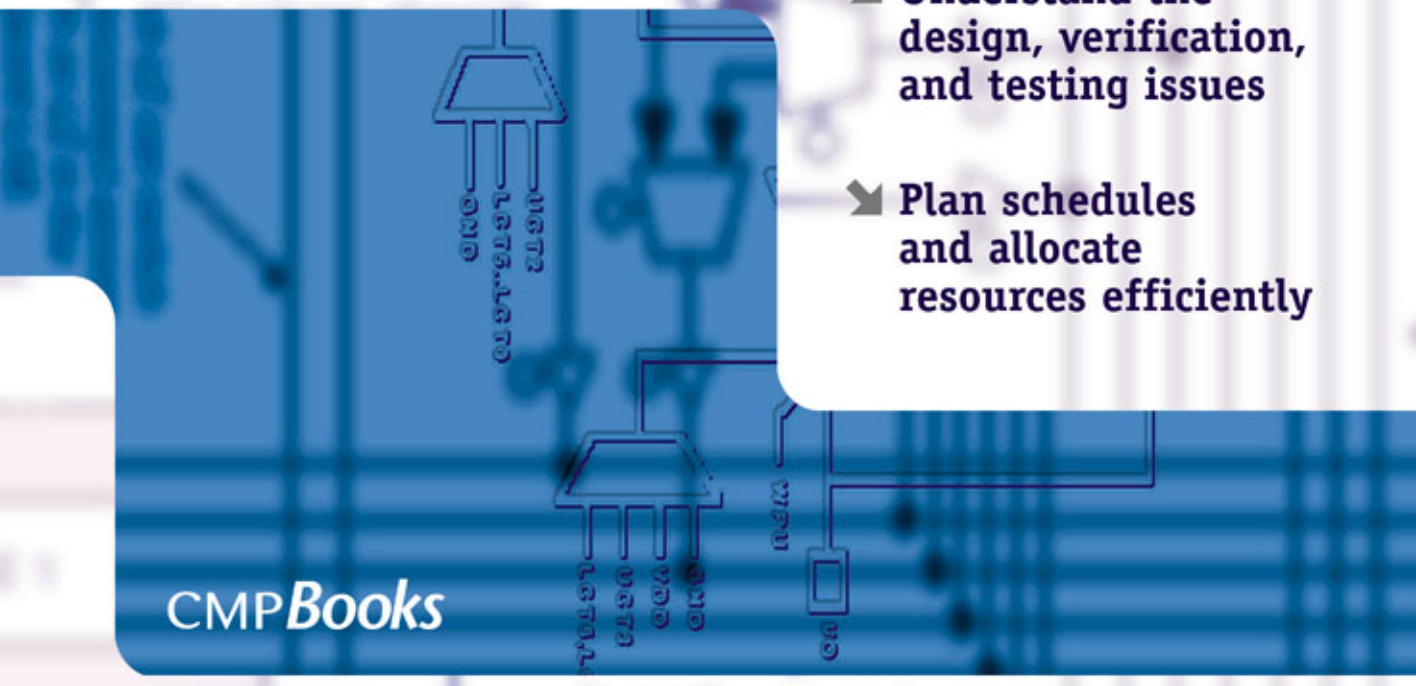
Designing with FPGAs & CPLDs

- Choose the right programmable logic devices and development tools
- Understand the design, verification, and testing issues
- Plan schedules and allocate resources efficiently



CMPBooks

Bob Zeidman

In this chapter...

- *UDM and UDM-PD*
- *Specification and specification review*
- *Devices and tools*
- *Design, simulate, and review*
- *Testing*

Chapter 4

Universal Design Methodology for Programmable Devices (UDM-PD)

This chapter describes the Universal Design Methodology for programmable devices. This method, based on my years of experience designing many types of programmable devices for small and large companies in different industries, lays out the entire process for designing a programmable device. Following the UDM guarantees that you will not overlook any steps and that you will have the best chance of creating a chip that functions correctly in your system.

