APPENDIX A: Field Programmable Gate Arrays: An Introduction

A.1: Introduction

Field Programmable Gate Arrays and related devices have been revolutionizing microelectronic system design. This chapter provides an overview of FPGA technologies and tools.

A.2: Basic Technologies

Describing Field Programmable Gate Arrays is a little like the four blind-folded men who are presented with an elephant. Each of the men grope around feeling different parts of the elephant and each gives a completely different description of what they believe they are feeling. In an analogous manner we will try to describe FPGAs or at least the parts we've been feeling around in. There are basically two divisions which separate FPGAs at an underlying technology level, these are one-time programmable and reprogrammable devices. The one-time programmable devices are non-volatile and remain configured even when the device is powered down. On the reprogrammable side there are devices that resemble EPROM, EEPROM, and SRAM technologies. This first level of distinction is illustrated in Figure 157.

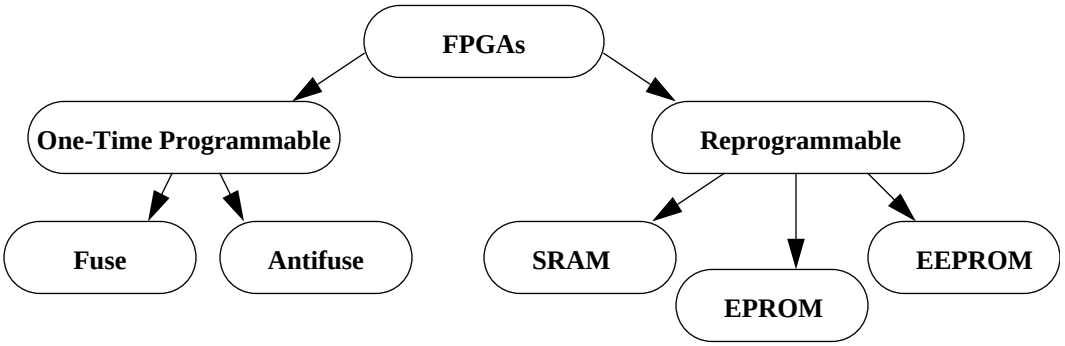

Figure 157 One-time and Reprogrammable FPGAs

Just to add to the confusion there are also one-time programmable devices that are based on EPROM technology with the rational being economic. In these caes it makes more sense to develop your proto-type in a more expensive package (eg. ceramic with a quartz window) that allows for reprogramming and committing to a more inexpensive plastic package for production.

Many of the basic ideas behind FPGAs can be found in the literature dating back several years and in a variety of forms. Early researchers in the area of wafer scale integration looked at a variety of connection methods which ranged from laser interconnection schemes to electronic switches. In addition, as engineers added flexibility to commodity ICs they became inherrently more programmable. Certainly the widespread use of PALs throughout the eighties provided insights into the role of programmable logic in system integration. Also traditional gate arrays contributed in a significant manner to the development of FPGAs. Traditional gate arrays consist of uncommitted logic and rout-

ing resources that are connected up by the ASIC designer. In a similar manner an FPGA consists of uncommitted logic and routing resources that are connected by the FPGA-ASIC designer. One of the main differences is that design customization for a gate array takes place at a mask level in the manufacturing process while the design customization for an FPGA takes place electronically on your desk, (the former requiring a "clean room" the latter a cluttered desk). The following table contrasts FPGAs with more traditional Gate Arrays.

Table 16 FPGA - Gate Array Comparison

Catagory	FPGA	Gate Array	Comment
Gate Count	O(10K)	O(100K)	Order of Magnitude
NRE	<$10,000	~$100,000	Order of Magnitude
Performance System Clock	<50MHz	<150MHz	Factor of 2
Design Tools	Full Suite	Full Suite	pc -> workstation
Programmability	Field	Factory	ms vs. month
Development Time	Weeks	Months	
Design Revision	Easy	Hard ($$)	

Table 16 is suitably vague in many of its comparisons which should allow for ample discussion. One thing that is evident is that the traditional ASIC landscape is changing and that field programmable devices are a major reason for the change. Although FPGAs often utilize extremelyaggressive IC processing technology it is unlikely that they will ever compare with the density of "state of the art" mask programmable gate arrays. This is due to the fact that some silicon area overhead is required in making the device programmable. This is illustrated in Figure 158 where a simple interconnection is shown diagramatically as a masked interconnect, fused link, and a pass gate interconnection.

