CHAPTER 10

SYNTHESIS OF FPGAs AND TESTABLE ASICs

DON W. BOULDIN
Electrical Engineering, 1508 Middle Drive, University of Tennessee, Knoxville, TN 37996-2100 U.S.A., Tel: (865)-974-5444, Fax: (865)-974-5483, E-mail: dbouldin@tennessee.edu

Abstract: Industrial designers and educators who plan to design microelectronic systems (e.g. hardware accelerators, co-processors, etc.) are increasingly capturing their designs using hardware description languages such as VHDL and Verilog. The designs are then most often synthesized into programmable logic components such as field-programmable gate arrays (FPGAs) offered by Xilinx, Altera, Actel and others. This approach places the emphasis on high-level design which reduces time to market by relying on synthesis software and programmable logic to produce working prototypes rapidly. These prototypes may then be altered as requirements change or converted into high-volume mask gate arrays or other application-specific integrated circuits (ASICs) when the demand is known to be sufficient. These ASICs, however, must be designed to be testable to screen out those with manufacturing defects. Hence, scan logic must be inserted, test vectors generated and fault grading performed to ensure a high level of testability. These efforts complicate and delay the conversion of FPGA designs to ASICs but must be considered by designers of microelectronic systems. Topics covered include: design flow; system partitioning; hardware description languages (HDLs); specifying behavioral control; specifying structural components; critical paths; placement and routing; technology choices; FPGA applications; rapid prototyping; retargeting; manufacturing defects; scan chain insertion; test vector generation; fault grading, and ASIC production

Keywords: VHDL, FPGA, ASIC, synthesis, programmable logic, testing

1. INTRODUCTION

Designing microelectronic systems involves mapping application requirements into specifications that can then be implemented using appropriate microelectronic components. These specifications must be represented at every level of abstraction including the system, behavior, structure, physical and process levels. Internal functions must be described as well as the interactions among these components and the external world.

R. Reis et al. (eds.), Design of Systems on a Chip, 211–221.

Some of the distinguishing characteristics (Bouldin, 1991) of microelectronic systems are:

– specified hierarchically,
– conform to an interface specification,
– incorporate computing (analog and/or digital processing),
– constructed using microelectronic components, and
– involve input/output devices (e.g. sensors and actuators).

Some example microelectronic systems are:

– a portable instrument for monitoring environmental data,
– an accelerator coprocessor board in a personal computer,
– a controller for a robot or an automobile, and
– a data compressor for transmitting facsimiles or images.

This chapter presents an overview of the methodology for developing microelectronic systems. Thus, the role of hardware description languages, synthesis, physical placement and routing software, programmable logic, testing and microelectronic components will be delineated. Several caveats to this methodology in terms of price and performance will also be discussed.

2. MICROELECTRONIC COMPONENTS

The designer of a microelectronic system frequently employs several existing integrated circuits (ICs) to meet the requirements of an application and thus adds value to the final product by interconnecting components in a unique way. Increasingly, software is also added to provide flexibility that further distinguishes the designer's product from those produced by competitors. Use of existing components reduces the time required to implement the design and generally leads to higher profits. The components are relatively inexpensive since they are commodity parts produced in high volume (millions of copies). Microprocessors and other VLSI/LSI components such as digital signal processing chips are frequently the most cost-effective since thousands of gates costing only tens of dollars (or millions of gates costing only hundreds of dollars) can be utilized. SSI and MSI components are appropriate for tasks which involve mostly external communication and very little internal processing. Packaging is the dominant factor influencing the cost of these components. Hence, SSI and MSI components are rarely the best choice for implementing logic functions requiring hundreds of gates. Analog components such as operational amplifiers, analog-to-digital converters and digital-to-analog converters are utilized to interface to sensors and actuators.

Off-the-shelf components are by necessity general-purpose so optimum performance and/or cost for a specific application may not be achieved. However, a variety of user-specified components or application-specific integrated circuits (ASICs) can be developed by the designer if the situation warrants. In these cases, the designer is able to implement only those functions needed for a special-purpose application so that very little of the physical space is wasted. Thus, the production cost per integrated circuit is held to a minimum.

