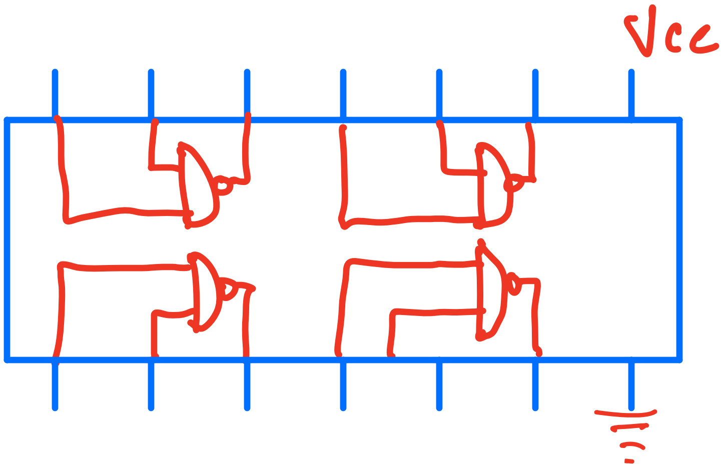
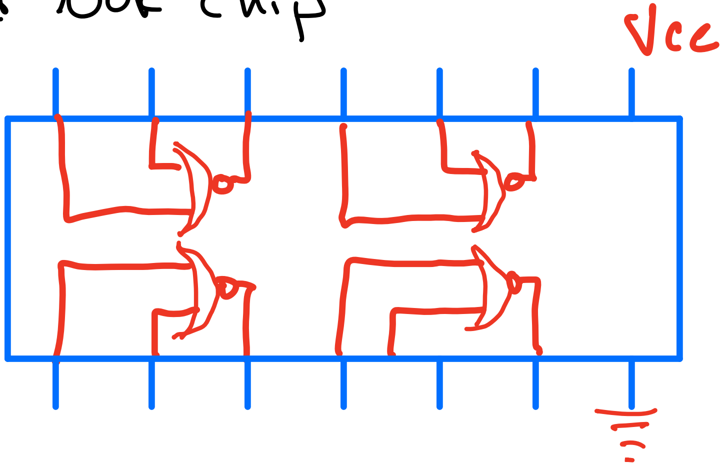
Implementation - build the circuit

1950s → 1980s, build integrated circuits

ex: quad NAND chip



quad NOR chip



process for building:

1. construct truth tables, use FOP → network of gates

2. simulate the logic, look at simution

wave form output

3. iterate until converges

4. build PC board w/traces to connect it all together

5. debug:

a) simulation needs to be exact to reproduce hardware

ex: • rise times of gate transistors
• delay time for signals to travel along traces

b) your ability to drive simulation with signals the same as for operations

garbage in ⇒ garbage out

Even after testing & debugging...

⇒ might still have errors

⇒ might want to make changes

At this point you have real HW boards

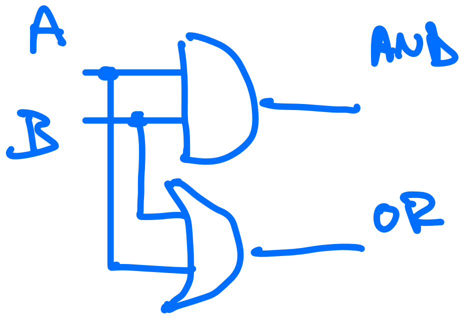
=> cut traces, solder "flying wires"

=> add more chips or use spare chip gates
more flying wires

There must be a better way!

