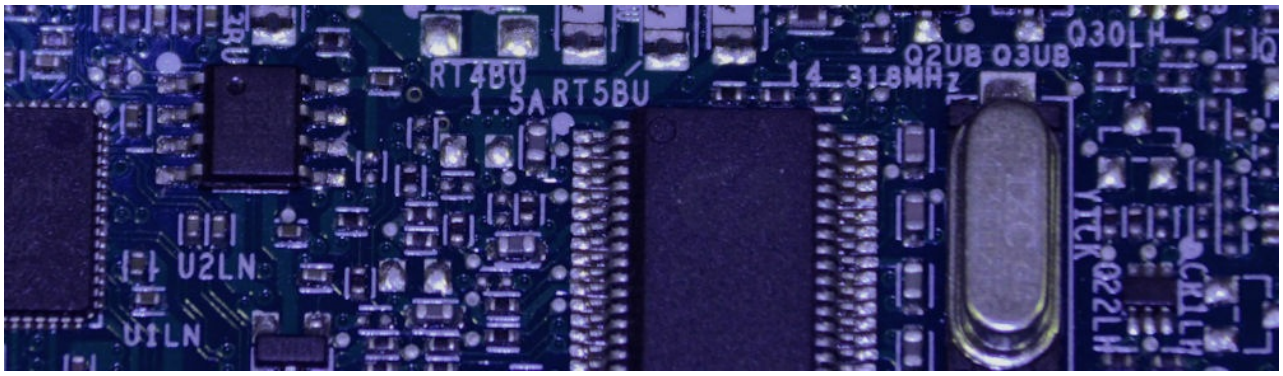
H-Bridge Drivers

modularcircuits.com/blog/articles/h-bridge-secrets/h-bridge_drivers/



Introduction

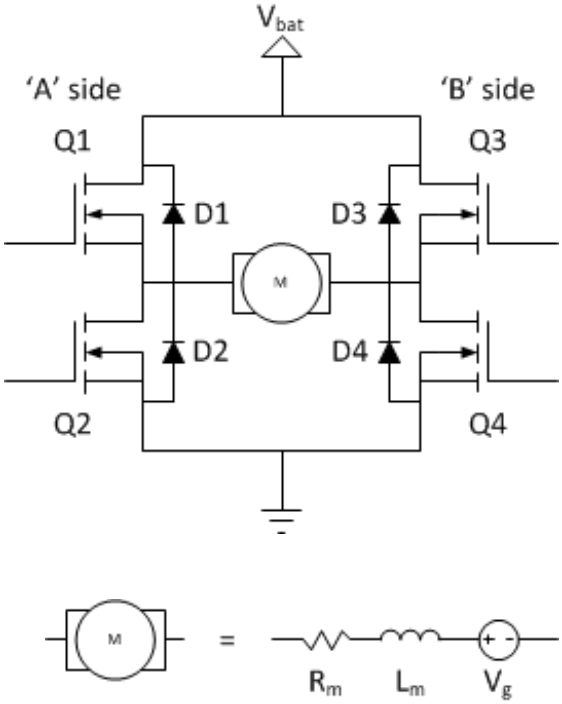
In the previous installment of the series we've gone through the high-level design decisions that you have to make when designing an H-Bridge, and we've discussed the considerations for selecting the MOSFETs and the catch diodes that will make up the bridge.

In this article I will go through the available options for drive circuits. We will discuss the trade-offs between them and what influences the various parameters of the drive circuits.

You will take the most out of this write-up if you are already fairly familiar with H-Bridge basics, so if you aren't, I suggest you read the introductory piece of the series first. Understanding of the various drive-modes will also be useful, so reading the sign-magnitude drive, the lock anti-phase drive and the asynchronous sign-magnitude drive articles isn't a waste of time either, though those pieces go into quite a bit of more detail than what is needed to follow this text.

To make referencing easier, let's review the H-Bridge circuit:

and our motor model:



Drive circuitry

The drive circuitry for an H-Bridge is basically the electronics that sits between the PWM (and potentially other) digital control inputs and the MOSFET gates. It has two major purposes:

- Translate the input voltages to suitable levels to drive the gates
- Provide enough current to charge and discharge the gates fast enough

On top of that, many drive circuits include additional functionality:

- Translate the input command into gate-drive signals according to the drive mode
- Provide shoot-through protection
- Generate voltages for the high-side gate-drive circuitry (for N-channel drivers)
- Provide additional safety functions, like over-current protection
- Control the turn-on and turn-off times of the FETs

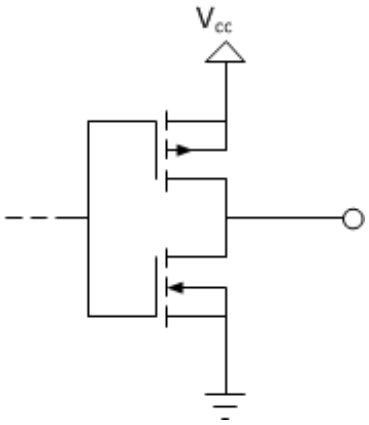
Drive circuits can come in many shape or form.

