The XC6200 FastMap^{TM} Processor Interface

Stephen Churcher, Tom Kean, and Bill Wilkie
Xilinx Inc.
(presented by Raj Patel)

Abstract

The Xilinx XC6200 is the first commercially available FPGA to be specifically designed for use within microprocessor-based systems. This paper discusses the architecture of the key element in this device: the *FastMap^{TM}* processor interface. Salient features of the user-programmable part of the XC6200 are also described.

1 Introduction

Almost all modern digital systems consist of three major functional components: microprocessors, memories and logic ICs. Logic ICs interface microprocessors to physical devices such as screens, keyboards and networks and perform computations which are unsuited to the microprocessor. Logic may be implemented using mask programmed devices or Field Programmable Gate Arrays (FPGA's) [1]. An interesting question is: What would the ideal logic part, from the point of view of a microprocessor, look like?

- Processors view the world as a sequence of memory locations. Therefore the interface to the FPGA should be through memory mapped registers.
- Processors run different programs at different times. FPGAs should therefore be able to be reconfigured for different tasks at different times.
- Since processors are synchronous devices the FPGA should be able to operate from the same clock as the processor, thereby allowing predictable interactions between them.
- The FPGA should be dense and fast and have good I/O capability.

2 Memory Mapped I/O

The concept of memory mapped I/O is simple: a particular register within the user's design selected via the processor address bus is connected to the processor data bus and a read or write operation is performed. The address selection and connection to the external bus is typically implemented using multiplexors and gates within the FPGA's user resources, although this has several disadvantages.

- The interface to a modern processor is complex and high speed and can use up large quantities of user logic and IOB resources.
- If the registers which are to be accessed from the processor are placed near the centre of the device, long routing wires are required to reach user IOBs. If many registers are to be provided, complex selection circuitry is necessary.

When one considers that the control memory of an FPGA is itself a static RAM, with address and data busses available on the device, it becomes clear that these resources could be presented off the chip as the primary programming interface rather than being hidden behind a serial channel. Naturally, this requires many more pins to support wide address and data busses, and so is not appropriate in many applications. However where the device must work with a microprocessor it is advantageous.

Register resources on the FPGA can be addressed using bit and word lines within the RAM array in the same way as configuration memory cells, and their contents presented on the external data bus. This technique was first implemented in the Algotronix CAL1024 chip [2]. Algotronix technology was acquired by Xilinx in 1993, and has been continuously developed since then resulting in the XC6200 family. Figure 1 shows the first device in the family, the XC6216.

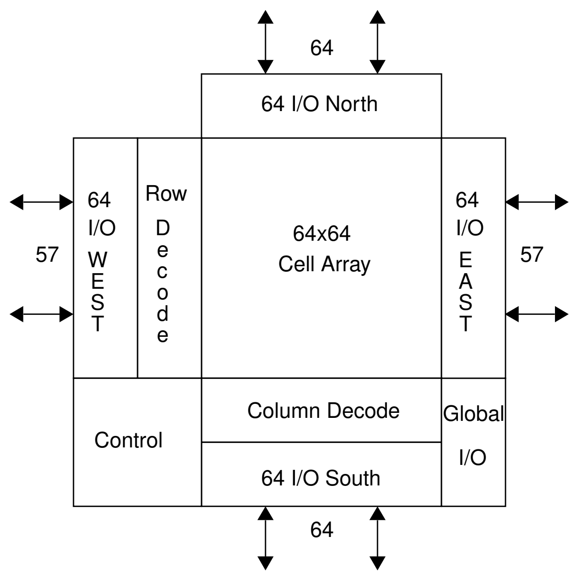

Figure 1 : XC6216 Architecture

Several additional steps are required to turn user registers, mapped into the device configuration memory, into a high bandwidth channel transferring 32 bit words between the processor and the user design on the FPGA. Firstly, the register bits, which will be physically dispersed among the configuration bits, must be collected together in the address space so that complete words of register memory are formed. This is achieved in the memory row and column decoders. Secondly, some flexibility must be provided in the mapping of register bits which are to be accessed in this way onto device cells - ideally a set of user registers located in arbitrary parts of the device could be grouped into a word of memory accessible through the processor interface.

