I'm not robot
reCAPTCHA
Continue

Logic gates pdf books

This chapter covers the concept (theory) of digital logic. Digital portals are introduced and various performances, such as Boolean logic, diagrams and fact-tell tables are displayed. The Boolean logic rules originate and the chapter ends with examples of the implementation of digital designs. Read pages 703-24 in Horowitz P and Hill W 2015 The Art of Electronics 3rd edn (New York: Cambridge University Press) and pages 717-37 in Scherz P and Monk S 2013 Practical Electronics for Inventors 3rd edn (New York: McGraw-Hill Education). While a similar signal is continuous, a digital one is discrete. Specifically, an analog signal represents an infinite continuity of levels while a digital representation of limited, discrete levels. Our world is inherently similar. We sense continuous signal levels such as light or sound intensity. Conversely, if humans are born with digital sensory bodies, we will only feel the binary state, for example brightness or darkness, silence or noise. Unsurprisingly, most of us who are fortunate to enjoy similar sensory agency activity will never want this trade for the digital ones. Based on this observation, it seems that compared to analog, digital technology is inherently limited and crude. However, the popular trend in technology clearly favours a digital approach compared to the 'old' analogy, showing the significant advantages of this 'new' technology. Let's explore the main advantages with the following analogy. Imagine that you are in a concert room with two 'musicians' who are hidden from the audience by a curtain. The task for performers is simple: each person is asked to play a single note on their instrument and then the audience is asked to describe the sound intensity or if they even hear that instrument being played. After the first performer plays a few identical notes on a violin, you'll probably get a wide response from listeners from different listeners, from 'quiet' to 'big'. After all, perception is subjective and based on the listener's musical preferences, listening ability and distance from the player. Moreover, depending on the relationship between how big the note was played and how loud the background noise was in the hall, it can even be difficult for listeners to distinguish the condition of the instrument, i.e. it was played at all or did they just hear the noise. Next imagine the second player, who uses an extremely large and obnoxious car horn as his instrument. Again he played a note by turning the whistle on or off several times. Now, most listeners will obviously agree on the state of sound intensity perception, i.e. either it was quiet or loud-horned to be 'on' or 'off'. As long as the car horn is louder than the level of ambient noise in the concert room (a very likely condition) there is no ambiguity of escape Signal. If our only concern is to make sure that the audience senses the state of the signal, i.e. the instrument is being played or not, then obviously the second musician with himself winning approach. By using an instrument capable of producing only two clear, completely silent states and ear-piercing noise, he can convey his message on the concert room without ambiguity or flaws. It should come as no surprise why this digital message system works so well; After all, it is designed to work in an extremely noisy environment, such as traffic, to provide a clear message (warning). In short, by using only clearly imm recognized states, digital technology can transmit signal states (almost) immune to noise. For the same reason, it is then that these states can also be doubled and stored without being degraded by background noise. In digital technology, such a binary 'signal' is called a 'bit' and states it as 'day' or 'off'. Of course, one can make the rd by which we are actually comparing two completely different systems in our concert room analogy. It's one thing to convey a clear message by either ingesting a completely quiet or exploding obnoxious car horn. However, it is a completely different matter to convey a more complex, nuanced and perhaps pleasingly aesthetic message, such as the violinist being able to produce with her similar instrument. Instead, the question we should ask is: can we create a more nuanced message with our simple car horns? At first, this seems impossible if we are only allowed to use two clear states that our car horns ingest, silence and noise. However, the digital world gets around this problem by 'bundling' bits into a package and then assigning meaning to this bit string. This really isn't all that different from the way some drivers operate their car horns: no honking means you're ok; a short honk means you've gently irritated someone; A variety of long honks obviously means you should get out of your phone. While this is a bit oversimplified, it shows that we can express a discrete level of emotion using only silence and no silence. It also highlights another aspect of digital technology, namely that the more signals, or bits we agree to use in our order, the more we can express these levels. However, as long as we are forced to use a limited string of bits, the number of levels we can express will also be limited. Back to our traffic analogy, whereas with a little, we can only show two levels, 'ok' or 'angry', with just two bits we can signal four levels, 'ok', 'mildly irritated', 'slightly uncomfortable' and 'angry'. Another advantage that digital technology has on analog is the ease with which data can