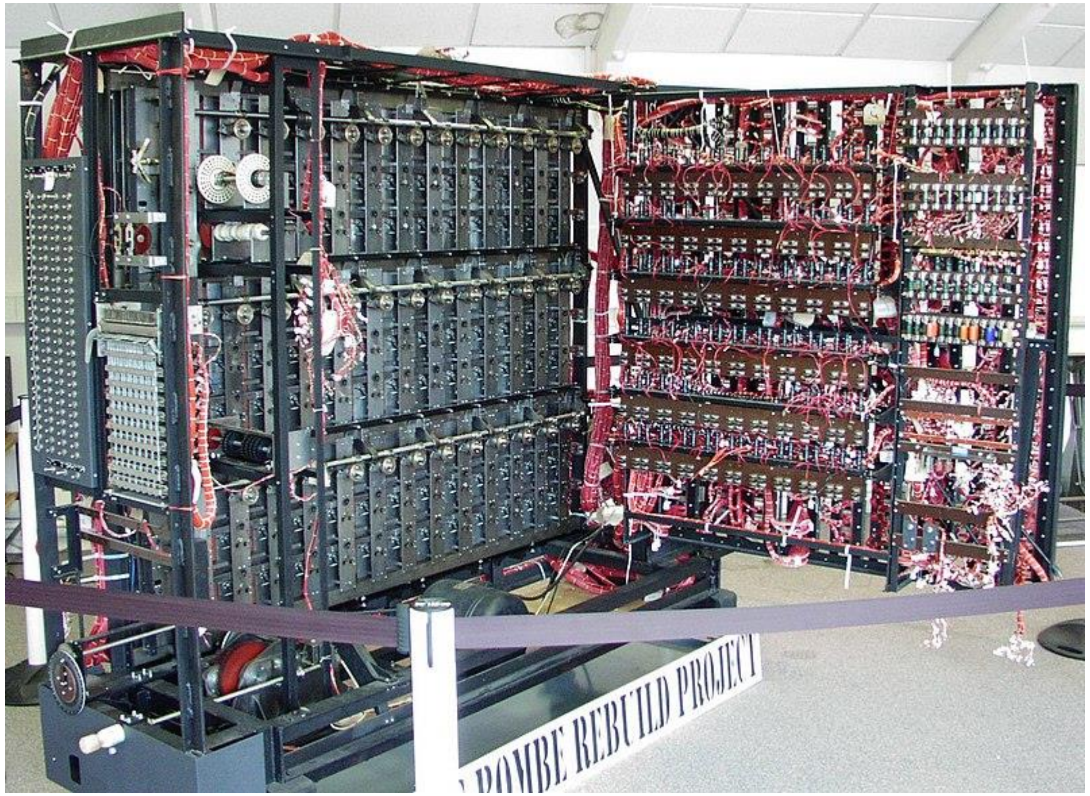
Intro to Computation

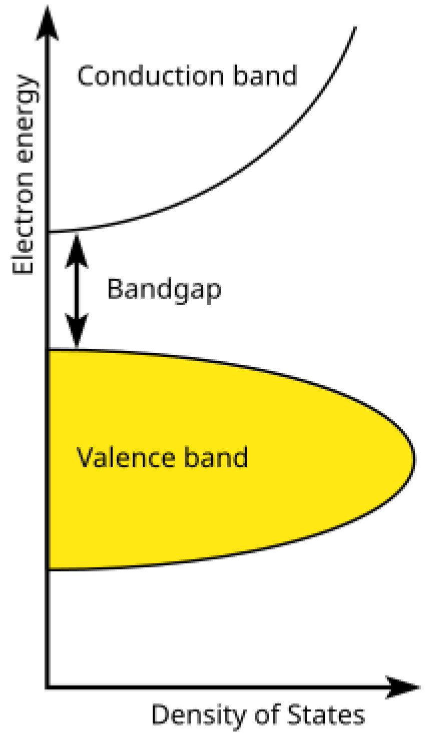
Early Computers

- One of the earliest computers was invented by the British mathematician Allen Turing.
  - It was used to break the cryptographic codes being used by Germany during world war 2.



