CPLD Based Simple Microprocessor for Data Acquisition

W.A.S. Wijesinghe[1], M.K. Jayananda[2] and D.U.J. Sonnadara[2]
[1] *Department of Electronics, Wayamba University of Sri Lanka*
[2] *Department of Physics, University of Colombo, Colombo 00300, Sri Lanka*

ABSTRACT

This paper describes a simple microprocessor developed using a Complex Programmable Logic Device (CPLD), with an instruction set optimized for data acquisition applications. Most low-cost data acquisition systems are built using general purpose microcontrollers with large instruction sets. However, only a very small number of instructions are actually needed in data acquisition applications. Therefore, with the aim of minimizing the cost and lowering the overheads, a simple microprocessor was developed with a tiny instruction set containing only the instructions required in the data acquisition applications. Due to the optimization of the features, it was possible to fit both the CPU and the program memory in the 36 macrocell Xilinx XC9536XL CPLD costing approximately RS. 500. The design of the CPU was carried out using the hardware description language VHDL. Reconfigurability of the CPLD using VHDL makes it possible to change the features of the CPU, including the instruction set, to suit the user requirements. An example data acquisition system implemented using this CPU is also described in this paper.

1. INTRODUCTION

Computerized Data Acquisition Systems are often employed in modern research activities. However, most commercially available data acquisition systems are quite expensive for local applications. Therefore, many Sri Lankan researchers have resorted to developing their own low-cost data acquisitions systems using microcontrollers. This paper describes an attempt to reduce the cost of such data acquisition systems even further while increasing the flexibility, by developing a microprocessor with minimum features required in such applications.

The development of complex digital circuits such as microprocessors without sophisticated chip manufacturing facilities has been made possible with the advent of Programmable Logic Devices (PLD). Among the various types of PLDs, Complex Programmable Logic Devices (CPLD) and Field Programmable Gate Arrays (FPGA) are the ones most suitable for digital circuits with large number of logic gates.

2. THE CPLD

A Programmable Logic Device [1] is a device with configurable logic and flip-flops which can be interconnected by the user. The basic building block of a CPLD is a macrocell. Each macrocell contains a user programmable combinatorial logic function and a flip-flop. A group of macrocells are arrange as a single configurable function block. The outputs of the macrocells are available through I/O pins of the CPLD. In addition they are fed to a global switch matrix which interconnects the macrocell outputs to macrocell inputs of all the function blocks. By programming the individual logic function of each macrocell and configuring the interconnections through the switch matrix, the user can build very complex multilevel logic functions.

For example the XC9536XL used in this work consists of two configurable function blocks, each containing 18 macrocells [2]. A total of 36 external I/O pins and 36 inputs from the two function blocks (i.e. outputs from the 36 macrocells) are connected to the switch matrix while each function block receives 54 inputs from the switch matrix. The macrocells and the switch matrix can be programmed by downloading a bit pattern through the JTAG port to the in-system programming controller.

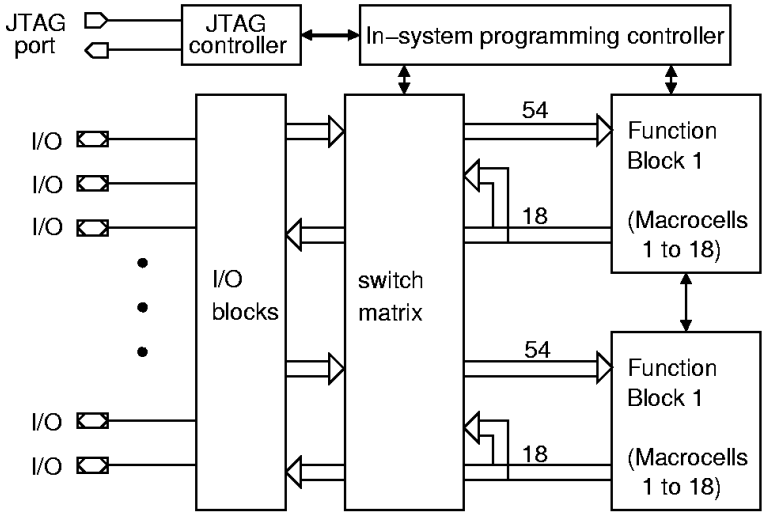


Figure 1: A simplified block diagram of the XC9536XL.