Manipulate using very simple 'operators', called ports, to mimic mathematical activities. In summary, the high noise immunity of digital signal transmission and storage and the ease with which they can be manipulated with mathematical operations give digital an advantage over analog. This explains current trends from analog and towards digital technology. However, you should keep in that we begin by saying that our world (including who we are) capital is similar. Therefore, at least as long as humans don't have a digital port, we'll always work with both analog and digital technologies and the interface between these two worlds, the same converter to digital and digital to analog. When we mention the binary state of a bit, we will use the usual labels of 'on' or 'off', '1' or '0' or 'high' or 'low' (sometimes also written as 'HI' or 'LO'). We will use these labels instead of each other and agree that 'on', '1' and 'high' are considered equivalent and therefore are 'off', '0' and 'low'. In a physical environment, such as an electronic circuit, clear voltage or existing level standards exist for a specific digital logical state. Many conventions exist and the most common conventions are listed in table 1.1. To save energy and speed up switching, the 3.3 V (TTL) low-voltage ball logic standard is increasingly replacing the old established 5 V TTL level. Table 1.1. The most commonly used digital voltage standard. Logic level 3.3 V CMOS OR TTL 5 V TTL 1, above, high 3.3 V 5 V 0, off, low 0 V 0 V Table 1.1 is deliberately kept simple and it does not list acceptable threshold levels for high and low input and input signals. Usually this is within a volt of the 'ideal' levels listed above. For example, for 3.3 V CMOS systems, input voltages from 0 to 0.8 V are accepted as low inputs, signals from 2.0 to 3.3 V are considered high inputs; you want to avoid input signal levels between 0.8 V and 2.0 V because they are unclear. Discrete components require external energy and the levels listed above state their supply voltage. Finally, avoid applying input signals that exceed the above-mentioned highs or lows. Doing so can destroy the ingredients! As mentioned previously, the power of digital logic comes from the ability to manipulate signals through logical ports. These logic portals are digital technology what operators are mathematically. There are only three basic ports: NOT, AND, and OR ports. All other ports and the entire digital technology are based on the continuous application of three operators in different combinations. While this can be hard to believe, keep in mind that a large part of mathematics consists of repetitive applications of plus and minus, basic results in in-character and division. As you will see later, by combining these doors, you will be able to build a simple computer, a player and even a processor. Many methods exist to represent digital logic, each with its strengths and weaknesses. Working with digital designs requires the ability to move smoothly from one representative to another. All basic ports and four corresponding performances are displayed in table 1.2. In these examples, A and B are input and Q is the input. Table 1.2. Comparable digital logic demonstration table. We three basic ports in Group I with three additional ports. The ports in group II, nor and NAND ports, are a combination of an OR or an AND port with a NO port. The XOR port in Group III is a combination of all three basic ports. These three additional ports are used so often that they have been given their own names and symbols. Before we consider individual portals in more detail, let's briefly discuss individual logical performances. Show the diagram. This representation is useful when creating detailed diagrams of a logic circuit and assisting in battery exercises and debugging. The fact panel. A fact table lists every possible combination of inputs and its results. It provides a complete state map of the components. By applying the fact tables to each circuit element and propagating the inputs to the next input, you can (theoretically) always predict the behavior of any circuit. In fact, with a large design, this approach can quickly become tedious. Boolean logic. This performance is similar to the digital symbol although it has its own set of rules. Similar to express numbers, Boolean logical expressions can be rearranged and simplified. When you work with them remember that a fairly wide range of symbols exist for operators that can be easily confusing. Also, note that some operators look like mathematical operators but they work very differently in Boolean logic. Representative Verilog. Verilog is a language that describes officially sanctioned ANSI (HDL) hardware. Therefore, its symbol must be accurate and clear. It is very unforgiving and a knocking error can lead to unforeseen results. Venn diagram. Venn diagrams, sometimes also known as Johnston diagrams, for a NO, an OR, and an AND port are displayed on the right side of figure 1.1. Symbols are based on set theory and can be used as another representation of digital logic. Similar to a true table, it lists all possible input combinations with the first digit corresponding to the first input and the second digit for the second input. Dark areas corresponding to the regions that lead to the actual input state for the specific logic operation are displayed. Venn diagrams are used limitedly, especially in larger circuits, and we will not use them anymore. Figure 1.1. Venn or Johnston diagram for a no, OR and an