Sborník vědeckých prací Vysoké školy báňské - Technické univerzity Ostrava
číslo 2, rok 2006, ročník LII, řada strojní
článek č. 1543

János LAZÁNYI[*], Béla FEHÉR [**]

DISTRIBUTED EMBEDDED SYSTEM USING SINGLE FPGA

DISTRIBUOVANÝ EMBEDDED SYSTÉM S VYUŽITÍM FPGA

Abstract

This paper presents an evaluated configuration with three microprocessors inside a single FPGA. The main processor is running an embedded operating system enabling to implement sophisticated software algorithms, meanwhile the other two 32 bit microprocessor runs separate, dedicated hardware related real-time tasks.

Abstract

Příspěvek popisuje zhodnocení konfigurace s třemi mikroprocesory a jediným FPGA. Embedded operační systém běží na hlavním procesoru, který dokáže implementovat sofistikované softwarové algoritmy, zatímco další dva 32-bitové mikroprocesory běží odděleně a obhospodařují úkoly v reálném čase.

1 INTRODUCTION

FPGA technology has been improved rapidly in the last decade. Multiple microprocessor cores, embedded bus systems, on-chip memory and many others along with dedicated peripheral unit and hardware accelerators can be implemented on a single FPGA. Major chip vendors offer complex design environment for implementing such complicated distributed **SoPC** (*System on a Programmable Chip*) systems. A high range of silicones from low-cost simple solutions, up till devices with 4 hardwired and tens of software-configurable processor cores are available on the market.

Current embedded system designs use a single microprocessor in general. The time critical and real-time functions are usually implemented using interrupts, meanwhile the auxiliary functions are running in a main program loop. Increasing the complexity of the given design intricate the nested interrupt structure which results in unpredictable response time and problematical system verification.

By implementing "network-of-processors" on a single FPGA the functional requirements can be separated into well defined tasks, each of them can be assigned to dedicated on-chip microprocessors. Therefore response time of the time critical task can be predicted which is not affected by the overall system performance.

2 EVALUATED SYSTEM ARCHITECTURE

A multiprocessor SoPC was built to evaluate the proposed network-of-processor system architecture. The main PowerPC processor may run Nucleus RTOS allowing to implement complex system tasks, like communication with host computer over Ethernet, file system management etc. The underling two Microblaze microprocessors run dedicated real time task without operating system.

[*] János Lazányi, Department of Measurement and Information Systems, Budapest University of Technology and Economics, Magyar tudósok krt. 2., Budapest, (+36) 1-463-2066, lazanyi@mit.bme.hu
[**] Béla Fehér PhD, Department of Measurement and Information Systems, Budapest University of Technology and Economics, Magyar tudósok krt. 2., Budapest, (+36) 1-463-2686, feher@mit.bme.hu

Two-fold bus architecture serves as a communication channel. There are direct point-to-point links between the secondary processors, and a shared memory area between the main PowerPC and the slave Microblaze processors.

A Virtex 4 FX Evaluation Board was used for test purposes. The PCB includes 32 MBytes of DDR SDRAM memory, 10/100 Ethernet, serial port and XC4VFX12 Virtex 4 multi-platform FPGA with approximately 5500 slices and 36•18Kbit Block RAMs.

2.2 Typical application

The above described architecture can be useful in many applications, where a central processing unit and several slave computational units are required.

A typical application could be image or audio compressing / decompressing where the given algorithm is separated into complex functional blocks that are subservient to each other, enabling pipelined operation. Alternatively, secondary functional units can serve logically independent functions too. A usage scenario could be a multi-protocol analyzer, where each Microblaze processor owns dedicated hardware interface for external communication. PowerPC processor gathers data measurements, transfers to the host computer and stores it on board. Moreover computation throughput can be improved using multiple functional units performing identical operation.

3 HARDWARE ARCHITECTURE

To understand the capabilities of the distributed architecture, both microprocessors and the interconnection possibilities should be investigated. (*Figure 1.)*

3.1 Microblaze and PowerPC processors

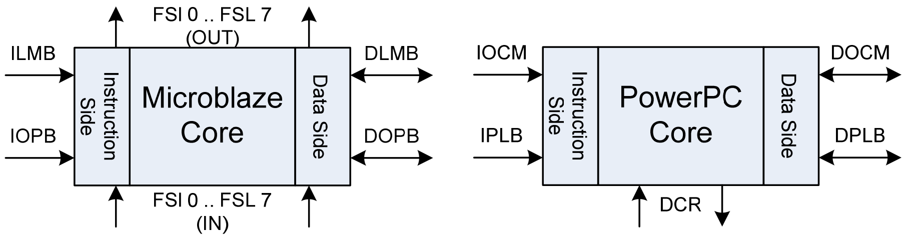

Fig. 1 Microblaze and PowerPC processor cores

Microblaze embedded soft core processor, running RISC instruction set on 32 bit architecture is especially optimized for Xilinx FPGAs. It has separated instruction and data busses. On-chip BRAMs can be interfaced easily over instruction (ILMB) and data side (DLMB) *Local Memory Busses*. Peripherals and memory cores can be connected to the IOPB and DOPB ports. (*On-Chip Peripheral Bus*)

There are eight dedicated FIFO based, bidirectional, point-to-point buses called: FSL (*Fast Simplex Link*) for inter-processor communication. All interfaces are 32 bits wide.

