WP167 (v1.0) December 10, 2002

Field Programmable Controllers for Cost Sensitive Applications

By: Richard Griffin

Staying ahead of the competition is getting tougher everyday. Cost pressures, changing standards, and device obsolescence are just a few of the challenges. To stay ahead, you need a low cost, competitive processing solution that's customizable throughout the entire design cycle and can quickly be brought into high-volume production.

The Xilinx Field Programmable Controller (FPC) solution allows you to create low-cost, customized processors with peripherals, memory, and logic — all on a single cost-optimized Spartan™-IIE FPGA. The FPC solution is ideal for applications in which cost and integration within a system is critical. With the flexibility to allow integration of other IP on the FPGA fabric, the Spartan-IIE family presents an ideal embedded solution. This white paper presents the end markets, FPC solution, and its associated tools, end applications, and the Spartan-IIE performance advantage.

Introduction

The introduction of embedded soft processors has offered substantial benefits to the world of digital electronics. The world's huge appetite for increasingly intelligent and sophisticated control systems has dictated the rapid advancement of processor technology. Perhaps most prevalent of all is the huge leap in demand of embedded microcontrollers. These processor-based devices can now be found in a colossal number of modern electronic devices. Microprocessors and microcontrollers are frequently confused; indeed the two terms are often incorrectly equated. For the purposes of this paper, it is first necessary to clearly define the two terms and establish some important yet subtle differences.

Microprocessors

Microprocessors are single chip devices containing typically the core processor technology. This includes the execution units, the register file, program counter, memory interface, interrupt controller, and in some higher performance examples, the cache units and a larger peripheral set. In order to correctly function, the microprocessor requires a plethora of external components. Blocks of RAM and ROM are typical examples whether they exist to store executable code or provide a scratchpad memory area. Other examples include input / output (I/O) ports, timers, and serial communication ports. Figure 1 shows the typical layout of a microprocessor system.

