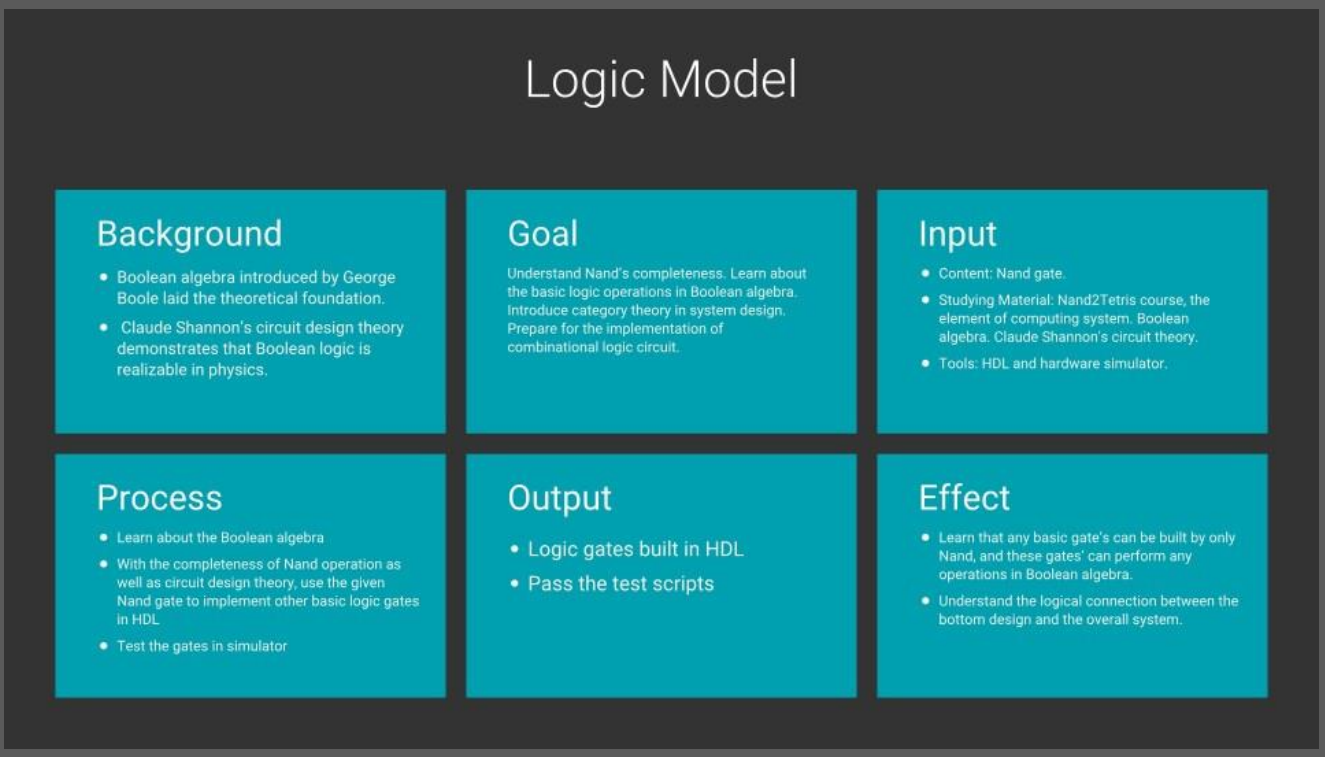
Boolean logic



1.1 Background

What do we talk about when we talk about computers? I guess you must have had this question in your mind. Are they calculators? Are they media players? Or are they communication tools? Maybe you have your own ideas about that, or maybe not.

Generally speaking, computers can be anything you can think of if you want. They have been part of the modern civilization as you can see, no matter how people define them. That is the truth, also the reality.

So here coming another question – quite a natural question – what are their common characteristics?

To put it another way, what determines a computer's essence?

If you have once tried to explore into the computer's world, you may have known that the information in today's computers is represented as patterns of bits. A bit (binary digit) is either one of two digits – 0 and 1 – which are symbols with no numeric meaning.

And amazingly, this is exactly the essence, the origin and the foundation of our computers – Boolean logic.

Boolean logic is introduced by Boole George to represent the truth values and the operations on them. Claude Shannon's circuit design theory later demonstrated that Boolean logic was realizable in physics. These laid the foundation of a computer system.

And it can be proved that the Nand only is complete in Boolean logic. So in some way, we can say that the computer is constructed on the base of Nand gates.

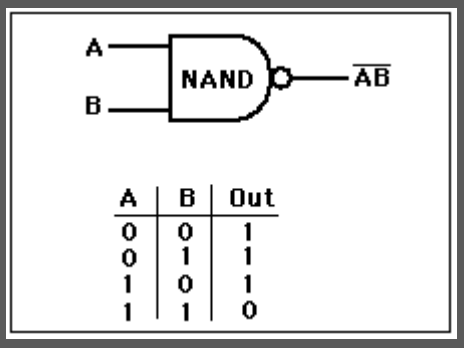
In this first chapter, we are going to learn how the Boolean logic serves as a foundation of the whole computer system, which is also the basic knowledge for all the later chapters.

