the life cycle of product, also avoiding the losses due to updating and upgrading of processor [1].

Using Nios II processor has the following advantages: (1)There are a series of processor cores available, including core which has more than 200 DMIPS performance; you can put as many processors together as you need and you can combine different processors on a board; you can add one or more Nios II soft-core processors; you can reduce system costs, complexity and power consumption by integrating processor, peripherals, memory and I/O interfaces into a single FPGA; you only need to spend 10-20 RMB by embedding Nios II processor into low-cost Cyclone FPGA. (2)It provides easy to use tools for design, which quickly push products into market; it provides permanent, free license, so that products based on Nios II processor avoid losses due to updating and upgrading. (3)By using Nios II integrated development

[+] Ma Meng-ya
mmytiankong@yahoo.com.cn

environment (IDE), software development is accelerated; a system can be built in minutes by taking advantages of powerful SOPC Builder system development tool of Altera and Quartus II design software [2].

The Nios II structure graph is shown in Fig. 1[1]:

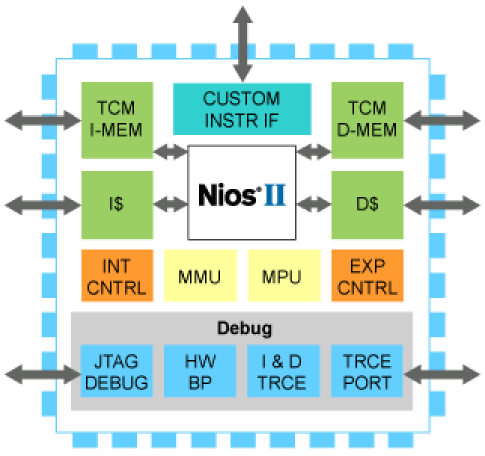

Figure 1. Structure of Nios II

The Nios II architecture defines the following functional units:

- Register file
- Arithmetic logic unit (ALU)
- Interface to custom instruction logic
- Exception controller
- Internal or external interrupt controller
- Instruction bus
- Data bus
- Memory management unit (MMU)
- Memory protection unit (MPU)
- Instruction and data cache memories
- Tightly-coupled memory interfaces for instructions and data
- JTAG debug module

Nios II processor is designed for 5-stage pipelines, with separate data and instruction Harvard structure. Nios II has its own dedicated architecture and instruction set to support 32 bits hardware multiplication and division instructions. It has 32 general purpose registers. Users can also customize up to 256 instructions according to their needs [3].

Nios II uses Altera's own Avalon-chip bus standard for connecting the timer, UART interfaces, LCD interface, memory controller and Ethernet interface chip modules. Nios II also provides a Debug module which supports on-line JTAG debug. Altera Corporation provides Nios II with a very comprehensive software and hardware development environment. Nios II processor program is based on the construction of HDL source, providing three different basic soft cores with performance and resource consumption. They are Nios II/f (fast), Nios II/s (standard) and Nios II/e (economy). Using SOPC Builder system development tool of Quartus II development software, users can conveniently configure their Nios II cores to meet their own needs on the basis of any kind of soft-core.

The company also provides toolchains based on GNUC/C++ and Eclipse IDE software development environment for Nios II. Users can in this development environment to facilitate the completion of coding, simulation and debugging work. Nios II development kit supports an uc/OS-II free real-time operating system, while Nios II also supports uClinux, Nucleus Plus, KROS and other third-party operating systems. Users can use free Nios II development platform, but Nios II only supports Altera Stratix and Cyclone devices. Users can only use free Nios II on Altera's FPGA, and can't get Nios II HDL source code. If designers use Nios II kernel in ASIC design, you need to pay licensing fees.