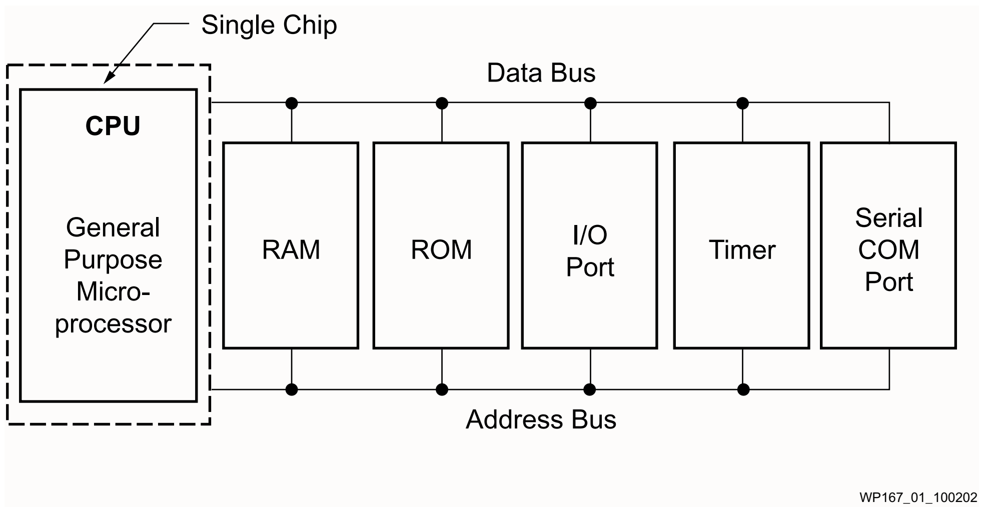


Figure 1: **A Typical Microprocessor System**

Microcontrollers

Microcontrollers, on the other hand, offer a considerably more integrated solution. Rather than enforcing a building-block platform upon the user, a microcontroller system is comprised of blocks within the boundaries of the device package. The CPU, RAM, ROM, I/Os, and peripherals are all contained within the device, closely integrated using localized internal connections. The inputs and outputs of the microcontroller system are coupled to the other hardware blocks within the design via the pins on the microcontroller device. Microcontrollers present the designer with a solution that is far easier and faster to use. Figure 2 shows a microcontroller implementation. A single device prevents the user from making any mistakes connecting the CPU to the memory and other peripherals, while affording them rapid development time and ease of implementation. In essence, a microcontroller provides a solution to which only I/O signals need be connected and executable code be written. The flexibility of the microcontroller is reduced when compared to the use of a full microprocessor, but this is quite often an acceptable loss for the application in question. The primary issue of importance is one of cost per unit, which often sways the decision in favor of the microcontroller solution.

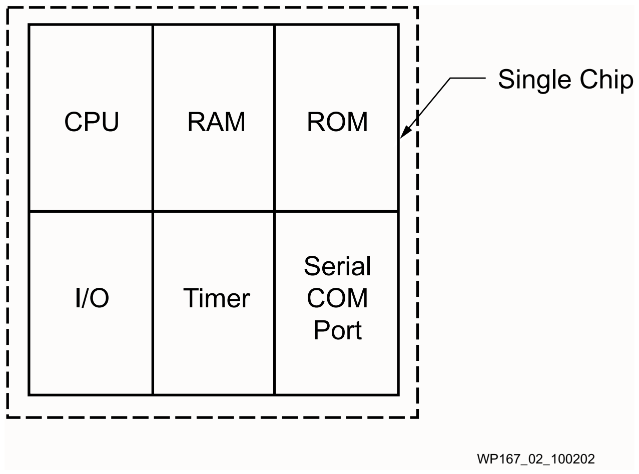


Figure 2: **Microcontroller**

End Markets

Microcontrollers are used extensively in three major market areas: automotive, industrial, and consumer products. The modern automobile is replete with microcontroller-based systems providing automated control for just about every conceivable part of the car. Braking systems use microcontrollers to deliver advanced safety features like ABS and traction control to the driver. Windshield wipers are controlled to bring us timed interval wipes and even rain sensitive automatic wiper activation. Heating controls for the vehicle interior are frequently found monitoring multiple zones within the passenger compartment, adjusting the supply of air to the interior to maintain a user chosen ideal temperature. Seats even adjust electrically to remember the favored positions for different drivers of the car! All of this technological wizardry is achieved and implemented under the watchful eyes of a collection of microcontrollers.

The industrial world has been a changing place in recent years. Gone are the days of a large workforce all trained to monitor a specific area of a production plant. They have now been replaced by a series of microcontroller based monitoring modules, often linked to a central station, which is monitored by a supervisor. The reduction in cost in microcontroller technology has permitted these modules to replace their human counterparts who were ultimately prone to lapses of concentration and the potential dangers of human error. Microcontrollers can work tirelessly around the clock without lapses in concentration, requiring only the most minimal maintenance and supervision.

The consumer market is also the home of a heavy concentration of microcontrollers. Walk into any electronic store and you will be hard pushed to find a product that does not have some kind of microcontroller lurking beneath its covers: video recorders, televisions, dishwashers, video games, set-top boxes, refrigerators, washing machines, remotely controlled lighting dimmers, telephones, answering machines, ovens, toasters, printers, scanners, even children's toys are microcontroller equipped. So huge is the demand that microcontrollers are now outselling the more conventional microprocessor by around six units to one as shown in Figure 3. By the year 2005, it is expected that over eight thousand million microcontroller units will be shipped in electronic products around the world. Microcontrollers are here to stay, it is therefore important that they are able to meet our ever increasing demands for the future. With these quantities in mind, the cost of microcontrollers becomes a very big issue indeed. The only way to keep up with the demands on lower costs is to incorporate the microcontroller into existing hardware devices to form system-on-a-chip platforms. For this to succeed, a flexible platform is needed into which the microcontroller can be embedded.

