

## EGR 215 Topics in EE VLSI

**VLSI or Digital Systems deals with the design and use of digital electronics, most notably in computer hardware and computer subsystems.** VLSI are the "gate stringers" in a world divided between them and "code slingers".

To place it in context:

**Computer Science deals with the theory of computation, and with programming languages.**

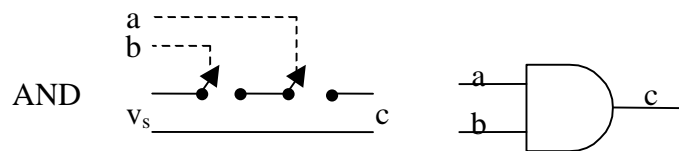
**Computer Engineering** used to deal exclusively with **instruction set architecture**. This involved tradeoffs between hardware (larger instruction sets are more expensive to implement) and software (smaller instruction sets are less efficient to execute). Computer Engineering has expanded in one direction to deal with **the applied science of programming ("Software engineering")** and in the other direction to overlap some of the **hardware issues**.

**Digital Systems deals with the hardware implemented using digital electronics.**

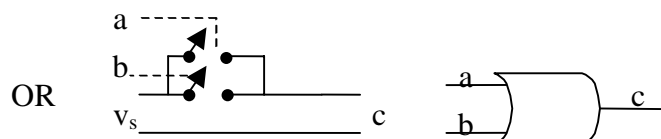
VLSI actually refers to the current technology of hardware implementation. VLSI stands for **Very Large Scale Integration**, and refers to putting many, many components (~20 Million) on one substrate, usually a silicon wafer. VLSI has brought us the electronic computer and the microprocessor, which have found almost universal application, and many other digital electronic applications.

Digital Electronics is the electronic implementation of **digital logic**. Digital logic has been around since the Greeks in 300 BC. It is based on four operations.

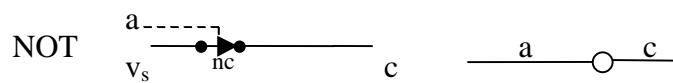
a	b	c
0	0	0
0	1	0
1	0	0
1	1	1



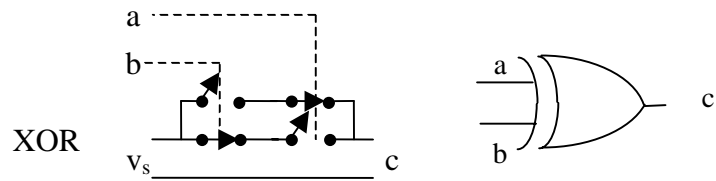
a	b	c
0	0	0
0	1	1
1	0	1
1	1	1



a	c
0	1
1	0



a	b	c
0	0	0
0	1	1
1	0	1
1	1	0



The drawing shows a truth table, a controlled switch implementation of electrical logic circuits, and the corresponding logic symbol. In the **AND operation**,  $c = a \text{ AND } b$ , that is, when both  $a$  and  $b$  are high voltage,  $c$  is high. Otherwise,  $c$  is low voltage. In the **OR operation**,  $c$  is high when either one of  $a$  or  $b$  are high. In the **NOT operation**,  $c$  is low when  $a$  is high. (And the little circle usually appears at the output of an AND gate, making it a NAND gate. All the other logic operations can be created from NAND gates.) Finally, in the **XOR (exclusive OR) operation**,  $c$  is high when  $a$  and  $b$  are opposite each other.

Note that the values in these diagrams are all either high or low - one or zero - binary, base 2.

As you may imagine, you can **implement** these functions **using** paper and pencil, mechanical linkages, electromagnetic relays like those used by Morse in the telegraph, vacuum tubes, or **transistors**. And in fact all of these methods have been and continue to be used (except tubes).

Digital mechanical calculating machines (as opposed to analog calculators, known as slide rules) date back to the French mathematician Blaise Pascal in 1642. Herman Hollerith in the 1920's in the United States pioneered the use of punch cards and electrical sorters as a form of tabulating machine. The operator would put a punched card in a "press". Holes would allow electrical contact, which would pop open the lid of one of many bins, and the operator would put the card in the bin - an automated sorting method. Computers still spend a lot of time sorting. As time went on, Hollerith worked on making the sorting function more and more automated, and more and more sophisticated.

In 1937 Alan Turing, a genius mathematician in England, had a huge insight when he realized that with a very simple state machine, he could compute (solve) any logical problem, including mathematics. In his imaginary machine, there was a paper tape divided into blocks. Each block could have a mark, or not have a mark (1's and zeros). The machine would interpret the marks as instructions for one of four functions: Move the tape one block to the left, one block to the right, make a mark or erase a mark. It was the first **stored-program computer**.

