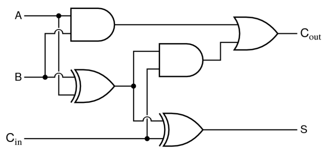
Eric Roberts
CS 54N

Handout #7
October 10, 2016

Binary Arithmetic and Digital Logic

Binary Arithmetic and Digital Logic



Eric Roberts
CS 54N
October 10, 2016

Great Ideas in Computing Hardware

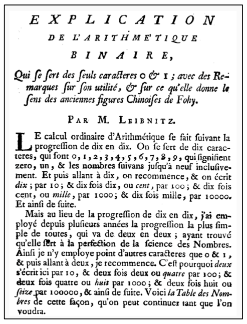
1. All data processed by computers—including the instructions used to create computer programs—can be represented using collections of individual binary digits, or ***bits***.
2. The amazing complexity of modern computers arises largely from the use of extremely simple components replicated on a massive scale.
3. The best way to manage the complexity of modern computers is to define a hierarchy of ***virtual machines***, each of which hides at least some of the complexity of the underlying layers.

The Power of Bits

- The fundamental unit of memory inside a computer is called a ***bit***—a term introduced in a paper by Claude Shannon as a contraction of the words *binary digit*.
- An individual bit exists in one of two states, usually denoted as **0** and **1**.
- More sophisticated data can be represented by combining larger numbers of bits:
  - Two bits can represent four (2 × 2) values.
  - Three bits can represent eight (2 × 2 × 2) values.
  - Four bits can represent 16 (2^4) values, and so on.
- This laptop has 16GB of main memory and can therefore exist in $2^{549,755,813,888}$ states. If you were to write that number out, it would contain more than fifty billion digits.

Leibniz and Binary Notation

- Binary notation is an old idea. It was described back in 1703 by the German mathematician Gottfried Wilhelm von Leibniz.
- Writing in the proceedings of the French Royal Academy of Science, Leibniz describes his use of binary notation in a simple, easy-to-follow style.
- Leibniz's paper further suggests that the Chinese were clearly familiar with binary arithmetic 2000 years earlier, as evidenced by the patterns of lines found in the *I Ching*.



E X P L I C A T I O N
DE L'ARITHMETIQUE
B I N A I R E,

Qui ſe ſert des ſeuls caracteres 0 *&* 1 *; avec des Remarques ſur ſon utilité, & ſur ce qu'elle donne le ſens des anciennes figures Chinoiſes de Fohy.*

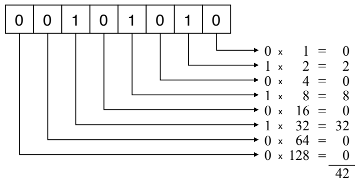
PAR M. LEIBNITZ.

LE calcul ordinaire d'Arithmétique ſe fait ſuivant la progreſſion de dix en dix. On ſe ſert de dix caracteres, qui ſont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, qui ſignifient zero, un, & les nombres ſuivans juſqu'à neuf incluſivement. Et puis allant à dix, on recommence, & on écrit *dix* ; par 10 ; & dix fois dix, ou *cent*, par 100 ; & dix fois cent, ou *mille*, par 1000 ; & dix fois mille, par 10000. Et ainſi de ſuite.

Mais au lieu de la progreſſion de dix en dix, j'ai employé depuis pluſieurs années la progreſſion la plus ſimple de toutes, qui va de deux en deux ; ayant trouvé qu'elle ſert à la perfection de la ſcience des Nombres. Ainſi je n'y employe point d'autres caracteres que 0 & 1, & puis allant à deux, je recommence. C'eſt pourquoi *deux* s'écrit ici par 10, & deux fois deux ou *quatre* par 100 ; & deux fois quatre ou *huit* par 1000 ; & deux fois huit ou *ſeize* par 100000, & ainſi de ſuite. Voici *la Table des Nombres* de cette façon, qu'on peut continuer tant que l'on voudra.

Using Bits to Represent Integers

- Binary notation is similar to decimal notation but uses a different ***base***. Decimal numbers use 10 as their base, which means that each digit counts for ten times as much as the digit to its right. Binary notation uses base 2, which means that each position counts for twice as much, as follows:

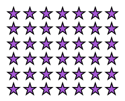


Numbers and Bases

- The calculation at the end of the preceding slide makes it clear that the binary representation 00101010 is equivalent to the number 42. When it is important to distinguish the base, the text uses a small subscript, like this:

$$00101010_2 = 42_{10}$$

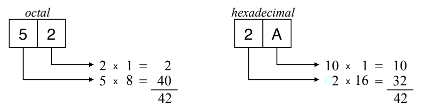
- Although it is useful to be able to convert a number from one base to another, it is important to remember that the number remains the same. What changes is how you write it down.
- The number 42 is what you get if you count how many stars are in the pattern at the right. The number is the same whether you write it in English as *forty-two,* in decimal as 42, or in binary as 00101010.