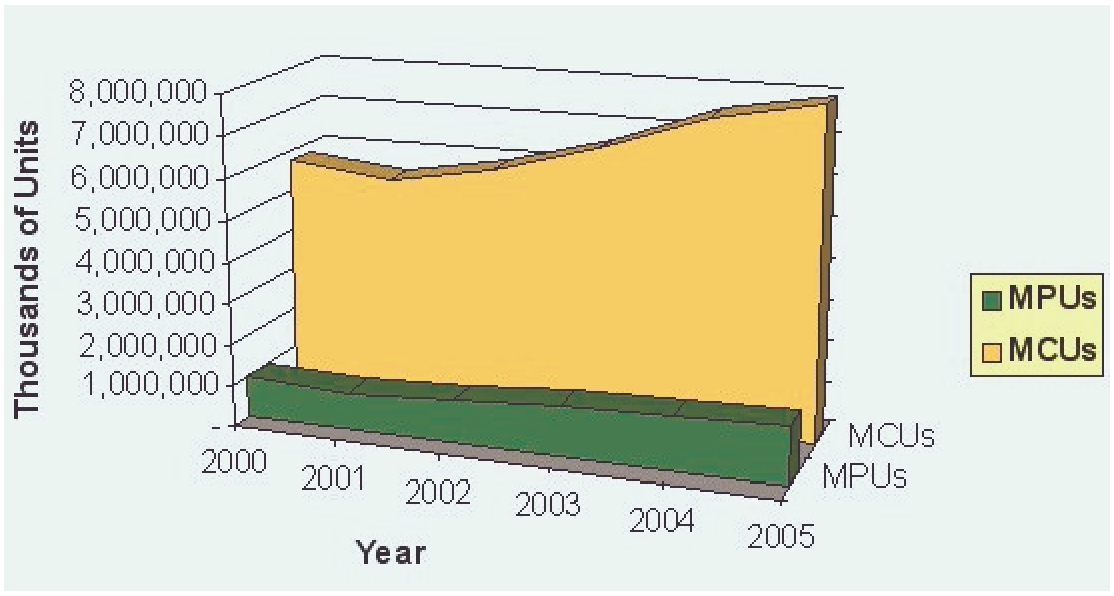


Figure 3: **Worldwide Microcomponent Unit Shipment Forecast**
Source: Gartner Dataquest (January 2002)

The FPC Solution

The Spartan-IIE family of FPGAs have a regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), surrounded by a perimeter of programmable Input / Output Blocks (IOBs). Two columns of block RAM lie on opposite sides of the die between the CLBs and the IOB columns. A powerful hierarchy of versatile routing channels interconnects these functional elements. Spartan-IIE FPGAs are customized by loading configuration data into internal static memory cells while permitting unlimited reprogramming cycles to become a viable upgrade path for future product enhancements. Therefore, Spartan-IIE FPGAs are ideal for shortening product development cycles while offering a cost-effective solution for high-volume production. Spartan-IIE FPGAs achieve high-performance, low-cost operation through advanced architecture and semiconductor technology. Spartan-IIE devices provide system clock rates beyond 200 MHz and offer the most cost-effective solution while maintaining leading edge performance. In addition to the conventional benefits of high-volume programmable logic solutions, Spartan-IIE FPGAs also offer on-chip synchronous single-port and dual-port RAM (block and distributed form), DLL clock drivers, programmable set and reset on all flip-flops, fast carry logic, and many other features. Figure 4 shows a block diagram of the Spartan-IIE FPGA device.