=> Programmable logic

Start w/ truth tables for AND & OR

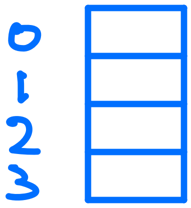


| A B | AND | OR |
|---|---|---|
| 0 0 | 0 | 0 |
| 0 1 | 0 | 1 |
| 1 0 | 0 | 1 |
| 1 1 | 1 | 1 |

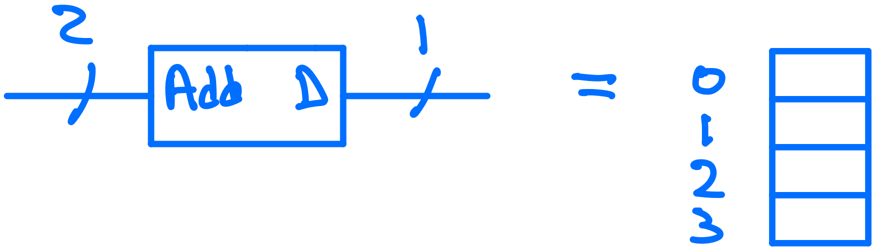
next construct small memory chip:

=> 4 locations

=> store 1 bit in each location



so need: 2 bits input for the address
: 1 bit output data



lets store bits like this: store 0's in 1st 3,
1 in last

=> to address, drive 2 address bits w/ A, B
=> output mimics result of AND gate

take same chip & store:

| | |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |

mimics OR gate!

so if you set up an array of these then you can change digital network structure by reprogramming memory

This memory is called Look-up table, or LUT

=> Basis for "programmable" logic

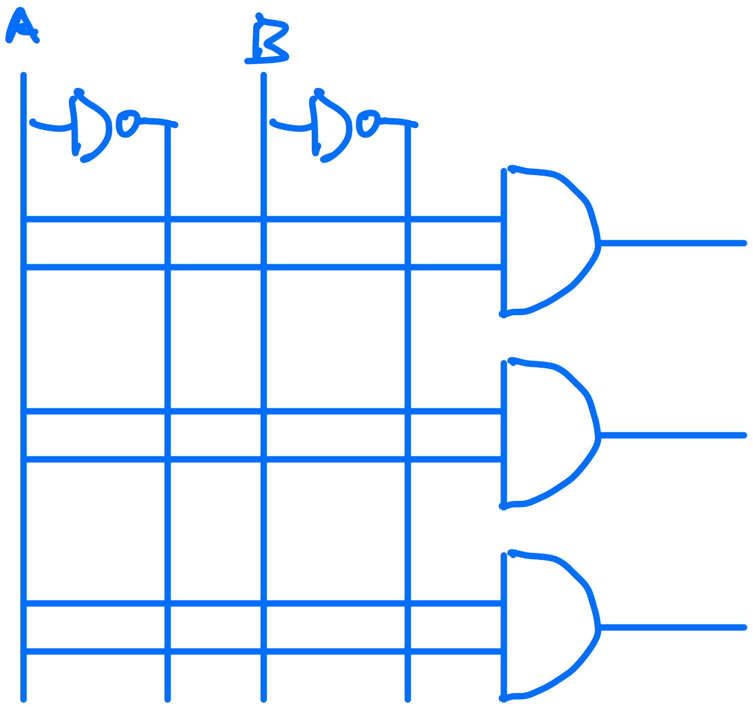
| A B | | | AND | | OR | | XOR | | NOT | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 0 | => | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 1 | => | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 0 | => | 2 | 0 | 2 | 1 | 2 | 1 | 2 | 0 | 2 | 1 |
| 1 1 | => | 3 | 1 | 3 | 1 | 3 | 0 | 3 | 0 | 3 | 0 |
| | | | | | | | | | $\bar{A}$ | | $\bar{B}$ |

"Logic array"

