University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Storing a Bit: the Gated D Latch

Everything So Far Has Been Combinational Logic

So far, we have talked only about **combinational logic**.

Combinational logic allows us to solve the following type of problem:

- given a set of bits as input,
- how can we combine them to produce other sets of bits (Boolean expressions)?

But where do the input bits come from?

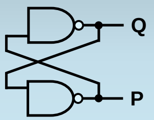
Now Let's Look at Sequential Logic

Today, we will start to look at **sequential logic**.

Sequential logic

- **stores bits as state**, and
- its **behavior depends on the state** (the values of the stored bits),
- just like the behavior of a C program can depend on the current values of variables.

A Dual "Inverter" Loop Serves a Specific Purpose



What is a 1-input NAND gate?

An inverter / NOT.

Remember the gate structures?

What does the circuit here do?

It has no inputs!

How can we analyze it?

Start Solving by Picking a Value for Some Variable

First, **write a truth table.**

Then **pick a value**.

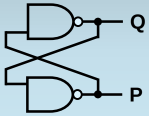
Say **Q = 0**.

Which implies what about P?

P = 1.

Which implies what about Q?

Q = 0 (be sure to check!).



Q	P
0	1

Trace Logic Values to Find Stable States

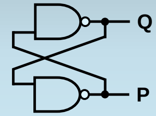
We say that this state (**Q = 0**, and **P = 1**, as shown in the truth table) is **stable** because the values do not continue to change forever.

What if we instead pick **Q = 1**?

In that case, what is P?

And what does P = 0 imply for Q?

Again, be sure to check stability.



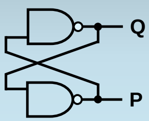
Q	P
0	1
1	0

The Dual-Inverter Loop Stores One Bit

We say that this circuit is **bistable** because it has **two stable states** (bi- = two).

Bits on a chip are typically stored using this kind of dual-inverter loop.

But … **how do we set a value?**



Q	P
0	1
1	0

Use an Extra Input to Set the Bit Q

Let's add an input.

We will call it **S'**.

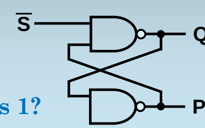
What happens when S' is 1?

The new input has no effect! (green is the previous truth table)

What if S' = 0?

Q = 1! And **P = 0**.

So **S' Sets the bit Q to 1**.



S'	Q	P
0	1	0
1	0	1
1	1	0

Active Low Inputs are Named with “Bar” (NOT)

Why did we call the input **S’**?

(Call it “S bar,” by the way.)

The action induced by **S’**,
- to **S(et) the bit Q**,
- occurs when **S’ = 0** (S’ is low).

We say that the input S’ is **active low**.

And we name it **S’** instead of **S** to indicate how the input should be used.

What About Resetting Q to 0?

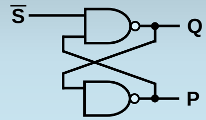
So we can set **Q = 1**.

But … what if we want **Q = 0**?

Keep flipping the power on and off until we get lucky?

(Maybe not.)

Any ideas?



S’	Q	P
0	1	0
1	1	0
1	0	1

Use an Extra Input to Reset the Bit Q

Another input? Sure.

We will call it **R’**.

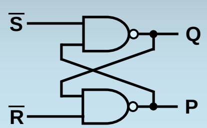
What happens when R’ is 1?

The new input has no effect! (green is the previous table)

What if R’ = 0 and S’ = 1?

P = 1! And **Q = 0**.

So **R’ Resets the bit Q to 0**.



R’	S’	Q	P
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0

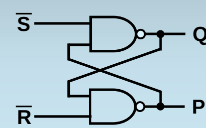
An R’-S’ Latch Consists of Two NAND Gates

This circuit has a name!

It’s an **R’-S’ latch** (“R bar, S bar latch”).

Store a **1 bit** by lowering **S’** to 0.

Store a **0 bit** by lowering **R’** to 0’.



R’	S’	Q	P
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Avoid Setting Both S' and R' to 0 Simultaneously

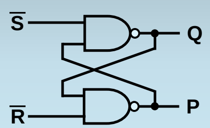
What if we set both **S'** and **R'** to 0 at the same time?

Q = 1 and **P = 1**.

But when we raise the inputs, we may leave the stored bit in either state.

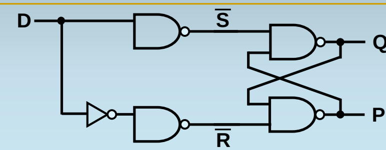
Or, worse, the loop may not settle into digital voltages (**metastable**).

Do NOT lower both at once!



R'	S'	Q	P
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Extra Gates Prevent the Forbidden Input Combination



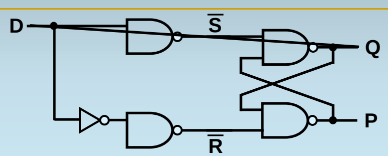
We can add a couple more NAND gates to prevent setting both **S'** and **R'** to 0.

Let's check the truth table.

So **Q** stores **D**…

D	R'	S'	Q	P
0	0	1	0	1
1	1	0	1	0

We Can Simplify the Design to Copy D to Q

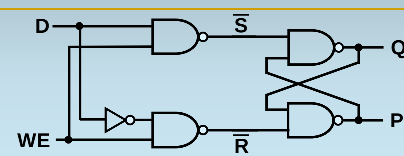


There's an easier way to implement such a circuit, though…

So **Q** stores **D**…

D	R'	S'	Q	P
0	0	1	0	1
1	1	0	1	0

Extra Inputs Control Copying of D to Q



Let's add more inputs to the new gates, too.

Now our design only copies **D** to **Q** when **WE = 1**.

WE	D	R'	S'	Q	P
1	0	0	1	0	1
1	1	1	0	1	0

The Circuit Can Also Store a Bit

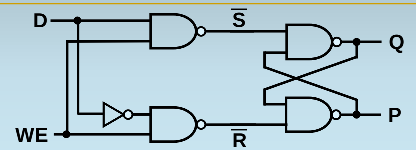
What happens when WE = 0?

The circuit stores the last bit from **D** (same truth table as before!).

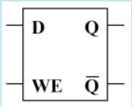
This circuit is called a **gated D latch**.

WE	D	R'	S'	Q	P
1	0	0	1	0	1
1	1	1	0	1	0
0	x	1	1	0	1
0	x	1	1	1	0

This Design is Called a Gated D Latch



Symbolically, a **gated D latch** is drawn as shown here.



Notice that **P** has been replaced by **Q'**, since they are always complements of one another.