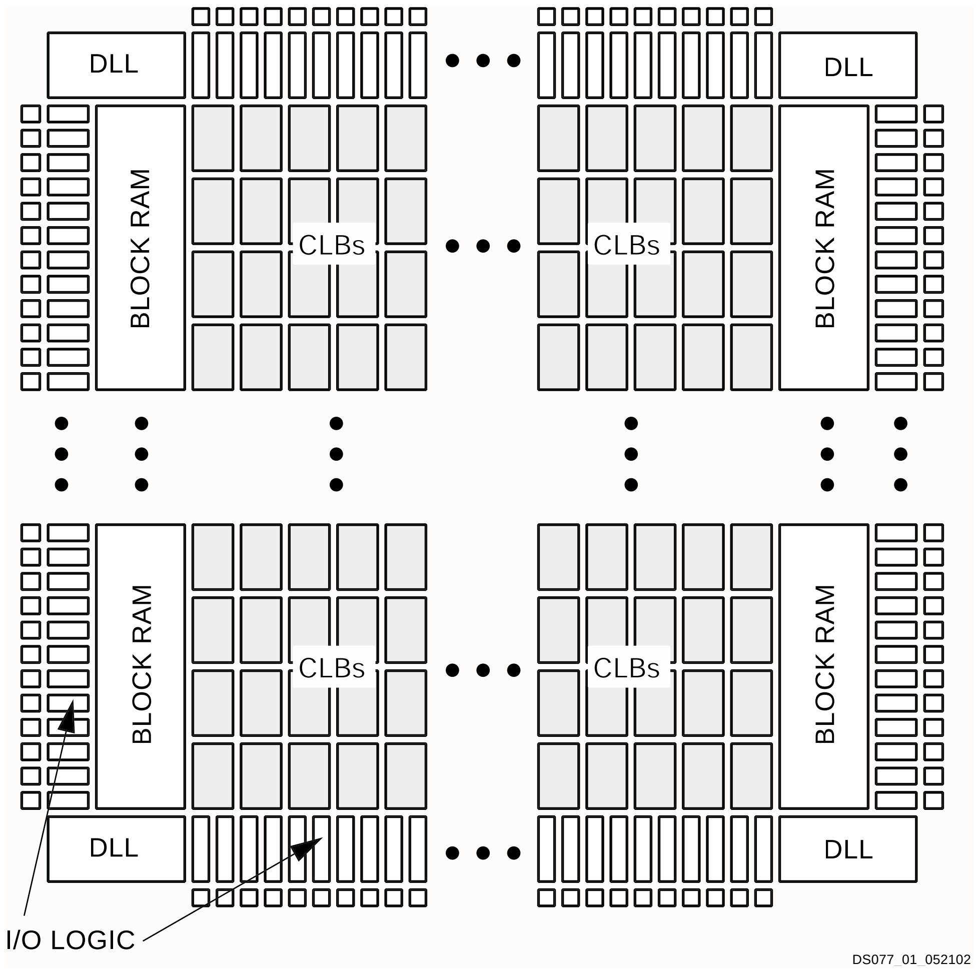


Figure 4: **Spartan-IIE FPGA Block Diagram**

This flexible platform is an ideal base upon which to implement a microcontroller system. We previously looked at the way in which microprocessor systems and their peripherals and memory were all integrated onto one device, providing huge cost savings to many modern electronics products. An *embedded* microcontroller takes the concept of integration one stage further by permitting the designer to embed the microcontroller system into a small section of a programmable device. No longer does the microcontroller have to exist in a stand-alone package; it can now be embedded deep within custom hardware. This technology offers colossal advantages to the designer in terms of functionality, cost, performance, circuit board area, and most importantly, flexibility. The MicroBlaze soft processor core has taken this concept and pushed back the boundaries of what can be achieved in a programmable logic device.

The introduction of customized soft processor systems for Field Programmable Gate Arrays (FPGAs) has offered huge flexibility, but with this flexibility comes some new challenges for the designer. Traditionally the designer has approached the task of processor selection by comparing the needs of their system specification to the features listed on the processor datasheet. While this may often sometimes be a trivial task, there are times when unusual processor configurations are desired by the system specification.