Objectives

In this chapter, using Universal Design Methodology as a reference model, you will learn what steps are necessary to create working, reliable chips that function correctly in your system. The first section of this chapter explains UDM and specifically UDM-PD for programmable devices. The following sections detail each step of the design process and how it relates to the method. Reading this chapter will help you:

- Understand the UDM and UDM-PD methods.

- Discover the importance of writing a specification and performing a specification review.
- Learn how to choose appropriate programmable devices and software tools based on your specification.
- Recognize the issues to consider when synthesizing your design.
- Understand the need for proper simulation, review, and testing techniques during and after the process.

4.1 What is UDM and UDM-PD?

After years of designing circuit boards, ASICs, FPGAs, and CPLDs for a large number of companies, I noticed that engineers rarely followed a complete methodology, i.e., a complete set of standard procedures and steps. In many cases, companies simply assumed their engineers somehow knew what to do next — and the outstanding engineers often did. If the methodology was in some brilliant engineer's head, though, when that engineer left the company the methodology left, too, leaving the company to rediscover and redevelop the knowledge through costly trial-and-error. In those companies that had established procedures, the process documentation often consisted of scattered notes and files; rarely was the documentation compiled in one place.

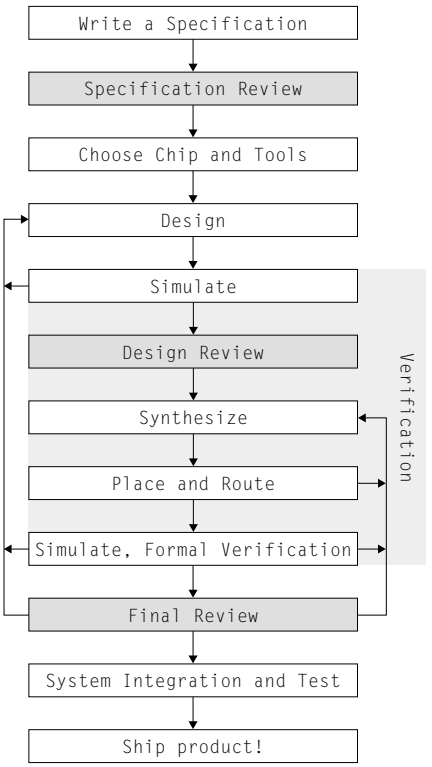


Figure 4.1 Design Flow

Even in companies that followed a defined procedure, often parts of the procedure were not well understood or completely implemented. For example, a company might emphasize the design process and not understand the testing process.

Eventually, as a consultant going from company to company, I decided I needed a methodology that I could take with me for each new design. It would have to be one that applied universally to large and small companies in any industry. Rather than being a method that described very specific steps, it would need to be a methodology that could be generally applied to different designs. I called this methodology the Universal Design Methodology. As technology

changed, the methodology was adapted. I did find that different types of devices required slightly different methodologies. ASICs and printed circuit boards required slightly different methodologies than FPGAs and CPLDs. The methodology described here is specifically for programmable devices, and I call it UDM-PD.

4.1.1 The Goals of UDM

The goals of the Universal Design Methodology are these:

- Design a device that
- Is free from manufacturing defects
- Works reliably over the lifetime of the device
- Functions correctly in your system
- Design this device efficiently, meaning
- In the least amount of time
- Using the least amount of resources, including personnel
- Plan the design efficiently, meaning
- Create a reasonable schedule as early in the process as possible
- Know all necessary resources up front and allocate them as early in the process as possible

4.1.2 The Design Flow of UDM-PD

UDM-PD specifies a design flow that enables you to reach the UDM design goals when designing a programmable device. The design flow consists of the steps shown in Figure 4.1. Each particular design will have slight variations in the specifics of each step, but essentially the steps will be the same. By understanding the entire flow before the project begins, you will be able to meet goal number three, a better understanding of how to schedule and allocate resources for the development process.

A Living Document

My last full-time employer (before becoming a full-time consultant) was a telecom startup. The company had lots of problems and became a great example of what not to do when designing a product or running a company.

