

DIGITAL PRODUCT DEVELOPMENT

Digital product development aims at a **prototype**. The prototype is eventually **manufactured** as a new **chip** or as a new **printed circuit board** (PCB) in mass quantities. If it is a new chip it has to be “mounted” on a new PCB.

Digital product development for a new chip involves three cycles : (i) the development cycle on computers, (ii) the development cycle with FPGA chips and (iii) the development cycle on a prototype chip. Digital product development for a new PCB involves three cycles : (i) the development cycle on computers, (ii) the development cycle with off-the-shelf chips and (iii) the development cycle on a prototype PCB. Most of these cycles have three main steps : design/mount, test and modify. Often, these cycles occur as design/mount, test, modify, test, modify,..., test, modify and test. Once the product passes all three development cycles, it is ready for **mass production**. The chip or the PCB is manufactured in large quantities !

1) The development cycle on **computers** consists of designing the product, testing and modifying it, by using a Computer Aided Design (CAD) software package. The product is designed, tested and modified piece-by-piece (block-by-block), following the rules mentioned in the *General Design Rules* handout. To test, input combinations (**test vectors**) are applied and outputs are observed. If a minor error is detected during the testing of a block, the block is modified (a simple change). After the modification, the circuit is tested again for correctness. If major errors are encountered, the design step is revisited. *In the worst case a complete redesign is performed.*

While testing and modifying circuits, engineers may realize that slight design changes can be employed to satisfy design goals better. For example, components in the circuit can be eliminated (the minimization), saving hardware and reducing cost. After the minimization, the circuit is tested. There is no rule that specifies exactly when the minimization should be performed. In general testing for correctness should precede the minimization to give the designer a chance to become familiar with the circuit.

The CAD software package must be “industry-strength,” to be capable of simulating real circuits with possibly millions of transistors or tens of chips, depending on whether a new chip or a new PCB is developed. The CS2204 CAD software is the Xilinx Foundation software which is industry strength. Other widely used software packages include the Cadence, Synopsys and Mentor Graphics which are used for research and education at Polytechnic University.

2) a) If the design passes the test on computers and the goal is to manufacture a **new chip**, the development cycle **with FPGA chips** starts. It consists of three steps : mount, test and modify. One or more FPGAs are mounted, tested and modified. The development cycle on computers may be revisited if problems are encountered during testing.

An FPGA chip is a “hardware programmable” chip. It is “programmed” to function as the intended circuit. To program the FPGA, a file generated by the CAD software package is “downloaded.” Mounting FPGA chips means the chips are placed on special boards or breadboards, then wired and finally the FPGA chips are programmed. Testing means input combinations (**test vectors**) are applied to the circuits and outputs are observed. If unexpected outputs result, we might rewire the FPGAs and/or mount new FPGAs and/or modify the circuit on the CAD software. If the circuit is modified on the CAD software, we do simulations and verifications on the computer until the new design seems correct. Then, we return to the development cycle with FPGA chips. We test the set up until we are satisfied, continuing as such. *But, in the worst case, a complete redesign is performed.*

The FPGA would display many expected characteristics of the target chip. But, the FPGA chip would be slightly slower, larger than the eventual chip and consume more power. Experienced designers can estimate the differences, infer on them to decide when to stop this development cycle and start the development cycle of the prototype chip.

2) b) If the design passes the test on computers and the goal is to manufacture a **new PCB**, the development cycle with off-the-shelf chips starts. A prototype circuit is mounted on special boards or breadboards with off-the-shelf chips, the chips are wired, tested and modified. *In the worst case, a complete redesign is performed.*

3) a) If the design passes the test with FPGAs and the goal is to fabricate a **new chip**, the development cycle on the prototype chip starts. The circuit implemented on the prototype chip is tested for correctness.