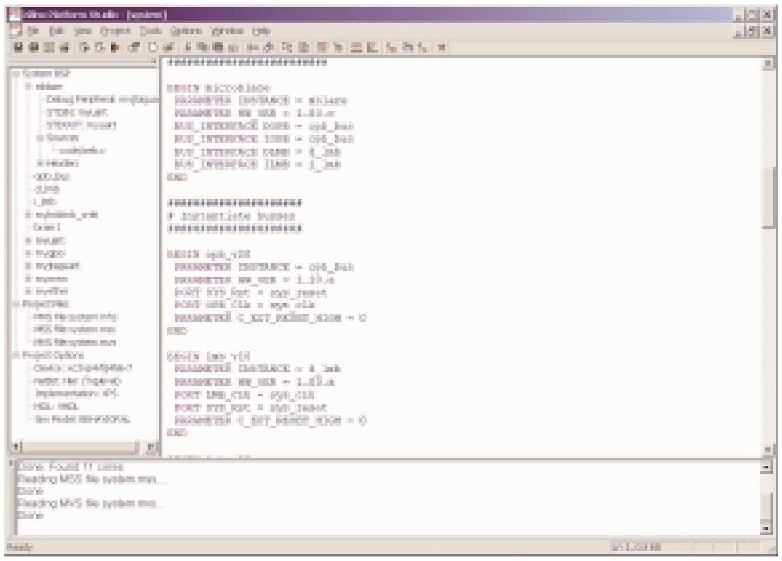
For example, the designer may desire a processor with 10 UARTs, an interrupt controller, and access to a block of external FLASH. Although many off-the-shelf processors offer multiple UARTs and the other desired peripherals, they would typically be of sufficient complexity to have numerous other peripherals that would be unused in this system. Not only is the designer paying for the additional peripherals, it is often necessary that unused peripherals in this type of processor have to be placed into a safe mode or otherwise disabled via software.

An additional burden now exists on the software design team. Not only do they have to make the used processor peripherals operate correctly; they also have to write code for the parts of the processor which are not being used. It is clear that purchasing an off the shelf solution for this scenario would be highly wasteful not only in terms of initial cost, but also in wasted engineering time during the design process.

With the MicroBlaze soft processor, the designer has the luxury of a different approach. They can now start with a processor core and build the peripheral set to meet their exact requirements. Silicon wastage is reduced to zero since the designer will only implement what they need. Software design complexity is reduced because no code needs to be written to disable unwanted processor functionality. The creation of unusual processor configurations, which can be changed at any time to suit changes in the specification, is reduced to a simple task.

Customized processors, from their very name, requires that someone performs the customization! This is where design automation tools and intellectual property play a key role (Figure 5). The processor core is placed at the heart of the system, and the required peripherals are then added to the system from a catalog of Intellectual Property (IP) cores. As each block of IP is added, the customization process deepens by allowing the user to select the behavior and functionality of each peripheral. UARTs can be configured to operate at the correct baud rate, communicate using the desired number of data and stop bits, and employ the required parity checking. External memory controllers can be customized to insert sufficient wait states for correct and efficient memory device access; multiple (independently configurable) banks of memory are supported from a single controller giving the designer access to SRAM, FLASH, and EPROM memory. Interrupt controllers can be configured to respond to rising or falling edge inputs, or indeed to adopt a level triggered response.

Bus structures are added to connect the entire system, which are again configurable to meet the needs of system clock speed or silicon area.