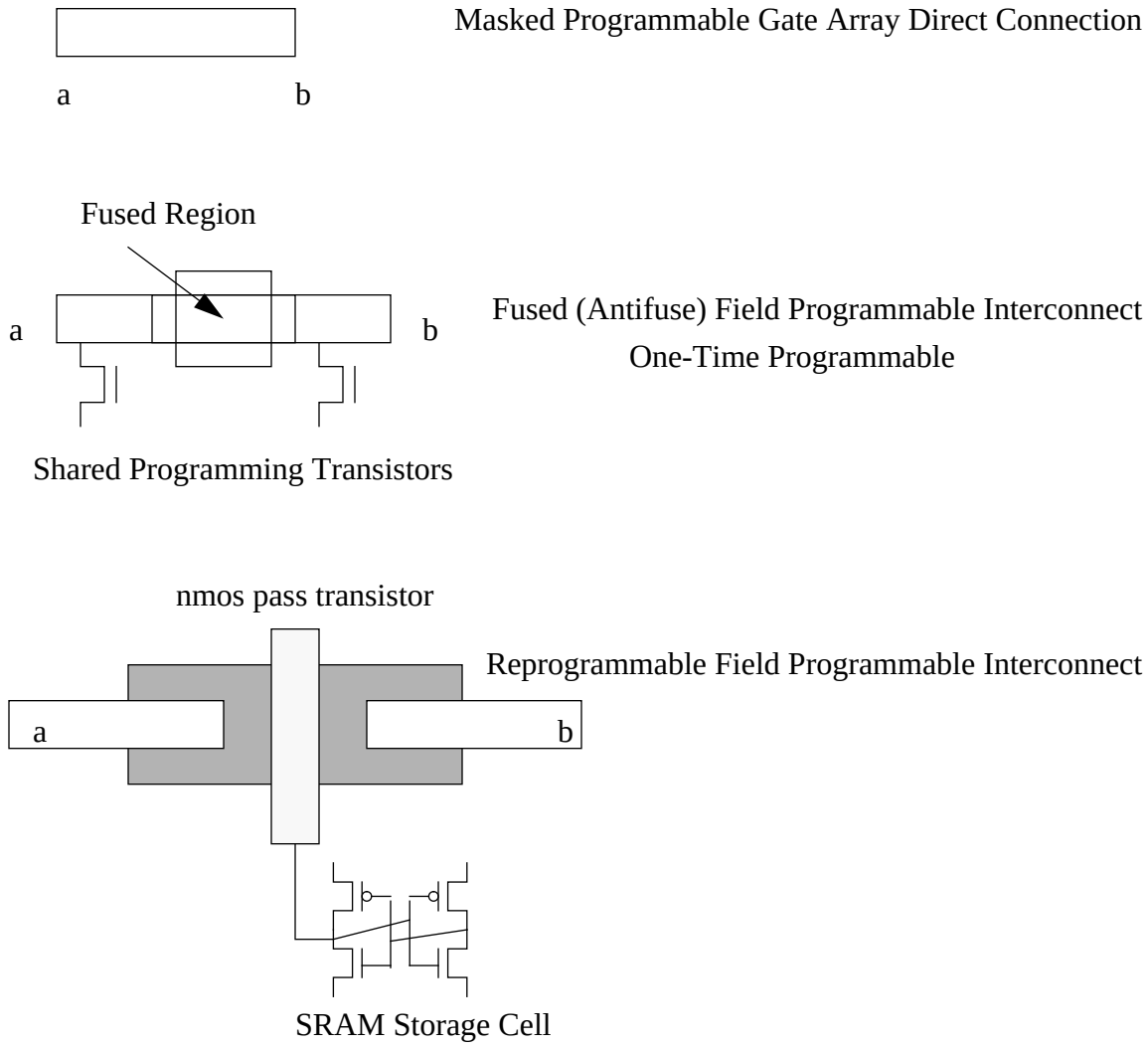

Figure 158 Silicon Overhead for Programmable Gate Arrays

Although Figure 158 is highly schematic in nature it illustrates many of the basic trade-offs amoung field programmable devices at a very low level which driectly effect performance. For example, the most common model for first order performance estimations are derived from lumped element models of transistors and interconnect. The masked programmable interconnect would have lower values of both parasitic capacitance as well as resistance as compared to either of the FPGA devices illustrated.