- There are low-side drivers, that are designed to drive Q2 or Q4 on our bridge.
- High-side drivers in turn are designed to drive Q1 or Q3.
- Half-bridge drivers combine one low- and one high-side driver, so they can drive Q1 and Q2 (or Q3 and Q4) together.
- Full-bridge drivers obviously have two low-side and two high-side drivers so they can drive all four FETs.

As we've discussed in the previous article, low-side MOSFETs are always N-channel ones, while on the high-side we can you either P-channel or N-channel devices. This means that when we discuss high-side drivers (or half- and full-bridge drivers) we have to create

two sub-categories, one for each channel-type.

Before we dive into the details, let's get familiar with the mother of all driver circuits, the complementer CMOS driver:



In this circuit, a high-side PMOS and a low-side CMOS FET are combined to provide a clean digital logic output: if the input (the gates of the FETs) is grounded, the low-side FET is off, while the high-side is on. The output is connected to V_{cc} through the relatively low r_{rdson_high} of the high-side element. When the input is connected to V_{cc}, the opposite happens, and the low-side FET starts conducting, while the high-side FET is off, so the output is connected to ground through a similarly low r_{dson_low}. This topology is fairly common amongst not just bridge drivers, but logic gates and in general digital logic. We will use it as our starting example, and expand to more complicated circuits as we discuss the problems that come up.

In case you were wondering how is this driver stage different from one side of an H-bridge:

- The FETs are much smaller, so their gate capacitance is really small. Even a relatively weak source can quickly charge and dis-charge them.
- These smaller FETs also have a much higher r_{dson} value (several ohms) so the dynamic shoot-through currents are low enough not be a headache.

This drive stage will be the building block for all of our low- and high-side drivers, but with some modifications on occasion. I'll start with low-side drivers and discuss the problems you will face with them. Some of the discussion will be applicable (with slight changes) to high-side drivers. After those topics are cleared, we can continue on to high-side drivers.

As I have said, this drive configuration is the same that is used in CMOS digital logic. (TTL and some other logic technologies are significantly different!) Because of this, as a concrete example, I'll use the output stage of the AHC logic series to discuss the features and characteristics of this type of driver.

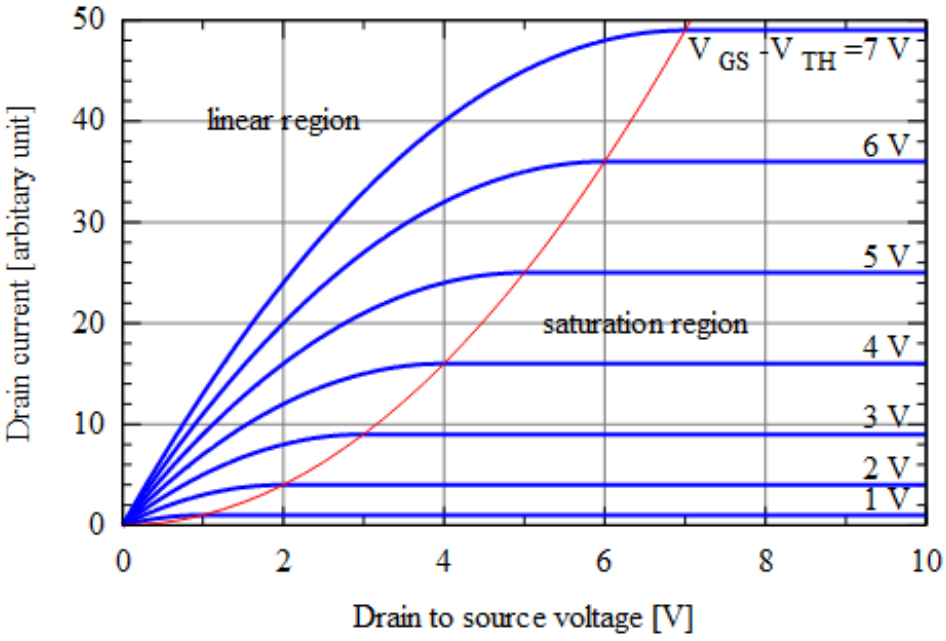
Calculating turn-on and off times

As we've discussed it in the previous part of the series, the gate capacitance of the MOSFETs together with the available drive current from the drive circuit will determine how fast the transistor can be turned on or off. Let's investigate that topic a little more in detail!