A variety of tools are available for CPLD design. They can be broadly categorized into two groups as schematic-based tools and hardware description language based tools. Available hardware description languages include VHDL, Verilog and ABEL. It is also possible to design systems by mixing the two approaches.

In the work described in this paper, VHDL approach was used. The design was carried out using ISE WebPACK software available free of charge from Xilinx [3]. The configuration of the device was done by downloading the binary file generated by Webpack through a Xilinx parallel port cable connected to the JTAG port of the CPLD.

3. THE CPU ARCHITECTURE

In most of the low-cost data acquisition systems, the main function of the microprocessor or microcontroller is to read data from one or more ADC channels and send them to a computer or store data in an on-board memory for retrieval later. Usually most of the instructions available in a typical microcontoller are not required for such applications. The most important instructions required are memory read and memory write. With memory mapped I/O, the same instructions can be used for accessing ADC and DAC channels.

As the basis for a very simple processor with a tiny instruction set, the VHDL implementation of an 8-bit minimal CPU described by Bocke [4] was adopted.

Implementation of the Instruction set:

In the initial design, the instructions were encoded using two bits, which made the maximum size of the instruction set to be four. However, only three instructions were actually required. The mnemonics, opcodes and descriptions of the implemented instructions are shown in Table 1.

Table 1: Instruction Set

Mnemonic	Opcode	Descriptions
RD	00	Read 8-bit word from the ADC and store it in Data Register
WR	01	Send the value in Data Register to the memory module
JP	10	Set PC to 0

Internal Registers:

The register set of the CPU is listed in Table 2 and the block diagram in Figure 2 shows the data flow within the CPU. The program counter PC holds the address of the next instruction to be fetched. Just before an instruction is fetched, the content of PC is transferred to the address register AR and PC is incremented by one. After fetching, the instruction is loaded to the instruction register IR. Data read from memory (by RD instruction) or data to be written to memory (by WR instruction) is stored in DR. The register MA contains the address of the next free location of data memory.

The operation of the CPU is controlled by a simple state machine consisting of three states (see Figure 3). The three states represent fetching, decoding and execution phases of the instructions.

Table 2: Register Set

Register	Description
AR	2-bit Address Register
DR	8-bit Data Register holds the 8-bit data
IR	2-bit Instruction Register holds the current instruction being executed
PC	2-bit Program Counter
ST	2-bit to hold the state of the process
MA	6-bit to hold the address of the next data memory location

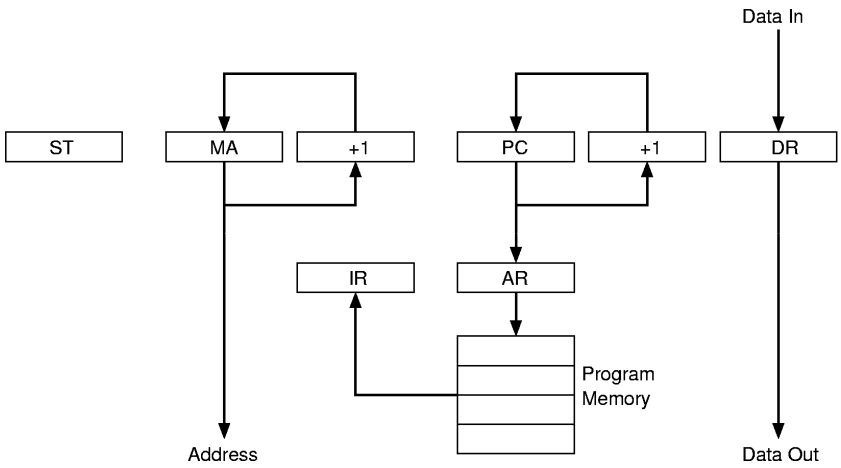


Figure 2: The register set of the CPU.