- Numbers do not have bases; representations do.

Octal and Hexadecimal Notation

- Because binary notation tends to get rather long, computer scientists often prefer ***octal*** (base 8) or ***hexadecimal*** (base 16) notation instead. Octal notation uses eight digits: 0 to 7. Hexadecimal notation uses sixteen digits: 0 to 9, followed by the letters A through F to indicate the values 10 to 15.
- The following diagrams show how the number forty-two appears in both octal and hexadecimal notation:



- The advantage of using either octal or hexadecimal notation is that doing so makes it easy to translate the number back to individual bits because you can convert each digit separately.

Exercises: Number Bases

- What is the decimal value for each of the following numbers?
- As part of a code to identify the file type, every Java class file begins with the following sixteen bits:

1	1	0	0	1	0	1	0	1	1	1	1	1	1	1	0

How would you express that number in hexadecimal notation?

Bits and Representation

- Sequences of bits have no intrinsic meaning except for the representation that we assign to them, both by convention and by building particular operations into the hardware.
- As an example, a 32-bit word represents an integer only because we have designed hardware that can manipulate those words arithmetically, applying operations such as addition, subtraction, and comparison.
- By choosing an appropriate representation, you can use bits to represent any value you can imagine:
  - Characters are represented using numeric character codes.
  - Floating-point representation supports real numbers.
  - Two-dimensional arrays of bits represent images.
  - Sequences of images represent video.
  - And so on . . .

Representing Characters

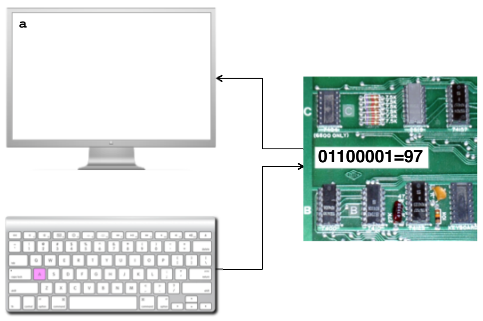
- Computers use numeric encodings to represent character data inside the memory of the machine, in which each character is assigned an integral value.
- Character codes, however, are not very useful unless they are standardized. When different computer manufacturers use different coding sequence (as was indeed the case in the early years), it is harder to share such data across machines.
- The first widely adopted character encoding was ASCII (*American Standard Code for Information Interchange*).
- With only 256 possible characters, the ASCII system proved inadequate to represent the many alphabets in use throughout the world. It has therefore been superseded by Unicode, which allows for a much larger number of characters.

The ASCII Subset of Unicode

The following table shows the first 128 characters in the Unicode character set, which are the same as in the older ASCII scheme:

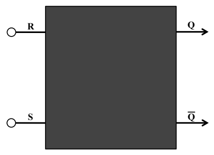
	0	1	2	3	4	5	6	7
00*x*	\000	\001	\002	\003	\004	\005	\006	\007
01*x*	\b	\t	\n	\011	\f	\r	\016	\017
02*x*	\020	\021	\022	\023	\024	\025	\026	\027
03*x*	\030	\031	\032	\033	\034	\035	\036	\037
04*x*	*space*	!	"	#	$	%	&	'
05*x*	(	)	*	+	,	-	.	/
06*x*	0	1	2	3	4	5	6	7
07*x*	8	9	:	;	<	=	>	?
10*x*	@	A	B	C	D	E	F	G
11*x*	H	I	J	K	L	M	N	O
12*x*	P	Q	R	S	T	U	V	W
13*x*	X	Y	Z	[	\	]	^	_
14*x*	`	a	b	c	d	e	f	g
15*x*	h	i	j	k	l	m	n	o
16*x*	p	q	r	s	t	u	v	w
17*x*	x	y	z	{	|	}	~	\177

Hardware Support for Characters

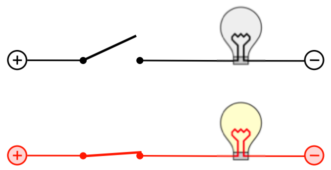


Implementing Bits

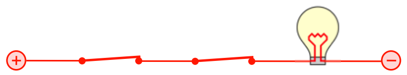
- If you want to implement a bit inside a computer, you need to come up with a mechanism that
  - Can exist in either of two states.
  - Allows other parts of the computer to *read* the state.
  - Allows other parts of the computer to *write* the state.
- Conceptually, these capabilities are captured by a hardware unit called a ***flip-flop***, which can exist in either an *off* or an *on* state:



