CHAPTER 1

Phases of an ASIC Project

1.1 Introduction

This chapter provides an overview of the entire Application-Specific Integrated Ciruit (ASIC) development process. Starting from feasibility, it breaks the process down into a set of project phases that must be carried out to obtain fully validated silicon that can be used by production. The main activities in each phase are described, as are the roles and responsibilities of the different members of the project team during these phases. Because this chapter is intended to serve as an overview, the reader is referred to subsequent chapters in the book for detailed treatment of many of the issues covered here.

1.2 List of Phases

Different companies and design teams may use slightly different names to describe what is essentially the same set of steps in the ASIC design process. The phases listed below, or slight variations on them, are followed by most ASIC design teams today. The names or titles used for each phase are intended to capture the main activities during the phase. However, within each phase, there are always miscellaneous tasks that do not fit neatly under the title associated with the main activity. These miscellaneous tasks are, of course, still described because they are all necessary activities that take place during the phase in question.

The main phases of an ASIC project are:

- Top-Level Design
- Module Specification
- Module Implementation
- Subsystem Simulation
- System Simulation and Synthesis
- Layout and Backend
- Preparation for Testing of the Silicon
- ASIC Sign-Off
- Testing of the Silicon

The phases are shown diagrammatically in Figure 1-1.

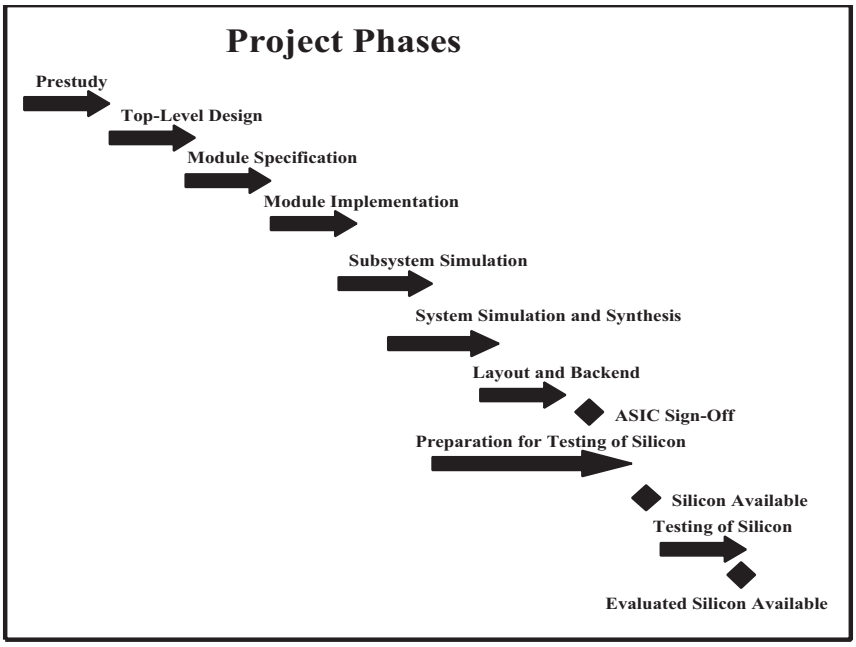

Figure 1-1 List of Project Phases

The following subsections describe the phases of the project in greater detail. The phases are generally sequential in time, although there can be some element of overlap. The initial prestudy, top-level design and specification stages have a crucial impact on the chances of success of the entire project. It is, therefore, important to allow adequate

time and appropriate resources for the initial phases and to avoid the tendency to rush too quickly into the implementation phase.

1.3 Prestudy Phase

Outputs from This Phase:

- Estimate of project timescales and resource requirements
- Estimate of silicon area
- Estimate of product cost
- Initial architecture design
- Analysis of risks
- Identification of project design goals, deliverables and milestones
- Initial decisions on design route and tools

General Tasks:

- Initial architecture design
- Initial planning and estimation of resource requirements
- Risk and cost analysis

The prestudy phase is the initial stage of the project where development and marketing work closely together. The prestudy work must identify a business opportunity and some initial product architectures to satisfy the opportunity. The opportunity may arise as a result of a new market area or a new design to reduce costs or add new system features. Often, ASIC projects integrate a number of functions into one chip, reducing cost and power and typically increasing performance and providing additional features. This is a key characteristic of the emerging SoC (system-on-a-chip) market, where entire systems are integrated onto one chip, typically by reusing a combination of in-house and bought-in third-party IP (intellectual property).

The deliverable from the prestudy phase will be a business case that projects the financial returns, development timescales and risks. The development part of the business case should provide estimates for product costs, timescales and resources needed. It should also provide a detailed assessment of the risks.

If the purpose of the ASIC is to replace a currently successful product, cost and/or feature enhancement is often the driving requirement. When the purpose is to address a new market area or to replace a struggling product, timescales are often the highest priority. The project leader should establish the driving factors before starting the pre-

study phase, because these will affect the architecture options.

The project leader will usually require resources during the prestudy phase. These should include experienced engineers who are good at top-level design. It is important to have access to both hardware and software resources. This is a good time to build the core of a strong team. The team that works on the prestudy phase should ideally be involved throughout the full duration of the project and especially during the top-level and design phases.

During this prestudy phase, the ASIC team must identify the major building blocks of the ASIC and obtain some initial ASIC cost quotations. This will involve talking to ASIC vendors to get a current range of prices. It is useful to consider a number of product solutions and architectures that differ in terms of cost, development timescales, resource requirements, etc. A set of options can then be presented to the management team with a recommended option identified. The team should justify why a particular option is recommended.

For SoC designs or very large devices, any external third-party IP block requirements must be identified and initial cost quotations obtained. If any such IP blocks are not already predeveloped by the IP provider, development timescales and delivery dates must be determined. The team should be wary of selecting IP that has not been proven on silicon because it presents an obvious additional risk to the project.

Large, deep submicron designs may require the use and purchase of new EDA (electronic design automation) tools. Any such requirements should be identified and discussed at this stage with the ASIC vendor and the IP provider because the costs may be significant and should, therefore, be incorporated in the business case.

As part of the process of identifying product solutions, the team should weigh the merits of an ASIC-based solution against the possibility of using third-party silicon. The ASIC may be cheaper or may provide important features or a performance advantage. It must provide some marketable benefit. Otherwise, it is pointless to embark on an ASIC development.

There are no major risks during the prestudy phase. As always, however, there are two conflicting pressures; the first is to generate an accurate business plan defining a winning product, and the second is the pressure of timescales. The prestudy phase can take a reasonable length of time. In this phase, product innovation is very important. Defining the right product and the right development approach makes a winning product. If the prestudy phase is rushed, the chances of success are reduced.

The project leader can reduce the risk of lengthy prestudy timescales by requesting appropriate resource in a timely manner and having all relevant information available when the management team assesses the business case.

Good product or development ideas can be generated when a number of engineers brainstorm ideas together. Several brainstorming sessions should be held. These should include both hardware and software representatives, and participation should not be limited exclusively to those working full time on the prestudy. It can prove beneficial to draft additional contributors because of their known expertise in the area or because of their known capacity for innovation and creative thinking. Competing products should be compared with the product ideas.