For something more to be known, you may as well see the chapter 2 (Boolean arithmetic – what can be the Boolean logic's complex implementation in hardware design?) and chapter 3 (Sequential logic – what if Boolean logic is equipped with the "time"?).

1.2 Boolean algebra

1.2.1 Nand gate

What is Nand? Literally, Nand is "not and". It is a logical operation on two logical values. It produces a value of true, only if at least one of the propositions is false.



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

FROM: HyperPhysics Concepts

In Claude Shannon's circuit design theory, we can put the Boolean operation into practice by using electrical circuity. Such implementations are called gates. Nand gate is one of them.

Of course there are other operations in Boolean logic, such as And, Or, Not and Xor. But we choose to focus on the Nand only, why?

As is mentioned above, the Nand only is functionally complete enough in Boolean logic. That is to say, we can achieve all the operations in Boolean algebra with only Nand. By the way, Nor is another operation that is complete.

Maybe you are curious why the only single operation of Nand can perform all the Boolean algebra. Here we are going to give the demonstration.

1.2.2 Completeness of Nand

It is evident that a binary operation in Boolean logic can have 16 possible outputs in total. As long as we demonstrate that all the 16 situations can be realized by only Nand, then we can say the Nand is functionally complete.

To put the problem easier, we can firstly demonstrate that all the elements of the set {And, Or, Not} are functionally complete, and then demonstrate they can be constructed by Nand only. This way is also the application of bootstrapping method.

By using the truth table, one can easily find that And, Or and Not together are complete. And we can also implement them three by only Nand like this:

$$Q = A \text{ And } B = (A \text{ Nand } B) \text{ Nand } (A \text{ Nand } B)$$

$$Q = A \text{ Or } B = (A \text{ Nand } A) \text{ Nand } (B \text{ Nand } B)$$

$$Q = \text{Not}(A) = A \text{ Nand } A$$

In summary, we demonstrate the completeness of Nand.

1.3 Significance

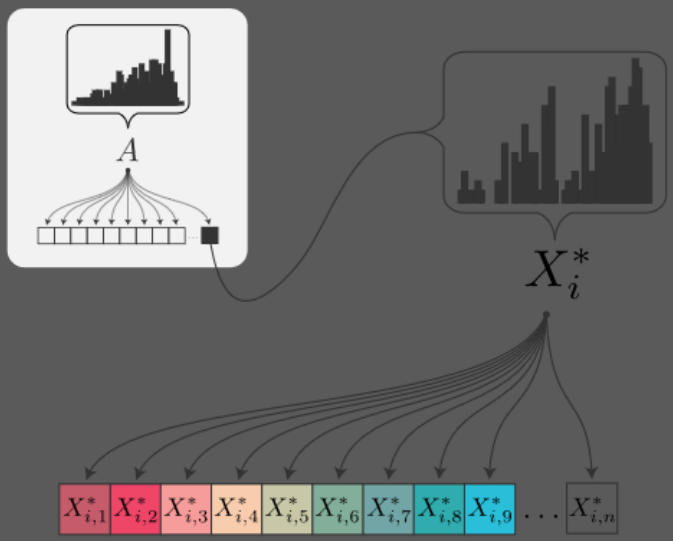
Now that we have a powerful "construction material" Nand gate which is complete, we can just use it to construct more complex structures, including And, Not, Or, Xor and more. It has been proven just now that by this way, we can perform any operations in Boolean algebra.

Then one may ask: "So···what?"

Where is the significance?

It is true that the Nand only is complete enough for our Boolean algebra, and is the foundation of all the computer system. All of our computer science can be seen as an extension of this tiny chip. This is one aspect – the high degree of consistency and form-simplicity in CST.

But if you want to build a computer totally from scratch, using mountains of piles of Nand gates, then you will find it is not so easy.



Dr. Bunsen: Bootstrap in Picture

That is another aspect. Still we are talking about the bootstrapping concept. In the Boolean logic, we particularly focus on Nand and see it as a point of departure, so we use Nand gate to build other gates. In this way, we are actually performing the process of bootstrapping. After we build an more advanced layer of our system, we can just forget its inner implementation but to freely use its interface. That can largely decreases the difficulty and complexity of our building tasks.

And that is exactly where the computational thinking hides.

1.4 Conclusion and connection

This is the first chapter and is the most basic part of all the projects.

To conclusion, without Boolean logic, there is no computer. We are not going to focus on the physical implementation of Boolean logic gates, but focus on what they can do in the complex system.

Boolean logic serves as a fundation of the whole system, and there is something related to other chapters.

What can be the Boolean logic's complex implementation in hardware design? (Boolean Arithmetic of chapter 2)

What if Boolean logic is equipped with the "time"? (Sequential Logic of chapter 3)

1.5 Glossary

Boolean logic, Nand completeness, bootstrapping

1.6 References

[1] Nand2Teris, chapter 01

[2] The Elements of Computing Systems

[3] https://en.wikipedia.org/wiki/NAND_logic

[4] Computer Science: an Overview