A.2.1: Antifuse Devices

One popular family of FPGAs are based on "antifuses", which is an unfortunate term used for two materials

that are "fused" together. Typically, a dielectric with a very high resistivity is sandwiched between two conductive interconnect segments. The sandwich is then stressed with an applied voltage sufficient to break-down the dielectric thereby connecting the two segments. The dielectric sandwich must have a sufficiently high break-down voltage such that connections are not created during normal operation yet it must not require an enormous voltage that would break-down field oxide regions or isolating dielectrics. Also a decoding scheme is required such that the applied voltage is seen across the antifuse we want to program. This can be accomplished by decoding the transistor pair shown in figure 2, tieing one of the segments at ground and the other at the programming voltage (Vp). All other segments during programming would be held at some intermediate voltage (e.g. Vp/2) such that any other potentally programmable connection would only see Vp/2 volts across it.

Devices which employ antifuse technology are one-time programmable and non-volatile.

A.2.2: SRAM Devices

Perhaps the most popular type of FPGAs are those based on SRAM technology to maintain device configuration. Although larger and slower than masked programmable gate arrays they offer considerable advantage due to their inherrent reprogrammability. As illustrated in Figure 158, two segments are connected via an nmos pass transistor whose state is determined by a cross coupled latch. This latch would be configured as part of a large shift register when the device is being programmed. As with the antifuse devices the CMOS proccessing can be very agressive with some fine tuning associated with the pass gates thenselves. For example, the nmos pass gate processing may be modified such that a reduced threshold voltage degradation is seen when passing a logical one. One of the immediate advantages of configuring the SRAM cells on a linear array is that only a small number of I/O pins are required when configuring a device.

A.3: Common Features of FPGAs

There are several features common to all FPGAs; programmable I/O, logic, and interconnect. The basic architecture of the antifuse and SRAM devices are illustrated in Figure 159.

Programmable I/O allows the designer to specify whether a pin is to be an input, output, or bi-directional port. There may be several other programmable options under the designers direct control such as configuring pull-up devices and slew rate control. Assignments of I/O pins is often accomadated on the schematic or explicitly specified when placing and routing the design. In most cases a design can be more efficiently placed and routed if the I/O are not made location specific giving the automated place and route tools more flexibility in optimizing the "layout". In many proto-typing applications however the designer is constrained by predetermined I/O.