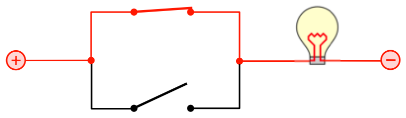
Switching Circuits



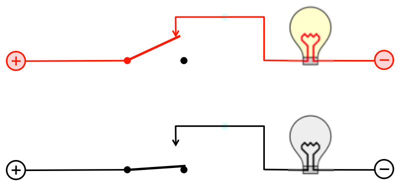
The AND Circuit



The OR Circuit

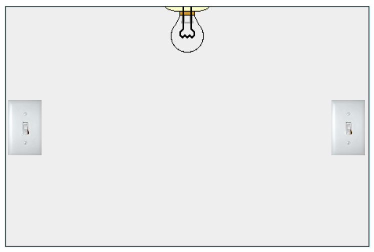


Implementing NOT with Switches

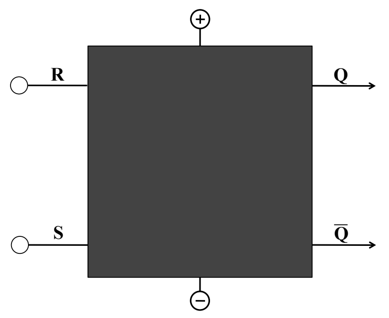


Exercise: Controlling Room Lights

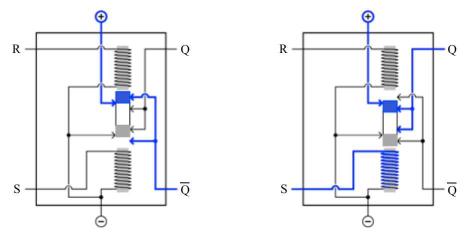
- Suppose you have a room with one light that you want to control from two switches at opposite ends of the room. How would you implement this functionality with switches?



Exercise: How Might You Build a Bit?

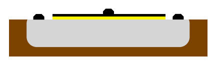


A Solution Using a Relay



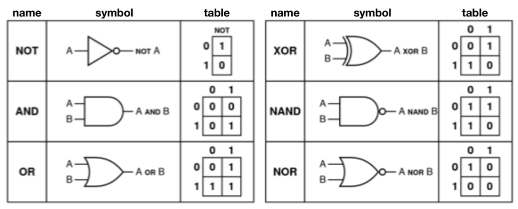
Implementing Switches in Silicon

- In modern computers, switches are usually implemented in chips made based on a ***semiconductor***, which is simply a material (typically some form of silicon) that is conducting in some circumstances and insulating in others.
- One of the most common components in today's computers is the ***field-effect transistor*** (or ***FET***), which you can implement in semiconductor using a structure that looks like this:



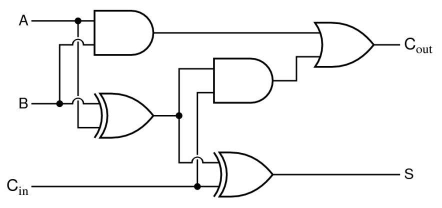
- Field-effect transistors come in two types. In a **n**-type FET, the presence of a charge on the gate allows current to flow across the transistor as a whole. In a **p**-type FET, a charge on the gate inhibits the current flow.

Logic Gates



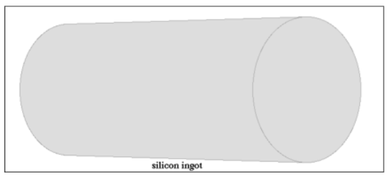
name	symbol	table
NOT	A — NOT A	$\begin{array}{c\|c} & \text{NOT} \\ \hline 0 & 1 \\ 1 & 0 \end{array}$
AND	A, B — A AND B	$\begin{array}{c\|cc} & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array}$
OR	A, B — A OR B	$\begin{array}{c\|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 1 \end{array}$
XOR	A, B — A XOR B	$\begin{array}{c\|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$
NAND	A, B — A NAND B	$\begin{array}{c\|cc} & 0 & 1 \\ \hline 0 & 1 & 1 \\ 1 & 1 & 0 \end{array}$
NOR	A, B — A NOR B	$\begin{array}{c\|cc} & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 0 \end{array}$

Full Adder



Chip Fabrication

The manufacture of a semiconductor chip begins with a crystal of extremely pure silicon. The silicon itself is extracted from beach sand and then grown into a cylindrical ingot in such a way that impurities represent no more than one of a trillion atoms in the crystal.