1.4 Top-Level Design Phase

Outputs from This Phase:

- Functional requirements specification reviewed and agreed on.
- Top-level architecture document reviewed and agreed on.
- Initial plan and resource requirements reviewed and agreed on.

General Tasks:

- Write functional requirements specification.
- Brainstorm a number of architectural options.
- Analyze architecture options—consider technical feasibility, resource requirements, development timescales, etc.
- Generate top-level architecture document.
- Identify critical modules—start them early, if necessary.
- Identify possible third-party IP block requirements.
- Select team members.
- Identify new processes and tools.
- Agree design route/design flow.
- Brainstorm risks.
- Estimate silicon area, pin-out, cost, power, etc.

Project Manager-Specific Tasks:

- Generate project plan.
- Obtain resources (project team, equipment, tools).
- Organize training courses.

This is a creative phase, during which the top-level product architecture is defined.

Many of the classic engineering trade-offs are made in this phase. Factors such as cost of the product, cost of the design, time to market, resource requirements and risk are compared with each other as part of the process of developing the top-level design. Innovation at this time can have dramatic effects on the success of the product. The innovation can take the form of product ideas, top-level architecture ideas and design process ideas. The resource requirements during this stage will again be a small number of skilled engineers who are talented architects and system designers.

The team should make trade-offs between functions performed by standard off-the-shelf chips, existing ASICs, software and the new ASIC. Chapter 8, "Planning and Tracking ASIC Projects," outlines some ideas for reducing timescales through careful architectural design.

One of the deliverables at the end of this stage is a top-level architecture document, which clearly defines the partition between board, software and ASIC functions. Often, the ASIC represents such a significant part of the design that the top-level ASIC functions are also defined in the architecture specification. It is important to review the top-level architecture specification with selected experts within the company, including representatives from other project teams.

At this stage, an initial ASIC plan should be developed. This plan and the resource requirements should be reviewed and agreed with senior management. At the same time, the design route document should be written. This defines the tools, techniques and methodologies that will be used at each stage. It will aid the planning process because it forces the team to think about required tools and equipment early on. It may be necessary to start booking these at this stage. From the point of view of the design team, the design route document provides a good overview of the design flow and explains the reasons for some of the project rules. The design route document is described in Chapter 3, "A Quality Design Approach." Any new tool identified in the design route document should be tested before the tool is needed, and the plan should include time for testing and contingency options if any of the tools are not sufficiently reliable or simply do not work.

Toward the end of this stage, the team members are selected. The project manager should identify any training requirements and ensure that courses are booked. It is helpful at this stage to assign an engineer familiar with the ASIC architecture to prepare a presentation explaining the architecture. This can then be used to get new team members up to speed as they join the project.

During this phase of the project, many factors will be unknown. The project leader must make educated guesses about resources, timescales, costs, risks and what the competitors are likely to do. Modules, which are critical from the point of view of

timescale or risk, should be identified and, if necessary, work should be started on these early. One method of reducing risk and improving timescales is to consider design reuse (see Chapter 2, "Design Reuse and System-on-a-Chip Designs," for details).

For SoC designs or very large designs, IP reuse is a common design approach. It may be necessary to source IP blocks from third-party vendors. Such large designs will typically require significant effort during the top-level design phase because the size of the chip often implies a high level of complexity. One or more team members should be assigned the task of identifying IP blocks from other companies, analyzing their specifications and assessing the impact on the chip architecture. This task should actually begin in the prestudy phase, but the level of detailed analysis is greater during this phase. The project manager should begin contacting IP providers to start negotiating price, deliverables and support. Some of the issues associated with the use of external third-party IP are considered in Chapter 2, "Design Reuse and System-on-a-Chip Designs."

Equally important, existing IP blocks from within the company should be analyzed. The advantages of using in-house IP are lower cost, in-house expert support and the possibility of making minor modifications to the IP (although, wherever possible, reusable blocks should not be modified). The disadvantage associated with in-house IP blocks is that if they were not originally designed and documented with reuse in mind, they may not facilitate easy reuse. In extreme cases, reusing blocks not designed with reuse in mind can take longer than redesigning them from scratch. In less extreme cases, when the blocks are being used in a different environment or in a different way than before, they may exhibit previously undiscovered bugs. The project manager should ensure that either sufficient in-house expert support is available for the project or, alternatively, that the design has been certified as a reusable block that is compliant with the company's reuse standards.

For projects that are using deep submicron technology, the design route should be identified at this stage. Any new tools required should be sourced and tested, and any necessary training programs on the use of these tools should be organized.

During the top-level design phase, there is a risk of moving too quickly to the next stage, before the top-level design document and a top-level plan are available. The next phase of the project is when the majority of the team comes on board. Time can be wasted when many people join a project if the top-level architecture is not properly defined and documented.

Toward the end of the top-level design phase, it is useful to start thinking about which engineers will work on which blocks. This can be particularly useful for the less experienced engineers. They can be overwhelmed by the system in its entirety and may

feel more comfortable focusing at an early stage on specific parts of the design.

When the resource requirements are not available within the organization, the project manager must hire new staff. This recruitment process should start early in this phase because there is usually a significant lead time in recruiting good engineers. Chapter 13, "Project Manager Skills," provides some guidelines on interviewing.

1.5 Module Specification Phase

Outputs from This Phase:

- All lower-level modules specified
- Accurate project plan available and reviewed

Tasks:

- Decompose architecture into lower-level modules.
- Document module functions and interfaces.
- Review project plan and top-level architecture document.
- Analyze risks—modify architecture/plan to reduce risks, if necessary.
- Make team aware of any standard approaches (coding style, directory structure, synthesis scripts, etc).
- Check chip design rules (die temperature, package cavity, power pin requirements, etc).
- Re-estimate silicon gate count.

Project Manager-Specific Tasks:

- Start team building activities and individual motivation techniques
- Analyze and manage risks.
- Update the plan, assigning resource to the lower-level module design tasks.
- Start to think about silicon evaluation/validation.
- Define a quality project framework that explains what information is stored where and how updates are controlled.

Risks:

- Some team members can feel isolated during the architecture design.
- The team may not understand the project goals.

During this phase, the majority of the team joins the project. It is convenient to split the task descriptions into two sections. Tasks for the bulk of the team are described first. This is followed by a summary of the other tasks that are also part of this phase.

1.5.1 Tasks for the Bulk of the Team

During this phase, the top-level architecture is decomposed and partitioned into smaller modules, and the interfaces between the modules are defined and documented. Ideally, the hierarchy of the architecture should be captured diagrammatically, using a suitable graphical tool. Some such tools can convert the captured diagrams into structural VHDL or Verilog.

It is useful to have the entire team participate during this phase. First, the top-level architecture should be presented and explained. Then the ASIC is decomposed into smaller and smaller blocks. Each block should have its interfaces defined and its function clearly stated. This process helps ensure that the entire team has a reasonable understanding of the overall design and, in particular that they have a good understanding of the specific blocks for which they have design responsibility and the blocks that interface directly to their blocks. This process encourages team spirit and motivation because the entire team is assembled and everyone gets an overview of the project. One of the deliverables at the end of this phase is to start building team spirit and motivation.