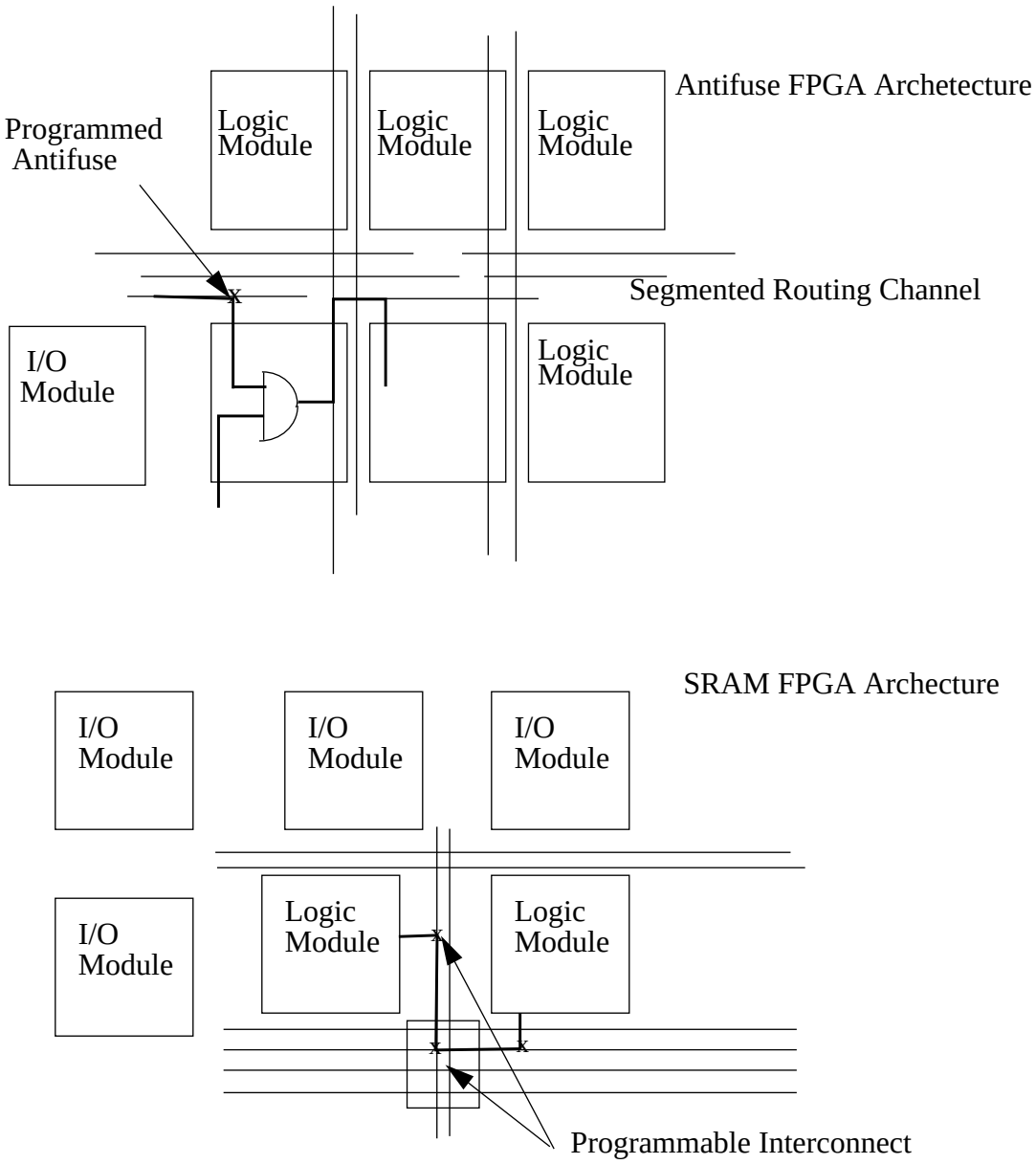

Figure 159 Typical Architectures of Antifuse and SRAM FPGAs

Programmable interconnect is also common to all FPGAs. Implementations vary but the basic notion is to be able to connect I/O to logic modules and logic modules together. Interconnect resources may also include a hierachy of interconnect allowing one to take advantage of local connections or buffered long lines for the distribution of global signals. There is always a trade-off between silicon area dedicated for routing and area dedicated for logic. As FPGAs are very general purpose devices they may not be ideal for all applications which may be either logic or routing intensive. There are however a wide variety of alternatives offered within families of devices as well as an increasing number of device families.

The final common element to all FPGAs are the logic modules themselves. With the antifuse devices they are typically MUX based and within SRAM devices they are typically look-up table devices. In either case there is provisions for sequential elements such as latches and flip-flops. Figure 160 illustrates schematically a logic cell implemen-

tation of a simple function in both an antifuse and SRAM technology.