One of these problems should have been an early warning sign for me. I joined the company as the main ASIC designer--employee number nine and the only one with any experience designing chips. One of the chips that needed to be designed was a simple buffer between a board and a backplane. The logic requirements of the design were extremely simple, but the current drive requirements — 48 mA per signal for 8 signals — was more than most ASIC vendors could produce at the time. We found that one company, NCR Corporation, had the technology. Because the logic design was so simple, we decided that I would produce a written specification that would be handed off to the engineers at NCR. I would supervise their progress while I did the real design work on another important chip.

At the end of the design review, one of the engineers at the company adamantly stated that we should demand that NCR meet the specifications that we had written. If not, he proclaimed, we would not pay them. That would give them the incentive they needed to design the chip to our specifications. I said that it made more sense for us to work with them to understand any issues that might arise. If NCR couldn't meet the specification, we would work with them to get the best results possible and see how we could change the specifications, and the specifications of our system, to get the best results possible. He was insistent. We argued for some time, the other engineers at the review sitting quietly and uncomfortably. Finally, I told him that I agreed. If NCR couldn't manufacture the chip, we should refuse to pay them. We would then save the $20,000 NRE fee. He smiled. I then said that we could all go home because we had saved all that money, had no chips for our boards, no working product, no company, and no jobs. I think I then called him an idiot.

Luckily the president of this company saw my logic immediately and agreed that we should work with NCR to produce the best product that we could. If, along the way the specifications needed to be changed, we would change them. The written specification is not gospel; it's a living, changing roadmap that gets you where you're going even if a lightning storm suddenly throws a tree along the road. You must be specific enough to create your design, and flexible enough to make changes when necessary.

And by the way, don't call anyone an idiot at a company meeting, even if it's true. It makes an enemy for a long time — an enemy who can take revenge in secret ways, refuse to cooperate on important projects, and blame you for all of his mistakes. I'm speaking from experience.

4.3 Specification Review

At the end of the specification phase, it is very important to have a design review. As many people as are interested should take part in this review, because this specification is the basis for the entire chip. You want to know right now if anything is wrong or has been left out. Invite people from all departments who are specifically involved with the chip and the system, including marketing, sales, software, applications, etc.

4.4 Choosing Device and Tools

Once a specification has been written, the design team can use it to find the best vendor with a technology and price structure that best meets their requirements. They can also choose tools to work well together and with the device and technology they have chosen.

One important case in point for choosing the device and tools at this early stage is that of synthesis. The selection of a synthesis tool will determine the coding style for the RTL HDL. A specific coding style is necessary because synthesis tools cannot do a good job if they must handle too many code possibilities. For example, there are many ways to code state machines. If the synthesis tool needs to understand each possible way, it becomes more complex and thus more error prone. To avoid problems like this, synthesis tool vendors specify coding guidelines guaranteeing that their tools will recognize the structures you are designing and synthesize them correctly to gate level descriptions.

4.5 Design

When designing the chip, remember to design according to the rules that are discussed in detail in Chapter 5. These are

- Use top-down design
- Work with the device architecture
- Do synchronous design
- Protect against metastability
- Avoid floating nodes
- Avoid bus contention

4.6 Verification

Verification is a “super-phase” because it consists of several other phases of the design process. Which exact phases that make up verification is open to argu-

The Correct Way To Calculate Power Consumption

Engineers often ask me about power calculations for circuit boards or systems. When they add up the worst-case power consumption numbers for all of the parts in the system, they end up with a number so large that it seems unreasonable. They can't justify designing in a power supply that large, or the necessary cooling needed to get the predicted temperature down to something acceptable. When they put the system together, the power numbers are often much closer to the typical power consumption numbers than the worst-case power consumption numbers. Many engineers simply shrug or apply some rule of thumb such as cutting the worst-case power number by a factor of two. Some engineers are bothered by the fact that the numbers are off. Some know the secret to correctly calculating worst-case power consumption, and I'm about to reveal that secret to you.

The power numbers which manufacturer's report for individual components are statistical numbers. The best case and worst case numbers are usually values three sigma in either direction from the mean (looking at a standard Bell curve). In other words, there is an insignificant probability that the power consumption is more than the worst-case numbers or less than the best-case numbers. If you treat the individual components statistically, you should treat the entire circuit board or system statistically.