The initial architecture design meetings may take several days (depending on the complexity of the ASIC). The main objective of the initial meetings is to develop and document the lower-level architectural details. This provides a baseline from which the architecture can be refined and developed. The process normally works best if it is broken into two parts. First, there is an initial design session with everyone present. Then, after spending some time documenting and analyzing the initial work, a second meeting is held to review and further refine the architecture.

The architecture hierarchy diagram is not a static document at this stage, and care should be taken to keep it up to date. After the initial sessions, the team can split into subgroups to develop the architectures for the submodules independently. The system architect should attend as many design sessions as possible.

With the breakdown of the architecture, the timescales for the submodules can be estimated and the team members assigned to develop specific modules.

The architecture sessions usually involve defining the modules and their interconnections on a white board in a form of schematic capture process. The project leader should try to involve all the engineers in this process, rather than let it be dominated by a handful of senior team members. Junior team members may have valuable contribu-

tions to make. Even if they do not, keeping them involved improves the learning process for them. Every team member is important and should be made to feel part of the team.

1.5.2 Other Tasks

During this phase, the ASIC vendor should be selected. Chapter 10, "Dealing with the ASIC Vendor," includes advice on selecting vendors. Regular meetings should be established with the selected vendor, and the ASIC architecture and design route should be discussed. Particular attention should be paid to sign-off requirements and tools, especially if the vendor or the sign-off tools are new. This is important so that testing of the vendor tools can be added to the plan.

For SoC or large designs that use IP blocks from external companies, contracts should be signed with the IP providers during this phase. Some initial testing of these blocks should be done and analysis of simulation times, programming interfaces and test coverage used to provide input to testbench generation and simulation planning.

The project manager should be considering techniques for derisking the project and reducing timescales. These topics are discussed in Chapter 9, "Reducing Project Risks." This is the stage of the project where ideas for risk reduction and improving timescales must be generated and analyzed. Some ideas may require further investigation. These investigative tasks should be planned right away so that the ideas can be evaluated and any decisions dependent on the evaluation made as early as possible.

One of the key roles of the project manager is to build team spirit and motivation. There are many techniques that the project manager can employ, and these are discussed in Chapter 11, "Motivation and People Management." The project manager should explain the importance of the project for the business and ensure that the contribution of each engineer is valued. An important technique that helps sustain motivation among team members is the provision of relevant training programs throughout the project. One form of training that is beneficial and relevant to the project is to organize presentations by senior engineers on subjects such as design techniques, system partitioning, tools, etc.

At this stage in the project, the team members must be allocated roles and responsibility for design modules (see Chapter 12, "The Team," for a description of the different roles in the team). These roles should be clearly described to the team. It is a good idea to discuss the role allocation with some of the senior engineers. Then, following an initial assignment of roles resulting from these discussions, each individual engineer should have his or her role explained before making it public. This will strengthen the relationship between the project leader and the senior engineers, and more than likely

will result in better engineer role-matching than would be the case if the project leader assigns the roles without consultation.

One of the key roles in the project is that of the testbench engineer. During this phase, the testbench top-level objectives and architecture must be defined. The testbench engineer should also attend the architecture design sessions because this will add to his or her understanding and result in a better testbench. It will also help make testbench engineers feel part of the team, because they tend to work more in isolation than those working directly on the chip design itself.

1.6 Module Design

Outputs from This Phase:

- All modules designed, coded and tested at module level (with reviews)
- Initial trial synthesis of each module
- Agreed pin-out

Tasks:

- Module design, coding, testing and synthesis
- Chip-level testbench design, coding and testing
- Generation of a more accurate silicon area estimate

Project Manager-Specific Tasks:

- Provide documentation guidelines and set expectations on standards and levels of documentation.
- Explain the reviewing process and identify what will be reviewed and when.
- Review the quality framework approach with the team.
- Run weekly project meetings, tracking and closing actions continuously.
- Agree on an approach for trial layouts with the vendor.
- Agree the test vector approach and required test coverage.
- Book resources for prototyping and testing the silicon.
- Source third-party simulation models.

Risks:

- The timescales can slip during this stage—hold early reviews, monitor plan.

- The silicon gate count may exceed maximum estimates—consider modifications to architecture.

During this phase, most of the team members are assigned to address a number of mainstream tasks, while the rest of the team undertakes the remaining miscellaneous tasks. Again, it is convenient to subdivide the phase into two activity streams. Tasks for the bulk of the team are described first, then the remaining tasks are covered.

1.6.1 Tasks for the Bulk of the Team

The architecture design will have partitioned the ASIC into a number of submodules. In some cases, the submodules will be small enough for a single engineer to design, whereas in other cases, the submodules will require a number of engineers working together. The usual trade-offs between resources and project delivery dates must be considered.

Block design can be broken down into the following five tasks:

- Detailed specification
- Design of the module
- Coding
- Simulation
- Synthesis

Detailed specification involves capturing the design function for the block or blocks in question and tying down the interface definition. Team members should frequently consult each other during this period if there are any ambiguities in interface signal definitions or implemented functions, or if it becomes apparent during this low-level block specification phase that new interface signals or design functions are required.

The design itself involves translating the specification into a design solution. There are several different methodologies for doing this but, in essence, they all involve similar sets of steps. Typically, the blocks can be described by process flowcharts and subsequently defined by block diagrams, timing diagrams and state machine diagrams. There is a temptation to start coding before this initial paperwork design stage is fully worked through. This should be avoided because it can often require the reworking of code to cover the function of the module fully. The design process during this phase is described in detail in Chapter 3, “A Quality Design Approach.”

Generation of appropriate design documentation is part of a quality design process. This design documentation should be reviewed before coding has started in ear-

nest. The process of translating the design documentation into code often leads to minor changes in the detailed design approach—perhaps because there were some undetected flaws in the initial approach or perhaps because the designer has thought of a better or simpler way of implementing a particular function. Any design changes made during the coding should be reflected back into the design documentation, and the complete documentation and coding package should then be further reviewed.

It is important to review the code before exhaustive testing is carried out, because there may not be time to rework sections of the code at this later stage.

Simulation and initial trial synthesis should be done in parallel. There is no point to having a fully functioning module that will not synthesize to the correct speed or that translates into an unacceptably large gate count. Encouraging early synthesis also has the advantage that the size of the chip can be monitored from an early stage in the design. If the ASIC size becomes too large, changes to the higher-level architecture or to the design of the module may have to be made. The earlier any such changes are implemented, the smaller the impact will be on the end date.

After the initial synthesis, some initial gate-level simulations should be carried out. This will prove that the RTL (register-transfer-level) code not only functions correctly but that it synthesizes as intended. It also derisks the later top-level gate-level simulation phase.

1.6.2 Other Tasks

The synthesis approach should be documented and reviewed with the ASIC vendor and the synthesis tool application engineers. The documentation should include a generic synthesis script to be used by all the project engineers.