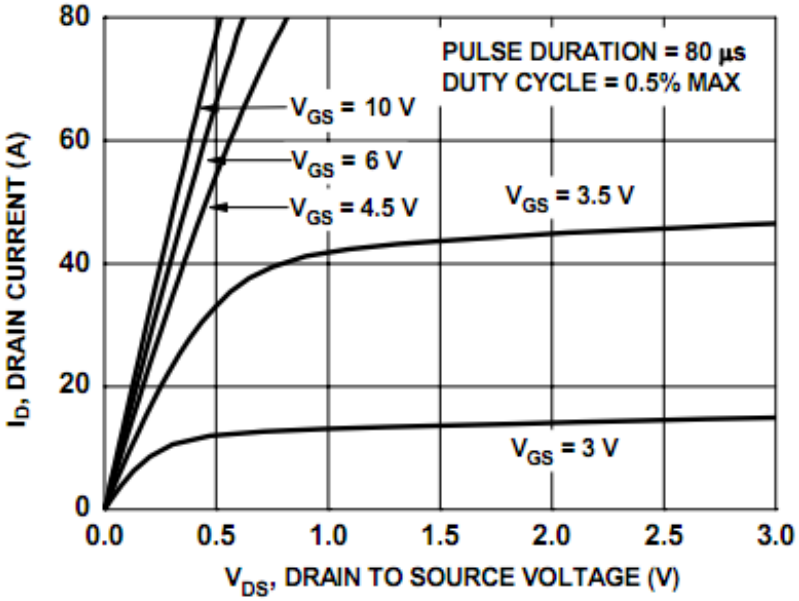
In the following I will only concentrate on a low-side, N-channel device and it's driver. You can easily convert the results to a high-side drive situation for both the N- and P-channel cases.

FET datasheets specify the gate capacitance, sometime called the input capacitance. The capacitance varies a lot depending on the size of the device. For example, this relatively large MOSFET (the PHK31NQ03LT from NXP) has an almost 5nF gate capacitance. At the same time this transistor has a less than 5mΩ on-resistance. Another example would be IXTY 01N80. This 50mΩ on-resistance transistor only has a roughly 60pF gate capacitance. The FETs used in my Servo Brain µModule project have a roughly 100mΩ r_{dson} and a roughly 350pF gate capacitance.

To calculate the turn-on and off times we need to know a couple of things: how high the gate-voltage needs to be and how fast the drive circuitry can charge and discharge the gate-capacitance. Let's address these questions in order: the minimum gate-voltage that is needed to turn the FET fully on is specified in the datasheet, but also depends on the drain current. It is usually specified in the form of a chart like this (this is from Wikipedia):



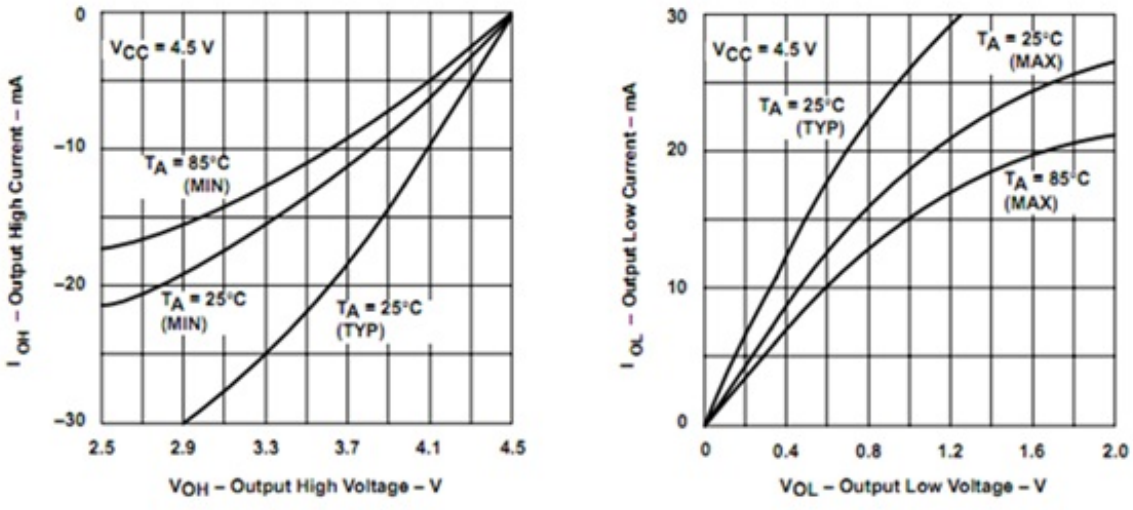
As you can see, for relatively low drain-currents (Y axis) the FET operates as a small resistance (the curve is linear and goes through the origin). For high currents however, the FET transitions into so-called saturation where the current is pretty much constant. For our application, we want to keep the FET in it's linear (resistor-like) region. So, if you know the maximum current the FET (which is the current limit of the bridge), you can figure out the minimum gate-voltage that is needed to keep the FET linear. To take a practical example, let's use the FDMS8880 FET and lets assume we want to build a bridge with a 20A current limit. The same diagram for this particular FET looks like this



You see that if the gate voltage (V_{gs}) is only three volts, the FET would not even be able to conduct 20A. It saturates at around 15A. With a gate voltage of 3.5V, you can get the device back into it's linear region for 20A current, but it's resistance is still a bit high. If you however increase the gate voltage to about 4.5V, you'll see that the resistance at 20A does not depend on the gate voltage too much any more. So, for our case, we would need a gate voltage of at least 4.5V.