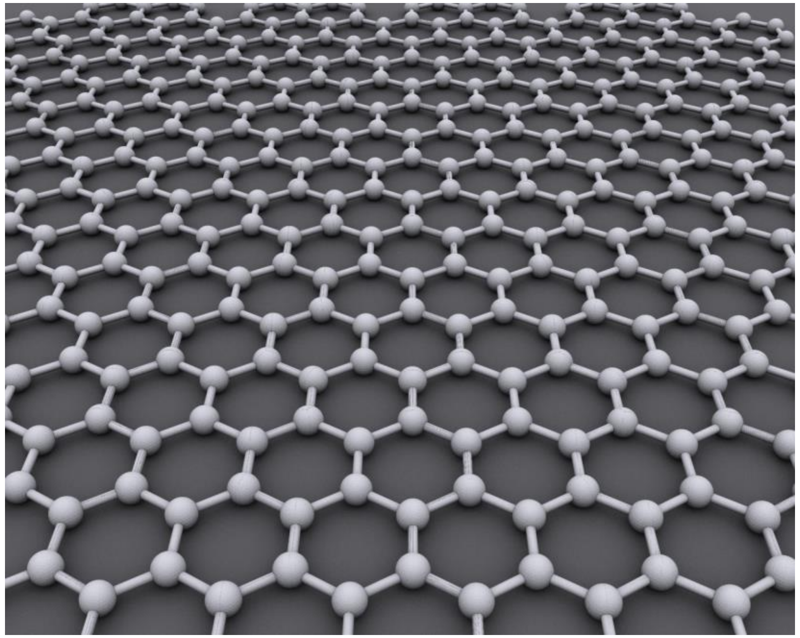
Semiconductors

- The building block of modern computing.
- Semiconductors are somewhere between a conductive material and an insulator.
- Electrons around an atom exist in one of 2 electron bands.
  - Valence Band: When the electrons are at their lowest or “ground state” energy, the electrons fill the valence band.
  - Conduction Band: When energy is added to the electron’s they jump up to the conduction band if enough energy is added.
- Metals (conductors) have no energy gap between bands.
- Insulators have a very large energy gap between bands.
- Semiconductors have a small energy gap between bands.