The simulation approach should be defined during this stage of the project and resource allocated to the testbench tasks. The testbench is typically used for both system and subsystem simulations. The development plan should include all testbench work. The testbench should be designed using a flow similar to that used for designing the ASIC modules. The architecture and code should be reviewed. The list of functions that the testbench provides should be documented and reviewed by the team. The plan should allow some time during the system and subsystem simulations for the testbench to be modified or enhanced, because it is difficult to envisage a completely comprehensive set of tests in advance of implementing the detailed design. Ideally, the testbench should be written as synthesizable code, because some tools, such as cycle-based simulators, and other techniques, such as hardware acceleration, are more effective with synthesizable testbenches. As an aid in developing testbenches, a number of testbench generation tools and high-level testbench languages are available from EDA tool ven-

dors (see Chapter 14, “Design Tools”).

Any third-party simulation and/or synthesis models required should be identified, and sourcing of these models should be entered like all other tasks into the project plan. Two versions of third-party simulation models may be required—one version that models the functionality to the required maximum level of detail and a second, simpler model.

Simpler models simulate faster and are often adequate for the majority of simulations. The speed difference is not usually particularly noticeable in individual module or subsystem simulations, but the effect can be significant when simulating at the system level. Conversely, detailed models simulate more slowly. This is due to the level of detail being modeled and checked. However, detailed models are required for at least a subset of the simulations because these allow for a more rigorous level of testing and parameter checking.

The project leader should explain the design review process to the team early in the design phase and set out expectations for these reviews. This involves providing guidelines on the level of design documentation required and drawing attention to any design procedures and coding styles being adopted. For companies employing a structured design approach, this type of information is often available in company standards documentation. However, it is usually worth reminding the design team of the most important aspects of these standards and explaining the reason for them. It is also important to explain the reasons for the design review—it is an essential part of a quality design process and not a performance review of the designer.

Full team meetings should start at this stage. They should be held at regular intervals—ideally once per week—at a time that is convenient for all team members. A good time to do this is before lunch because this tends to limit the length of the meetings, and the participants are still fresh. The purpose of the meetings is to ascertain progress against the plan and to keep the team informed of general developments in the projects that are of interest to them. During the team meetings, open communications between team members should be encouraged. It is a good opportunity for team members to highlight any issues or problems that have arisen or are foreseeable.

During this phase of the project, the project leader should start planning the testing of the silicon. This initial planning mainly involves estimating the laboratory space and identifying the equipment that will be needed. These resources should be booked or purchased, if not available.

A further activity, that arises at this stage of the project is that of dealing with the ASIC vendor. The following list identifies some important items that need to be considered at this point and tracked from here on in:

1) Device pin list: The pin list has to be generated, reviewed and frozen many weeks before the final netlist is delivered. The pin list should be agreed by the ASIC vendor, the manufacturing team and the printed circuit board (PCB) design engineers.

2) Package: If the package is new to production, space models (chips with the correct package but with basic test silicon rather than the real silicon) can be obtained from the vendor so that trial PCBs can be manufactured. The quality of the solder joints can be analyzed and the production techniques refined, if necessary.

3) Sample and preproduction volumes: The vendor normally dictates the number of initial samples available. There are a number of different types of samples available with different turnaround and delivery times. For initial testing, it is very important that a sufficient number of development units are available to verify the silicon and the system quickly. With good negotiating, the preproduction volumes can be increased, which can be useful to improve production ramp-up times.

1.7 Subsystem Simulation

Outputs from This Phase:

- Successful run of first subsystem simulation
- Reviewed subsystem simulation specification
- Subsystem module testing complete

Tasks:

- Write and review test list document.
- Write test “pseudocode” (i.e., CPU register accesses, testbench configuration).
- Run simulations.

Project Manager-Specific Tasks:

- Closely monitor plan; arrange regular, quick meetings to discuss progress.

Risks:

- Poor communication of simulation issues between team members can increase timescales unnecessarily.

Subsystem simulation is that part of the project where a collection of separately designed but logically related modules are now connected together and tested as subsystems. For certain designs, subsystem simulation may not be appropriate or necessary. However, for larger designs, it often pays dividends. This phase will typically run in parallel with the module design phase. The subsystem simulation can be carried out by a team containing some of the design engineers who designed the modules in question with the assistance of the testbench engineer. This section is treated under two separate headings: First we look at the tasks and activities of the subsystem simulation team, then we look at the project leader's responsibilities during this phase.

1.7.1 Tasks for the Subsystem Simulation Team

The individual unit modules have already been simulated in the previous module design phase. The reason for an intermediate subsystem simulation phase, rather than just jumping straight from unit to system simulation, is threefold:

First, the subsystem simulations should run significantly faster than the system simulations, so more testing can be done in a given amount of time.

Second, the interfaces to the submodule might allow easier testing of infrequent events. For example, in a communications chip, we may need to process several thousand consecutive data packets in subsystem *X* to trigger a particular error condition, such as a FIFO overflow. We are interested in establishing that a downstream subsystem, for example, subsystem *Y*, can detect and manage this error condition gracefully. It may take several hours to simulate several thousand packets to get to this state. In a simulation of subsystem *Y* on its own, testing the ability of subsystem *Y* to detect and manage this error condition properly may be as simple as asserting a testbench input representing the error condition from block *X*. This is all we are interested in from the point of view of testing block *Y*—we don't want to have to send in several thousand packets through block *X* each time to achieve this.

Third, the system simulations might require that a particular subsystem is operating correctly before another module or subsystem can be tested. By planning subsystem simulations carefully, tested subsystems will be available when needed for the system simulations.

The subsystem testbench and simulation strategy should be carefully designed and planned. Ideally, the tests will be a subset of the system simulation tests, and many of the same configuration, testbench command and input files can be used during both subsystem and system simulations. This will save time and reduce risks of slips against the plan.

1.7.2 Project Leader Tasks

When the subsystem simulations start, there is often a number of small issues to be resolved to get the first simulations working. The project manager should hold a short meeting every day, so that the issues can be discussed and priorities set. It is very easy for small problems to lie unresolved. Identifying, prioritizing and tracking issues and problems on a daily basis will lead to faster progress and minimize bottleneck hold-ups where a significant number of the team members can be held up by the same problem.

The subsystem simulation phase is the first point where the testbench engineer's work is being used and relied upon. From working in a relatively independent capacity, these engineers suddenly find themselves the focus of attention. They are expected to act in a "help-desk" type role to get the first simulations running and are expected to sort out bugs in the testbench quickly. Any such bugs can hold up several team members at the same time. This is a stage where the testbench engineer is often under considerable pressure from the rest of the team. Given that development of the testbench is frequently assigned to more junior members of the design team who are likely to be less familiar with and less equipped to deal with such pressure, the project leader should ensure that the pressure does not become unreasonable. The daily progress meetings at this stage are a good way of monitoring this and keeping it under control.

The ASIC development plan should be updated at the start of the subsystem simulation phase with a list of the major test areas. The plan should have a milestone of "first subsystem simulation working." This is a significant achievement. It always takes a few days to get the first test working. Tracking each test closely provides a good measure of progress and regular (possibly daily) subsystem test meetings tends to encourage the team to resolve the initial problems encountered quickly.

In parallel with the subsystem testing, work will be carried on combining the entire individual design modules and subsystems together in an integrated full chip-level netlist. For all but the simplest designs, this initial netlist will almost inevitably require rework following simulation. However, it can and should be used in its initial form to carry out trial layouts.