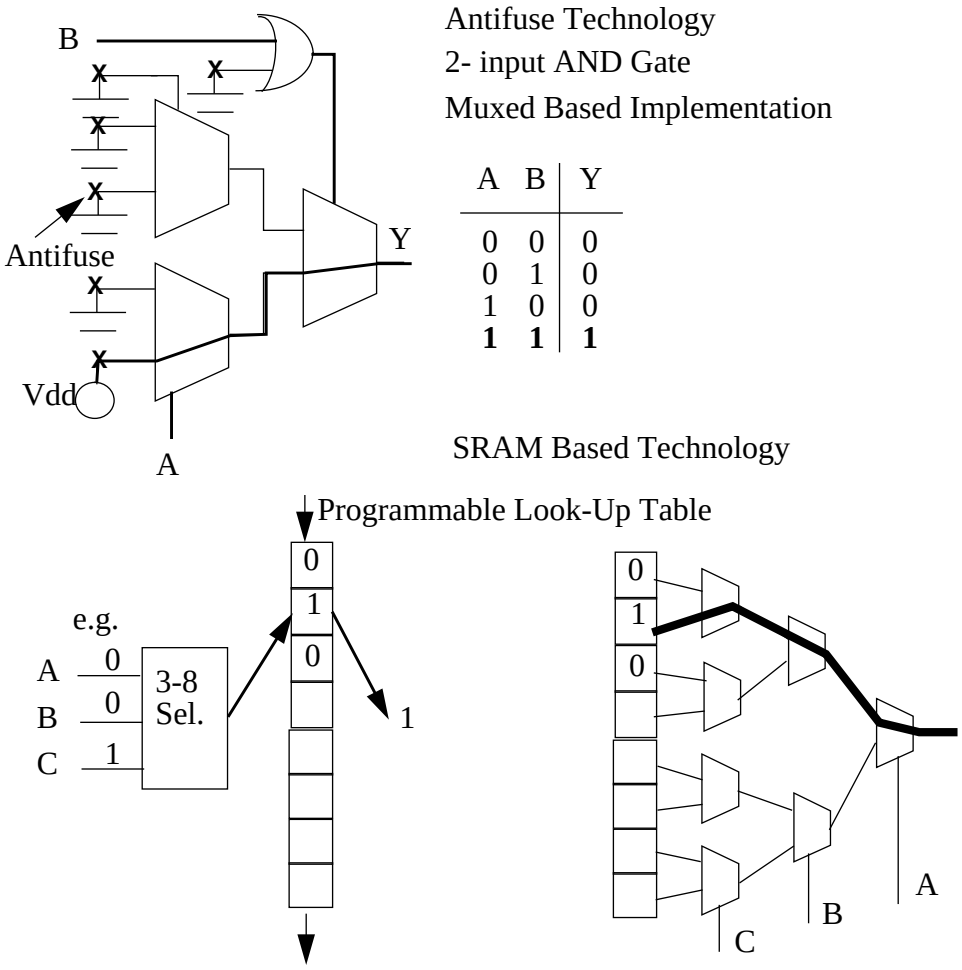

Figure 160 Logic Implementations in Antifuse and SRAM Technologies

A.4: Applications of FPGAs

Field programmable gate arrays are finding a variety of applications in a wide number of areas primarily due to their flexibility. The following lists several areas where FPGAs are playing an increasingly important role. This list is only representative and expandable through engineering creativity.

A.4.1: Logic Integration

In many cases there is an opportuinty to migrate an existing design from small and medium scale integration to FPGAs. It may not be viable to move to a masked gate array but the economics of a more integrated system may allow for the deign to be migrated to FPGAs. The flexibility of being able to design and prototype the newer FPGA design may also lead to improvements in design and in production test. With the relatively low cost of FPGAs additional features may be added and evaluated and integrated into the product if desired. This may prove to be very efficient as interfaces continue to evolve across a spectrum of "standards".

A.4.2: Rapid Prototyping

With increased emphasis on epeditious system design to get your product to market FPGAs are and will continue to play a major role. FPGAs allow for rapid hardware design revisions to be made without the risk of waiting several months and several 10s of thousands of dollars in first-time correct silicon. This is particularly important in fields where standards are either evolving rapidly or don't yet exist.

A.4.3: System Level Testing

System level testing is in part the functional and structural test of a microelectronic product. FPGAs and in particular reprogrammable devices afford the system designer the resources that may not otherwise be available or economically feasible in other technologies. Boundary scan is one example where FPGAs are very usefull in system level test. For example, with respect to PCB testing the designer may develop or utilize the boundary scan resources available from the vendor. The advantage of developing ones own scheme is in incresing the flexibility of boundary scan. For example, in testing a PCB one FPGA may become a boundary scan master in effect interfacing the board to a boundary scan test environment. Pattern generators and signature analyzers may be easily implemented providing patterns to test the board at-speed as well as compacting responses. Application specific test programs such as testing memory or measuring memory access time may be developed to allow the designer to in-system test a non-boundary scan compliant memory device with a production test such as walking ones or march test. These types of hardware overheads would be difficult to justify in the majority of products if the overhead were to remain resident, the advantage of a reprogrammable FPGA here is that these test programs can be stored in memory and used as required. Once these tests are completed the system can in many cases be reconfigured to eliminate the fault or design error.

A.4.4: In Field Modification, Repair and Updating

In a similar manner to that of system level testing FPGAs offer a considerable advantage in being able to be reconfigured "in-system" or in the case of an antifuse device reprogramming the device to account for design revisions or modifications. Again the advantage of the reprogrammable device is the potential of in-field repair in a remote manner. The basic idea is for the engineer to remotely login to the user's product, run a series of diagnostics and attemt to repair the system by reconfiguring the the hardware by downloading a new configuration bit stream. This is a more likely scenario in a telecom application where access to the product is part of its system features. The cost of system repair for a company is definitely more if an engineer or technician has to make a service call as opposed to repair from and engineers desk. He or she may repair a product in several cities in one day without ever leaving their workstation.