3. Dhrystone benchmark

Currently there is no uniform standard of performance testing for embedded processors; we chose a DhrystoneV2.1 Benchmark evaluation criterion which is widely used by various embedded processor vendors. The Dhrystone benchmark was created back in 1984 by Dr. Reinhold P.Weicker. Today Dhrystone 2.1 is the current version, which was written in 1988. Weicker's intention with writing Dhrystone was to measure the performance of computer systems, and since the computer systems of that era were focused on integer performance, Dhrystone primarily targets integer performance. Dhrystone is a synthetic benchmark composed of a, of that time, "typical" application mix of mathematical and other operators. The Dhrystone benchmark basically consists of a main loop executed a number of times. The output of the benchmark is the time spent in the main loop [4].

Running Dhrystone 2.1 benchmark program on different processor platforms, the parameter values can be obtained from Dhrystone per Second, with VAX-11/780 for reference value. We convert this value to the Dhrystone 2.1VAXMIPS (DMIPS) value. Because the test result is 1757 Dhrystone per Second for the VAX-11/780, as a reference, draw DMIPS = Dhrystone per Second/1757. Because the processor's performance is closely related to the operating frequency, which DMIPS is different measured at different operating frequencies, usually DMIPS / MHz is used as a standard to assess the advantages and disadvantages of the structure and the performance level for each processor [5].

In Dhrystone benchmark, we can see two parameters. The first is Microseconds which means the execution time of running Dhrystone for one times. This is an equation shown in (1):

$$\text{Microseconds} = \frac{T_{user\_time}}{N_{number\_of\_runs}} * K \tag{1}$$

Parameter T_{user_time} is the total execution time of running Dhrystone program and its unit is second. Parameter $N_{number\_of\_runs}$ is the total number of running it. And K is a constant number, which equal to 1,000,000. It represents the number of microseconds in a second. The lower the value is, the better the processor is. The second is Dhrystones_per_Second. Its equation is shown in (2):

$$\text{Dhrystones\_Per\_Second} = \frac{T_{user-time}}{N_{number-of-runs}} \tag{2}$$

The parameters are as same as that in the previous equation. And Dhrystones_Per_Second means execution times for processor running program in a second. The higher the value is, the better the performance is. The result we wanted is DMIPS. Its equation is shown in (3):

$$\text{DMIPS} = \frac{\text{Dhrystones\_Per\_Second}}{1757} \tag{3}$$

4. Exploration on improving performance

4.1. The Architecture of Nios II and System

This paper shall use Dhrystone benchmark to assess the Nios II/s performance. This paper chose Altera Cyclone II development board (EP2C35F484I8) as hardware testing platform.

Nios II/s core is designed to maintain a balance between performance and cost. The configuration of Nios II/s as follows:

- Instruction cache of 4Kbytes
- Optional tightly coupled memory for instructions
- Five-stage pipeline
- Static branch prediction
- Hardware multiply
- JTAG debug module (1 level)
- resource usage: 1200-1400 LEs

These are several components in this system including on-chip memory of 44Kbytes, JTAG-UART, Timer and SDRAM controller. About 44Kbytes memory bits can be used as on-chip memory, but the size is not enough for running Dhrystone program; so we need to add off-chip memory. This paper add SDRAM controller which governs SDRAM chip with size of 64Mbits. Internal timer is needed for running Dhrystone program. And JTAG-UART is used for communicating between PC and FPGA. The external clock frequency is 50MHz with the same as the board clock. This paper also adds a PLL so that one of output clocks from PLL connects clock input of SDRAM chip and another feeds the system in FPGA. The frequency of both is not change.

This system can be easily generated but PLL in SOPC Builder and compiled the whole system in Quartus II. The result of compilation is this whole system consumes 3416 logic elements and 374,400 memory bits. They take up 10% and 77% of the total number respectively.

This paper shall talk about the factors that can affect the result of running Dhrystone program. Here the result means DMIPS value. After we build up our system in SOPC Builder and Quartus II, we can download this famous benchmark from Internet and compile it in Nios II IDE after appropriate modification. We can probe into the factors that may affect DMIPS value by testing in different conditions.