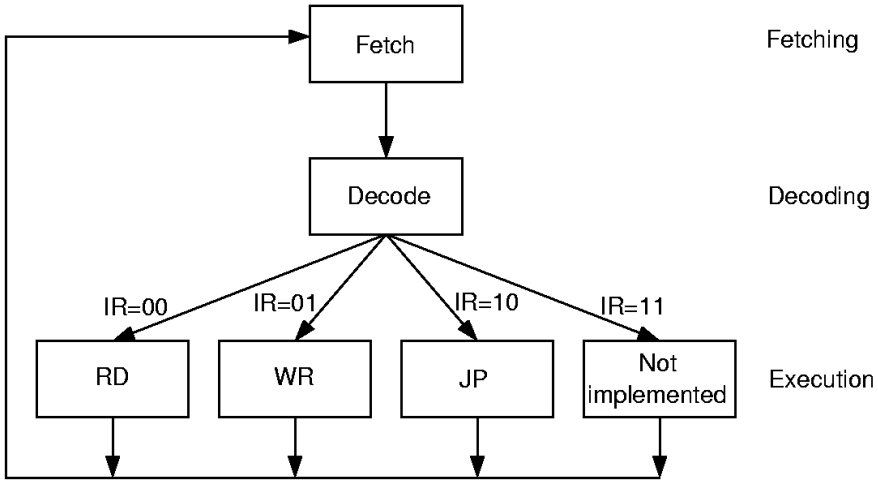


Figure 3: The operation of the CPU as a state diagram.

Table 3: State registers

State	Function	Operation
00	Fetch	**AR** ← **PC**
01	Decode	**PC** ← **PC** + 1
		IR ← Memory[address]
10	Execute the instruction:	
	if IR = 00 read data from ADC	**DR** ← Data In
	if IR = 01 write data into memory	Data Out ← **DR**
	if IR =10 set PC to zero	MA ← MA + 1
	(IR = 11 not implemented)	**PC** ← 00

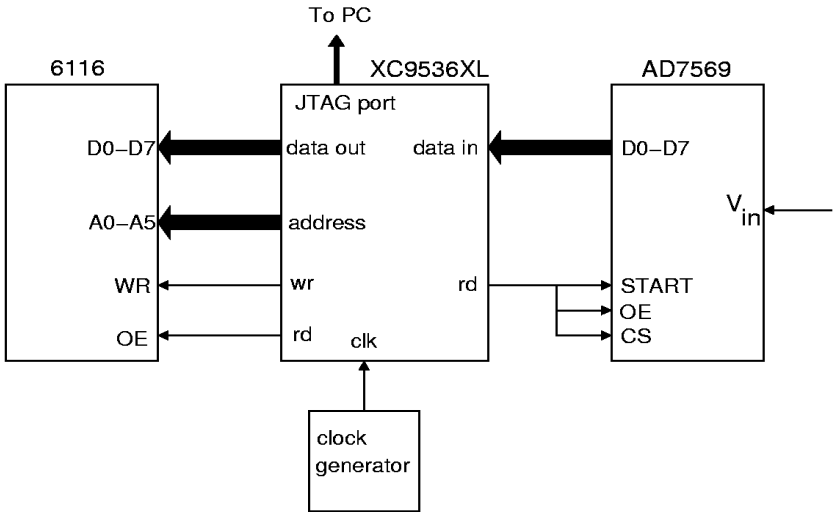
The implementation of the CPU required only a small program memory. In the initial design, the size of the memory was limited to four two-bit words. The incorporation of memory inside reduces the external device count and increases the speed of program execution. Although this constitutes to hard-wiring the program in to the CPU, there is no loss of flexibility because the CPLD can be easily reconfigured.

4. DATA ACQUISITION SYSTEM

The above described CPU was incorporated in a prototype data acquisition system for evaluating the performance. The data acquisition system comprises of three basic components – the CPU, a memory module and an Analog to Digital Converter (ADC). A block diagram of the data acquisition system is shown in Figure 4. The program stored in the program memory of the CPU repeatedly reads the ADC and stores the data in the memory module. The ADC is configured so that each read operation initiates a new conversion. Each time a data word has been stored in the memory module, the memory address register is incremented by 1. The program used for this process consisted of just three instructions: RD, WR and JP.

A 6116 SRAM chip which contains a 2k×8 memory arrangement was used for the memory module. The 6116 has eight I/O lines and 11 address lines. As shown in the Figure 3, all 8 data lines and 5 of the address lines are connected to the CPU. Chip Select ($\overline{CS}$), Write Enable ($\overline{WE}$), and Output Enable ($\overline{OE}$) pins are also connected to the CPU.