Therefore, to get a realistic number of the best-case, typical-case, and worst-case power consumption for a system, use the following formulae (which are standard formulae for the mean and standard deviation of a discrete random variable):

Minimum system power $P_{min} = \sum P_{i,typ} - \sqrt{\sum (P_{i,typ} - P_{i,min})^2}$

Typical system power $P_{typ} = \sum P_{i,typ}$

Maximum system power $P_{max} = \sum P_{i,typ} + \sqrt{\sum (P_{i,max} - P_{i,typ})^2}$

where

$P_{i,min}$ is the minimum power of component i

$P_{i,typ}$ is the typical power of component i

$P_{i,max}$ is the maximum power of component i

Using the formula for worst-case maximum power consumption of a system, you'll find that the more components in the system, the closer this maximum power is to the typical power. And this makes sense. For a system with many components, it is more likely that most of them are running under typical conditions and only a few of them are running at worst-case conditions.

ment, but generally verification can be broken into several phases, each of which is essential to the entire process. These steps consist of

- Simulation
- Design review
- Synthesis
- Place and route
- Formal verification

The components of verification are discussed further in the following sections and in much greater detail in Chapter 6.

4.6.1 Simulation

Simulation is an ongoing process; some kind of simulation will be performed as part of nearly every stage of the design process. Small sections of the design should be simulated separately before hooking them up to larger sections, as described in Chapter 7. There will be many iterations of design and simulation in order to get the correct functionality.

4.6.2 Design Review

Once design and simulation are finished, another design review must take place so that other engineers can check the design. It is important to get other engineers to look over the simulations. At this stage, they should be looking for missed details and improper assumptions. This is one of the most important reviews because only with correct and complete simulation will you know that the chip will work correctly in your system.

In addition to the chip design team, you should invite engineers who were not involved in the actual chip design to this review. Often, the chip designers will inadvertently make assumptions that outsiders would question. These outsiders can also suggest corner cases that were not already simulated that will often uncover additional problems in the design.

4.6.3 Synthesis

The next step is to synthesize the chip. This involves using synthesis software to optimally translate your RTL design into a gate level design that can be mapped to logic blocks in the FPGA. This may involve specifying switches and optimization criteria in the HDL code, or playing with parameters of the synthesis software in order to ensure good timing and utilization.

After synthesis, the chip must be resimulated using the gate level output of the synthesis tool. If everything has gone well up to this point, the gate level sim-

A Testing Horror Story

If you're frightened easily, don't read further. Some years ago I was called into a small startup company that was designing medical imaging equipment. The founder of the company, an engineer, had been kicked out by the investors. The senior members of the engineering team, maybe three or four people, left with him. The company called me in as a consultant to help them figure out why the hardware they had didn't work. Through a painful reverse engineering process (the fleeing engineers had left no documentation), I found a slew of reliability problems that I corrected one by one. At that point, although we had a working board covered by blue wires, I suggested that we redesign the board. The company agreed. I then submitted a plan, including a test plan, whereby the software engineers would write code to test the hardware.

When the board had been redesigned and it came back from fabrication and assembly, I went to the software engineers to get the code to begin testing. The software engineers told me that the new Vice President of Engineering had changed all of the priorities so that the test code was dead last on the list.

I went to the VP's office for an explanation. He explained that getting the next round of funding was contingent on getting a beta site for our system. If we had a system working in a hospital, then we could get funding to keep going. Otherwise the company would be out of business. How could we get the system working, I asked, if we couldn't test the hardware. Wouldn't it be better if we put a working system into a beta site than a nonworking one? He told me that he trusted my abilities so much that he was sure my hardware worked perfectly. I told him that I appreciated his confidence, but even I don't deserve that kind of responsibility.