1.8 System Simulation/Synthesis

Outputs from This Phase:

- First system simulation runs successfully
- Reviewed system simulation specification available
- All RTL and gate-level simulations completed

- Initial synthesized netlist available (trial netlist)

Tasks:

- Write and review test list document.
- Write test "pseudocode" (i.e., CPU register accesses, testbench configuration).
- Run RTL and gate-level simulations.
- Record and track resolution of bugs using a fault-reporting system.
- Check chip design rules.
- Write chip device user guide.
- Create synthesis scripts and first trial netlist.
- Create layout floor plan and documentation.

Project Manager-Specific Tasks:

- Closely monitor plan; arrange regular, quick meetings to discuss progress.
- Arrange layout meeting with ASIC vendor.

Risks:

- Poor communication between engineers can significantly increase the time required to get first system simulation running.

The two main tasks during this phase are the top-level simulations and the synthesis of the netlist. As with some of the earlier phases, it is useful to examine separately the activities of the team and those of the project leader. Many tasks during this phase will overlap with the layout and the backend phase that is covered in the next subsection.

1.8.1 Tasks for the System Simulation Team

A number of tasks must be completed before system simulation can start. A prerequisite of the top-level simulations is a full top-level netlist that includes the I/O pads. The top-level netlist should be carefully reviewed prior to the start of the simulations to eliminate simple connection errors. The synthesis team typically generates this netlist.

The top-level testbench is a further prerequisite for system simulations. The testbench engineer should provide a basic user manual for the testbench, detailing such things as how to set up testbench configuration files and how to set up different operat-

ing modes for complex transactors. The testbench engineer should also give a formal presentation to the team before simulation begins, explaining the architecture of the testbench and how best to use it.

It is usually a good idea to split the team into a number of subteams of one to three people each for each of the areas in system simulation. Each team is assigned a different area to focus on during the system simulations. Each team must define a list of tests that comprehensively test its area. The list of tests should be reviewed by the full team to ensure that there are no obvious omissions. The test list will typically include a subset of the tests carried out during unit and subsystem simulations as well as any additional tests that can be run only at the chip level. For some modules, the time taken to create truly representative input stimuli at the module level can be prohibitively long. For these modules, chip level testing is the only way to validate the module fully.

The system simulations should be run first on the RTL code and subsequently on the gate-level netlist, when the gate-level netlist is available from the synthesis team. The RTL simulations should be carried out first, because these simulate faster and the RTL description is always available before the gate-level netlist. The RTL simulations prove that the entire chip implements the functions and algorithms defined in the top-level specification. Gate-level simulations are discussed in the layout and backend phase that is covered in the following subsection.

For large designs, where it is difficult to plan the coincident completion of all blocks, initial top-level system simulations can be done without all the modules being available. In this case, simple dummy modules to allow the netlist to elaborate or compile can replace any missing modules.

The project should develop some guidelines to ensure a systematic and consistent approach to system simulation. These can be drawn from the best practices and experiences on previous projects and should be improved and updated with each passing project. Among other things, the guidelines should cover test directory structure, test naming conventions, netlist version management, result logging, etc. One method of approaching directory structures is to have separate directories for each test category. These can be further divided into subdirectories, as appropriate. Ideally, the test file names will match the test names in the test list to allow easy cross referencing. For example, a test file named *error_corrector_4.3* would correspond to test 4.3 in the test list. Clearly from its title, it involves testing some aspect of the error corrector.

It is important that tests are run on the latest version of the netlist. There are, of course, limits to this. If the netlist is being updated on a daily basis, it may not be practical or efficient to rerun all tests on all new versions of the netlist. In such a scenario, perhaps releasing the codebase once a week, for example, is more appropriate. The

RTL and gate-level code should be under official revision control, and the simulation teams need to understand the simulation directory structure and to know how to link to relevant versions of the netlist. During the system simulations, changes to the VHDL or Verilog code should be done in a controlled way so that everyone knows when changes have been made and what tests they are likely to affect. Time is often lost trying to understand why a particular module under test has stopped working when the problem has been caused by changes to other modules. The team should agree when and how code is released, and who is allowed to release it. These issues are discussed further in Chapter 7, "Quality Framework."

Officially tracking problems and their resolution is essential to ensure that problems are fixed before the final netlist is sent for fabrication. The project manager can also use tracking statistics as an indication of how well the netlist has been tested. At the start of system simulations, the rate of problems identified will be high. Toward the end, the rate should tend toward zero. Clearly, the netlist should not be released if the rate of problem detection is still rising. Last-minute changes to the netlist may not affect all modules. In such cases, it is valid to rerun only a subset of system simulations to reduce timescales.

There is a number of steps that can be taken to reduce the duration of the system simulation phase. Create self-checking scripts that will automatically run a series of system simulations and log results. This is useful for rerunning the tests following a change to the netlist. Because the tests are invoked via scripts, it is easy to set them off overnight, for example, instead of having to run each in series interactively. Wherever possible, use the same tests for RTL and gate-level simulations. Limit the use of internal probing and node forcing. Nodes that are available in the RTL code may not be available in the gate-level netlist. Start trial system simulations early, with the testbench engineer and a single design engineer. This ensures that time taken to iron out the basic testbench problems does not affect the entire system simulation team. It is a form of testing the testbench and gives an early indication of how user-friendly the testbench is and whether the documentation on using the testbench is adequate.

Typically, the system simulations will take several man-months of effort to complete. For complex designs, the system simulations can never fully test the netlist without incurring very long simulation times. In these cases, an early sign-off can significantly reduce the time before production silicon is available. The idea here is to fabricate the netlist before the all system simulations have been completed. However, a subset of both RTL and gate-level simulations must have been completed before the early sign-off takes place. This early silicon is used to test the functionality of the design rather than testing at all environmental conditions. Therefore, the netlist can be

sent for prototyping before the gate-level netlist has achieved the required synthesis speed. The advantage of this approach is that, once the trial silicon is available, testing with test equipment is much faster than testing by simulation. Consequently, a very large number of tests can be executed in a fraction of the equivalent simulation time. There should be a planned second version of the silicon that is fabricated only after initial testing of the first silicon has been completed. Avoid the tendency to fabricate the second silicon too early before the initial testing has been completed, because bugs will often be found toward the end of the testing.

Toward the end of the top-level system simulations, the list of tests and the results should be carefully reviewed, because the quality of these tests have a direct bearing on the success of the project. Often, as the test teams become more familiar with the complete design, they will come up with additional tests that were not thought of at the time of formulating the initial test list. If these new tests are testing functionality that was not otherwise being tested, they need to be added to the test list.

