SBC System Implemented On FPGA Technology

Laurentiu-Cristian Duca*.

**Politehnica University of Bucharest,*
Automatic Control and Computers Faculty,
Str. Splaiul Independentei no. 313,
060042, Bucharest, Romania (Tel: 40-765-310-869; e-mail:laurentiu.duca@cs.pub.ro).

Abstract: This paper describes a SBC system implemented on FPGA technology. The system is made by processor, a memory, an UART and an unit for performance evaluation, interfaced to a system bus. The processor runs a firmware program which is used to initialize and diagnose the system. The technique used for simulating the system operation like in the real environment is, also, described. The product should be considered as an open source.

1. INTRODUCTION

The current application is part of a project that involves the construction of an embedded system dedicated for optimal implementation of embedded applications. The target is to obtain a high grade of efficiency and low cost in implementing embedded applications with a higher grade of complexity. The system is implemented with reconfigurable hardware circuits and development software distributed with royalty free license. The control logic of the system will be modelled in software by using languages of type HDL and then implemented on FPGA circuits. In this way, will disappear the costs of production between different versions which exists in the case of manufacturing the classic components and the classic system. The general structure of a system dedicated to embedded applications is presented in Fig. 1.

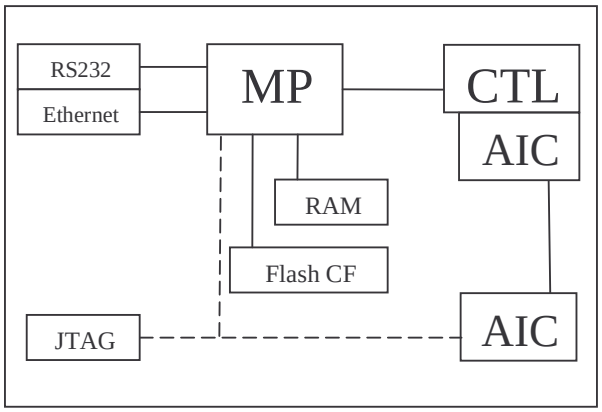

Fig. 1. The general structure of an embedded system

We meet, in function of the complexity of the system, up to three groups of processors, each type with its own functions. From the first group are the processing units dedicated to the intensive computational part of the application (AIC). In the case that is necessarily a high grade of configurability and connectivity with the external medium, it can be utilized a processor dedicated for the activity of administration and management of the system (MP). This kind of applications are meet in the embedded systems which operate in telecommunications, production fabrics and medical industry. The processor that manages the system is utilized for deployment, configuration and control of the applications. When more systems are required to work together, it is utilized a processor which merges the ccommunication tasks with the intensive computational tasks (CTL-AIC). This paper describes the implementation of the principal system without the processors for intensive computational processing.

2. THE HARDWARE ARCHITECTURE

The implemented system is presented in Fig. 2. The intensive computational processors (AIC) are not included in this system.

In this moment, exists many implementations on FPGA of processors of type RISC. The most accessible are the GPL licensed solutions which have a compiler for high level programming. From a study made on Internet I observed that much from the community open-source has attention on the following CPUs: OpenRISC1k, LEON (SPARC) and MIPS.

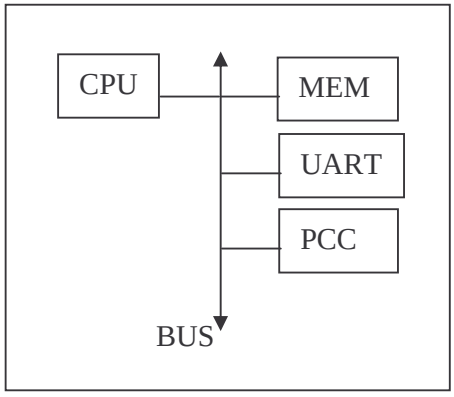

Fig. 2. The hardware architecture

All the components are written in Verilog. The processor is a MIPS R2000 equivalent and implements the MIPS I instruction set architecture. The processor and the other components communicate using a WishBone bus. This standard has license GPL. The processor address the peripheral units using a memory address decoding scheme. The memory unit contains read-only memory and read-write. There is the boot code and its variables. The input/output console will be represented by the UART unit. This unit has the role to connect the system to a personal computer. The performance count unit (PCC) permits the measurement of the execution time for a sequence of program executed by the CPU to perform a computation.
The system bus is the wishbone bus (WB). In the following figures is shown the basics about the wishbone bus. In Fig. 3 are shown the main signals of the wishbone bus. In Fig. 4 is shown the basic read operation.