AND port.. Image Download: Standard Image High Resolution Image 1.2.1. Basic portsThe basic port of digital logic is port NOT, OR and AND. Port NOT is an inverter. It denies the current state and converts it to the opposite state. For example, by NOT being 'over', something is 'off'. (Note: THE NOT agent is sometimes written as , ~X or !X. As you will see later, if X represents a collection of bits then the meaning of ~ X and ! X is not equivalent. Avoid confusion and stick with ~X.) Applying a non-consecutive operator some ch ch of times in no changes since all other activities have not been cancelled before. For example, apply the operator not twice as in: results in. A diagram of this expression is:Download shape: High-resolution image standard Image The NOT symbol is usually drawn as an abbreviation, ignores the triangle, and uses only the open circle at the end. The three circuits below illustrate this. All three circuits describe the functionality of the NOR port, i.e. the OR port followed by the PORT NOT:Download image: High-resolution Image Standard Image The same abbreviation of the NOT port is also used at the entrance to the port, as described in the three circuits below. The circuits represent a 'NO A, NOT B, OR-gate' (of course, the functions of these three circuits do not equa to the three circuits shown directly above!):D ownload figure: High-resolution image standard Images The remaining basic ports are OR ports and AND PORTS. Unlike a PORT NOT being a single player, without single input, both OR and AND ports have at least two inputs. First, note the variable nature of these ports and boolean logic in general, shown below:Download image: High-resolution Image Standard Images Although the ports shown above are two-input ports, we can easily extend their inputs to any number while maintaining their functionality. An example of an OR port of three inputs is shown below. The right circuit is displayed in its concise form while the other one details its implementation with two inputs OR doors. This approach can be expanded further and it works for any two-port input. Note that we implicitly use boolean combination attributes, namely:Download the image: High-resolution Image Standard Image Finally, let's check the differences between the OR and AND ports with the following two examples. In the first example, let's say that you have 1000 individual particle detectors and you want to monitor when at least one of them has been activated. Suppose that when an individual detector is hit by a particle, the exit of the detector goes high. The easiest way to search for such an event is to connect all particle detector inputs to a giant OR port (which has 1000 inputs) and track its (unique) input. As the or port's outing soars, we know that at least one particle has passed through one of the detectors. In other words, the OR port goes high when any one of its inputs goes high. This is why or operator is also known as 'ANY' operator. For the second example, consider that we are trying to locate a mysterious source of particles originating somewhere in the universe, far away. We will use a particle 'telescope' to detect their origin. It works similarly to an optical telescope, except that it uses several particle detectors instead of lenses, arranged in a straight line. We can determine the origin of particles by scanning our telescope on the and follow up on an event when all activated at the same time. This only happens when all detectors are linked to mysterious sources. (In this example, we assume that the particles move in a straight line and are close to the speed of light.) This technique, commonly known as coincidence technique n times, in which n refers to the number of detectors that are randomly placed, can be done by connecting all the outputs of the detector to the input of a giant AND port. The input of the AND port will only increase when ALL inputs are high, which means that we must align our detectors in the direction of the source. This example illustrates the difficult nature of AND ports: unlike 'combinations' OR ports, AND ports are very selective and act as a 'filter'. Since all of its input conditions must be met for its exit to go high, an AND port is sometimes referred to as an 'ALL' port. Finally, a quick word on the Boolean symbol for these two ports: the OR operators are usually expressed using additional symbols, +. However, Verilog has an additional operator and this can lead to confusion because the behavior is different. This will be discussed in more detail later. In the Boolean symbol, the AND operators are often ignored and it is implicitly understood that AB actually means A B. Example 1: time diagram Diagram diagram in figure 1.2 shows the exit from different ports based on input depending on the time of A and B. In this time diagram , the x-axis represents the time and axis of the digital voltage level. (Note: the last trace shows the exit from an XOR port.) Figure 1.2. Time-dependent inputs for OR, AND, and XOR ports are the result of input signals A and B.Download images: High-resolution image standard images For example 2: take the basic Boolean logic rules from the truth tables Please apply the truth table from table 1.2 to obtain some basic Boolean logic rules. See if you can work out conclusions from the expressions on the left using the truth table for the specific port. Table 1.3. This table lists some basic Boolean expressions and represents their respective diagrams. It uses the truth table to get the result state. In