The fabrication of the prototype chip is such that the CAD tool generates an output file (a graphic data system II or GDSII file) that is sent to the fabrication facility (called **taping out**). It may take weeks to a few months to fabricate the chip. The chip is extensively tested with many different input combinations (test vectors) and the outputs are observed. *In the worst case, if there are errors, the chip is discarded!* This is because one **cannot** modify the chip. If errors are found, a **new** prototype chip has to be fabricated, meaning a loss of months or more!

3) b) If the design passes the test with off-the-shelf chips and the goal is to manufacture a **new PCB**, the development cycle on the prototype PCB starts. A prototype printed circuit board is built, chips and other components are mounted. Then, the PCB is tested.

Again, the CAD tool generates the output (file) that is needed to fabricate the prototype PCB. It may take a few weeks to fabricate the prototype PCB. The PCB is extensively tested with different test vectors and the outputs are observed. If errors are found, the solution might be a simple rewiring on the PCB. If the modification is extensive, one might have to design a new prototype PCB. *In the worst case a complete redesign is performed.* This means loss of months.

CS2204 Lab Target

In the CS2204 lab, an FPGA chip is used. It is already mounted on the FPGA board. In addition, **no** off-the-shelf chip is mounted. Thus, in the CS2204 lab we learn the development cycle with FPGA chips. *That is, in the lab, our goal is developing a new chip. During lectures, however, we learn how to develop both new chips and new PCBs.*

In CS2204, the mount step of 2(a) requires only programming of the FPGA chip (“downloading” the bit file to the FPGA chip). The testing includes using switches and push buttons that input different values (test vectors) and observing the outputs connected to the LED lights and 7-segment displays. The modification means the circuit (the schematic) is changed and the bit file is downloaded to the FPGA chip again.

Digital Product Development on Computers

This is step 1 on the first page! Typically in industry, the design, test and modify steps of the development cycle on computers overlap to a certain extent to speed up the development. In CS2204, we will do the same. Also, both in industry and CS 2204, *block-based* approach is used where each team member is assigned one or more (sub)blocks. So, parts of the circuit are designed simultaneously by team members, speeding up the development even further. Recall that step **1** consists of design, test and modify. Below, we describe them in detail.

DESIGN : When a digital circuit is designed there are two main tasks : obtaining the precise input/output relationship and the implementation. We repeat these two steps until all (sub)blocks are implemented and tested. The figure on the next page shows the design phase in a graphical way.

