Computers Debunked

A Box of Switches

Techno-Plaza -Oct 17, 2022

Introduction

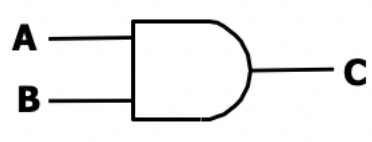
When someone that should know explains what a computer is, you either hear "It's a box of switches" or "It's all just 1s and 0s." That's like saying, The sun warms the blood." There a bit more information that is needed to understand the sun. What I want to do here is put the spade in the ground and dig a little deeper into the topic of what makes a computer compute.

During and just after World War II, engineers realized a digital machine could be built that could automate mathematical calculations that originally were needed to make quick direction calculations to point a cannon and hit an enemy target or continually track with a radar antenna, another recent wartime invention. There were various algorithms in literature dating back to the Greek philosophers about how logic worked that could be applied in an electronic world and implementation was just beginning to happen. So in our modern world when we abstract away all the basic meaning of what a computer is we totally miss the point and confuse anyone new to computers or those who just want to understand the underlying principles going on inside the box. So let's think inside the box for once.

First and foremost concept to understand is a computer is a state machine. Each and every step it takes has to be controlled and sequenced properly or the machine defaults to a failed mode. Next, way before we get to an answer on a screen, there are many circuits, whether they be a bunch of relays, vacuum tubes, or micro-electronics that are preforming logic decisions. Yes, basic logic like, "If this signal is the only one present, then no output. But if this signal is accompanied by that signal, then we get an output." This is a logic decision point made by a machine better known as an AND operation. We only need a few more logic building blocks to really get something that can do much of what a modern computer does. We need an OR operator that will pass various signals on without judgement. We need an XOR or exclusive OR that only passes a signal if one of many is present. If more inputs are present, then the input, in general, is not exclusive so no output. We need and inversion component to flip signals to their opposite state. We need a circuit the can hold on to a signal value while other circuits around it are allowed to change. You could call this a memory but in this context it is better labelled a register. We also need a circuit path selector, called a multiplexer, to route signals and that's all we need to get way down the road. We also need imagination because we need to visualize circular paths with their state controlled, and a machine that can repeat operations, switch to other operations, or service operations temporarily and then return to a known state. All of this can be accomplished with straight forward logic provided by the simple circuits described above. All of this "control" circuitry has

to be in-place before we get to the fancy algorithms that add, subtract, multiply, divide, do logarithms, or process images. All of this is rather simply in principle.

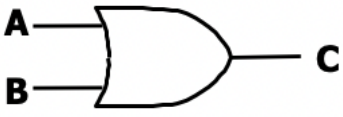
Let's look at a circuit that does the AND operation. Its attributes include a name: AND gate, implying it's a gate that only opens with all the inputs signals are present whether two or more and remains closed otherwise. Typically the operation is represented by a "truth table." It has the following symbol:



AND gate

A	B	C
L	L	L
L	H	L
H	L	L
H	H	H

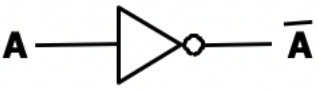
Notice our signals are labelled as either high or low. So where's the 1's and 0's? Here's our first abstraction: High and low or 1 and 0 or true and false or a high voltage and low voltage all imply where a signal is present or not. We'll get to where 1's and 0's represent numbers later (See TechnoNote 04 for details about Numbers).