A.4.5: Hardware Software Co-design

FPGAs offer an exciting opportunity in the area of hardware software co-design as they do in rapid prototyping environments. FPGAs make the distintion of hardware and software somewhat blurry. The advantage is that there is increased opportunity for the design of hardware and software to be more concurrent as opposed to the hardware engineers developing the hardware first and the software designers writing code for essentially fixed hardware. This does not make the problem simpler but gains made here will defintely expedite product development. In addition, implementation of often vague specifications do not have to be cast in stone during the preliminary satges of system design.

A.4.6: Dynamic Reconfiguration

Dynamic reconfiguration applies essentially to re-programmable devices. However, the overhead associated with changing a design dynamically has to be considered. For the serial SRAM based devices reconfiguration may take several milliseconds. This overhead would have to be justified as a hardware context switch. It would be desirable to be able to reconfigure the hardware in one clock cycle. Duty cycle time may be used to download the device but the context switch should proceed quickly. The SRAM based devices could accomadate this if shadow registers were employed. There is also a question of application and economics which would have to be addressed by the FPGA vendor as schemes such as shadow registers would consume considerable silicon realistate.

A.4.7: Application Specific Co-Processors

Application specific co-processors are devices configured to run off a host such as a workstation. These co-

processor boards would then be responsible for specific operations which are more easily perfomed in hardware although un-economical to be resident hardware. As an example, there are many operations which are well suited to hardware in Genetic Algorithms that can be off-loaded to a reconfigurable FPGA board that would speed operations such as string cross-over and string copying. There are other applications in number sieving that could utilize specialized reprogrammable hardware as a co-processor. Artificial Neural Networks represents another area where various applications require different topologies and where reprogrammablility would be very useful. The nice thing about a reprogrammable co-processor is that it would have a potential use in several application areas that on their own not justify a dedicated co-processor. Another reprogrammable application may be an interface card that supports a wide variety of interfaces.

A.5: More Field Programmable Devices

There are a number of relatively new devices that are either emerging or will emerge over the next few years.

A.5.1: Analog Field Programmable Devices

In a similar manner to digital gate arrays and subsequent FPGAs analog FPGAs will also become available following analog gate arrays. Several basic question remain such as the mode of operation. Should these devices provide basic building blocks such as operation amplifiers withj programmable interconnect, resistors and capacitors, or perhaps more fundamental blocks such as current mirrors and conveyors? Certainly in an education environment these devices would be very nice to have. Likely also of considerable value in prototyping.

A.5.2: Field Programmable Interconnect Devices

A relatively new comer to field programmable device arena is field programmable interconnect devices. These devices are typically very large pin-out devices which when configured connect various inputs to outputs. Examples of programmable interconnect devices are provided by Aptix and I-Cube. Although applications are just emerging it seems fairly clear that they could play a major role in proto-type development and in large hardware emulators. One of the advantages for microelectronic proto-typing is having a flexible PCB such that FPGAs could place and routed without locking down specific I/O. This typically allows the automated FPGA place and route tools to more efficiently utilize available logic and routing resources. Configuring the I/O would then become the reponsibility if the programmable interconnect device. Similarly the use of reporgrammable interconnect in applications where in-service repair is required would be an attractive application.

A.5.3: Asynchronous Field Programmable Arrays

Although most microelectronic systems rely on synchronization and clocking there is an incresing interest in asynchronous logic design. It is inherrently more difficult to design and test asynchronous devices but there are also several advantages associated with power distribution and speed that make asynchronous devices very attractive. There is also considerable reserach in synthesis and formal verification which will make asynchronous design more attractive. It is not unreasonable that an FPGA would fundamental building blocks such as arbitors, micropipelines, and C-elements become commercially available. There have been several efforts where commercial FPGAs have been utilized for asynchronous designs with a logical extension being commercially available FPGAs with fundamental asynchronous building blocks.

A.6: Research Areas

There are an increasing number of research areas associated with the area of field programmable devices. These range from improving synthesis methods targeting FPGAs, improving the basic architectures associated with logic blocks and routing resources, improving place and route tools, basic research on sub-micron devices and technologies. Figure 161 illustrates a modeling and prototyping environment involving FPGAs in the design cycle.