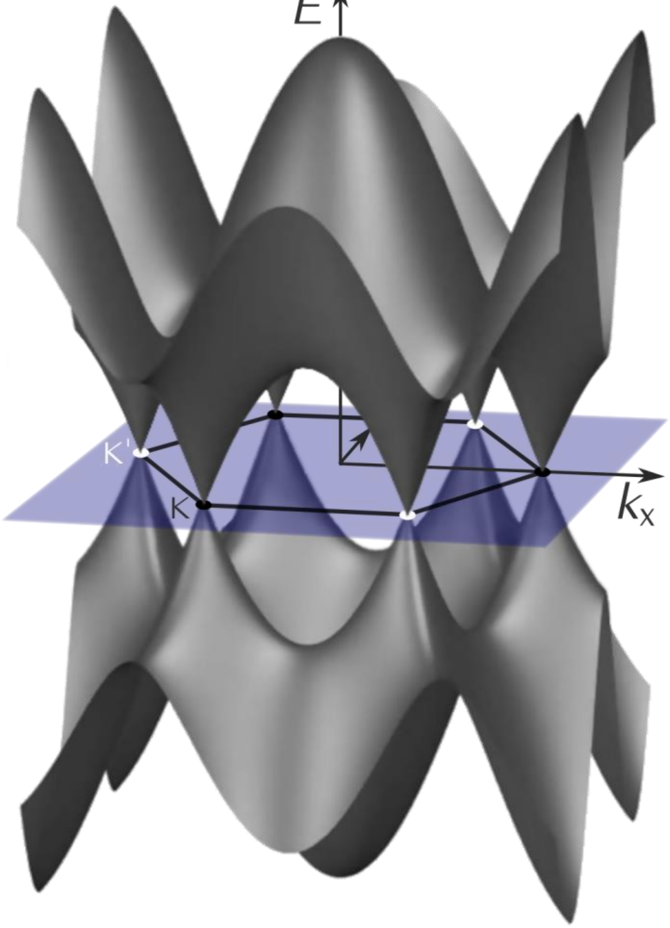


Graphene

- Graphene is an example of a semiconductor.
  - Due to its special honeycomb lattice arrangement and some quantum mechanics, we know that it has a band gap characteristic of a semiconductor



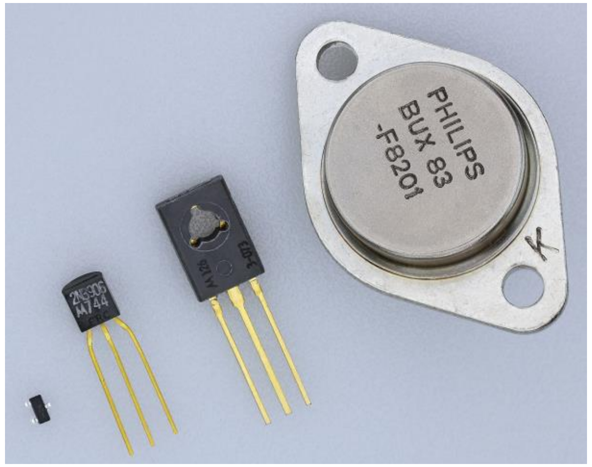
Graphene Honeycomb Lattice

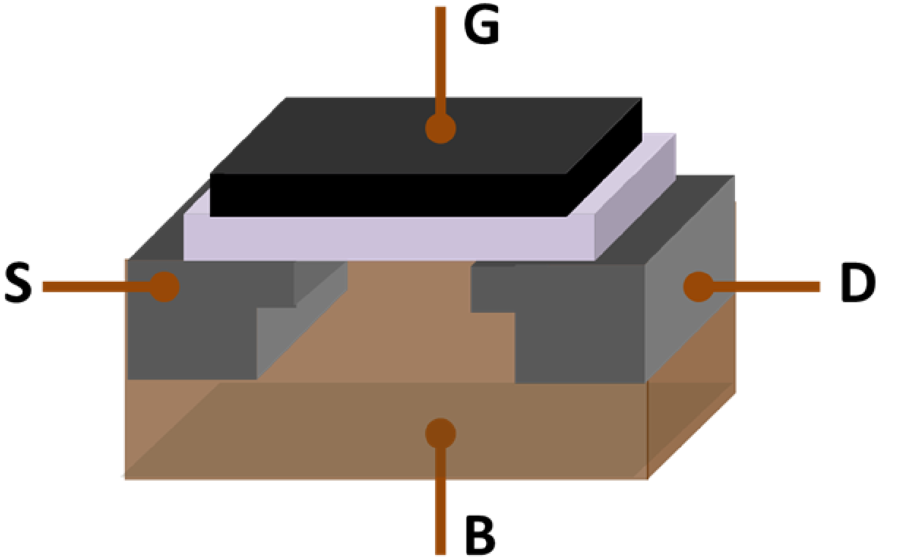


Electron band structure of graphene.

Transistors

- Computers are built out of transistors.
  - Transistors are made of a semiconductor material.
  - Because of this they can act as a switch.
  - Exist in a state of “on” and “off” represented as 1 and 0, respectively.
  - Each transistor is like a bit





Metal–oxide–semiconductor field-effect transistor (MOSFET), showing gate (G), body (B), source (S) and drain (D) terminals. The gate is separated from the body by an insulating layer (white).