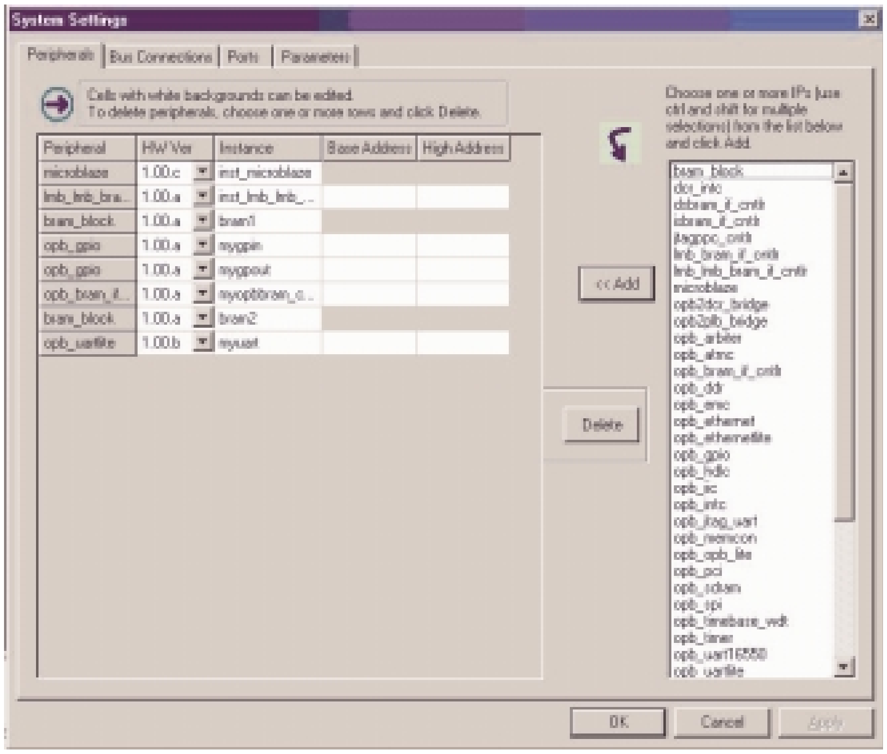
```
########################

BEGIN microblaze
 PARAMETER INSTANCE = mblaze
 PARAMETER HW_VER = 1.00.c
 BUS_INTERFACE DOPB = opb_bus
 BUS_INTERFACE IOPB = opb_bus
 BUS_INTERFACE DLMB = d_lmb
 BUS_INTERFACE ILMB = i_lmb
END

####################
# Instantiate buses
####################

BEGIN opb_v20
 PARAMETER INSTANCE = opb_bus
 PARAMETER HW_VER = 1.10.a
 PORT SYS_Rst = sys_reset
 PORT OPB_Clk = sys_clk
 PARAMETER C_EXT_RESET_HIGH = 0
END

BEGIN lmb_v10
 PARAMETER INSTANCE = d_lmb
 PARAMETER HW_VER = 1.00.a
 PORT LMB_Clk = sys_clk
 PORT SYS_Rst = sys_reset
 PARAMETER C_EXT_RESET_HIGH = 0
END
```

```
Done. Found 11 cores.
Reading MSS file system.mss...
Done
Reading MVS file system.mvs...
Done
```

Figure 5: **Xilinx Platform Studio (XPS) Tool**

Adding System Level Parameters

Once the basic structure of the processor system is in place, the designer can enter phase 2 of the design process and begin to allocate system level parameters to the processor design (Figure 6). These include the desired address map for the processor system, the selection of interrupt priorities, the allocation of the standard input and standard output devices from the included peripheral set, and so on.



Cells with white backgrounds can be edited.
To delete peripherals, choose one or more rows and click Delete.

Peripheral	HW Ver	Instance	Base Address	High Address
microblaze	1.00.c	inst_microblaze		
lmb_lmb_bra...	1.00.a	inst_lmb_lmb_...		
bram_block	1.00.a	bram1		
opb_gpio	1.00.a	mygpin		
opb_gpio	1.00.a	mygpout		
opb_bram_if...	1.00.a	myopbbram_c...		
bram_block	1.00.a	bram2		
opb_uartlite	1.00.b	myuart		

Choose one or more IPs (use ctrl and shift for multiple selections) from the list below and click Add.