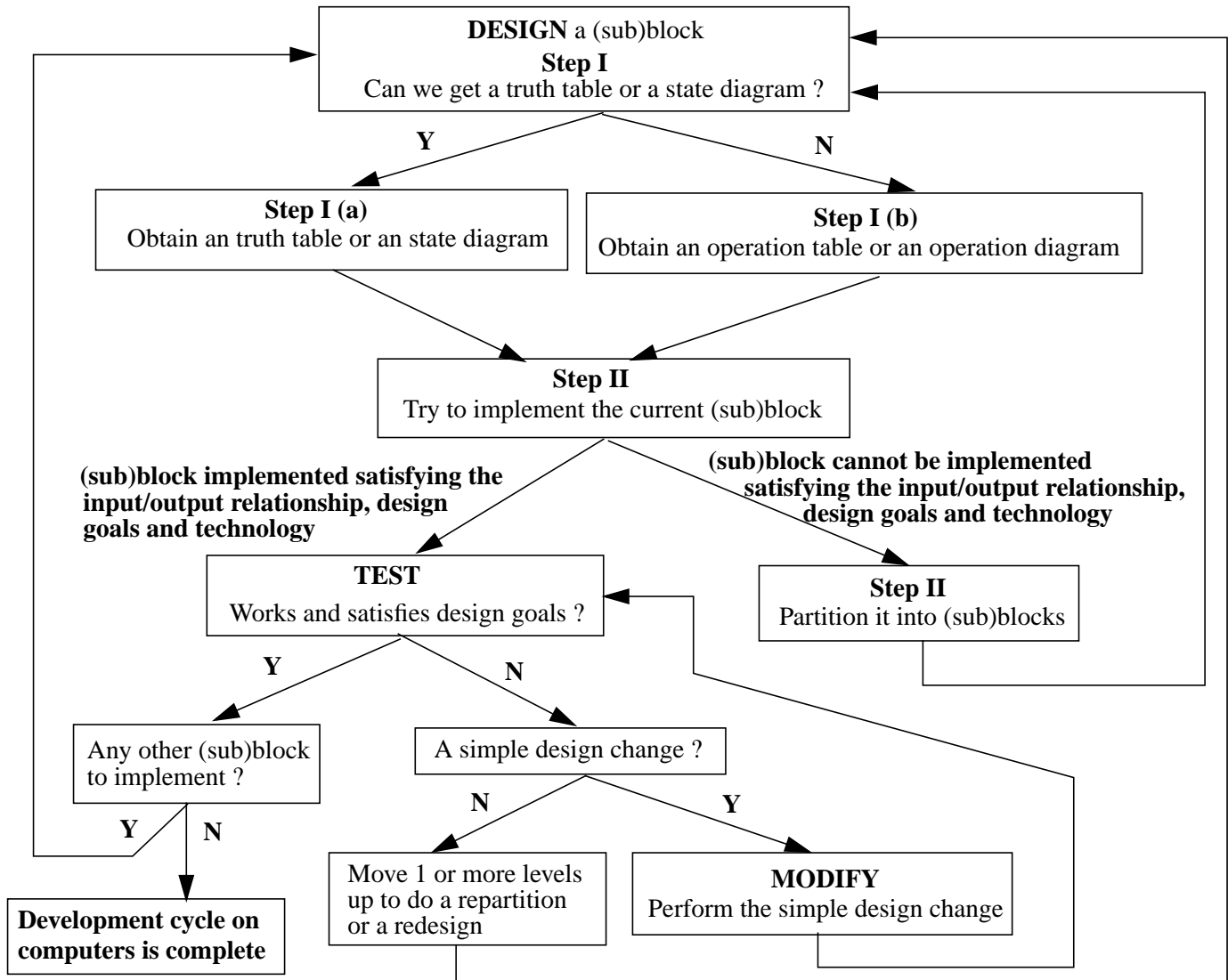
I) Input/Output Relationship : the (sub)block is first considered as a black box. Then, we relate the outputs to the inputs by using a **truth table** or a **state diagram** or an **operation table** or an **operation diagram**. First, we determine if the (sub)block is a simple one. Then,

a) If the (sub)block is a simple combinational circuit (it has less than five inputs), we get the truth table and circuit expressions quickly. Similarly, if the (sub)block is a simple sequential circuit (the sum of the number of flip-flops and inputs is less than five), we get its state diagram and circuit expressions quickly. Then, we move on to the implementation step.

b) If (sub)block is complex, we obtain an **operation table** or an **operation diagram**. After that, we move on to the implementation step to implement it, i.e. to determine the components.

II) Implementation :

The implementation means determining the components based on the given product goals : The input/output relationship, design goals (speed, cost, power consumption, reliability, size, weight, etc.) and the technology. What these components are depends on whether we are developing a new chip or a new PCB. If it is a new chip, these components are simple/core/Xilinx macro circuits. If it is a new PCB, these components are chips. The product goals are needed, because the implementation has to be consistent with the predetermined product goals. For the sake of simplicity, throughout the semester, we will aim at a minimal implementation, meaning minimal number of components in the circuit. Below, we will give the implementation steps to develop for both a new chip and a new PCB.



For a new chip, the implementation steps are specifically given below for the CS2204 lab. To implement a (sub)block we search the Xilinx library of components of the schematic editor and see if there is a component that immediately implements it, satisfying the textual input-output relationship, design goals and technology of the (sub)block. The Xilinx library contains three types of components : Gates, flip-flops and Xilinx macros known as Xilinx design blocks or XDBs. Since an XDB already implements a circuit, using it reduces our number of components and saves time. Thus, we have to try to use XDBs as much as possible. Note that here we imply that we have confidence in XDBs : They are as good as we can design and satisfy our product goals.