In order to turn the same FET off, we need to lower the gate voltage below the so-called threshold voltage. This is again something that's specified in the datasheet, for this particular device, it's 1.2V (minimum).

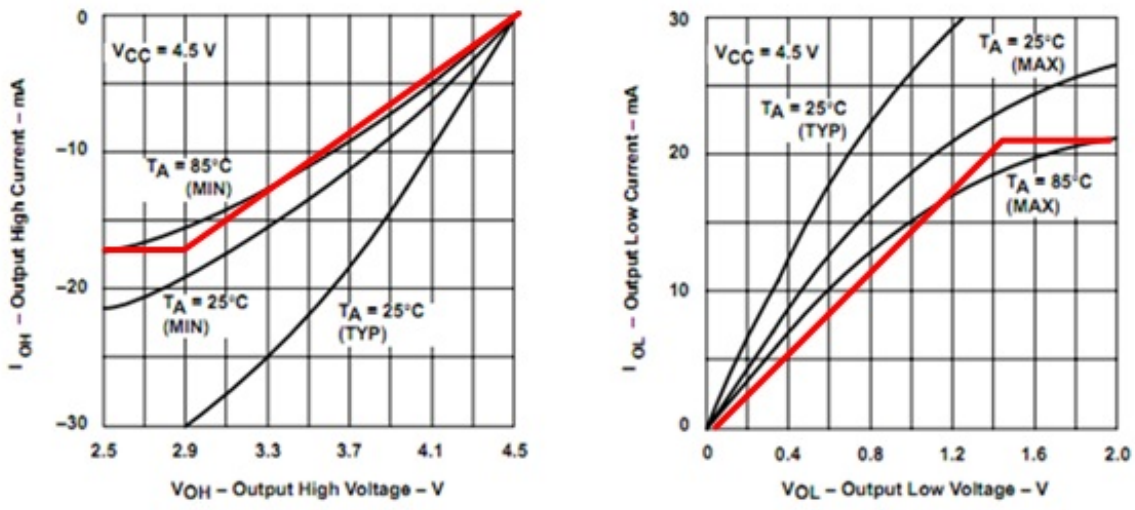
Now, on to the second question: how fast can the driver charge or dis-charge the gate of the MOSFET. Driver characteristics are usually quite complex, and they are specified using charts, like this:



Here you see how the output current changes as the function of the output voltage, or the more useful way of looking at it: if you want to draw a certain amount of current out

of the pin, how much the output voltage will deviate from its ideal value. This particular chart is from TI and specifies their AHC-series logic output characteristics (page 16). Notice how these charts are fairly similar to the charts above for MOSFET characteristics. This is not a coincidence, as the AHC series uses our complementer FET output drive circuit.

To approximate these curves, we can use a very simple model (again, using the MOSFET terminology): The output is in saturation mode for high currents – effectively acting as a current-source – and as the current decreases, it transitions into a linear region where it acts as a resistor. Graphically, we approximate the curve with two lines:



You can easily see that for the case of driving high voltages, the current source is at around 17mA, and the resistance is around 100Ω. When the output drives low, it can output 21mA and has roughly 70Ω resistance. (It is typical that an output stage has a somewhat weaker high-side driver, being a P-MOS device.)

In many cases you can even further simplify the picture and assume either only the current-source or the linear region. We will go though all three approaches, using an example: as we've seen before, the FDMS8880 MOSFET can be turned on completely by a 4.5 or greater gate voltage. So, in theory it can be driven directly by a 5V digital pin. Let's say we use the previously studied AHC-series to drive the gate of this transistor.

Constant current drivers

The constant current approach works the following way: we try to charge up a capacitor with a constant current source to (at least) a certain voltage. That will take some time:

$t_{on} = V_{gate}*C_{gate}/I_{source}$, where V_{gate} is the gate voltage we need to turn the FET completely on, C_{gate} is the gate capacitance, and I_{source} is the current the driver can source.

The off-time could be calculated like this:

$t_{off} = (V_{gate}-V_{th})*C_{gate}/I_{sink}$, where V_{th} is the threshold voltage.

Substituting the numbers for out example we get the following:

t_{on} = 4.5V*1585pF/17mA = 419ns

t_{off} = (4.5V-1.2V)*1585pF/21mA = 249ns

Constant resistor drivers

The constant resistor approximation is more complex, because of the exponential response of the RC circuit. The turn-on time will be the following:

$t_{on} = -R_{source}*C_{gate} * \ln(1-V_{gate}/V_{source})$, where R_{source} is the source resistance and V_{source} is the high-level no-load voltage of the driver.