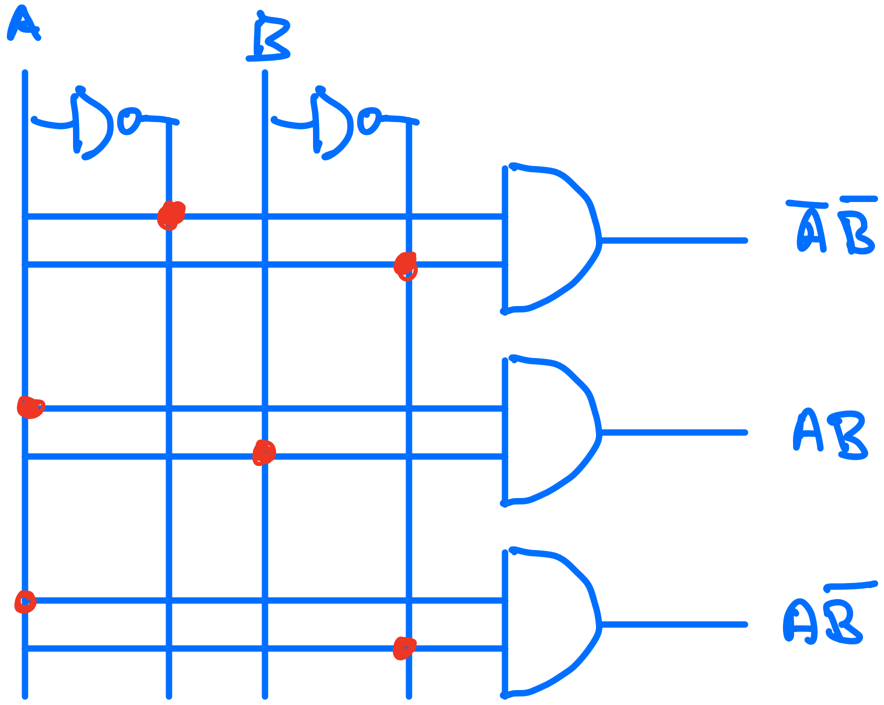
build circuit to perform $\bar{A}\bar{B}$, $AB$, $A\bar{B}$

need array of AND & array of OR

construct array of interconnects

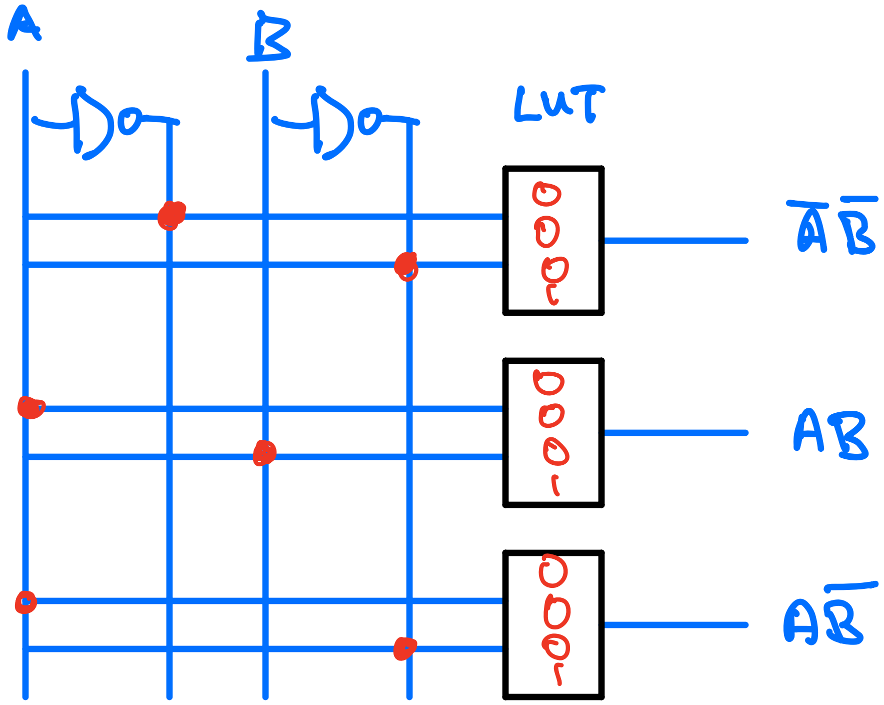


at each junction put FET
=> current into base decides if
connection is made



Turn on the right FET's to make 3 logic results

By using LUT's, device is fully programmable



or change LUT to be any AND, OR, XOR

then build up in larger structures

Now start using Vivado

1. show how to run & start project
2. go over synthesis, implementation, bitstream
3. show console, errors, etc.

4. show how to add source files

5. discuss constraints files (how to assign pins to wires)