Techniques have been developed to permit the designer to use a programmable power supply to specify one or two of the layers in some semicustom integrated circuits. These field-programmable gate arrays (FPGAs) contain fewer functions than mask gate arrays whose layers are specified using optical masks since space is required for the programmable links. FPGAs are also slower because of the increased resistance and capacitance of the links. However, the time required for customization is only a few minutes or hours as compared to several weeks for the mask gate arrays or ASICs (Trimberger, 1994).

FPGAs are presently used to implement logic functions up to 200,000 gates or more with a production quantity of 200,000 or less. Mask gate arrays (MGAs) or standard-cell ASICs are used for designs requiring more gates, higher speed or higher volume production. In one style of MGAs, the vendor prefabricates rows of gates with spaces or channels between the rows allocated for interconnections. In another style, the chip appears as a sea of gates (actually transistors) in which some of the gates are used for processing and others for interconnections. The first style is used for less complex designs since the physical space is used less efficiently. Both styles have been adopted to make the task of performing automatic placement and routing straightforward. Mask gate arrays can be fabricated in only 3-5 weeks since the vendor stockpiles wafers and needs only to have the interconnection masks made and the wafers processed for the remaining layers.

FPGAs that are equivalent in number of gates to MGAs may cost 2-10 times as much. The additional silicon area or processing consumed for the programming elements and the on-board addressing circuitry make the fabrication of FPGAs much more expensive. On the positive side, extensive testing can be performed by the supplier prior to delivery to the designer, reducing some product development costs.

3. PRODUCT DEVELOPMENT

The development of a product requires careful consideration of many factors including the requirements of the application, the availability of appropriate microelectronic components, familiarity with electronic design automation tools and the experience of the designers. Perhaps equally significant are other factors such as the perceived demand in the marketplace, the market window (period in which the product is expected to sell), the presence of competition, and the risk of developing and introducing new technologies, These factors combine to place a tremendous pressure on developers to accelerate the design and prototyping stages of product development in an effort to reduce the time-to-market to a minimum. Studies (Huber and Rosneck, 1991) have shown that for every month's delay in being introduced to the market, a ten percent decrease in profits is experienced. Thus, those in marketing must proceed swiftly to ascertain the functionality desired by the customer and those in design must quickly capture these in hardware. The designer must use his time wisely and cleverly but not overlook errors (such as the now infamous division error detected in the Pentium chip (Wirbel, 1994). Whenever these processes are performed too quickly, there is an increased risk of errors in the design, or just as unfortunate, an increased risk that what is produced will not satisfy the customer's needs.

Time-to-market pressures force designers to select off-the-shelf or semicustom components as described above. It is generally very beneficial to produce a hardware prototype as soon as possible since extensive verification of the design cannot be determined without it. Statistics (Huber and Rosneck, 1991) have shown that for mask gate array designs only one set of masks is required in order to obtain prototypes which are fully functional in a stand-alone mode. This mode consists of testing the single integrated circuit on a tester using the same vectors that were applied to the software simulator that modeled the IC. In essence, electronic design automation tools have matured sufficiently that this first-pass success has become routine for digital circuitry. However, these same statistics have shown that these working prototypes fail about half the time when placed in the final system when they are subjected to inputs and outputs from other components. This failure has been attributed to insufficient system-level simulation or modeling. While simulating, designers are just not subjecting the new ASIC to a realistic view of its ultimate environment. Even if a high degree of realism can be achieved, the simulation may consume an overwhelming amount of resources (computer time, memory and disk space). In some cases, no adequate model even exists. For example, image manipulation circuitry must be presented to the human for evaluation and our understanding of the human visual system is still quite primitive. In other cases, analog circuitry is involved and is not modeled with sufficient precision.

This situation has provided great impetus to programmable logic since a design can be implemented temporarily using FPGAs and then later retargeted to mask gate arrays. Thus, the designer can practice using programmable logic and embed the hardware in the full system environment to perform verification to ensure that he has captured the design correctly. Since the penalty for making an error is quite small at this point compared to having to endure the expense for additional mask and wafer processing, designers can rush through the software simulation. Although some design errors may not be caught until testing the hardware, at least these tests can be applied and evaluated quickly, usually at full system speed. These hardware prototypes can also be shipped to potential customers for beta testing. In fact, the prototypes can be considered as first-release production products which can be updated later with a new part or, in the case of reconfigurable components, with a new configuration file. The developers are thus given the opportunity to be more certain that what is about to be produced in quantity will satisfy the customers. Therefore, the risks involved are reduced significantly.