Similarly the off-time can be calculated as follows:

$t_{off} = -R_{sink}*C_{gate} * \ln(V_{th}/V_{source})$

To be able to use this model for anything meaningful, we will have to assume that V_{gate} is lower than V_{source}, in other words, the FET is fully on with a gate voltage that's less the drivers no-load output voltage.

Doing the calculations for our example we get:

t_{on} = –100Ω*1585pF * ln(1-4.5V/5V) = 364ns

t_{off} = –70Ω*1585pF * ln(1.2V/5V) = 158ns

You can see that there's quite a difference between the two estimations. Both are actually under-estimating the times: the constant current approach will assume more current than the driver can actually deliver at low output voltage drops, while the constant resistance approach (at least the way I did it here) over-estimates the current for the saturated region.

Piece-wise linear model

A more precise estimate can be made by combining the two methods, and assume constant current charge and discharge until the knee-point (2.9V and 1.4V respectively for our example) and assume constant resistance only for the remaining portion:

$t_{on} = V_{knee\_on}*C_{gate}/I_{source} - R_{source}*C_{gate} * \ln(1-(V_{gate}-V_{knee\_on})/(V_{source}-V_{knee\_on}))$

$t_{off} = (V_{source}-V_{knee\_off})*C_{gate}/I_{sink} - R_{sink}*C_{gate} * \ln(V_{th}/V_{knee})$

With this approach, we get:

t_{on} = 497ns

t_{off} = 286ns

These calculations can be done for P-channel MOSFETs and drivers as well, but of course you have to slightly change the equations to accommodate for the negative gate-source voltage of those devices.

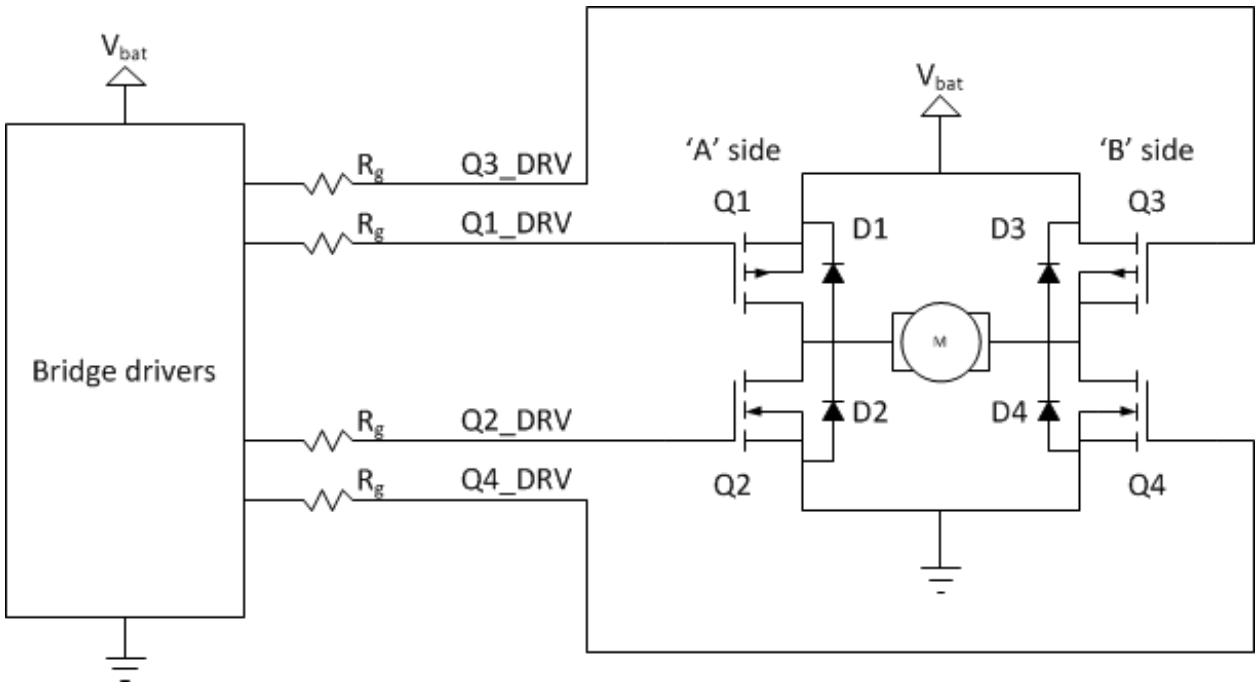
Controlling turn-on and off times

So far so good, we have several ways now to calculate the transient times, with various accuracy. But what if you're not satisfied with the results? What tools do you have to influence these numbers?

If you want to make the time shorter, your pretty much have two choices: either change the FET to one with a lower gate capacitance or you change the driver to one that can provide more current.

If you want to make the time longer, you have more options. One thing you can do is to add more capacitance to the gate by adding an extra capacitor towards ground for example.