The story gets worse. I decided to write some software tests at that time. It was difficult because I didn't understand all of the software running on this system. I was able to get a test running that looped random data throughout this imaging system. Lo and behold, sometimes the data sent out came back corrupted. I showed this to the VP and to the CEO of the company. Their response? Don't worry. It's only a few pixels that get corrupted. No one will miss them. What if these pixels were enough to look like a tumor to a doctor reviewing the image, I asked. Or worse, God forbid, they covered up a real tumor? Don't worry, was the reply I got again. We'll fix it after we get the funding.

I finished up my work at the company, collected my check, and left when the contract expired. A year later they had been turned down for more funding and went out of business.

ulation results will agree with the RTL results. Another possibility is to use a formal verification tool that will logically compare the RTL description to the gate level description. Whichever method is used, the design team will need to address any discrepancies. Discrepancies discovered at this stage often are the result of processing RTL code that doesn't conform to the style rules expected by the synthesis tool. Correcting these problems is usually a straightforward matter of rewriting the code in the correct format.

The Need For Specification Reviews

After about two years at the large semiconductor company that employed me right out of school, I was given my first responsibility as a project leader designing a dual universal asynchronous receiver transmitter, or dual UART. I had another engineer reporting to me, and I was pretty excited about this. The project had some visibility within the division and people would periodically stop me to find out about my progress. What confused me, though, was that different people from different departments had different ideas about what I was designing. One day, a sales person would ask me about the quad UART. I would explain that it was a dual UART. They would say something about customers needing four, not two, on a chip. Another day, a marketing person would ask me about the UART. I would explain that it was a dual UART. They would say something about the customers who had been waiting for a UART to drop into their existing systems. When I mentioned a quad UART to the people in semiconductor processing, they would laugh, saying it wasn't possible to put a device that large on a single die and still fit into an existing package.

I finally decided to hold a specification review — something that I then realized should have been held at the beginning of the project. I invited the managers from engineering, sales, marketing, testing, and production. In going over the specifications I found that the product I was in charge of met no one's expectations and could probably not be sold. Had we known this at the beginning, we could have modified the specifications or spent our energies on a product that would actually make money.

Afterwards, I was told to continue working on the chip, despite the fact that we couldn't sell it, because we had already announced it and wanted to show that at least once we could produce something we had publicized. Demoralized about my new project, I continued with the company long enough to get the chip design on the right track and put it in the hands of my colleague. The design was fully documented, a rarity at this company, and I left to find other work. Later I found out that the design was cancelled right before production because someone higher up found out that no one really wanted to buy it.

4.6.4 Place and Route

The next step is to lay out the chip, resulting in a real layout for a real chip. This involves using the vendor's software tools to place various functions into the available blocks in the device and to route the blocks together. The software will figure out the bits needed to program the chip to implement the design. If you cannot successfully place the design into the device and route it, you may need to tweak the design. In some cases, you will need to make radical changes, such as eliminating some functionality or using a larger device. If you have followed all of the procedures outlined in this book, the chances of a major problem at this stage, resulting in a major design change, will be minimized.

Once the place and route is successful, the design team must perform timing analysis. The timing analysis will determine whether the designmeets the timing goals. Typically, the design team will need to redesign and resimulate certain paths in order to get the correct timing. In some cases, they will need to change functionality, or they will need to change the timing specifications of the chip, in order to get the design to work.

4.6.5 Resimulation and Formal Verification

At this stage, the design team must check the results of synthesis to make sure that the RTL design that was fully simulated is functionally equivalent to the gate level design that was produced after synthesis. In some cases, it may also be necessary to show that the configuration of the programmable device behaves identically to the RTL description. There are two ways to do this: by resimulating the lower level design or by using formal verification. In some cases, both techniques can be used for further certainty.

Resimulation

The most common method of determining that the input circuit and the final circuit are correct is to resimulate the final circuit using all of the tests that were used to simulate the original circuit. These tests are called *regression tests*. The results of both circuits should be identical for each clock edge.

Equivalency Checking

The formal verification that takes place at this stage is known as *equivalency checking*. This kind of formal verification involves a software tool that compares the RTL design with the gate level design created by the synthesis software, or possibly even compares it with the resulting configuration of the FPGA or CPLD. This software performs a mathematical comparison of the functionality of both circuits in order to confirm that both circuits will operate correctly.