Now, Turing's work was computer science. Actually building the machine (which did not happen due to WWII) would be digital systems. And getting it to do anything useful clearly needed it to **operate at very high speeds** - far faster than any mechanical calculator.

In 1937-42, John V. Atanasoff, a professor of math and physics at Iowa State University, built the first electronic digital computer, with EE grad student Clifford Berry. He never completed the patent. Used vacuum tubes to do binary mathematics (add, subtract, carry) and logic. Used rotating drum memory, 2 x 50 numbers of 30 bits each. Used a rotating base 10 to base 2 converter. Was not (apparently) a stored program machine. Programmed by punch cards.

1944 Howard Aiken built the Harvard Mk I, a 35 ton vacuum tube based electronic calculator that could add, subtract, multiply and divide. Programmed by paper tape. Used for ballistics computation (firing tables).

1946 John Mauchly, an EE professor at Penn, and his grad student J. Presper Eckert, built the first electronic stored program computer, ENIAC. They used math and logic circuits from Atanasoff. It had 18,000 tubes, and was used for Army firing table calculations. By 1948 they had founded the Univac company and were selling the first business computers.

Transistors were an obvious improvement on transistor logic. Computers required many duplications of circuits, for example, Atanasoff's calculator has a dozen "add-subtract modules" (ASM). WWII had already led to the idea of **printed circuit boards** used to make many **identical modules** from individual components. It was inevitable that someone would think of simply putting several components on one chip. It took until 1958. The first integrated circuit was built by Fairchild semiconductor, founded by engineers leaving Shockley transistor. The simultaneous independent inventors were Robert Noyce and Jack Kilby. Kilby was working at Texas Instruments. After a patent fight Fairchild got the patent rights, and became rich. The integrated circuit (IC) had two transistors on it. Very quickly the transistor count increased to 10.

By the 1970's, the number of transistors was up to 1000 per chip, fitted into the same size as the two transistor chip of 1958 by making the feature size smaller. A new term was coined, Large Scale Integration, or LSI. By the 1980's, transistor counts had reached **20 Million**, again in the same space, and the name **VLSI** emerged. There was some discussion of ULSI (Ultra large scale integration) in the 90's, but people seem to have realized that they were running out of superlatives. (What comes after Ultra?).

In 1965, Gordon Moore, an engineer at Fairchild who was one of the 8 who left Shockley, was asked to predict the future of integrated circuits. He predicted **that the number of transistors on a chip of constant area would double every year. Over 40 years, it's been more like every 18 months (Moore's Law)**. Still, this exponential increase in performance has made the digital electronics industry a major force for transformation of society, as well as making a lot of money for the engineers involved.

In 1968 Noyce and Moore left Fairchild and founded Intel to focus on microprocessors.

What do you do with all those transistors?

At first, people developed different logic circuits. You could buy a chip with four whole NAND gates on it! And wire them together to do interesting things.

After a while, people started doing the wiring on the chip as well, and functionality became both more complex and more specialized.

The cost of making a chip to do one specific thing, however, was becoming prohibitive. The first (working) chip cost a lot of money in engineering and in setting up fabrication. The second chip was incredibly cheap. You needed products that would sell millions to break even, and **as transistor count increased, the specialization limited sales.**

Prior to 1971, computers were made from bits and pieces. You had boards full of chips with logic and math circuits on them, and you put them together to make a computer. Some supercomputers are still made this way. But in 1971 Intel marketed the first **microprocessor**, combining the computational logic storage on a single chip, and that's been the trend in computer implementation ever since. **The advantage is that you can have a very complex product - some microprocessors are among the most complex of ICs today - that still has general applicability to a wide variety of applications because you can change the programming. The hardware is general purpose, and the chip is specialized to its task by software.** These microprocessor-based systems are becoming more and more common.

Microprocessors are very good at dealing with digital inputs and outputs. Unfortunately, the real world stubbornly insists on being analog. An increasing trend is the quick conversion of analog data to digital form (ADCs), followed by extensive digital processing (which can include transmission of data), and reversion to analog form (DACs). That's what goes on in a digital wireless phone. There are both cost and quality advantages to operating digitally, but in the past there has had to be an analog front end, a converter to digital, a digital processor, and an analog back end. That's too many chips! And so we find Mixed Signal integrated circuits, which combine analog and digital processing on the same substrate, to be a hot topic. The device demands of analog and digital are significantly different, so that reconciling them on one technology (substrate material and fabrication method and resolution) can be a challenge.

We also find lots of interest in **Application Specific Integrated Circuits, ASICs.** These are circuits designed for a specific application. The challenge is to make the engineering and fabrication costs low enough to reap the advantages of specialization.