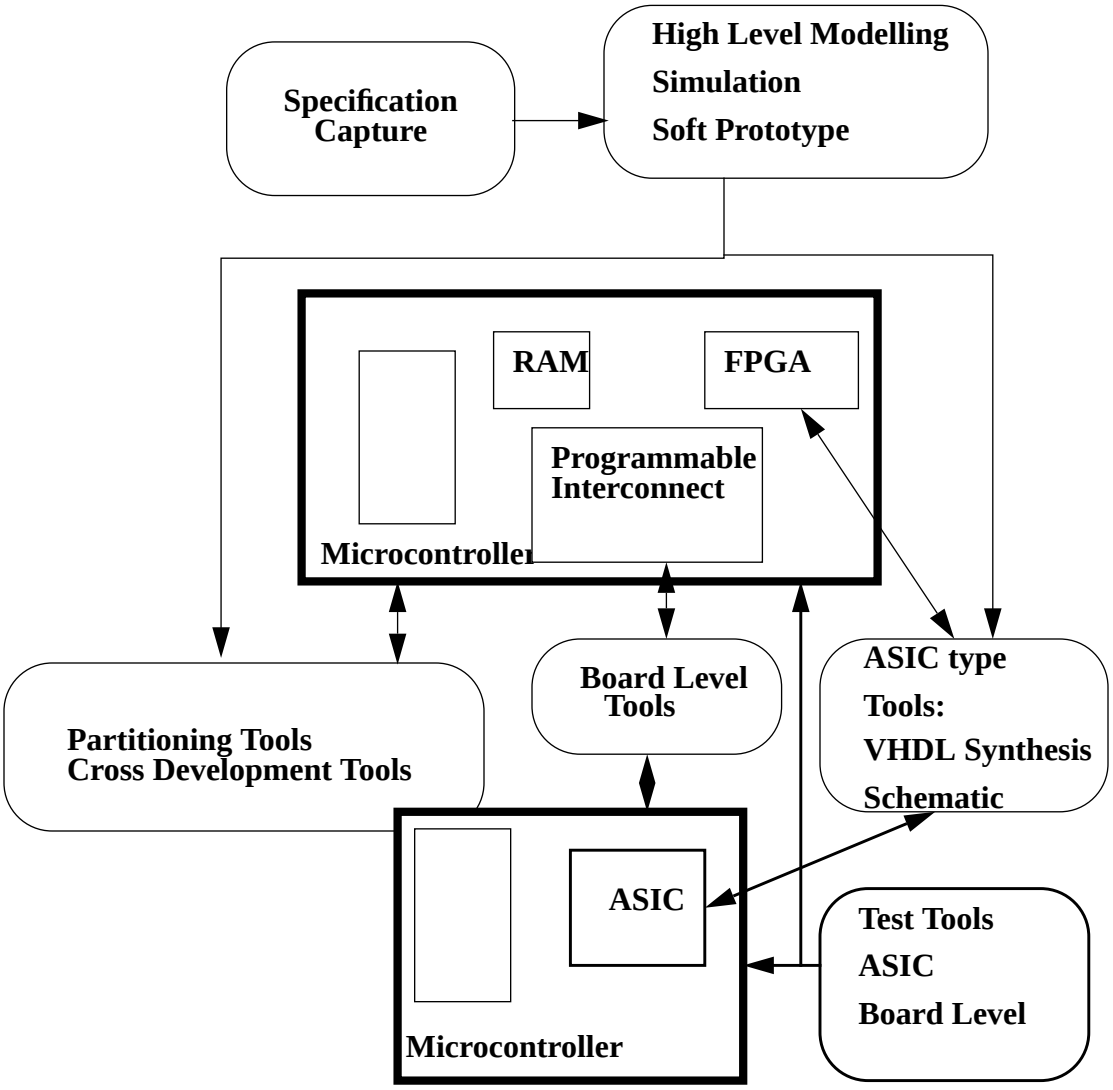

Figure 161 Research and Education FPGA Proto-typing Environment

For Further Reading

There is an increasing number of journal papers and conference papers being presented each year with emphasis on FPGAs and programmable devices. A good starting place for a literature review is the special issue of the IEEE Proceedings July 1993. There is a special section on FPGAs with numerous references.