For a new PCB, to implement a (sub)block immediately, we search data manuals to see if there is any chip that immediately implements it, satisfying the textual input-output relationship, design goals and technology of the (sub)block. These chips have different densities, ranging from ULSI to SSI chips. They are standard, off-the-shelf chips, therefore, they are cheap. Since we want to reduce space, size, power consumption, etc., we have to try to use **higher** density chips as they contain many circuits in a small area. We have to avoid using SSI chips.

If the (sub)block can be immediately implemented by satisfying the product goals, we move to the **TEST** step. Otherwise, we partition the (sub)block based on the product goals and go back to **step (I)** above. Here is what we do for each (sub)block :

i) For a new chip : Search the Xilinx library of components (gates, flip-flops and XDBs) for one or few high density XDBs that can be immediately used to implement the (sub)block, while satisfying the product goals. These XDBs

are standard blocks, therefore, they are known. That is, we immediately know their input-output relationship. If several XDBs are needed their interconnection is straightforward. A few extra gates and flip-flops might be needed which is fine. Their logic expressions are very simple. If there are alternative XDBs, choose the ones that satisfy the product goals the best. After the XDBs/gates are chosen move on to the TEST step of the (sub)block implemented.

For a new PCB : Search data manuals for one or few high density chip(s) (ULSI/VLSI/LSI/MSI chips) that can be immediately used to implement the (sub)block, while satisfying the product goals. If several chips are needed their interconnection is straightforward. A few extra gates and flip-flop chips (SSI chips) might be needed which is fine. Their logic expressions are very simple. If there are alternative chips, choose the chips that satisfy the product goals the best. After the chips are chosen move on to the TEST step below.

ii) For a new chip : If the answer is no, either because there is no XDB available to do the task or there are XDBs but they do not satisfy the digital product goals, search the Xilinx library of components to see if there is one or more XDBs that can be “programmed” to implement the (sub)block, while satisfying the product goals. Examples of these XDBs on the Xilinx library are ROMs and RAMs. Note that using them is plausible if the number of inputs to a programmable XDB is less than 11. A few extra gates and flip-flops might be needed which is fine. Their logic expressions are very simple. If there are alternative XDBs, choose the ones that satisfy the product goals best. After the chips are chosen move on to the TEST step of the (sub)block.

For a new PCB : If the answer is no, search data manuals to see if there is one or more chips that can be “programmed” to implement the (sub)block, while satisfying the product goals. Examples of these chips are PLAs, PALs, ROMs, GALs and FPGAs. A few extra gates and flip-flop chips (SSI chips) might be needed which is fine. Their logic expressions can be simple and straightforward. If there are alternative chips, choose the chips that satisfy the product goals best. After the chips are chosen move on to the TEST step described below.

iii) For a new chip : If the answer is no, check to see if this (sub)block is simple enough to be implemented by a few gates and/or FFs while satisfying the product goals. That is, it has less than 5 inputs or less than 5 FFs. If yes, select the right ones. After they are chosen move on to the TEST step of the (sub)block implemented.

For a new PCB : If the answer is no, check to see if this (sub)block is simple enough to be implemented by a few gate and flip-flop chips (SSI chips) while satisfying the product goals, i.e. it has less than 5 inputs or the sum of the number of flip-flops and inputs is less than five. If yes, select the right SSI chips. After the chips are chosen move on to the TEST step described below.

iv) For a new chip : If the answers have been “no,” check to see if a custom design block (CDB) can be borrowed as a core circuit. This is a black box whose internal structure is designed by another company. When used in schematics, it is simply a CDB. Licensing would be justifiable if this (sub)block is complex and designing it would delay the project completion. One would search companies which might have similar circuits and ask them if they are willing to license their circuits. These circuits must satisfy the input-output relationship you are looking for, the design goals and the technology. If you license the circuits, you have the core circuits. After the CDB is placed in the circuit, move on to the TEST step of the (sub)block implemented.