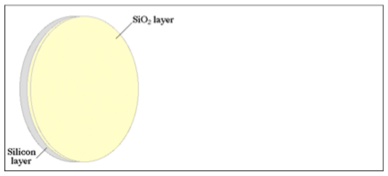

silicon ingot

Chip Fabrication

The silicon ingot is then sliced into individual wafers, which are typically 8-12 inches in diameter and less than a millimeter thick. Each individual wafer is then polished to remove any irregularities in its surface.

Chip Fabrication

The polished wafer is then heated in the presence of oxygen to create a thin layer of silicon dioxide (SiO_2). The SiO_2 layer is nonconducting, and thereby serves to insulate the silicon wafer from the various materials that will be layered above it.

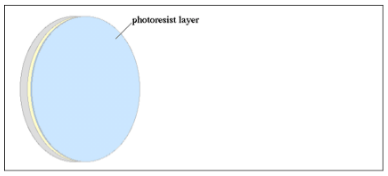


Chip Fabrication

The problem now is to remove just the right parts of the SiO_2 layer to expose the silicon underneath, so that these regions of the chip can be used as transistors. This phase of chip fabrication is accomplished using a process called ***photolithography***.

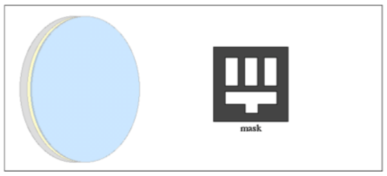
Chip Fabrication

The first step in the photolithographic process is to coat the insulated wafer with a layer called ***photoresist,*** which changes its chemical composition when exposed to light.



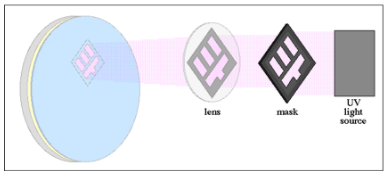
Chip Fabrication

The pattern to be inscribed on the chip is taken from an optical template called a ***mask***. The mask looks very much like an old photographic negative, with the transparent regions indicating which parts of the chip surface need to be etched away.



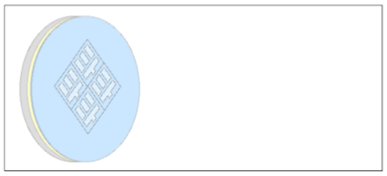
Chip Fabrication

The image on the mask is transferred to the chip surface by shining ultraviolet light through the mask and then using a lens to focus the image on the desired area of the chip.



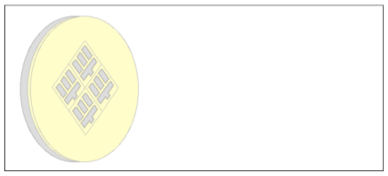
Chip Fabrication

The same pattern is then created in a new position in other parts of the wafer by repositioning the image. A typical wafer can hold thousands of identical chips, each of which can be etched into the wafer simultaneously.



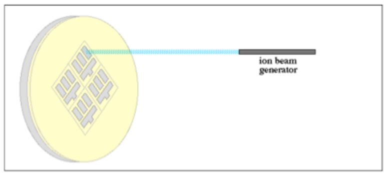
Chip Fabrication

The exposed photoresist is then removed chemically, exposing the SiO_2 layer underneath. A further etching step removes the SiO_2 in the exposed areas, revealing the silicon layer. The final step in the etching process is to remove the unexposed photoresist from the wafer.



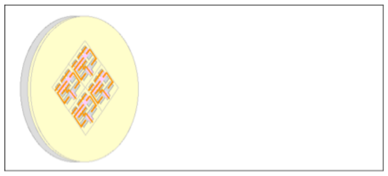
Chip Fabrication

The next phase in the chip-making process is to introduce new elements into the silicon wafer to make it semiconducting. An ion beam bombards the surface with the doping atoms needed to create **n**-type and **p**-type regions in the exposed silicon wafer.



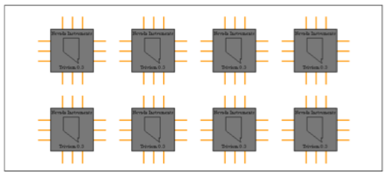
Chip Fabrication

To complete the fabrication of the chips on the wafer, additional photolithographic steps are used to lay down alternating layers of insulators and conductors. The conductors (typically aluminum and tungsten) create the wiring pattern for each chip.



Chip Fabrication

Once the wiring is complete, diamond saws cut the wafer into the individual chips, which are then encased in plastic, after connecting metallic leads to the appropriate contact positions on the chip.



Chip Fabrication

The individual chips are then tested to make sure that every internal circuit works correctly. Given the complexity and extremely fine tolerances of the manufacturing process, several of the chips on each wafer typically need to be discarded.