4.2. B The Experiment Without On-chip Memory

The on-chip memory can be removed and program can be running only in SDRAM. The structure of system is shown in Fig. 2:

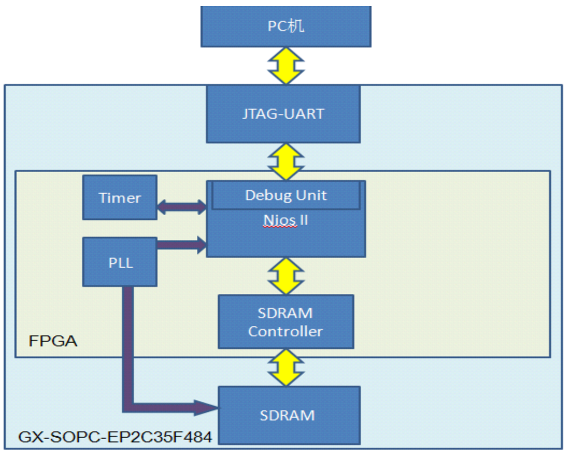

Figure 2. System structure without on-chip memory

At 50 MHz frequency here comes out result with extremely low number 3.696. The reason is easily understood because the speed of SDRAM is much slower than that of on-chip memory.

4.3. The Experiment With On-chip Memory

Next this paper adds on-chip memory with about 44Kbytes size into our system. Because we can’t put all the data on the on-chip memory, part of data are chosen to put on the SDRAM. Another reason to do this is that program data size is too large to fit the size of on-chip memory.The system structure is shown in Fig. 3:

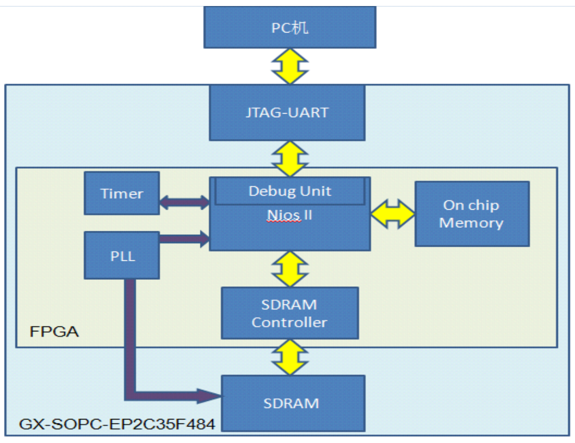

Figure 3. The whole system structure

▪ *4.3.1.* Poor Utilization of On-chip Memory

This paper put read-only data and read/write data onto the on-chip memory, leaving program data, heap and stack in SDRAM. This condition this paper named is poor on-chip memory utilization. The result is 3.988. It is not a good result.

- *4.3.2.* **Better Utilization of On-chip Memory**

In order to take full advantage of on-chip memory, all the data but program data are put on the on-chip memory. This paper calls this condition better on-chip memory utilization. With the superiority of on-chip memory, the result improves to some extent, reaching 10.396 on this board. But this value is still low, equivalent to performance of Nios.

4.4. The Experiment With REG Option

We notice that there is an option in the source file. This is REG option. Turning on this option or not may affect the result. The experiments above are based on the option without REG. So next we modify source file to open the REG option. The result value increases a little, reaching 10.702. The conclusion is that existence of REG slightly improves performance.

4.5. The Experiment With Optimization

The optimization of source files can greatly improve performance proved by others' experiments. We shall also prove this. After optimization the code size becomes much smaller, but it still cannot fit the size of on-chip memory; so we still put part of that on SDRAM and this part is same as above. In this experiment the DMIPS value markedly increases up to 32.893. The optimization of source files plays a more important role compared with addition of on-chip memory. That's the result that we wish to see. The reason that optimization could largely improve performance may be that the execution time of cpu decreases when the code size gets smaller. So the smaller the code is, the less time the cpu executes.

4.6. Experiment With SRAM as Off-chip Memory

The speed of SRAM is faster than that of SDRAM. But I get the same final result when I use SRAM as off-chip memory instead of SDRAM. However, I find that the SDRAM controller and the PLL can be saved, which can reduce the consumption of logic elements. This system structure is shown in Fig.4.