For a new PCB : If the answers have been “no,” either because there is no chip around to do the task or there are chips but they do not satisfy the digital product goals, think about designing a “custom” chip. That is, an application specific integrated circuit or ASIC. This would take you to the first page of the handout : You are now starting to develop a new chip ! Remember : This is a lengthy process taking many months. To shorten the chip design duration, you search if the design of most complex parts of the new chip can be avoided by licensing them from other companies. You search companies which might have similar circuits and ask them if they are willing to license their circuits. These circuits must satisfy the input-output relationship you are looking for, the design goals and technology. If you license the circuits, you have the core circuits. After the custom chip is fabricated, move on to the TEST step below to test the custom prototype chip.

v) For a new chip and a new PCB : If all the answers are no, you will need to partition (sub)block based on the product goals and then move to **Step I** to implement the new partitionings.

Note that the partitioning of a (sub)block often results in common operations that are implemented as high-density components/chips. That is, these components are available and there is no need to design them : ADDers, MUXes, Decoders, Registers, Counters. Sometimes, the partitioning can result in an unusual operation. In this case, the unusual operation can be immediately implemented by programmable components/chips so that we do not have to partition it further. During partitioning, keeping available components in mind (the technology) is necessary. Otherwise, one might have a (sub)block whose implementation by available components is inefficient, forcing a repartitioning. Considering the components all the time also helps us stop the partitioning early. Note that, this style of design by considering several layers at the same time is the **middle-out** design.

Note that the design and implementation steps are gone through many times, since this whole process is a trial-and-error process. Therefore, one should **not** feel disappointment if there are several iterations. Also, the following points are known by many practitioners of digital design :

- You might have alternative divisions of a block due to alternative implementations. Use steps (i) through (iv) above to come up with the best solution.
- Sometimes, it will seem the current block partitioning is implementable, but later you realize a subblock is not that easily implementable. This requires a different partitioning of the original block as we will see below.

TEST : Once a (sub)block is implemented, it is tested to see if it satisfies the product goals. That is, we check to see if it has logic and timing issues due to the current design or a modification we just performed.

The testing on computers is done by means of simulations which can be functional and timing. During simulations, input combinations (test vectors) are applied to a (sub)block we are testing and outputs of the (sub)block are observed. If the outputs are correct, new test vectors are applied and outputs are observed. If the design fails the test then we move to the MODIFY step. If we are satisfied with the design or modification, we check if all the (sub)blocks have been implemented. If yes, then the Development Cycle on Computers is completed.

Testing sequential circuits is different from testing combinational circuits. It is because sequential circuit outputs are a function of present and past inputs. Therefore, the testing involves not only determining the right test vectors, but also the right order of test vectors. Otherwise, the error in design or modification may not be detected.

For beginner designers and for complex (sub)blocks, it is highly recommended that each (sub)block be tested **separately** first. When two (sub)blocks pass the test, they can be connected to each other and then tested together. Then, another (sub)block which passed its test can be connected to the previous two (sub)blocks. We test these three (sub)blocks together, and so on. It is critical to remember that we **not** connect **all** the (sub)blocks and then test !

During the TEST, if we recognize that satisfying the design factors can be improved, such as we can reduce the cost if components determined to be redundant are removed, it is highly advised that we note them down first. We should **not** try to minimize a circuit when we are testing for correctness. This is because if the circuit has errors, minimizing would be wrong or irrelevant and so would be a waste of time. After the (sub)block passes the test, we can do the minimization in the MODIFY step.

➤ CS2204 students :

- ⇨ Perform **functional simulations** on individual (sub)blocks. If an output value does not match your expected result, you will need to go back to your schematic design to correct the error. Connect (sub)blocks one at a time to each other and perform simulations. Finally, perform simulations on the complete circuit. Note that the number of inputs is large, you cannot try all possible input combinations. Thus, carefully think about which input combinations (test vectors) to apply with your partner. Make sure to write down the test vectors and what you observe so that you can figure out the cause of an error quickly. This way you also not apply a test vector again and again unnecessarily.
- ⇨ If you realize you can minimize the circuit while testing, note what it is. Do the minimization after you think it is highly unlikely you would catch any more errors. After the minimization, continue doing the testing (simulation) until you are sure the minimization is correct and does not introduce new errors.
- ⇨ If you are not sure about the circuit response, open previous Ppm projects and observe their responses.