Bits and Logic Gates

- Computers are built out of bits, and logic is performed on those bits.
- Because bits can represent digits in binary numbers, they may be used to represent any information that may be described by integer numbers.
- We will use vectors to represent bits.
  - A bit in the 0 state is represented by $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
  - A bit in the 1 state is represented by $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
  - The notation $|0\rangle$ and $|1\rangle$ will make more sense later, but for now just thick of the bar and bracket to mean whatever inside is a vector.

Logic Gates

- A logic gate is an operation on one or more bits.
- The XOR operation is an example of a logic gate on 2 bits.
- When bits are vectors, then logic gates are matrices, and the bit(s) go through the logic gate by doing matrix vector multiplication.

NOT Gate

- We expect the NOT gate to switch the bit from 1 to 0 or 0 to 1.
  - This is written mathematically as $\mathrm{NOT}|1\rangle = |0\rangle$ and $\mathrm{NOT}|0\rangle = |1\rangle$.

- The NOT gate as a matrix is $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

- Lets check to make sure it works.

  - $NOT|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \cdot 1 + 1 \cdot 0 \\ 1 \cdot 1 + 0 \cdot 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$

  - $NOT|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \cdot 0 + 1 \cdot 1 \\ 1 \cdot 0 + 1 \cdot 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$

Multiple Bits

- Certain logic gates operate on multiple bits, like the XOR gate or the AND gate.
- How do we represent a vector of multiple bits?
- Consider a first bit of 0 and a second bit of 1
  - With brackets we write this as $|0\rangle|1\rangle$
  - With vectors we write this as $\begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1 \cdot \begin{bmatrix}0\\1\end{bmatrix}\\0 \cdot \begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$
  - $\otimes$ is called the Kronecker product. It takes vectors of size $n \times 1$ and $m \times 1$ and gives a vector of size $m \cdot n \times 1$

Kronecker Product

- To compute the Kronecker product, for each element in the first vector, multiply the second vector by that number. “Stack” the resulting vectors on top of each other.
- Examples:
  - $\begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0 \cdot \begin{bmatrix}1\\0\end{bmatrix}\\ 1 \cdot \begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$
  - $\begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\begin{bmatrix}1\\0\end{bmatrix}\\ 1\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}0\\0\\1\\0\end{bmatrix}\\ 0\begin{bmatrix}0\\0\\1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}$

AND Gate

- The AND logic gate operates on two bits, and should return 1 if and only if both inputs are 1, and 0 otherwise.
  - $AND|1\rangle|1\rangle = |1\rangle, \quad AND|1\rangle|0\rangle = |0\rangle, \quad AND|0\rangle|1\rangle = |0\rangle, \quad AND|0\rangle|0\rangle = |0\rangle$
  - As a matrix AND is given by $\mathrm{AND} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

More AND Gate

- Let's see how the AND gate works in an example
- $AND|1\rangle|1\rangle = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} =$

  $\begin{bmatrix} 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 \\ 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$

XOR Gate

- We saw the XOR before. If you need a recap go to the Block Ciphers presentation.
- $XOR = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$