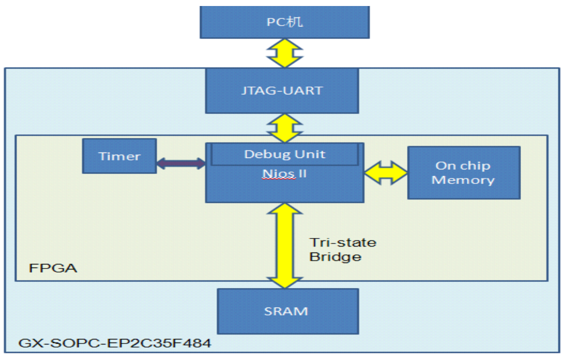

Figure 4. The system structure with SRAM

4.7. Combinational Experiments in Different Conditions

This paper has made 12 experiments under 4 different conditions. These conditions are REG option, optimization option, poor on-chip memory utilization and better on-chip memory utilization. The results are listed in Table I and Table II.

Table I. Results without REG option

REG: NO	**Optimization: NO**	**Optimization: YES**
On chip Memory: NO	3.696	11.159
On chip Memory: Poor Utilization	3.988	14.544
On chip Memory: Better Utilization	10.396	32.893

Table II. Results with REG option

REG: NO	**Optimization: NO**	**Optimization: YES**
On chip Memory: NO	3.639	11.159
On chip Memory: Poor Utilization	3.919	14.544
On chip Memory: Better Utilization	10.702	32.893

From these two tables we can see that better utilization of on-chip memory and optimization of code can improve performance result by about 3 times. Under optimization condition, turning on REG option or not cannot affect results. But in no optimization context, turning on REG option will increases slightly the value of DMIPS when on-chip memory is utilized well and turning off REG option will bring opposite result without on-chip memory and with poor utilization.

This paper still makes efforts to improve DMIPS value. Enabling the hardware divide may improve performance. After running benchmark we get 33.239. The increase is 0.346. This is a slightly amplification with the same as REG option at the cost of consuming about 300 logic elements. The tool may use these logic elements as special division unit. Anyway it improves performance though a little.

At 50MHz we get DMIPS value, which is 33.239, so the value of DMIPS/MHz is 0.66. When this paper increases frequency up to 75MHz by PLL, its result increases correspondingly. It is 49.860. The ratio of both is 0.66, which is as same as the previous. Increasing on frequency can improve DMIPS value, but cannot enhance value of DMIPS/MHz.

5. Conclusion

This paper adopts Dhrystone 2.1 benchmark to evaluate Nios II processor's performance in different architectures and conditions. This paper gets conclusions through experiments that better utilization of on-chip memory and optimization of code can improve performance result by about 3 times. When the REG option is turned on in source files and hardware divide feature is added into system, both of two can improves performance slightly. And when frequency is increased, the DMIPS value improves correspondingly, but the ratio of both is not change.

6. Acknowledgment

This work was supported by the National Natural Science Foundation of China (60976028) and the Startup Foundation for Doctors of BJUT (X0002014201101, X0002012200802, and X0002013201103).

7. References

[1] http://www.altera.com.cn/

[2] Lanying Li, SOPC design principles and application of Nios II embedded soft-core, 1st ed. Beijing University of Aeronautics & Astronautics Press, 2006, pp. 29-30.

[3] Altera Corp. Nios II Processor Reference Book. Altera, 2010.

[4] DANIEL MATTSSON, MARCUS CHRISTENSSON, "Evaluation of synthesizable CPU cores", Computer Science and Engineering Program.

[5] LIU Jun, GUO Li, ZHENG Dong-f ei, BAI Xue-fei, "Comparison and Analysis of Open-Source 32-bit RISC Processor IP Core", Chinese Jour nal of Electron Devices, Vo l. 28, Dec. 2005,