By far the most common way of controlling the trun-on and –off times is to add a series resistor to the driver outputs:



The series resistor method is ineffective if the driver is truly a current source, but that very rarely is the case. Normally, the effect is two-fold: one is that (by requiring more voltage for the same current) it gets the driver faster out of its current source region into it linear region. The second affect is that, once we're in the linear region, the effective source resistance of the driver will be higher, so the time-constant of the charge-up or down of the capacitor will be larger.

Let's see how to come up with the value of the resistor! If your model for the driver is a constant resistor one, the calculations are very simple: you simply express the required resistance from the on-time or off-time equations (here I'll use the on-time one):

$t_{on} = -(R_{source}+R_g)*C_{gate} * \ln(1-V_{gate}/V_{source})$

so the needed series resistance is:

$R_g = -t_{on}/(C_{gate} * \ln(1-V_{gate}/V_{source}))-R_{source}$

Note that if you do the same calculations for t_{off}, you usually get a different R_g value. Since you need to select a single value, it means that you can't independently control the on- and off-times.

If you used the combined current-source/linear model, you have a bit harder time, because first you have to figure out how long does the driver stay in current-source mode. The switch-over happens when the drive voltage is at it's knee point – 1.4V or 2.9V in our case. However at that point the series resistor drop $V_{Rg} = R_g*I_{source}$ voltage and only the rest is on the capacitor. So the time it takes to get out of the current-source region is this:

$t_{on\_current\_source} = (V_{knee\_on} - V_{Rg})*C_{gate}/I_{source}$

putting the above value in for V_{Rg} and doing some simplifications, we get:

$t_{on\_current\_source} = (V_{knee\_on}/I_{source} - R_g)*C_{gate}$

Similarly the time it takes to get out of the current-sink region for the turn-off time is:

$t_{off\_current\_sink} = (V_{source}/I_{sink}-V_{knee\_off}/I_{sink}-R_g)*C_{gate}$

After that time, the driver is in its linear (constant resistance) mode, so the previous equations can be used. The total turn-on and –off times for this approximation are the following:

$t_{on} = (V_{knee\_on}/I_{source} - R_g)*C_{gate} - (R_{source} + R_g)*C_{gate} * \ln(1-(V_{gate}-V_{knee\_on})/(V_{source}-V_{knee\_on}))$

$t_{off} = (V_{source}/I_{sink}-V_{knee\_off}/I_{sink}-R_g)*C_{gate} - (R_{sink} + R_g)*C_{gate} * \ln(V_{th}/V_{knee})$

But what is the right turn-on or -off time after all?

After all this math you might ask this question. The problem is that there isn't a clear-cut answer. The reasons you might want to lower the transients are the following:

- Reduced heat dissipation on the MOSFETs
- More precise PWM control of the motor (the bridge spends less time in the not-so-well-defined transient states)

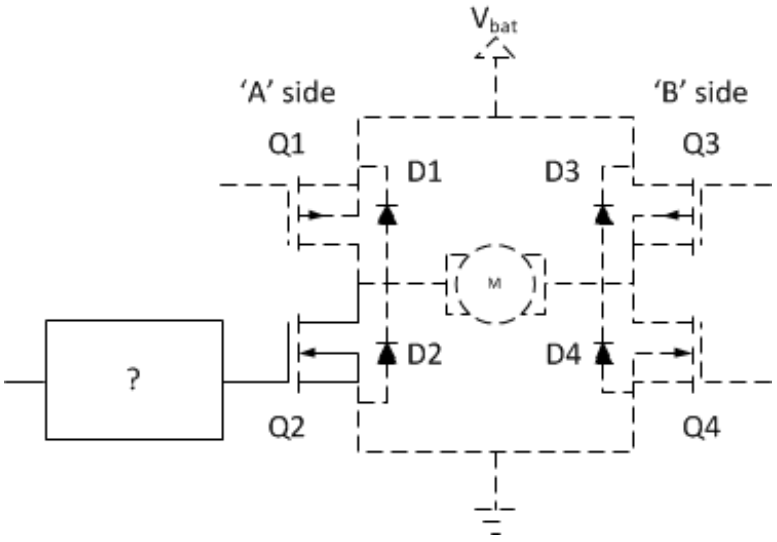
At the same time there are reasons to make the transients longer as well:

- The faster the transient is, the faster the catch diodes need to be
- Fast transients generate a lot of EMI noise
- Fast transients need high(er) current drivers

All in all, H-bridges are not the most demanding circuits as far as transient times are concerned: you've seen that a single AHC-series gate can pretty comfortably create sub-microsecond turn-on and –off times for a rather large FET. My guideline is that I try to keep the transient times to around 0.5-1% of the cycle time. This means that for a 20kHz bridge, I like to see 250-500ns transient times. This is a much more serious problem for higher switching frequencies that are normally found in high-power DC/DC converters, like PC motherboards.