1.8.2 Tasks for the Synthesis Team

Synthesis is the process of translating the RTL description into a gate-level netlist using a synthesis tool. The synthesis team has the job of integrating the top-level modules together into a top-level RTL description, then generating the gate-level netlist from this. The top-level RTL should be fully reviewed, paying particular attention to the pads. The definition of the power pins is particularly important because they cannot be fully simulated. Synthesis scripts are used to guide the synthesis tool and drive the synthesis process. The scripts provide information to the tool on physical and timing characteristics of the circuit and its signals. This information is essential to enable the tool to map the RTL code properly to a gate-level netlist that meets the required timing characteristics. The scripts contain information on clock signals (period, phase, duty cycle, aspect ratio, relative phase, etc.), input signal driving strengths, output signal loads, input and output timing requirements, multicycle and false paths, etc. The top-level synthesis scripts can be relatively complex and take a significant amount of time to perfect. Fine-tuning and tweaking of input parameters and constraints is usually required to get the desired end result. Synthesis and synthesis scripts are covered in Chapter 6, “Synthesis.” The scripts should be reviewed internally by members of the design team and in-house synthesis experts. They should also be reviewed externally by the ASIC vendor and the synthesis tool application engineers.

Test insertion and test vector generation are usually considered part of the synthesis process. Test vectors can take a significant amount of time to generate and simulate. The test vectors must be run on the gate-level netlist using best-, typical-, and worst-

case timings and can take many days to simulate, depending on the size of the netlist and the processing platform. Test insertion should be done as soon as the top-level netlist is available. This allows the maximum amount of time to get the tools working correctly. After test vector generation, the fault coverage can be assessed. Doing this early in the system simulation phase allows time for possible changes to the code to increase fault coverage. Functional vectors and parametric tests must be planned into the synthesis phase. These tests can be tricky to generate and can cause delays to the ASIC sign-off, if left to the last moment. Some ASIC vendors require extensions to the basic Verilog simulator to generate the test vectors in the required format. This requirement should be identified early and the extension incorporated and tested in the design environment.

The layout flow should be discussed and agreed on between the team and the ASIC vendor early in the project. Trial netlists should be sent to the vendor at this stage and the results analyzed (see the following subsection on postlayout simulations). The layout will be improved by ensuring good communications between the ASIC team and the ASIC vendor. The design engineers and the layout team should jointly develop a floor plan, and layout information should be generated and documented. Relevant information includes the clocking strategy, identification of blocks requiring specific placement on the die (typically for speed or noise reasons) and identification of critical paths. Timing-driven layout is a useful method of increasing the timing performance of the final silicon by focusing on the most time-critical paths first. Some synthesis tools are capable of automatically generating constraint files that can be read by the timing-driven layout tools. For deep submicron designs, the synthesis and layout tasks constitute a significant area of risk in the project. This is because the technology geometries are so small that interconnect delays become the dominant part of timing delays, rather than the gate delays. This can result in large differences between prelayout and postlayout timing figures, consequently making it difficult to achieve timing closure on the design. New tools that take floor-planning information into account as part of the synthesis operation are starting to appear on the market to address this particular problem. For large deep submicron designs, it is essential to agree early on with the ASIC vendor how the timing closure will be managed. It requires closer cooperation with the ASIC vendor than was traditionally the case in previous generation designs and requires careful planning of the division of roles and responsibilities between the ASIC vendor and the design team.

1.8.3 Project Leader Tasks

The ASIC team leader faces the same sort of issues during the system simulation

phase as were faced during the subsystem simulation phase (see above). Getting the first system simulation running properly is a major milestone. It is advisable to hold daily progress meetings from the start of this phase to identify and track problems. Common problems include testbench bugs or functionality limitations, inadequate testbench documentation, intermodule interface bugs, lack of hardware resources (hard disk storage and workstation memory), shortage of available simulation licenses, etc. These problems need to be prioritized and the task of resolving each problem assigned to various team members.

The management of the ASIC vendor becomes critical during this stage. This is covered in the following section, "Layout and the Backend Phase." Understanding the layout flow can help to reduce timescales. The project manager should arrange meetings to agree on the layout flow and discuss layout issues, synthesis scripts and delivery dates. It is useful to have an understanding of the task breakdown in the vendor's layout group and how it is addressing these tasks.

1.9 Layout and the Backend Phase

This section documents those tasks that are the responsibility of the ASIC vendor.

Outputs from the ASIC Vendor:

- Postlayout timing/capacitance information
- Information for silicon manufacturing

Tasks for the ASIC Vendor:

- Layout of the trial and final netlists
- Checking the netlist and test vectors for errors
- Generation of postlayout timing/capacitance information

This phase deals with the layout work that is carried out by the ASIC vendor. It also covers the work that the design team must do to support the layout team and the tasks that must be done following the layout. This subject is described in more detail in Chapter 10, "Dealing with the ASIC Vendor." This phase overlaps many other phases because layout should start as soon as an initial top-level netlist is available. Indeed, some critical modules may need layout before the top-level netlist is available.

The ASIC vendor takes the VHDL or Verilog netlist and converts it into a physical layout. The process involves a number of complex tools, and the level of risk increases

with the size and speed of the design. The risks are further increased if the ASIC is substantially different from ones that the vendor has previously processed. Differences could include different technology, very large gate counts, or the use of complex compiled cells, etc.

Trial layouts will reduce the risks by identifying any potential problems as early as possible. The project should plan for at least two trial netlists, and the number should be agreed on with the ASIC vendor at the start of the project. It is important that the trial netlists are passed through the complete layout flow.

The first trial layout should identify any major problems, such as design rule errors, modules that cannot be routed and major postlayout timing violations. The first trial netlist does not need to meet all final timing requirements. However, the extent to which timing requirements are violated at this stage will give an idea of how realizable the design is. It is also useful to examine the extent to which prelayout and postlayout timings differ.

The second trial netlist should meet all timing requirements. Meanwhile, some initial system simulations should have been run on the RTL and gate-level code. If the project is lucky, this second trial netlist becomes the final netlist, and the sign-off can be completed earlier than planned. If simple bugs are found during the system simulations, these can often be fixed in the layout, using an engineering change order (ECO). These ECOs define gate-level changes to the netlist, which can be done manually by the layout engineer. The ECO approach varies for each ASIC vendor, and the options available for ECOs should be discussed at the start of this phase.

The project team and the ASIC vendor should agree on a release naming convention for the VHDL or Verilog code. It is important that the ASIC vendor has enough time to complete the layout flow before the next netlist is sent, so the number of revisions should be kept to a minimum, even if the netlist seems to be changing on a daily basis.

The layout process involves floor planning, cell placement, clock-tree insertion, routing and timing analysis. Postlayout timing information is generated as part of the layout process, and this is used for postlayout simulation and synthesis.

As mentioned in previous sections, timing delays can be significantly longer after layout, especially for deep submicron technology. It may take several iterations of design, synthesis and layout to meet the target timing requirements. Newer synthesis tools can generate timing and routing constraints that are used to guide the placement and routing layout tools.

1.10 Postlayout Simulation/Synthesis

Outputs from This Phase:

- Final netlist sent for layout
- Test vectors (IDDQ, scan and functional) tested and sent
- Postlayout simulations and static timing completed
- Chip sign-off

Tasks:

- Synthesis, test insertion and test vector generation
- Generation of a layout document
- Support layout (floor planning, checking timing, etc.)
- Resynthesis after layout (fixing overloads, timings violations)
- Running gate-level simulations and static timing analysis with final netlist, using back-annotated timings

Project Manager-Specific Tasks:

- Arrange meetings with layout engineers/synthesis engineers.
- Review progress/milestones of layout.