The row and column (bit line and word line) addressing scheme used by memories makes collecting arbitrary registers on the device into a single word difficult: a realistic constraint is that all the registers should be in the same column of cells. An issue also arises concerning the manner in which bits in the processor word are mapped into registers. If full flexibility was allowed, 6 bits (to select one of 64 cell positions in a column) would be required for each of the 32 data bus bits. This is too much information to change quickly, in order to allow selection between different registers. A realistic limitation is that the bits appear in order in the processor word with the register with the lowest cell y-coordinate first. Given this constraint, the locations of the register bits can be specified with a 64 bit map register - a 0 in the map register (for example) indicates that the cell with the corresponding y-coordinate will take part in the transfer. This technique allows the registers to be spread out over the column in an arbitrary manner and can be implemented relatively efficiently in the RAM bit line logic. Figure 2 shows an example of accessing a register within user logic via an 8 bit external bus on the XC6216.

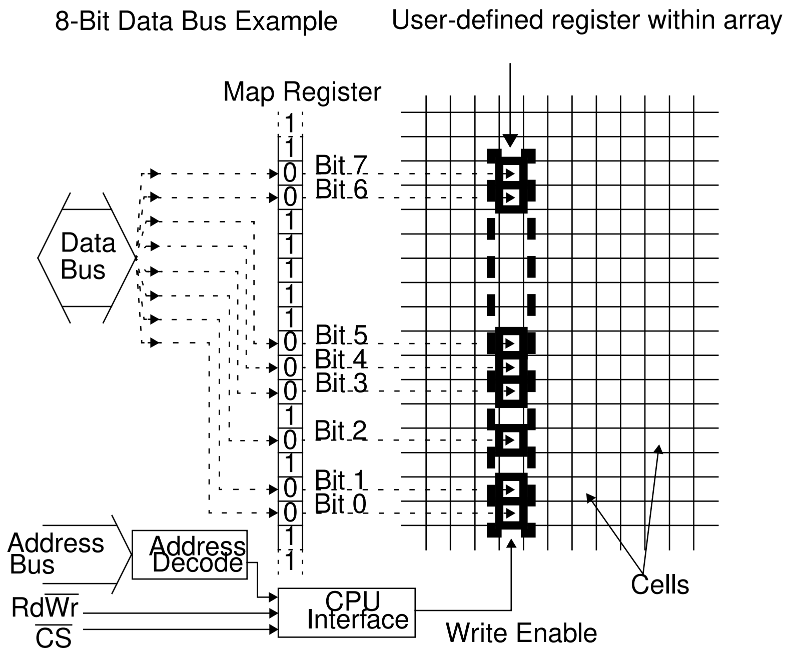

Figure 2 : Map Register - Principles of Operation

3 Dynamic Reconfiguration.

SRAM-based FPGAs are inherently capable of dynamic and partial reconfiguration; all that is required is to bring the internal RAM data and address busses onto device pins, as was done on the Algotronix CAL1024 part. The XC6200 family is configurable to provide an 8, 16, or 32 bit external data bus, and a 16 bit address bus. Using these features, the entire configuration memory can be programmed in under 100 µs.

Normally, however, it is not necessary to program the entire device. In such cases, the random access feature allows arbitrary areas of the memory to be changed. In addition, a mask register is provided which allows a subset of bits within a word to be masked out of a transfer to the memory. This is useful because often a user will wish to reconfigure a particular logical resource (e.g. a routing multiplexor) which will represent only a small fraction of the bits within a word of program memory. This feature is particularly attractive in combination with the wildcard registers discussed below.

Another property of FPGA configurations, particularly datapath-type designs, is that they are very regular, i.e. the same pattern of bits may appear at many locations in the memory (e.g. each slice in a 32 bit datapath). The XC6200 supports writing the same configuration information to multiple locations in the control memory simultaneously, using so-called 'wildcard' registers. These registers modify the row and column addresses supplied to the chip, putting 'don't cares' on corresponding address bits. For example when one address bit is subject to a 'don't care' condition, two memory locations will be written simultaneously on every transfer (Figure 3). Using the wildcard addresses, the configuration memory for all the cells on the chip can be cleared in a single memory cycle. Most user designs will use a fraction of the chip's resources, and in such cases it will often be faster to clear the configuration