Low-side drivers

In the following I will only deal with one-half of the bridge. The second half needs identical treatment, so I'm going to ignore that for a while. On the low-side, we only have one type of device to deal with: N-channel FETs. These need a low voltage to turn them off, and a higher voltage (typically in the 5...15V range) to turn them fully on. The question is: what to put in the place of the mystery circuit:



The simplest drive circuitry: none

For very simple, low-voltage designs, they might be completely missing, and the FETs are directly driven by logic level signals.

This technique however only works under limited circumstances:

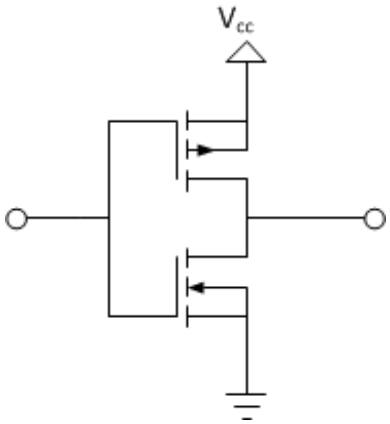
- You have to make sure that the output voltages of the digital logic are in fact capable of turning the FET fully on. This was true in out previous example, but if you've used lower-voltage logic (3.3V or even lower) or wanted to have higher currents through the bridge, it would not have been the case

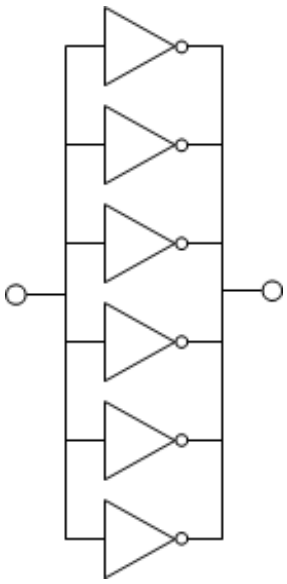
- The logic output has enough current to generate the required turn-on and –off times. This is especially important for logic output with asymmetrical drive capabilities, like open-drain CMOS outputs or TTL chips.

Higher current drivers

If for the above reasons, you want to have a driver that can provide more current (but you're still fine with the limited output voltage range), you can still use the complementer driver configuration, just use larger FETs, with lower r_{dson}.

Alternatively, you can gang together several output buffers from standard CMOS devices to increase the drive capability that way, for example by connecting all six of the available inverters in an 74AHC04 together:



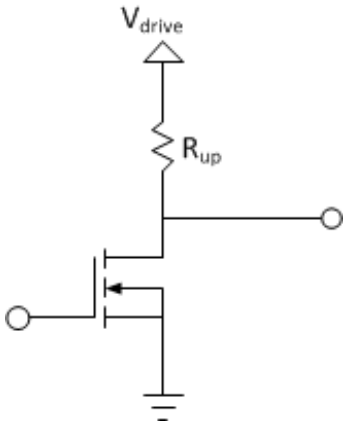


High-voltage complications

So far we've only talked about low-side drivers where the voltage level of the gate-drive needed was within the range of standard digital logic. We've seen that – at least in that one example – a standard 5V digital logic gate works reasonably well for closing a relatively large MOSFET. As lower voltage (3.3V and below) digital standards gain popularity or if you try to increase the current capability of the bridge, you'll find pretty quickly that direct logic level drive is inadequate.

When the gate-driver voltage of the FET is higher than your digital supply, at least a level-shifter will be needed in order to be able to drive the device. One of the simplest level shifters is this:

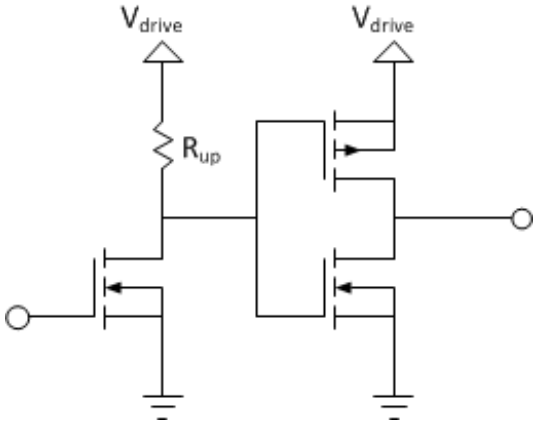
Here, the gate of the small-signal N-FET is driven by a suitable logic signal (and a logic level signal can easily turn this N-FET on) and the drain of it is pulled up to the gate-drive power supply, V_{drive}. When the FET is off (the gate is driven to logic '0'), the output will be pulled to V_{drive} by R_{up}. When the gate is driven to logic level '1', the FET turns on, and pulls the output to 0V. So in fact, the output is the logical inverse of the input, but the voltage levels are changed to 0 and V_{drive}.