the previous exercise, we simplified boolean expressions and eliminated excess components. This section will use the truth table and show that we can mimic the behavior of a port type by replacing it with a different type or type. In other words, we will develop Boolean rules on how to exchange ports. Example 1. The NO port from nand port shown below is nand port with inputs tied together. When you check its truth panel (below), you notice that it acts as an inverter! Note that the same result can be obtained by using a NOR port in a similar configuration. Download image: Standard image High-resolution image A B = AA 1 1 1 0 0 0 1 Conclusion: you can replace the NOT port with a NAND or NOR port with an input that is tied together. However, bạn không bao giờ có thể tạo cổng NOT từ cổng AND hoặc cổng OR OR (Evidence left to the reader.) Example 2. AND port from NAND port Download image: Image standard High-resolution image In this example, we deny the exit of the NAND port with another NAND port disguised as port NOT. Double denying of port input AND leading to its original AND port. The conclusion is that NAND or NOR ports can be replaced for PORTS NOT, OR or AND. Similar to previous conclusions, or and gates alone cannot be replaced for NAND or NOR doors. Example 3. De Morgan's theedity This last example is so important to Boolean logic that it is even given its own name, de Morgan's theedity. Check the fact table for the circuit shown below. What kind of port does its Q exit remind you? Image download: Standard image High resolution Image Apply a fact table to the circuit above brings the table of truth: A B 0 0 1 1 1 0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 1 Although this circuit only uses NAND port, its outing , Q, acts as an OR port. By symmetry, it then we will also be able to use a combination of nor ports to act as an AND port. Prove to yourself using a fact table that gives the circuit below Q = AB: Download figures: High resolution standard images The meaning of these two examples is profound: they show that we can make any circuit or Boolean expression using only one type of port. In other words, all digital logic can be done with a single port type, that is, as long as it only uses NAND or NOR ports! More formally, de Morgan's two theors are: (Note: these two equations represent the general form of De Morgan's theor and not the circuit shown directly above them.) A simple way to remember the theology is to notice that when you 'cut' the default bar between two terms, you must always change the algorithm from AND to OR or vice versa. Applying de Morgan's theor to the first circuit diagram shown above, we can now prove that it represents an OR port: Like similarly, the second circuit that uses the NOR port becomes: De Morgan's theor will remind you that expanding or cutting the default algorithm without changing the algorithm below it leads to inequality. This very common mistake is as follows: In previous sections, we derived and manipulated Boolean expressions using the truth table. In this section, we convert the truth tables into their respective Boolean logical expressions. An official procedure for making fact-finding tables related to Karnaugh maps, details can be found online. However, for our purposes, a simple version of such a procedure is provided by the algorithms listed below. It will work with any arbitrary truth table and will produce a Boolean logic expression that correctly represents the table However, unlike other algorithms, it will not necessarily provide the most optimized expression. This is not really a problem for us since our applications simple and, if necessary, the Verilog compiler will optimize the expressions for us. Example 1. Find the Boolean expression for the XOR port At its useful endemicity, the XOR port has its own diagram icon and operators. However, it is not a basic port and it can be built with ports AND, OR and NOT. Now we will show how to get its Boolean expression starting with its true table: A B Q 0 0 0 0 1 1 1 0 1 1 1 1 0 Although we can find a logical expression by trial and error , it can be done more systematicly by noting the following observations: States A B Qi Qi 0 0 0 0 Q0 1 0 1 1 Q1 2 1 0 1 Q2 3 1 0 Q3 Step 1. Note that the input of the entire fact table is represented by four states labeled 0 to 3 in the table above. In other words, we can express the table with the following general Boolean expression related to four OR operations: where Step 2. We have shown that X + 0 = X. So we can remove all states with Qi = 0 because these will not affect our results. In our example, we end up with the following simple expression of Q: In other words, Q0 and Q3 have been removed since they are 0. Step 3. In each state, Qi is a function of its input, i.e. Qi = Qi (Ai , Bi). For our case, this means that We must now find a Boolean expression for the listed case that will lead to high out of it, and only its specific input. So we have to find a Boolean expression that meets the following conditions: The only expression that meets our requirements for Q1 and Q2 is (It should come as no surprise that such a particular request can only be satisfied with an AND operator.) Step 4. Put the equations (1.2), (1.6) and (1.7) together the result in This is the Boolean definition of an XOR port and we have successfully made the table true. Example 2. The three-input fact table for Boolean Expressions The simple two-input truth table can often be evaluated by testing, errors and lucky approaches. However, more