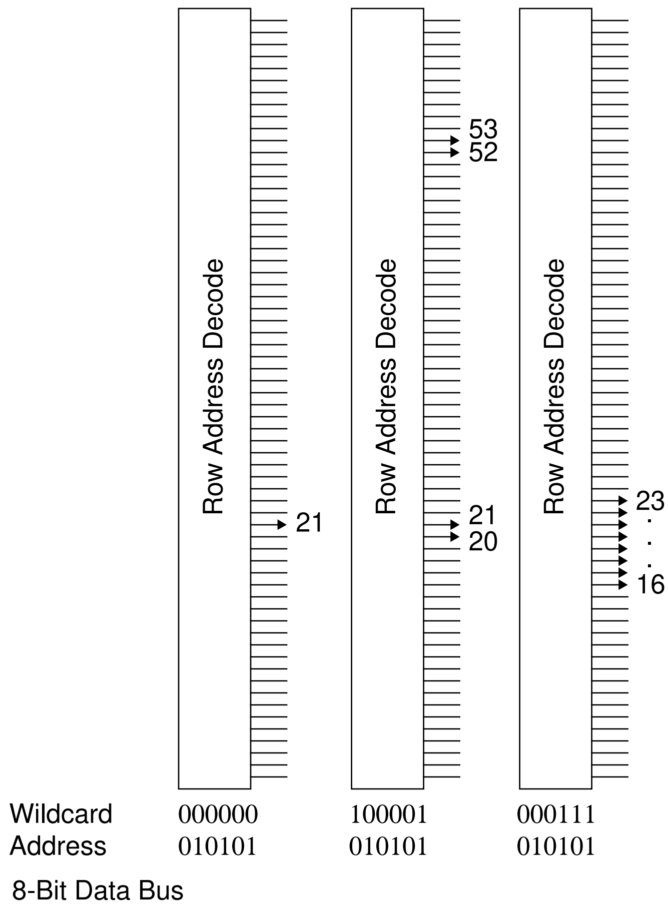

Figure 3 : Wild Card System

4 Synchronous Access.

The communication between the processor and the FPGA can function most effectively when they are both running from a common clock. This synchronizes input and output of data through the processor interface with computations running in the user logic and ensures that setup time requirements are met on write accesses to user registers and that read accesses obtain valid data. For this reason the clock signal for the processor interface is also supplied as one of four low skew global signals to the user logic [3].

As well as synchronizing FPGA and processor communication, it is useful to provide a mechanism for signalling to the user logic that a transfer to or from a register has occurred. This allows the user logic to begin processing an input value or to start computing a new output value. This function could be implemented by using a second user register as a flag to indicate transfers, but this would double the number of processor accesses required. Instead, bit and word lines used for transfers to user registers are made available as inputs to programmable routing switches within the array. By connecting these wires to logic gates within their design, users can monitor the transfers through the processor interface and take appropriate action.

5 Density and Performance.

The first member of the XC6200 family, the XC6216, contains a 64x64 array of fine grain programmable logic cells. Each of these can implement any logic function of two variables, or act as a 2:1 multiplexor. In addition to these combinatorial resources, each cell contains a register which may be used in consort with the logic function generator in a variety of ways, as shown in Figure 4. Finally, each cell in the array contains four interconnection multiplexors, which provide ‘nearest neighbour’ routing resources for up to four independent signals.