Risks:

- Pin-out errors are common—review this several times.
- There may be issues with layout (routing, timing after layout, etc.)—do trial netlist as early as possible.
- Test vector generation can take a long time—start generation of vectors early.
- Gate-level simulations are polluted with unknowns (Xs)—emphasize the need for reset conditions on all registers early in the design.

During layout, a number of files are generated that can be used to analyze the impact of the layout on the circuit. The files are back-annotated onto the netlist, and some processes are rerun. There is a number of steps in the layout process. Some of these files can be extracted after the initial layout stages of floor planning and cell placement. This gives an indication of the layout but is not yet fully accurate, because it

does not contain the routing information. After the layout has been completed, a custom wire-load file, a capacitance load file and a standard delay format (SDF) file are generated. Normally, an updated netlist is also sent back by the vendor, because the clock tree will have been added during the layout.

The custom wire-load gives the average wire length for a module or chip. Before a custom wire-load is available, the synthesis tool uses an estimated wire-load file. The wire-load values in this are estimated based on the synthesized gate count. The custom wire-load file from a trial placement can be used as the starting point for a complete resynthesis, if desired. Alternatively, the custom wire-load from a final or almost final layout can be used in combination with an SDF file for minor resynthesis or in-place optimization. In this scenario, the SDF timing information is used for all cells and nets that do not change, whereas the wire-load file is used in calculating delays for the small number of new nets created in the resynthesis.

The SDF file defines the time delays between each pin in the netlist. This includes delays through the cells and interconnect delays. It is used for both synthesis and simulation. The synthesis tool will use it to do static timing analysis and for resynthesis to solve timing violations. At this point, the synthesis script should now use a propagated clock, rather than the ideal clock that was used prelayout.

Postlayout timing analysis is a critical stage in the project and should be carried out as quickly as possible and the results analyzed. This is typically done using a static timing analysis tool. Results from the trial netlist could indicate the need for a different layout, changes to the code, or resynthesis. With current small geometry technologies, the interconnect delay is increasingly important in defining the critical timing path in large gate-count designs. The static timing analysis will highlight particularly long nets. Sometimes, altering the floor plan or cell placement, or simply rerouting some nets can resolve timing violations. Another approach to resolving timing problems is to resynthesize with back-annotated SDF, custom wire-load and capacitance files. The synthesis tool will try to meet the timing requirements based on the actual delay and capacitance file.

Anything other than trivial changes to the code to fix timing problems will inevitably require changes to the plan and possibly also changes to the test list. The test plan can be rescheduled to focus initial tests on parts of the design that will not change. This minimizes the effect of any redesign on the testing schedule.

The capacitance file defines the capacitance that each cell output drives. The technology library will define a maximum capacitance that the outputs can drive. If this is exceeded, the ASIC vendor's design rule check (DRC) will fail. The synthesis tool can read the capacitance file and rebuffer the overloaded nets. It can be set to allow

rebuffering only (preferred, because it has least impact on the layout but might not fix all violations) or to allow restructuring so that an overloaded net is replaced by a number of parallel nets. The overloaded nets should be fixed before timing analysis is done, because the overloaded nets result in poor timings. Traditionally, the project team, rather than the ASIC vendor, fixes design rule violations, but some ASIC vendors have tools that can automatically fix design rule violations that have arisen as a result of layout.

Tip: The cell timing models are based on standard industry-defined worst-case temperature and voltage. If the design team is confident in advance that the power consumption or operating voltages are, in reality, going to be more favorable than the standard worst-case conditions, custom worst-case operating conditions can be used for the cell timing models. This helps to achieve the required timing performance. If the cell models are not designed to take variable operating environment parameters, the ASIC vendor will need to provide a custom synthesis library. Even if the cell models can take variable operating environment parameters, it is important to establish with the ASIC vendor that the models are sufficiently accurate at these nonstandard operating points.

1.11 ASIC Sign-Off

Outputs from This Phase:

- Chip sign-off

Project Manager-Specific Tasks:

- Check sign-off documentation.
- Get agreements from different departments for silicon quantities.

After postlayout simulations and synthesis have been completed, the netlist is sent for fabrication. This is referred to as *ASIC sign-off* because there is typically a document that is signed by the team and the ASIC vendor. The document clearly states the netlist revision ID, the required test vector files, agreed sample quantities and commercial aspects. This document should be studied several weeks before the sign-off date so that all items can be completed on time. Representatives from a number of departments, including development, manufacturing and marketing, must complete the document.

Before sign-off, the ASIC vendor must run checks on the netlist, layout and test vectors. The vendor will normally require that the test vectors be simulated before

sign-off. This can be a lengthy process. With some persuasion, the vendor may agree to send the netlist for fabrication before these simulations have completed. However, because this prevents the vendor from properly testing the fabricated silicon, the vendor will disown responsibility for fabrication problems until the test vector simulations have all passed.

1.12 Preparation for Testing Silicon

Outputs from This Phase:

- Reviewed evaluation plan
- Test equipment commissioned before silicon available
- All tests prepared (hardware, software and automation)

Tasks:

- Write and review evaluation test list/plan.
- Write tests.
- Plan and implement test automation.
- Reserve test equipment.
- Design or outsource design of nonstandard test equipment.
- Commission test equipment before silicon is ready.
- Define requirements for evaluating silicon at different voltages and temperatures (environmental testing).
- Define method for recording, analyzing and resolving bugs.

Risks:

- This can be a time-consuming task—allow appropriate planning and start early.
- The specification for the ASIC is not available/inaccurate when planning this phase early in the project.

Preparation for testing of the silicon is, in principle, a straightforward task. However, it is all too easily ignored while focusing on planning the more complex stages of the ASIC development cycle.

The time required to verify the silicon can be significantly reduced if some preparation work is carried out before the device arrives. The preparation usually requires

significant effort and should be carefully planned. The work is often done in parallel with top-level simulations, and the risk is that it may be pushed aside as more pressing issues with the simulations arise. To avoid this, the preparation tasks are ideally assigned to engineers who have little involvement with the top-level simulations. The preparation requires the involvement of a number of different engineering disciplines, including ASIC, software, PCB and test engineering.

There is a number of aspects to preparation for testing. Laboratory space must be booked in advance. Standard and specialized test equipment must be booked and purchased or hired, if not available in-house. Special attention should be paid to the purchase of specialized test equipment, because lead times may be long.

Test PCBs must be designed, manufactured and populated in advance if test boards, rather than IC testers, are being used. If the ASIC is to form part of a production PCB, the design of the board should be done in conjunction with the ASIC engineers. The ASIC pin-out should be defined with help from the PCB designers because the pin-out can have a major impact on the PCB track routing. IO timings, output pad load capacitance, output voltage levels and input requirements should be discussed with the PCB designers. The ASIC should contain circuitry to allow easy debugging. This should allow access to internal status registers, access to internal nodes via output test pins, built-in self-test, error-checking circuits, etc. It is important that these debugging features are discussed with the PCB designers so that test equipment can be easily connected to the ASIC. IC clips can be bought that fit over some ASICs, giving access to the chip inputs and outputs. However, for high pin counts, these can be very expensive and not fully reliable.