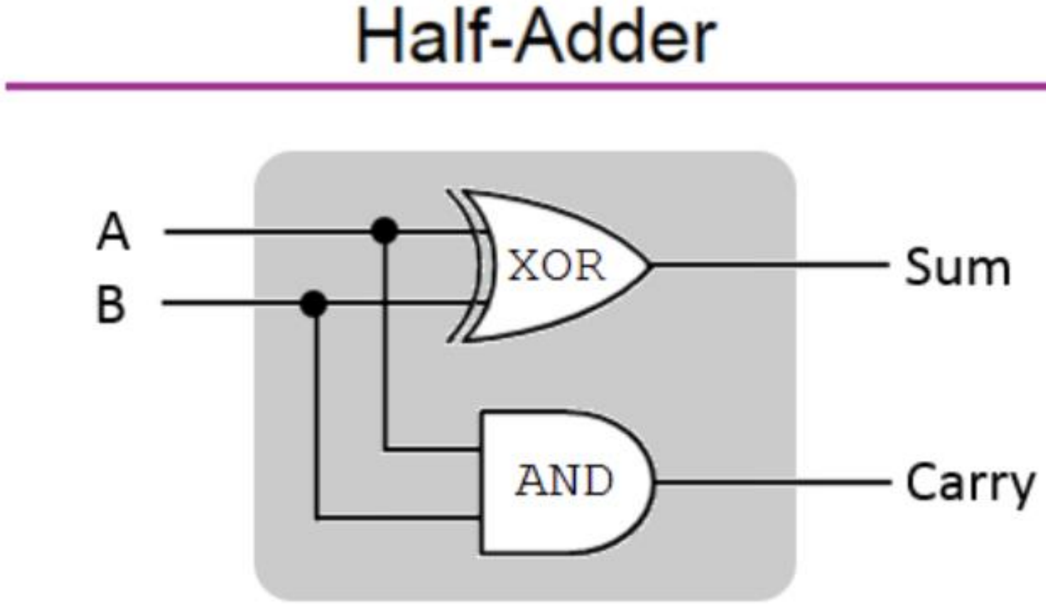
OR Gate

- The OR gate differs from the XOR gate because it should return 1 of both bits are 1.

- $OR = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$

Half-Adder

- This gate is built out of XOR and AND gates and helps perform addition.
- A and B are bits:

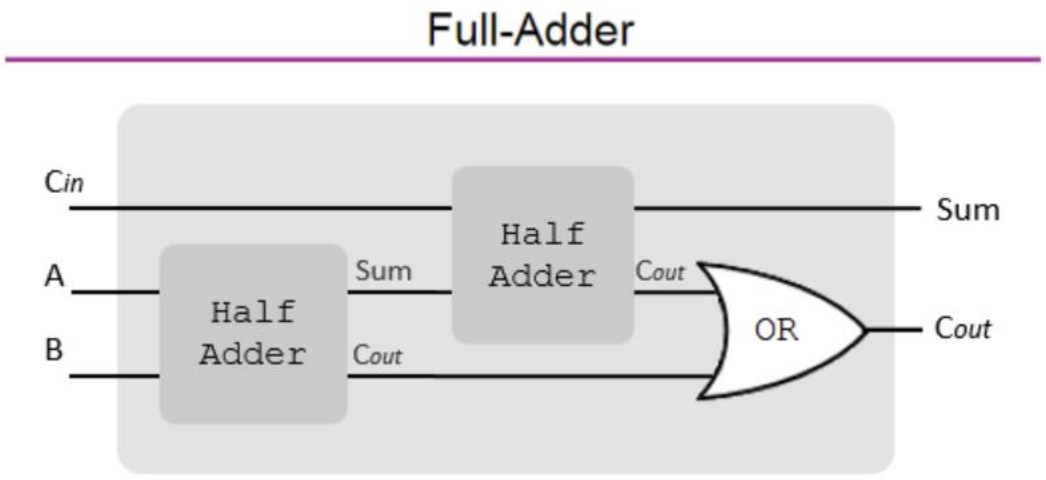


Half Adder

- When A and B are both 0 the half-Adder Returns 0 out of both gates,
- When Either is 1 and the other is 0, it returns 1 out of the sum and 0 out of the carry.
- When both are 1, the sum is 0 and the carry is 1.

Full Adder

- The full Adder circuit adds takes 3 bits, a previous carry (C_{in}) and 2 more bits to add, and gives the resulting sum and carry. It uses 2 Half Adder gates and an OR gate.



Binary Addition

- To perform binary addition of two n bit numbers, one may start with a half Adder on their rightmost digits, followed but $n - 1$ consecutive Full Adder's on the next bits and the previous carry.
  - You keep track of the returned sums, and the last carry, and that is the total summed number.

Binary Addition Example

num1		**1**	**0**	**1**	**0**	**1**
num2		**1**	**0**	**1**	**1**	**1**
sum		0	1	1	0	0
carry		1	0	1	1	1
Result	1	0	1	1	0	0

num1	**21**
Num2	**23**
Result	44 (correct)