bram_block
dcr_intc
dsbram_if_cntlr
isbram_if_cntlr
jtagppc_cntlr
lmb_bram_if_cntlr
lmb_lmb_bram_if_cntlr
microblaze
opb2dcr_bridge
opb2plb_bridge
opb_arbiter
opb_atmc
opb_bram_if_cntlr
opb_ddr
opb_emc
opb_ethernet
opb_ethernetlite
opb_gpio
opb_hdlc
opb_iic
opb_intc
opb_jtag_uart
opb_memcon
opb_opb_lite
opb_pci
opb_sdram
opb_spi
opb_timebase_wdt
opb_timer
opb_uart16550
opb_uartlite

Figure 6: **System Settings Window**

The tools can now be used to configure the memory in the system to contain the executable software code taken from the supplied C compiler. Memory values are allocated to the internal block RAM memory, located within the core of the FPGA. External memory can also be configured using the various utilities supplied with the Embedded Development Kit (EDK).

Once the designer has completed configuring their processor system with the the Xilinx Platform Studio (XPS) tool, the hard work is performed by the tool and in a matter of seconds the hardware side of the design will be constructed and made available to the designer as a black box module. This module is then ready to be instantiated into the FPGA design; the tool also assists the designer with this task by creating a Hardware Description Language (HDL) template. The driving force behind all of this hard work are the Xilinx "Platform Generator" (PlatGen) and "Library Generator" (LibGen) tools. Working from the text-based Microprocessor Hardware Specification file (MHS) and the Microprocessor Software Specification file (MSS), XPS will customize and synthesize each peripheral to meet the needs of the designer. Bus structures and banks of internal block RAM are also customized and synthesized in the same way before PlatGen connects all of the system components together. This task is transparent to the user, who no longer needs to observe the complicated but often essential design / protocol rules for the supported CoreConnect® bus structures.

However, the design automation does not stop here. In the background the LibGen will also have created software libraries customized to the processor design, containing no more and no less than the software functions which can be accessed given the chosen peripheral set. Inaccessible software routines are removed from the libraries to prevent wastage of valuable block RAMs within the FPGA. The aforementioned user selection of the standard input and output peripherals are used to tailor commonly used C software functions like "printf" and "scanf", offering the software design team ease of use. Device drivers are also created in C for each of the selected peripherals, using their previously determined instance names to create a selection of appropriately named C functions, each customized to the peripheral in question. Interrupt control and its associated housekeeping is also handled automatically, the user simply supplies the name of the C function that should be executed when the interrupt occurs. Interrupt handler routines can be assigned to each peripheral individually; interrupt priority is encoded automatically in the Interrupt Service Routine by the LibGen tool.

MicroBlaze Flexibility

The finished processor system, although complete, remains totally flexible. Any change in the system specification can be quickly reflected by the designer using the automated toolset, allowing maximum flexibility with zero wastage.

The MicroBlaze soft processor is based upon the successful RISC / Harvard architecture combination (Figure 7).