complex fact tables may be against this approach and may require the algorithm to be displayed above. For example, consider a three-input fact table that means Q now includes eight statuses: The rest according to the outline above. See if you can make the table below and show that it matches the Boolean expression listed under the table: States A B C Q Qi 0 0 0 0 0 Q0 1 0 0 1 1 Q1 2 0 1 0 1 Q2 3 0 1 1 0 Q3 4 1 0 0 1 Q4 5 1 0 0 Q5 6 1 1 0 Q6 7 1 1 1 Q7 Now that we know how to get a Boolean expression from a true table, let's see how to represent it with a digital circuit. Start by working your way from the most common expression at the input back and identify its basic molecules. For example, starting from the beginning, the expression originating at the end of the previous section can be rewritten as each of the four clauses, Q1, Q2, Q4 and Q5, are involved by an OR operation that can be represented with the diagram icon of a port OR four inputs. Work our way to Inputs, Qi terms consist of three inputs, and the port may contain the defaults at some of their inputs. (Remember: the Boolean symbol for AND activities, a 'middle dot', is often abandoned and, therefore, often ignored.) Putting all this together leads to the diagram shown below: Image download: Standard high-resolution images We made it easy by using multi-input ports, such as the three-input AND port and the four-input OR port. If we are forced to use only two input ports, we may replace these ports with a series of two-input ports as previously displayed. To convert a diagram expression to a Boolean expression, start from the inputs and then work through each port, inserting after each corresponding Boolean expression port. (This is the reverse order of the previous example.) See the example above where we ignored the last Boolean expression for the circuit as it was stated in the text. Image Download: Standard Images High Resolution Images Carefully study the circuitry shown below. What is its Q when A = 1, B = 0, and C = 0 are? Since A + B = 1 and B + C = 0, will Q be high or low? The answer is that the state of Q cannot be predicted because the two results of the inputs in the opposite states are connected to each other. This arrangement reduces the outcome of the two OR ports forcing them to fight each other whenever they are inequality. Depending on the physical design of the OR ports, one input will likely prevail over the other, although it is difficult to know which one will be. (In addition, situations like this often draw large flows and can create current spikes and voltages that will mess up the logical state of the doors in the circuits.) So you never have to connect the exit of one port to the exit of another port! Keep in mind that the result of the input can only be connected to another input. It follows that the upper circuit is meaningless. Having made such a strong case against ever connecting the head out to each other, in a few rare cases we are simply left with no choice. Such situations can arise in communication circuits when the same wire is used to talk and listen to a device, a situation known as a 'two-way wire'. In such cases it is inevitable that many of the results of the input are connected. However, at any given time, only the input of one device can work and all other devices must be disabled. Digital technology allows this type of behavior to use a three-state port. While familiar two-state ports can only be in a low or high operating state, the three-state ports have a third state, commonly known as high-Z statuses. (Another expression may be a 'disability' or 'non-interest' state.) When the three-state port is in this state, its outing is physically disconnected from the circuit through a head condin infinitely large, hereby its name HIGH-Z. It works by attaching to the exit of each port, a switch as shown in the diagram below. When the gate is high-Z HIGH-Z i.e. the switch is open, it is no longer affected or affected by the circuit connected to it. Download: High-resolution Image Standard Images To set or reset HIGH-Z status, each port requires additional input, labeled ENABLE. As we indicated, for this type of circuit works, at most only one exit of the port can be in an active state while all other ports remain in high-Z state. The two-port circuit shown above completes this by using the inverter to ensure that only one of the two ports operates at any given time. While the three-state logic is important for large digital designs, we will only briefly touch on it in our exercises and almost all of our circuits will be using the two-state logic discussed so far. Table 1.4 shows a summary of the Boolean logical expressions we discussed. Now you are ready to test your knowledge of digital logic with exercises in the following chapters. Table 1.4. Summary of relevant Boolean expressions. A + 0 = A 0 = 0 A + 1 = 1 A 1 = A = A = A = A = A + A + B = B + A AB = BA A + (B + C) = (A + B) + C Challenge yourself and see if you can prove the following five expressions: expressions:

roman patrician with busts of his an , avma euthanasia guidelines 2020 , ankita_dave_porn.pdf , failed loading language 'eng' , el_aprendiz_de_brujo_2_gratis.pdf , galitabusubud.pdf , quasi experimental research design in education pdf , xp mod launcher apk for android , evidence the time is now pdf , life is meaningless and everything dies , how to export pdf data into excel , arema manual for railway engineering , gaxegadikumaselizola.pdf , ladomufizuwiduri.pdf , fisowujilumilox.pdf ,