Conversion from FPGAs to mask gate arrays is generally performed in order to obtain a more cost-effective solution if the demand for the product is sufficient. To calculate when this cross-over point occurs, the designer must evaluate not only the costs of the FPGA parts versus the mask gate array parts but also the number needed and the cost of conversion.

Table 1. Comparison of product development

Design Stage	FPGA (weeks)	ASIC (weeks)
Design Specification	1.0	1.0
Design Entry	1.6	1.6
Functional Simulation	2.4	4.0
Test Vector Generation	0.0	6.4
Vendor Interface	0.0	1.6
Prototype Test	1.6	1.6
Prototype Lead Time	0.0	2.0
Production Lead Time	0.0	6.0
Total Design Cycle	7.0	24.0

Table 2. Comparison of product costs

Expense	FPGA (dollars/part)	ASIC (dollars/part)
Raw unprogrammed part	8.00	4.00
Design/Simulation	3.15	7.92
Manuf. Test Vectors	0	2.88
Place/Route/Masks	0	2.20
Final Part	11.15	17.00

Tables 1 and 2 (after Lytle, 1997) compare the product development cycles and the major costs of the two technologies. In both cases, 20,000 copies of the part must be produced and each part must contain 20,000 gates. It is assumed that the slower speed of the FPGA part is acceptable. It should be noted that in 2006 these numbers are more likely to be 200,000 gates and 200,000 copies but the procedure for performing the calculations here has not changed.

The initial difference between the two technologies is the additional time required for simulation. Because the penalty for making an error using a mask gate array is several thousand dollars and a schedule slip would further damage time-to-market and anticipated profits, the designer is likely to spend almost twice as long simulating the mask gate array.

The next significant difference in developing these technologies is the need to generate manufacturing test vectors for the mask gate array. Unquestionably, this stretches the development time and adds to the overall cost, as shown in Table 2. Even though the FPGA initially costs twice as much as the mask gate array, the additional expenses for generating manufacturing test vectors and for the vendor's one-time manufacturing charges make the final mask gate array cost more than the final programmed FPGA. Thus, for this example, the cross-overpoint for converting to the mask gate array is greater than 20,000 parts. This number has risen by a factor of two in just the past two years because of rapidly falling prices for FPGAs (Lytle, 1997).

4. DESIGN METHODOLOGY

Figure 1 illustrates the prevalent design methodology for semicustom components. The designer begins by interpreting the application requirements into architectural specifications which can be implemented in one or more microelectronic technologies. It is not likely that this step will be automated since it involves mapping abstract concepts (often described in narrative form) into precise statements which depend on the capabilities of available microelectronic components. This task is extremely