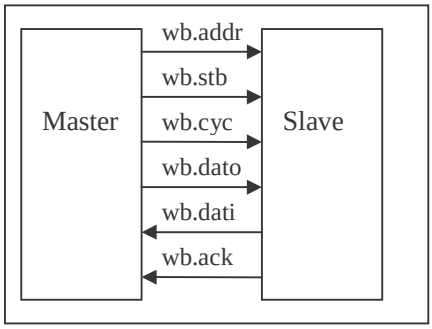

Fig. 3. The main signals of the wishbone bus

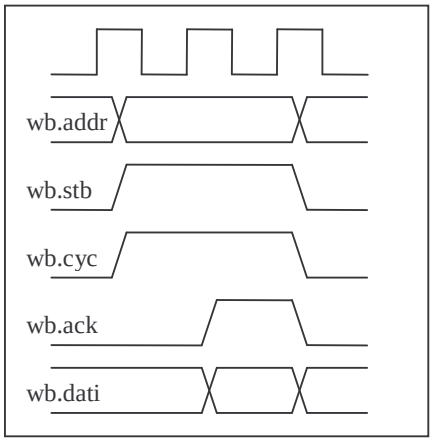

Fig. 4. The basic read operation of the wishbone bus

On a system bus can be present more units of type master and slave. In this situation is necessary an arbitration scheme.
This scheme is presented in the Fig. 5.

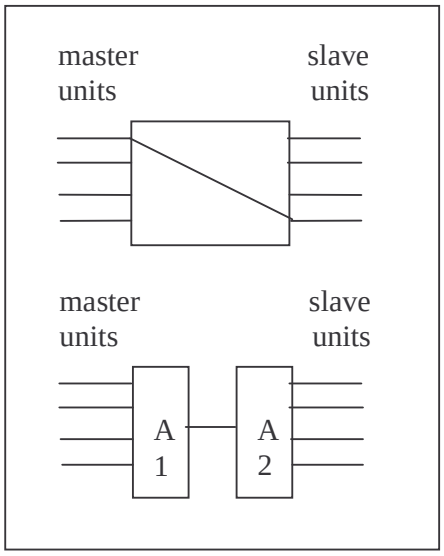

Fig. 5. Interfacing master and slave units on system bus

One single master has access to the bus at a given moment. This master is selected by A1. The master that gains the access from the arbiter will have the signal "slave ack" valid. The other units of type master will have the signal "slave ack" on 0. One single slave can be addressed at a givent time. The slave that will be addressed will have the signals "master stb" and "cyc" on 1. The other slaves have the "stb" signal on 0. The course TSEA02 from the University of Linkoping implements a scheme of this type.
Ucore is a processor implemented at the Chinese Academy of Science. It respects the set of instructions of the MIPS I ISA and the specification of the processor MIPS R2000. It is a processor with 6 levels of pipeline: IF, ID, RF, EX, MEM, WB. The processor had a Harvard architecture and it had a continuous request to read the instruction memory. I had to put a verification such that, at each WB.ack received, the unit that fetches the instructions - to verify if there exists an active request from the data unit and to allow the data unit to access the memory. In this way, the Von Neuman architecture was set.
The processor brings two units of master type at the bus interface: the unit to fetch the instructions and the unit to read and write data. By blocking the signal WB.cyc, a unit of type master can keep the access grant of the bus.
The processor in the native mode is big endian.The skeleton of the system is taken from the course of the Linkopings faculty. Here I have modified the interface of the processor with the bus and I had made the system little-endian.
The memory is implemented with a unit of memory with access at the bus. The processor accesses the memory only by the system bus. The disadvantage is made by the fact that, this way the access at the memory takes two clock cycles (stb+ack). The pipeline unit is blocked during the memory access. The arbitration of the bus delays the pipeline with another period of clock.
The serial communication of the system UART is taken from Jeung Jong Lee [3]. The sources had some problems. The receiver in the initial state was "ready" and signalizing the event of receiving a character. For clock frequencies of 25