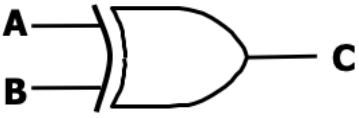
OR gate

A	B	C
L	L	L
L	H	H
H	L	H
H	H	H

OR gate truth table



Inverter

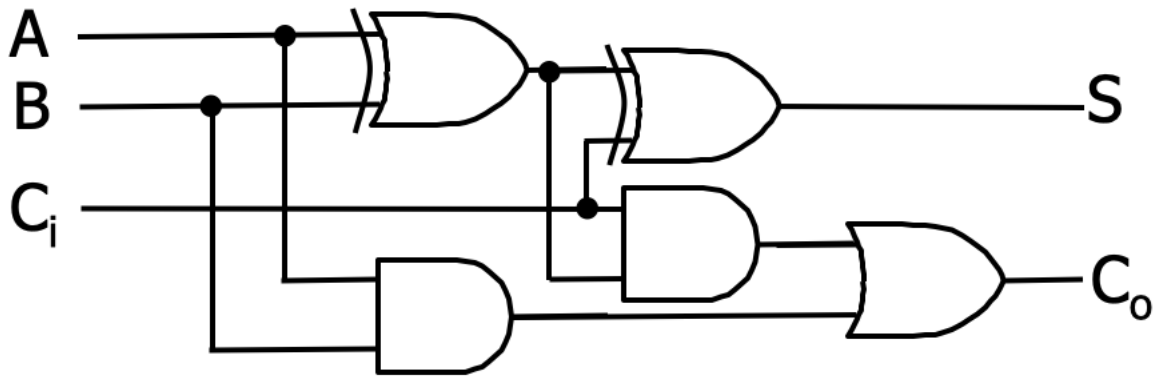


XOR gate

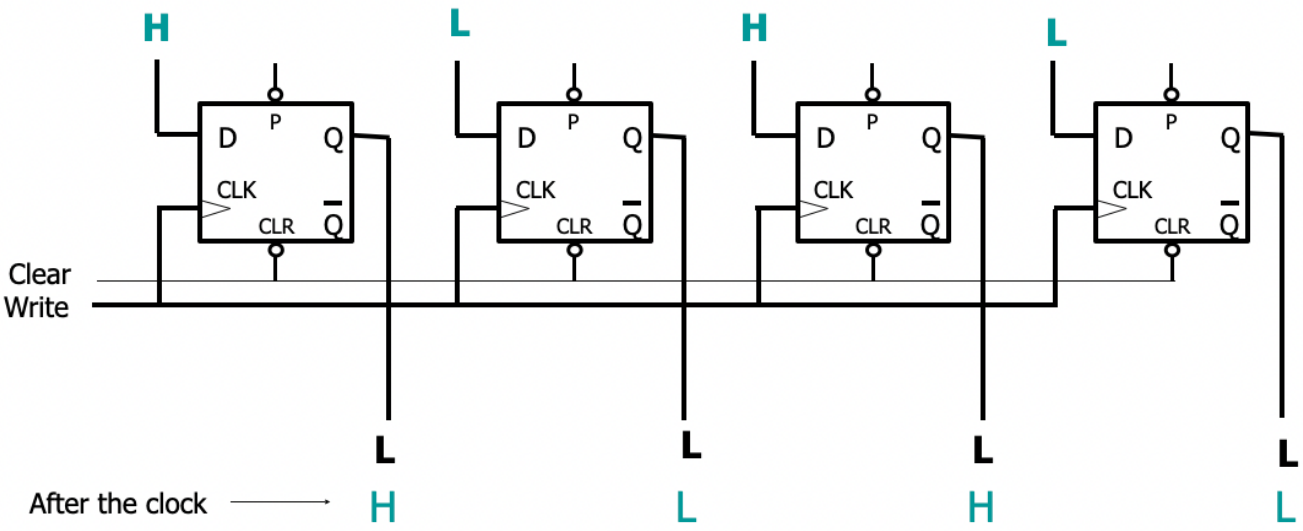
A	B	C
L	L	L
L	H	H
H	L	H
H	H	L

XOR gate truth table

So what can you do with logic gates -- math.



The first XOR provides a sum of two binary bits and the AND provides a carry if both A and B are both present. To account for a carry from a previous add Ci, we add again for the sum. This circuit repeats for each bit pair for the size of the data words involved.



This is a register made up of D-Flip-Flops. Any data on the D-pins can be held in the register once a clock pulse is issued. The output can be tied to a bus via an interface buffer. Memory circuits are complex designs unto themselves but contain control logic gates around the memory cells. Memory space is where we can store program instructions and data. We use registers to temporarily store data from main memory to be acted upon then returned to memory. The only difference between data and instructions is the location each are stored and the interpretation of instructions. Data and instructions are just bits representing on or off state. Instruction bits are arranged so the most significant end of a data word is an op-code that is read that sets up the control logic to perform a particular operation. Next bits in the

instruction typically are used to select registers to store operands or address information where to get data from memory. Once the appointed registers have their data, the chosen operation is performed by logic gate combinations and returned to a register to be transferred back to a memory location. All of this is done by logic gates. The memory space dedicated for a program is cycled through as each instruction is executed in sequence until the end of the program and the state of the machine is then returned to the operating system. There are a lot of logic gates involved with system clocking as well because of the need to have some clocking inverted or delayed and gated on or off.

It's true that logic gates are made up of switching circuits based on the transistor but to just call the whole computer a box of switches circumvents their logic implication. Their logic combinations are what make the computer tick, well and the system clock. The other distinction is to realize 1's and 0's used for binary counting and math operations are just data set in a memory word that could just as well represent the state of the machine, the address of more data or be used to set up an LED display pattern. If these same bits are meant to be boolean information, then the bits represent true of false values. If the bits are the result of reading an analog signal, they could be the voltage values at particular reading times.

It actually makes more sense to think of a computer as a state machine that has the ability to be programmed to do various things automatically. Computers can compute, control assets, read sensors, communicate over networks, and an endless set of tasks and all they ask in return is a little electricity. So don't get hung up on the 1's and 0's enigma. Electronic switching of logic operations is where the code meets the road.