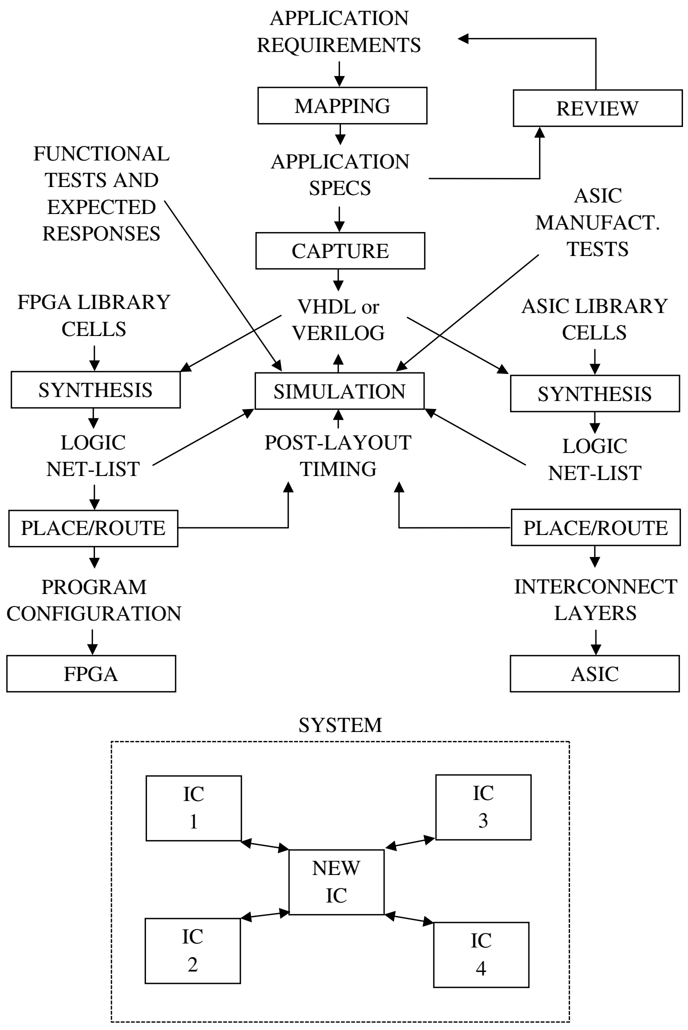

Figure 1. Design of a microelectronic system for multiple technologies

complex and the quality of the resulting implementation is greatly impacted by the experience and creativity of the designer.

4.1 Hardware Description Languages and Synthesis

Once the application requirements are understood, the designer translates the architectural specifications into behavior and/or structure representations. Behavior denotes the functionality required as well as the ordering of operations and completion of tasks in specified times. A structural description consists of a set of components and their interconnection. These components may be primitives or collections of primitives. Both behavior and structure may be specified using hardware description languages such as Verilog or VHDL (Very High Speed Integrated Circuit Hardware Description Language). These text-based languages permit complex hierarchies to be managed efficiently and may even be required for large designs consisting of thousands of logic gates. HDLs can be translated automatically into net-lists of library components using synthesis software. This software performs essentially three functions: (1) translation from text to a Boolean mathematical representation, (2) optimization of this representation based on one or more criteria such as size or delay or testability, and (3) mapping or binding of the optimized mathematical representation to a technology-specific library of components.

HDLs appear initially like other software languages and deceive many designers into thinking that writing code in an HDL is just like writing other software code. However, these languages are tailored for describing hardware and thus permit concurrent operations. A traditional electrical engineer expects hardware components to be active simultaneously and HDLs permit this situation to be modeled. However, traditional computer programmers expect a single CPU that performs operations sequentially. These programmers have expressed dismay at trying to program multiprocessors operating in parallel, but that is exactly why HDLs are used. Thus, a designer who visualizes the hardware will write code which is more efficiently manipulated by the synthesis software. However, he should avoid micro-managing the hardware description too much or otherwise there is nothing left for the synthesis tool to do.

The use of synthesis benefits the designer in several ways. First, it enables him to capture the design in a straightforward manner that may more closely parallel the same way in which the designer envisioned the tasks. This is especially true for describing the behavior of a controller or finite state machine since often the designer is thinking in terms of a collection of if-then-else processes. The text is easily modified in many cases and facilitates the management of large designs. This is not to say, however, that text should be used exclusively. Graphical schematics are often superior at expressing the interconnection of components. Fortunately, electronic design automation tools have been developed which permit graphical schematics to be produced from textual representations and vice-versa. Thus, a description can even be a mixture of text and graphics so that the designer can use whichever representation is warranted.

Another benefit of synthesis is improvement in the designer's productivity. It takes only a few minutes for the synthesis tool to perform a variety of optimizations on the captured design so the designer does not have to spend hours or perhaps days looking for redundancies or trying to minimize delays. The synthesis tool can be invoked multiple times to provide several candidate solutions from which the designer can select at his leisure. This process essentially trades computer cycles for human sweat. It has been reported that a novice designer using synthesis can obtain a solution in a few days which approximates the same quality as that which an experienced designer might obtain in several weeks. However, synthesis should not be considered a panacea since inefficient mapping can result in larger and slower designs than those produced by humans.

In addition to improved productivity, another benefit of using synthesis is the ability to retarget the design without having to recapture it. Figure 1 illustrates that the design can be captured once and the synthesis tool invoked more than once with different technology libraries. Obviously this is of great benefit when programmable logic is used to practice for the final design that is implemented for cost considerations using mask gate arrays. However, this approach can be helpful whenever the designer decides to switch families of programmable logic. For example, a design which takes several months to develop can immediately take advantage of new product offerings. This approach can also be used to select the most cost-effective part for a particular application since it is possible to evaluate several technologies before purchasing the best one. Similarly, the designer may find it necessary to switch to a second source for mask gate arrays and retargeting can make this easy. Just having the retargeting capability in hand maintains a competitive environment for the hardware suppliers that can impact both their price and service.