MODIFY : If a (sub)block fails to satisfy the product goals because there is a design error or the previous modification was incorrect, we modify the (sub)block to correct the logic/timing error. This implies a simple change. If it is not simple change then we do a repartition one level up or two levels up or in the worst case all the way which means a total **redesign** ! That is we go back to the DESIGN step.

If we see we can satisfy product goals better, such as there are optimization possibilities, we do so **after** we correct the errors in the circuit. After we modify, we go back to the TEST step to make sure our modification is correct.

➤ CS2204 students :

- Modify the circuit when errors are encountered. That is **modify the schematic design**. Then, go back to the TEST step above.

Digital Design Conventions

Industry and digital logic books use the following conventions. Students need to know them :

a. Digital Circuit Drawing Conventions :

- On the lower right corner of the printout (after taping if more than one sheet) information identifying the students that produced the work is shown. That is, the names of team members, “CS2204” the Lab section and the semester are placed on the **lower right** corner of the printout.
- When drawing lines to connect gates/FFs/XDBs, **horizontal** and **vertical** lines are drawn. Note that some CAD software does allow 45° angled wires. But, the Xilinx software does **not** allow such angled wires.
- Wires are **not** drawn over blocks, circuits, chips, buffers, pads, labels, etc. Wires are drawn around them horizontally and vertically.
- Remember to **beautify** the circuit on the screen after determining that it works. This means components that do similar work form horizontal and vertical lines, line tanglings are minimized and unnecessary wire turns are eliminated.

b) Logic Circuit Design Conventions :

- Following block-based design, circuits for individual blocks are drawn block by block. In order to help the reader, circuits for different blocks are placed with enough distance between them. A block may consist of subblocks in which case each subblock circuit must have enough distance from its neighbor subblocks. Also, a block or subblock should not have its components all over the screen : the components must be close to each other to signal a block or a subblock.
- No output can be short circuited to any other output unless the outputs are tri-state outputs. Thus, no totem-pole, gate, flip-flop and chip output can be short circuited with any other output.
- If an output is not needed, leave it **unconnected**.
- A high-impedance (Hi-Z) output value must be carefully examined to see if it is correct.
- CS2204 students :
 - ⇨ Make sure the schematic files are saved.
 - ⇨ Perform a **logic simulation** on each (sub)block. If you cannot figure out the cause, do a Xilinx IMPLEMENTATION and follow the error and warning messages in the Implementation Log File.
 - ⇨ Make sure that you continuously look at the experiment check list handouts to remember and follow design conventions and avoid errors.
 - ⇨ Important : if students cannot make progress, they are referred to the earlier handouts that are about simpler but similar circuits. Exercises in some of the handouts also give detailed coverage of input/output relationship determination and implementation possibilities. In addition, students should refer to the “*General Design Rules*” handout.

c. Digital Circuit Printing Conventions :

- The circuit printout must be readable : characters, labels, gates, FFs, blocks, chips must be readable.
- If the circuit is large, a readable printout requires more than one page. In that case :
 - ⇨ A one-page printout is not acceptable, unless otherwise indicated,
 - All sheets must be attached to each other (such as a scotch tape on **both** sides of the sheets),
 - Lines, chips, names must be continuous (requiring cutting sheets along the edges before taping).
- CS2204 students :
 - ⇨ If students are asked to identify blocks, they should do so with a color pen, a **green** pen. It is green in order not to cause confusion with the red pen the professor uses. Students should draw a circle (or rectangle) around each block. Students must make sure that they circle each block carefully, without leaving out any circuit that belongs to the block.
 - ⇨ If Students are also asked to name the blocks, they should use the same color pen and make sure the name matches the block.