Mhz, the unit was functioning at frequencies less that 57600 bps. Due to the fact that the length of a bit of data was wanted to be divided in 16 parts, but the division didn't care about the rest. The autor was considering that the system clock had a frequency much greater then the frequency of transmission. At this unit I have wrote the interface to the WB bus.
The unit for immediate testing permits the processor to access two system ports, both of 32 bits. In the current implementation, I have linked one port to 8 buttons and the other to 8 leds. And, next to this unit, I have put a timer which can be utilized for measuring the performance of the system.

3. THE SOFTWARE SYSTEM

The base software system is made in two parts. The first part is the compiler, and the second is the firmware
that runs on the embedded system.
The development environment is cygwin and gcc. Files of type Makefile are used for compiling the source code
and introduce it in the file ".bit" which will be loaded onto the FPGA. For synthesis and implementation of the hardware sources it is used the development system Xilinx ISE Webpack 9.1. This includes a tool named "data2mem" which can be used for translating an executable file from ".elf" format in that part of the memory that is utilized by the processor implemented on the FPGA. This portion takes a specific place from the ".bit" file which is told in the file of constraints.
The firmware is written in C. The compiler used is mips-gcc. This was compiled on the cygwin suct that it can run on Windows, by using the program cygwin.The compiler will generate object code for the processor R2000
having the set of instructions MIPS I ISA.
The firmware is used for starting the system and contains the routines for initialization and access to resources. At system start, the program counter has an initial value where it should be placed the first instruction. Then it must be initialized the registers SP and GP. Also, it must initialized the table of interrupts. Having stack and global data, can jump to the function main() placed in the zone of code. To the compiler it is told the zones in which must be placed the segments of data and code, in a file ".ld" used by the link-editor.
Because we work in firmware, the functions that access the console like printf, and those who handle text are wrote in a simplified manner and have characteristics dedicated to the platform. The unit UART is used for the interface with the user. The last step is to give the control to the program that is dedicated for the utilisation of the system.

4. STANDARD SIMULATION

The standard simulation of the system is made by supervising the diagram of transitions of the signals necessary to verify the basis functionality of the system.
The first step in testing the system is to verify that in simulation the system starts correctly and loads the first instruction from the memory. The start sequence of the system is shown in Fig. 6.

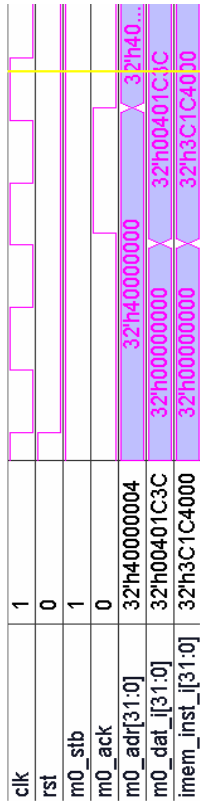

Fig. 6. The start sequence of the system

The boot address for the processor is 0x40000000. The unit that reads the instructions is one of the two masters that are attached on the system bus. This unit makes a request to access the system bus by setting the signal m0.stb on 1. And will set the address of the desired instruction on the bus by using the signal m0.addr. The data that comes from the memory driver is transmitted to the processor via the system bus. It is used a scheme that converts from little endian to big endian. This scheme is necessary because the system is little
endian and the processor is big endian. In the final, the processor has access to the instruction, which is placed in the internal signal imem_data. In the figure it can be observed that the program counter (m0.addr) is incremented and points to the next instruction. The code of the first instruction is 0x3C1C4000 and represents the instruction "lui gp, 0x4000". The next instruction is "ori gp,gp,0x5600" and is represented by the code 0x379C5600. As it was presented in the chapter referred to the software of the system, first are initialized the registers GP and SP.