The PowerPC utilises 64 bit architecture with a 32 bit subset. Virtex 4 FX series FPGAs contain one ore two PowerPC 405 cores in hard-macro format. Maximum operating frequency is 450 MHz. OCM (*On-Chip Memory*) buses serve as a dedicated interface between the PowerPC core and BRAMs. Data side bus supports 32 bit bidirectional interface, while instruction side (IOCM) offers unidirectional 64 bit transfer. The two PLB (*Peripheral Local Bus*) can be used to communicate with high speed 32 or 64 bit IP peripheral cores. DCR (*Device Control Register*) bus enables easy control register implementation.

3.3 Interconnect possibilities

Microblaze processors can use their dedicated FSL bus for fast and easy inter-processor communication. Special assembly instructions were introduced for both blocking and non-blocking access. A separate bit indicates whether the transferred 8/16/32 bit wide words are control or data type. FSL bus can be implemented both synchronous and asynchronous using BRAMs or SRL FIFO.

Hence FSL bus was selected for interconnecting the Microblaze processors. FIFO depth can be adjusted (up till ~8000 words), enabling easy communication balancing. A full mesh of eight Microblaze processors can be implemented using the dedicated FSL links. In addition, dedicated hardware accelerators, instruction set extension and peripherals can be connected easily over the FLS busses.

Shared data memory areas can be developed between two Microblaze processors utilizing both ports of the BRAM memories. Multiple processors can share memory over the OPB bus, using ready-to-use OPB arbiter and OPB to BRAM interface cores.

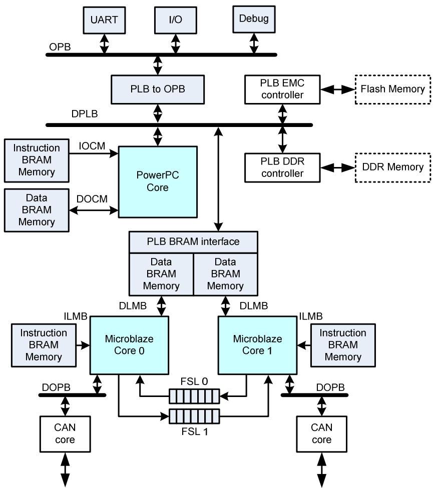

Fig. 2 Implemented system

Figure 2. depicts the implemented system memory and bus structure. The system is designed for embedded automotive protocol analyzer. Each Microblaze processor interconnects to a CAN bus through DOPB bus. Two Microblaze processors run high level message filter and packet assembly functions, meanwhile communicating with each other over the FSL bus. Results can be transferred to the PowerPC processor through the shared memory area that interfaces the entire system to the host computer.

3.4 Implementation results

Tab. 1 Table title

Flip Flops	2,341 of 10,944	**21 %**
4 input LUTs	3,336 of 10,944	**30 %**
BRAMs	32 out of 36	**88 %**

Table 1. shows that the system design bottleneck is caused by the limited number of BRAMs available in the Virtex 4 device. Each Microblaze processors uses four 16kbit BRAM memories on both instruction and data side due to the byte accessibility. At minimum four BRAMs must be used in

each Microblaze design, if data access and instruction fetch shares the same memory. In this case common shared memory area can be implemented on the OPB bus.

Number of used microprocessor cores and internal bus architecture varies project by project. However, implementation results shows that the slice resource permits to implement around 4-6 Microblaze inside the smallest Virtex 4 FX FPGA. (Please note that the synthesis result is calculated without any auxiliary Microblaze peripheral units.)

4 SOFTWARE DEVELOPMENT

Xilinx has special SoPC design software called EDK (*Embedded Development Kit*). It includes many different programs, ranging from Hardware Wizard, Platform Studio, GCC compiler, assembler and debug tools. The user has the possibility to develop programs independently for each microprocessor included in the system.

4.1 Nucleus RTOS

The scalable and configurable nature of Nucleus RTOS provides easy hardware/software codesign. Nucleus Plus, a fully preemptive and scaleable hard-real time operating system, is specially designed for embedded applications, enabled by small (15-20 k) kernel. The Nucleus kernel functions are provided as libraries. In the final image the required kernel functions are linked together with the application code. Preliminary evaluation of Nucleus Plus has been investigated on Virtex FPGAs.

Nucleus RTOS has a wide variety of available middleware libraries. Ready-to use Ethernet packages are available for complete embedded communication, ranging from TCP/IP protocol stack till HTTP Server. USB interface, file-system and graphical libraries are also available.

4.2 Debugging environment

The Embedded Development Kit offers necessary software tools, for both hardware and software debugging. The embedded “logic-analyzer like” ChipScope offers easy interface for signal debugging inside the FPGA over JTAG. The PowerPC processor has embedded JTAG interface for debug purposes which is laced together with the configuration JTAG chain. At maximum 8 Microblaze processors can be debugged with the MDM (*Microblaze Debug Module*) connected to the OPB bus (Figure 2.) and laced also into the JTAG chain.

5 CONCLUSION

In this paper I have presented an evaluated system of a three microprocessor platform implemented on a single Virtex 4 FX FPGA.

The basic idea is to separate the given problem into well defined functional blocks and run them on a dedicated microprocessor. The benefits are the following: The response time of each task is predictable, and independent from the overall system performance. Both hardware and software changes can be implemented easily. Special functional units can be implemented to accelerate the efficiency, and to interface with the external hardware units.

REFERENCES

[1] CLARK, C., NATHUJI, R., AND LEE, H. R. 2005. Using an FPGA as a Prototyping Platform for Multi-core processor Applications, Workshop on Architecture Research using FPGA Platforms, 2005.

[2] http://www.xilinx.com/

[3] http://www.acceleratedtechnology.com/

Reviewer: doc. Ing. Miluše Vítečková, CSc., VŠB – Technical University of Ostrava