The 8-bit AD7569 Analog to Digital converter was used in an infinite sample-and-hold configuration [6]. The sample signal is sent by the CPU. It drives the $\overline{CS}$, $\overline{RD}$ and inputs of the AD7569 . Although the CPU was capable of reading data at about 1 MHz rate, the sampling rate of the AD7569 is limited to about 500 kHz. Therefore, to ensure that the ADC has completed before reading data, a delay of 3 ms was incorporated into the



RD instruction. This brought the sampling rate of the ADC to approximately 333 kHz.

Figure 4: Block diagram of the data acquisition system

This data acquisition system can function as a standalone device and record data until the memory becomes full. For transferring the saved data to a PC, the JTAG boundary scan (IEEE standard 1149) [7] available in the XC9536XL was used. The main purpose of JTAG boundary scan is to provide a convenient method of diagnosing problems in complex circuits. Chips that support this standard allow the isolation of the core of the chip from its pins and setting values to output pins and reading the status of input pins using commands sent to the chip through the JTAG port.

Making use of this facility, the data transfer to the PC was carried out by connecting the JTAG port to the parallel port of a PC using the same cable utilized for configuring the CPLD. Each word stored in the 6116 SRAM was read to the PC using the boundary scan commands.

5. CONCLUSION

The CPU described in this paper provides a cheap alternative to general purpose microcontrollers for low cost simple data acquisition systems. Even the simplest microcontrollers contain many features that are not used in simple data acquisition applications. The CPU described above has only the bare minimum features required in such applications.

The incorporation of the program memory into the CPU itself was another important feature of this work. The stored program concept introduced in very early stages of the development of computers vastly increased the versatility of computers. Although, at first glance, hardwiring of the program into the CPU would appear as a step backward, the abandoning the stored program concept does not affect the versatility of CPLD based CPUs because they can be reconfigured very easily by downloading a bit pattern. The reconfiguration of a CPLD is almost identical to the process of downloading a program to a microcontroller based system.

In addition to the applications in data acquisition systems, the CPU described in this work can be very useful for teaching purposes. The lack of complex features makes it very easy to understand the basic operation of a CPU. Ease of reconfiguration allows students to learn lessons computer architecture by changing various features of the CPU.

The major advantage of CPLD based design is the possibility of reconfiguring the CPU to suit user requirements. Features such as number of registers, their size and size of program memory can be easily changed to suit different applications. The implementation of the CPU described in this paper did not require all the resources available in the XC9536XL, as shown in Table 4. The remaining resources can be used for adding more features to the CPU. If more than the remaining amount of resources is necessary for adding a new feature, the total gate count can be doubled by removing the XC9536XL from its socket and inserting a XC9572XL, without any other hardware change.

Table 4: Resource utilization of the XC9536XL.

Macrocells Used	Registers Used	I/O Pins Used	Function Block Inputs Used
33/36 (92%)	29/36 (81%)	**27/34 (80%)**	50/108 (47%)

In the prototype data acquisition card described here, advantage was taken of the JTAG boundary scan support available in the XC9536XL for transferring data to a PC. The alternative to this, would have been the implementation of a serial port, resulting in much higher resource requirements for the CPLD.

REFERENCES

1. David Van den Bout, *The Practical Xilinx Designer Lab Book*, Version 1.5, Prentice Hall, 1999.
2. *Xilinx Data sheet for FastFLASH XC9500XL High Performance CPLD Family*, Version 1.1, 1998.
3. http://www.xilinx.com/products/design_resources/design_tool/index.htm.
4. Bocke T., *A minimal 8Bit CPU in a 32 Macrocell CPLD*, http://www.tu-harburg.de/~setb0209/cpu/files/mcpu-doc.pdf
5. Carpinelli J.D, *Computer Systems Organization and Architecture*, Addison Wesley, 2000.
6. Byme M., *Implement Infinite Sample-and-Hold Circuits using Analog input/output Ports*, Analog Devices, Application Note, AN284
7. Nick Patavalis, *A Brief Introduction to the JTAG Boundary Scan Interface*, http://www.inaccessnetworks.com/projects/ianjtag/jtag-intro/jtag-intro.html.