5. THE EQUIVALENT SIMULATOR

This system runs in a real environment on a board with FPGA circuits. In reality, any change of the hardware sources implies the need to synthesize and route the design on the FPGA - to observe the behaviour of the system. In simulation, the advantage is that there are no processes for synthesis and routing the system on the FPGA. The code is synthesizable, so we can suppose that a correct simulator would have the same results like in reality. The disadvantage

of the simulator is that it works much slower, the speed of the simulation is with two orders of magnitude slower. That is why the simple components can be only modelated behaviouraly for wining time.
When the time to run the program on the processor is long in reality, the dimension of the database with the events of the simulation is very large.We will attach few signals or none in such a simulation, because we don't need to. The problem of functionality of the system has been solved at the begining, when we verified if the program counter and the instructions are correctly executed.
Now we solve the problem to simulate the hole system. In this sense, the interaction between the real user and the system is simulated by using a virtual console. See Fig. 7.

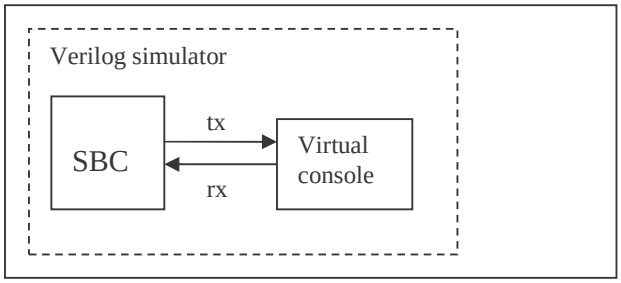

Fig. 7. The equivalent simulator

The virtual console is implemented using some functionalities of Verilog 2001. For the user, the simulation is transparently. He sees only that the speed of work is slower. In this way, the display of the messages on the screen are solved by using a mechanism of deserialising from the serial line the characters that are sent and display
them using the instruction "$display()". The commands from the user to the system are taken by using a modality of work with the file system. The user writes in a file the command in the same manner as the console and in simulation it is read the content of this file and whenever it changes, it will be sent the command on the serial line.

6. PERFORMANCE

This system is intended for having good configurability and access to the hardware implemented part of the intensive computational units. Those units bring up the real performance of an embedded system.
To compute de clock per instruction indicator, I choose a representative sequence of instructions to be run on the system. The measurement was done by using a hardware counter to count the clock periods elapsed from a given moment of time and another to count the number of instructions executed. These counters are accessible for reading and resetting by the processor at a given address of memory.
The code sequence generates the Fibonacci terms and for each two consecutive terms, computes the biggest divisor. The are no integer division instruction used. The counter measurement is presented in Table 1.

Table 1

Number of instructions	Number of time periods	Computed CPI
1754	8313	4.74

The memory access unit introduce delays in pipeline of 2 clock periods. Each time the instruction unit and data unit of the processor changes ownership on the bus, the pipeline is blocked another clock period. Considering this and the performance measurement, the processor pipeline has an efficiency of about 1.5-2 CPI. The system performances would be considerable improved by attaching a cache memory near the processor.

7. CONCLUSION

This paper presents an SBC system realised by using and porting the main components from open source architectures. The performance of the system is lower than the systems present on the market by means of clock per instruction and the fact that FPGA porting implies a clock frequency lower than then using an ASIC chip.
The advantage is that the entire system, hardware and software is contained at source level. In this way, the performances can always be improved and debugging is more facile.
The implementation of the system does not depend on the market availability of dedicated ASIC and memory chips. The HDL sources can be recompiled at any moment on a newer FPGA circuit.
With simple and usefull technics from Verilog 2001, this system can be simulated in a similar way with its real functionality.

7. REFERENCES

Bachle R., Building the gcc toolchain for the MIPS processor, www.linux-mips.org/wiki/Toolchains

Jeung J.L., *UART implementation in Verilog*, www.cmoexod.com

Petrescu A., *Course on Digital Computers*, www. csit-sun.pub.ro

Segger O. et al., *Course on Computer Hardware*, www.da.isy.liu.se/courses/tsea44/coursemtrl_07

Zhang F., *The ucore processor*, www.opencores.org/projects.cgi/web/ucore/overview