4.7 Final Review

The final review of the chip should be a formality at this point. If the design team has followed all of the other steps and the other reviews have taken place, this review should be a simple sign-off that verifies that the design has been coded, simulated, synthesized, laid out and routed, and is now ready to go into the system.

4.8 System Integration and Test

For a one-time programmable device such as most CPLDs or an antifuse FPGA, you simply program the device at this time, and you immediately have your prototypes. For a reprogrammable chip, such as an SRAM-based FPGA, you place a blank chip into your system, and the system programs it during power up. In either case, you have the responsibility to determine that the entire system, with your programmable device, actually works correctly.

If you have followed the procedure up to this point, chances are very good that your system will perform correctly with only minor problems. The design team can often work around these minor problems by modifying the system or changing the system software. The team needs to test and document these problems so that they can fix them on the next revision of the chip. System integration and system testing is necessary at this point to ensure that all parts of the system work correctly together.

When the chips are put into production, the production process should include some sort of burn-in test that continually tests the system over some long amount of time. If a chip has been designed correctly, it will only fail because of marginal physical defects that will usually show up with this kind of stress testing.

4.9 Ship Product!

At this point you are done. It is time to ship your product and take that long awaited vacation.

4.10 Summary

The Universal Design Methodology as it relates specifically to programmable devices ensures the proper design, review, and testing of your product. This methodology, called UDM-PD, has the following goals:

1. Design a device that
   - is free from manufacturing defects

- works reliably over the lifetime of the device
- functions correctly in your system

2. Design this device efficiently, meaning
   - in the least amount of time
   - using the least amount of resources, including personnel
3. Plan the design efficiently, meaning
   - create a reasonable schedule as early in the process as possible
   - know all necessary resources up front and allocate them as early in the process as possible

In this chapter, we have also studied the design flow that is associated with UDM-PD. This design flow has the following stages:

- Writing a specification
- A specification review
- Choosing programmable devices and software tools
- Designing the chip
- Simulating the design
- Design review
- Synthesizing the design
- Place and route of the design
- Resimulation and/or formal verification
- Final review
- Testing the chip
- Integrating the chip into the system and testing the system
- Shipping the product

Exercises

1. Select the three major goals of UDM-PD.
   (a) Design the device efficiently
   (b) Design a device synchronously
   (c) Plan to design a device
   (d) Design a device that works reliably over the lifetime of the device
   (e) Allocate a maximum number of resources in the minimum amount of time
   (f) Plan the design efficiently

2. UDM is a methodology to design a device that (choose all that apply)
   (a) Is free from manufacturing defects
   (b) Can be tested automatically
   (c) Functions correctly in your system
   (d) Works reliably over the lifetime of the device
   (e) Is inexpensive

3. UDM is a methodology to design a device efficiently, meaning (choose all that apply)
   (a) In the least amount of time
   (b) Using the least number of people
   (c) Using the least amount of resources

4. UDM is a methodology to plan a design efficiently, meaning (choose all that apply)
   (a) Writing a specification that will not change
   (b) Knowing all necessary resources up front and allocating them as early in the process as possible
   (c) Creating a reasonable schedule as early in the process as possible
   (d) Planning to ship products while testing the design

5. Put each phase of the design flow in the correct order.
   (a) Ship product!
   (b) System integration and test
   (c) Write a specification
   (d) Final review
   (e) Specification review

(f) Resimulation
(g) Choose device and tools
(h) Simulate
(i) Place and route
(j) Design review
(k) Design
(l) Synthesis

6. A design specification should include the following (select all that apply):
   (a) The name of the FPGA vendor
   (b) A description of the I/O pins, including output drive capabilities and input threshold levels
   (c) The estimated gate count
   (d) The target power consumption
   (e) Test procedures, including in-system test requirements
   (f) An external block diagram showing how the FPGA fits into the system
   (g) A notice that the document, once approved, cannot be changed
   (h) An internal block diagram showing each major functional section
   (i) Timing estimates, including setup and hold times for input pins, propagation times for output pins, and the clock cycle time
   (j) The target price
   (k) The package type