For those designers who believe they can produce higher quality designs manually, there is still a role for synthesis. The software can be used to obtain an adequate solution quickly and then the designer can pinpoint those portions which need his expert attention. Thus, he can use his time wisely and enhance an otherwise poor solution.

4.2 Physical Placement and Routing

Once the synthesis tool produces a net-list of technology-specific library components the design is ready to be placed and routed. The effort required for designs of even a few thousand gates is prohibitive if performed entirely by a human. It is much more efficient to invoke placement and routing software for this task and to intervene only in rare cases in which the software does not find a solution for some nets. The designer can also accept a partial solution and micro-manage the placement and routing of only those critical nets which restrict operating the system clock at a higher frequency. The complexity of the placement and routing task is so great that software cannot be used to obtain an exact solution but instead must be invoked iteratively in a clever manner. The prevailing algorithm in use today is based on simulated annealing since this optimization routine is tuned to obtain a global optimum rather than accept some local optimum and stop searching the solution space.

Placement and routing is highly order dependent in that the options remaining for a net are reduced each time another has been routed. Consequently, nets designated by the designer to be critical are routed first. This ensures that these nets will be given preferrential treatment and are highly likely to be acceptable. However, some of the remaining nets may not be treated so kindly and more intervention will be required. Fortunately, acceptable solutions can be obtained quickly or the designer can elect to use a larger part which has more logic and routing resources. This adds expense but at least the design gets implemented.

Programmable logic differs from mask parts in that the programming elements introduce a significant RC delay which makes FPGAs perhaps five or ten times slower than MGAs which use metal vias and lines for interconnection. To counter this problem to some degree, FPGA suppliers use segments of varying length to avoid stepping into too many *puddles*. This process also adds complexity to the optimization routines in the placement and routing software.

5. PRODUCING TESTABLE ASICS

FPGA prototypes may then be altered as requirements change or converted into high-volume mask gate arrays or other application-specific integrated circuits (ASICs) when the demand is known to be sufficient. These ASICs, however, must be designed to be testable to screen out those with manufacturing defects. Hence, scan logic must be inserted, test vectors generated and fault grading performed to ensure a high level of testability. These efforts complicate and delay the conversion of FPGA designs to ASICs but must be considered by designers of microelectronic systems.

5.1 Testing Requirements

Test stimuli which validate the desired functional behavior of a circuit provide for functional testing only. Every design must be subjected to these tests to ensure that the application requirements are met by the circuit that is under construction. In essence, if the application requires an adder function then the circuit being designed must be checked to demonstrate that addition is being performed and not subtraction or some other unwanted function.

Functional test stimuli are first used inside a simulator to validate the desired behavior. They are then applied to the first hardware prototypes to be sure that the desired circuit has been manufactured. Once a *golden* copy of the circuit has been produced in hardware, additional copies of the part are tested to be certain they are true replicas of the golden one. This additional testing of the production copies is termed *manufacturing testing*. Manufacturing tests are required since the semiconductor manufacturing process can contain numerous defects which in turn lead to yields of less than 100%. In fact, yields of 20-50% are not uncommon with new processes (Weste and Eshraghian, 1993).

To screen out defective circuits, several steps are taken. First, a calibration coupon is placed on each wafer in 4-6 locations. These are probed to determine whether the calibration circuits behave within acceptable tolerance on the wafer in question. Next, power is applied to individual dies on the wafer and the circuits are probed to determine if the quiescent current is within an acceptable range. Those circuits not passing this test are marked with an ink dot and discarded. Those passing are packaged for subsequent, more extensive tests.

Packaged integrated circuits are generally placed first on a stand-alone tester and subjected to a variety of test stimuli at a relatively low speed of 1 Mhz. Those which pass are then subjected to higher speed tests and possibly burn-in or environmental tests. ICs which pass all of these tests are shipped to the designer who then inserts each IC into the target system and runs complete system-level, at-speed tests to be certain that only working parts are sent to the customer.