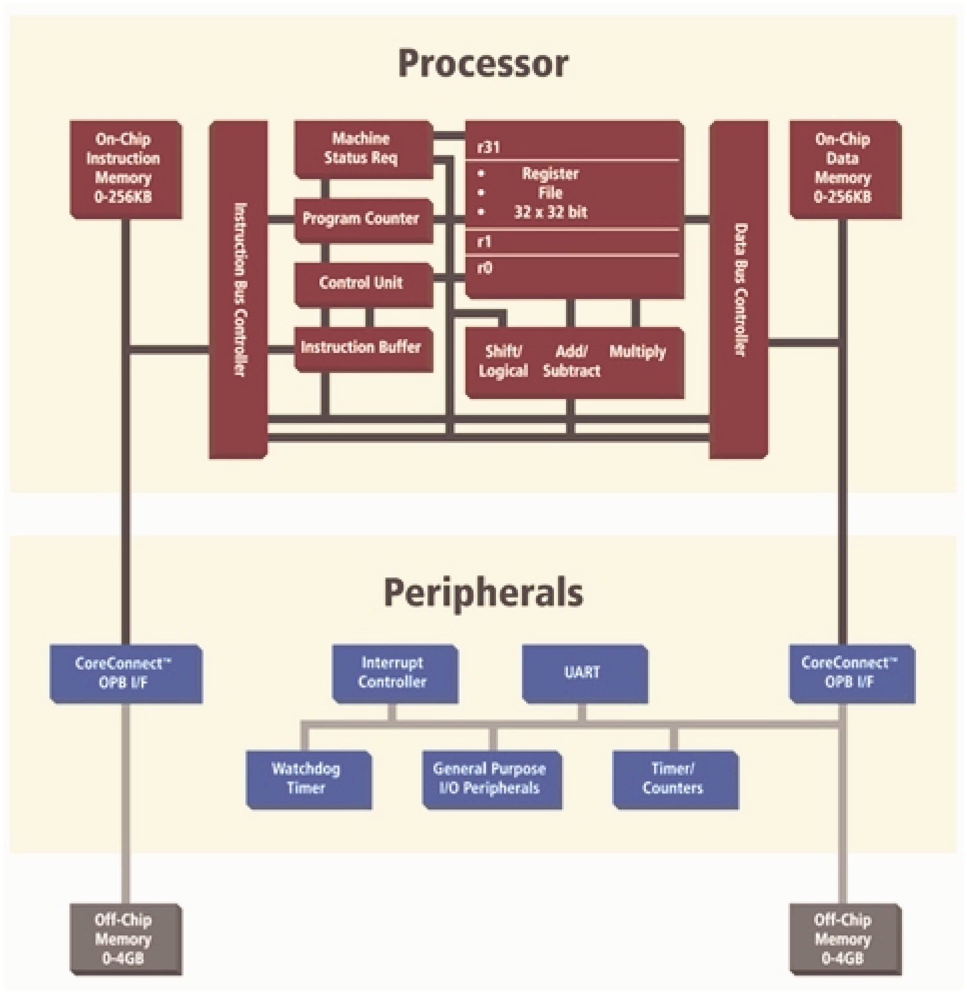


Figure 7: **MicroBlaze Block Diagram**

While remaining extremely compact, consuming just 1,050 logic cells in the Spartan-IIE FPGA, it delivers astonishing performance (49 Dhrystone MIPs at 75

MHz) to meet the needs of a huge variety of embedded processor applications. Performance of the processor core is guaranteed thanks to the use of the Xilinx Relationally Placed Macro (RPM) technology. Placement of the logic elements in the FPGA is predetermined guaranteeing effortless and repeatable performance figures.

The completed system can be implemented in a wide range of the Xilinx FPGA devices, indeed any member of the Virtex™ / Spartan-IIE family is a valid target for a MicroBlaze system. The use of multiple MicroBlaze processors sharing the same bus structure is possible, allowing the concepts of scalable distributed processing to become an effortless reality. Use of the Spartan-IIE device family permits low cost and high volume applications to be an ideal choice for systems under MicroBlaze control. Compatibility with the On Chip Peripheral Bus (OPB) from the IBM CoreConnect bus family permits the designer to include a vast range of existing IP into their MicroBlaze designs with little effort. CoreConnect compatibility also offers a simplified upgrade path should the designer wish to migrate their processor design to the industry leading Virtex-II Pro™ FPGAs containing the ultra high-performance embedded PowerPC processor cores.

Conclusion

The MicroBlaze soft processor core and its associated toolkit offers a new and powerful approach to embedded processor system design. Never before has this level of flexibility been available to the digital electronics industry. The combination of the high-performance MicroBlaze processor and low cost Spartan-IIE Field Programmable Gate Arrays make this solution invaluable to the modern digital designer.

Information on this subject can be found on the Xilinx Field Programmable Controller page at:

http://www.xilinx.com/fpc

Further information on the MicroBlaze product can be found in the Xilinx MicroBlaze lounge at:

http://www.xilinx.com/microblaze

Information on the Xilinx Virtex and Spartan-IIE FPGA products can also be found at:

http://www.xilinx.com

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/10/02	1.0	Initial Xilinx release.