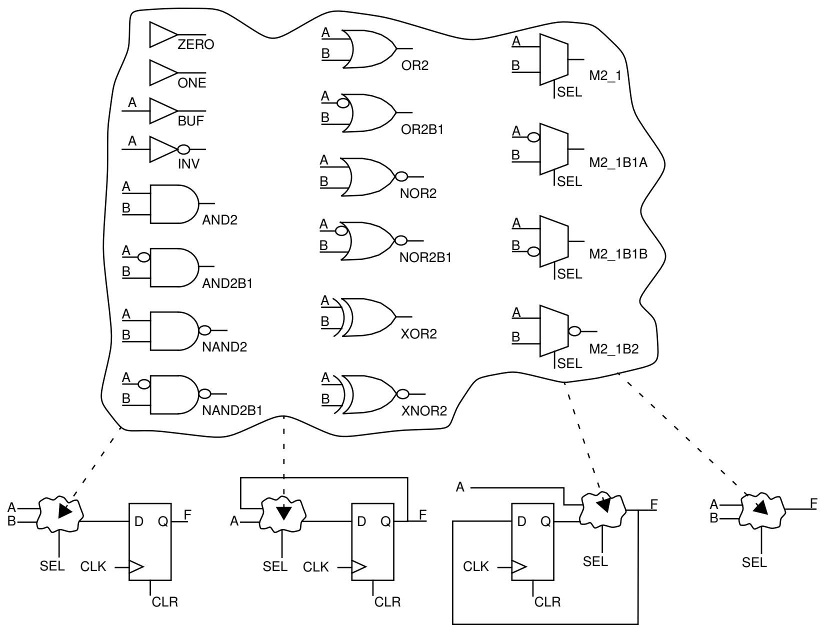

Figure 4 : XC6200 Function Unit Logic Configurations

The routing architecture of the XC6200 is depicted in Figure 5, and is a hierarchical structure with neighbour connections between cells, wires which span 4 cell blocks, wires which span 16 cell blocks and wires which cross the complete 64x64 cell array. There are also four global signals (clock, clear, and two user defined signals). These configurable connections, when combined with the ‘wireless’ I/O capability afforded by the *FastMap*^{TM} interface, provide users of the XC6200 with a powerful, flexible combination of routing resources.

The XC6200 family is implemented in 0.6 µm triple metal CMOS technology, and is expected to meet state-of-the-art system performance criteria. At the time of writing, full performance figures were unavailable. The equivalent gate count for the XC6216 is 16k gates for typical applications; however this figure is potentially as high as 50k gates for designs which are particularly register intensive. Smaller and larger family members will also be made available.

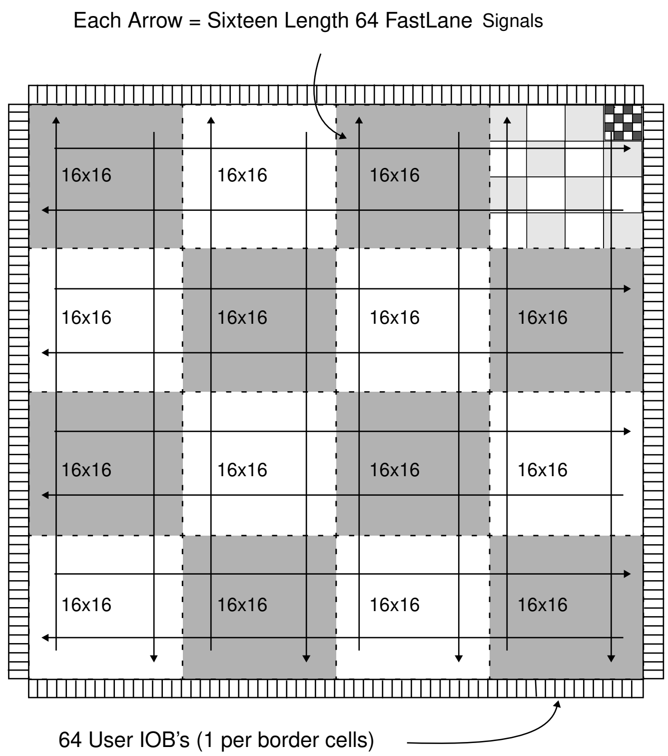

Figure 5 : XC6216 Hierarchical Routing

6 Summary

The XC6200 family [3] is the first commercial FPGA to address the requirements of interfacing programmable logic to microprocessors. With its combination of dedicated random access parallel interface, flexible hierarchical routing, and powerful fine grain logic function generator, the XC6200 is especially suited to 'embedded processing' applications where high bandwidth peripheral devices must be interfaced to a computer system, and processing of the incoming or outgoing data stream is required.

References

1. The Programmable Logic Data Book, Xilinx Inc, San Jose CA, 1994.

2. CAL1024 Data Sheet, Algotronix Ltd., Edinburgh UK 1990.

3. Xilinx XC6200 Family Preliminary Product Description, Xilinx Inc, San Jose CA 1995.