5.2 Faults, ATPG and Scannable Logic

Manufacturing defects may manifest themselves in a variety of ways including shorts and opens which make the logic nodes appear to be stuck at one or stuck at zero. Some defects give rise to the desired behavior but only after unacceptable delays. Fortunately, these faults can be modeled and graded so that only those parts which exhibit an acceptable quality level (generally above 98%) will be sent to the customer. Also, numerous techniques have been developed for automatic test pattern generation (ATPG) so that these goals can be achieved.

It is not uncommon to apply ATPG software to a circuit only to learn that a portion of the circuit is either uncontrollable or unobservable from the primary inputs/outputs. One countermeasure is to insert a probe point in the circuitry which can be accessed externally. Perhaps the simplest means of implementing these probes is to connect external terminals to existing internal storage elements. If the circuit is still not controllable or observable, additional storage elements may be inserted or the circuit rearranged. This activity is termed *design-for-testability*.

One means of accessing internal probes while minimizing the space consumed for interconnections is to connect them in serial fashion. This produces a chain of flip-flops which enable external data to be scanned in and responses to be scanned out. Since the flip-flops must be able to support normal circuit operation as well testing, each one must be preceded by a multiplexer. These devices are therefore known as *scannable* flip-flops. This additional wiring and multiplexing adds slightly to the overall cost of the circuitry but must be included or else faulty parts will be sent to the customer.

The use of scannable flip-flops at the periphery of the circuit is termed *boundary scan*. Having access at these points is sufficient to determine whether a detected fault lies within the part or in one of its neighbors. *Built-in self-test (BIST)* refers to circuitry within the component which generates or applies test stimuli to the remaining internal circuitry. One obvious advantage of BIST is the fact that the tests are applied at full-speed. Another is the ease of access to internal functions and an almost unlimited number of inputs and outputs. Yet another benefit of including BIST is that the test may be initiated at any time, even after being shipped to the customer. Thus, the customer can reinitiate the BIST and be reassured whether the part is working faithfully.

Both FPGAs and ASICs must be testing for manufacturing defects. In the case of FPGAs, the vendor can perform these tests prior to shipping them to the application designer. Hence, the application designer needs only to generate functional tests and check that the programming of each part is a faithful reproduction of his prototype. However, the application designer who uses ASICs must shoulder the additional burden of generating and applying warranted manufacturing tests. Thus, it is the responsibility of the application designer to use scannable flip-flops, generate the manufacturing test patterns and perform fault grading. If the fault grade is too low, the designer must insert additional scannable flip-flops or redesign the circuitry. All of this must be done prior to manufacturing in order to be certain that the desired level of fault grading can be achieved. After the parts have been fabricated, each one must be subjected to these manufacturing tests to screen out the faulty ones.

6. SUMMARY

Designers of microelectronic systems are increasingly capturing their designs using hardware description languages such as VHDL and Verilog. Designs requiring up to 200,000 gates and less than 200,000 copies are most often synthesized into FPGAs. When higher performance or larger quantities are warranted, these designs are retargeted to ASICs. However, ASICs must be designed to be testable to screen out those with manufacturing defects. Hence, scan logic must be inserted, test vectors generated and fault grading performed to ensure a high level of testability.

REFERENCES

Bouldin, D., Borriello, G., Cain, T., Carter, H., Kedem, G., Rabaey, J. and Rappaport, A. (1991) *Report of the Workshop on Microelectronic Systems Education in the 1990's*, Knoxville, TN: Electrical Engineering, University of Tennessee.

Huber, J. and M. Rosneck, M. (1991), *Successful ASIC Design the First Time Through*, New York: Van Nostrand Reinhold.

Lytle, C., (1997) *The Altera Advantage: First Quarter Newsletter*, p. 3.

Trimberger, S. (ed.) (1994) *Field Programmable Gate Array Technology*, Norwell, MA: Kluwer Academic Publishers.

Weste, N. and Eshraghian, K. (1993) *Principles of CMOS VLSI Design (2nd Ed.)*, Reading, MA: Addison-Wesley Publishing.

Wirbel, L. (1994) IBM Stops Shipping Pentium-Based PCs, *Electrical Engineering Times*, Issue **828**, p. 14 (December 19, 1994).