Modern ASICs can often have packages with very high pin count requirements. There is a range of packages available, such as BGAs, micro BGAs, flip-chip, or TAB. These give cheap, reliable packaging but require some experimenting to achieve high yields in the PCB assembly process. The production team can practice on empty packages before the silicon is available, which ensures that the first samples are mounted on the board as quickly and reliably as possible. It is important that the number of PCBs required is agreed on at an early stage to allow time for the purchase of components and for manufacturing resource to be allocated.

Development of test software must be planned and the code written and tested as well as it can be tested in the absence of the silicon. The testing can be done using a software emulation model of the ASIC/board. Alternatively parts of the test software can be tested within the simulation environment (see Chapter 5, “ASIC Simulation and Testbenches”).

Automating the tests will significantly reduce test times but, of course, requires

additional effort. The automation can be done using specialized test equipment or based on PCs/workstations with application-specific test software.

The test software is based on the test list specified by the design and test teams. The test list should be reviewed by the team to ensure that it is adequate to test the ASIC comprehensively. One approach to creating the test software is to provide a set of common low-level driver routines for setting up various chip configurations and register settings. If these routines are sufficiently comprehensive and well documented, it is then easy for members of the test and design teams to write a layer of higher-level test code to sit on top of these. For register-intensive designs with many possible operating modes or configurations, it can be useful to design the test code to initialize the chip based on test configuration files. Changing the operating mode can then be as simple as changing a number of ASCII text fields in a configuration file.

SoC or large designs may have IP blocks from third-party vendors. These blocks should be initially tested wherever possible in a stand-alone mode based on a test suite provided by the vendor. Once the block has been tested in a stand-alone mode, it can then be tested operating within the integrated device. The stand-alone tests may require specialist test equipment, and this should be sourced and tested well before silicon is available. Some IP providers produce reference boards that can be used to check the test equipment and test setup.

Sometimes, part of the test list includes interoperability testing. This means testing the silicon within its system environment with a range of existing products. These tests must be defined at an early stage so that the existing equipment can be acquired.

During testing, problems will be found. There is a number of potential sources of such problems, such as PCB errors, software bugs, test equipment failures, etc., in addition to any possible problems in the silicon itself. However, each failing test must be recorded in a database that can be easily accessed by the entire team. The fault-tracking database must be created and the bug tracking process defined. A template for the test report is also needed before the start of the evaluation.

1.13 Testing of Silicon

Outputs from This Phase:

- Fully tested silicon working in a real application
- Test report generated and reviewed

Tasks:

- Run the tests.
- Track test failures in a fault report database.
- Analyze failing tests.
- Identify workarounds for ASIC bugs.
- Identify netlist changes for ASIC bugs.
- Evaluate silicon at different voltages and temperatures (environmental testing).
- Do interoperability testing.

Risks:

- A shortage of working PCBs can significantly increase the time to complete the initial testing of the silicon.
- Modifications to the PCBs, as part of the debugging process, can make them unreliable, resulting in a shortage of working PCBs.

The preparation for the testing of the silicon should result in a test list and project plan for this crucial phase. The silicon will typically need to be mounted onto a printed circuit board. Ideally, a number of printed circuit boards with the new ASIC should be manufactured. It is important that an adequate number of working PCBs are available for the ASIC, software and PCB engineers.

The first tests should prove the basic operation of the silicon. Initial problems will always be encountered when running the tests. However, these may not be due to the design. Faults in the PCB, problems with manufacturing of the silicon, assembly of the board, software bugs or faulty test equipment can all cause tests to fail. To identify the problem area quickly, there should be an agreed-on approach when tests fail. Some options are suggested below:

- Run the same tests on multiple boards and ensure that the failure mechanism is repeatable.
- Check the test setup.
- Check register configurations (read the internal registers).
- Check internal ASIC debug registers.
- Check external ASIC signals and internal debug signals that can be multiplexed to output pins using logic analyzers and oscilloscopes.
- Create a simulation test that mimics the failing test.
- Analyze the VHDL/Verilog code for the failure mechanism.
- Brainstorm ideas for devising different tests that will identify the problem.

- Test at different voltages and/or temperatures to determine whether the problem is related to environmental conditions.

Which of the above options are followed and in what sequence will depend on whether the failure occurs during the initial tests or after many tests have been successfully run. Test failures should be noted in the fault report database.

One of the most important administration tasks is to keep an accurate track of the location and status of the PCBs. Whenever a test is finished, the result and the serial number of the board should be noted in the test report. This allows the test setup to be reproduced at a later date when tracking problems. A spreadsheet should be created which defines the status and location of each PCB, and this should be available on a common server/intranet. It is useful to record which boards have passed/failed which tests on a regular basis. It is also important to record the modification state of the PCBs. This is often the responsibility of the PCB engineers. These engineers should also be responsible for having PCB manufacturing problems fixed.

1.13.1 Project Leader Tasks

The evaluation of the silicon is a complex task because it involves ASIC, PCB designers and software engineers all working together. During the planning of the evaluation, the resources from the different disciplines should be identified and tasks assigned. The ASIC leader should ensure that adequate PCB and software resource is available, especially at the start of the evaluation.

For the first few days of the evaluation, the ASIC leader should check the progress several times a day. A list of issues and problems should be maintained, which should be analyzed daily to assign priorities and resources. After a few days, progress can be tracked daily. If a major bug has been identified within the silicon, the ASIC leader should arrange a meeting to brainstorm ideas for solving the issue without the need for changes to the ASIC netlist. It is important that PCB designers and software engineers are present at the meeting because they may be able to solve bugs by changes to the PCB or software. These types of changes allow the product to be launched earlier than if an ASIC respin is needed. However, they can often have some impact on the performance of product features or product cost. The ideas and consequences should be presented to marketing engineers and senior managers. Changes to the ASIC netlist should also be identified, and the ASIC leader should estimate the time before new silicon would be available.

The project leader should ensure that each engineer is completing the test reports and filing problem reports into the database. These can be somewhat tedious tasks, but

are necessary for the successful completion of the product. The list of open bug reports should be reviewed periodically with senior management. Two ratios are useful as a guide to the level of testing and the confidence in the silicon. The first is the number of completed tests, compared with the total number of tests. The second is the rate of finding bugs, compared with resolving bugs.

On completion of the silicon testing, the test results should be reviewed by the team.

1.14 Summary

With more and more of the electronic circuitry in new products integrated into ASICs, the ASIC design activity in a new product development can appear a daunting task to senior managers, project team leaders and ASIC specialists alike—the latter often being expert in one or more parts of the ASIC development process but lacking an adequate understanding of other parts. However, like most engineering problems, the way to approach this one is to break it into a series of smaller, more manageable tasks or phases. Each phase consists of a series of related design and support activities that can be collectively grouped and classified under one heading.

This first chapter attempts to capture the entire ASIC development process in overview form by dividing it into a number of mainly sequential phases and describing the main aspects of each such phase. It is a useful chapter for the entire project team to read before embarking on a new ASIC development. More detailed information on specific phases is provided in the subsequent chapters of the book.