The complication however is this: the drive strength (or current-delivery capability) of this level-shifter is significantly different in the 'high' and the 'low' case. When it drives low, its output resistance is pretty much r_{dson} of the FET. When the output is high, it's resistance is R_{up}. However R_{up} must be significantly higher than r_{dson} otherwise the low-level output voltage would not be close to 0V. This in turn means that the turn-on time (which is determined by r_{dson} for a P-FET) will be significantly – maybe even an order of magnitude – lower than the turn-off time, which is determined by R_{up}. This imbalance complicates shoot-through protection quite a bit and makes it very hard to turn the driven power FET off fast enough.

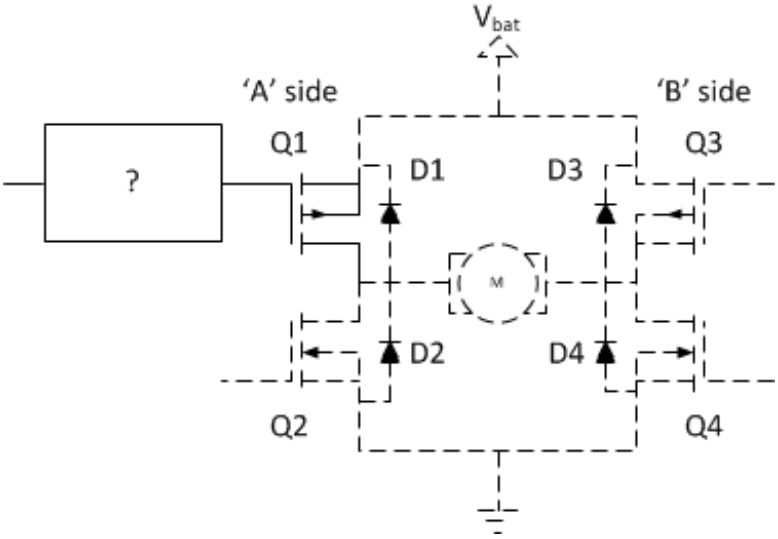
To overcome this problem, an complementer driver stage can be added between the level shifter and the power-FET:

This stage will make both the high- and low-level drive strength roughly equal, consequently making the turn-on and –off times much closer to one another.



High-side P-MOS drivers

So far we've only talked about driving N-MOS devices and driving them on the low-side. Let's consider now the high-side drivers, first for P-MOS devices:



This configuration present some complications: P-MOS transistors are open (non-conducting) when their gate is at close to the same potential as their source, and closed (conducting) when the gate is at a significantly lower potential, -5…-15V lower. This means that in order to completely turn off a high-side P-FET we'll have to drive it's gate as high as it's source, which is connected to the power supply. To turn the FET on, we have to lower the gate voltage by 5…15V below V_{bat}.

All the drive circuits we've discussed before can be used for high-side P-FETs with the following change: you have to power the driver stage from the same voltage as the bridge is operating on, that is V_{bat}. That way, the high-level output voltage will be V_{bat}, which will turn the P-FET off properly, and the low-level output voltage will be 0, that is almost always enough to turn the FET on. (You might have problems with extremely low V_{bat} voltages, where you would have to drive the gate to a negative voltage to turn the FET on properly. This however is a rare enough case to ignore simply because high-current H-bridges usually operate at higher voltages and low-voltage H-bridges have low-enough currents that a small logic-level FET can be used in them that can be turned on by $-V_{bat}$.)

Direct logic drive complications

The additional limitation of the driver operating from V_{bat} has a significant consequence for direct logic-gate driven bridges: a simple AHC series gate, like the one we've used before will only be able to function in this role if the bridge power supply is lower than the maximum supply the gate itself can be operated from, that is, less than 5V. This is a very serious limitation as most bridges but the smallest ones operate from higher voltages to maximize power delivery without requiring enormous currents.

With all that, for small motors this approach can result in a good, cheap solution. For a practical example, take a look at the Servo Brain µModule project.

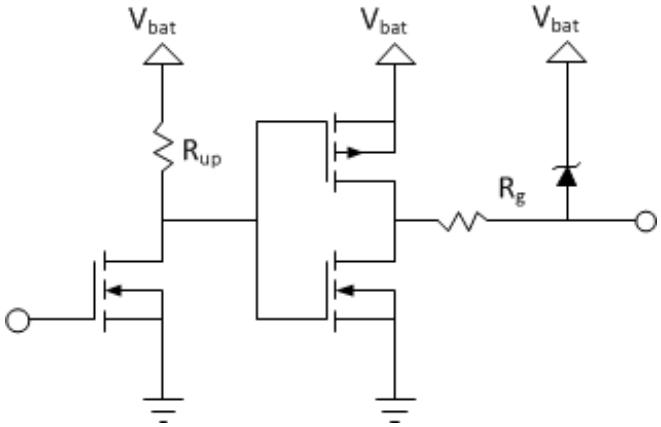
High-voltage drive complications

One of the major contributors to premature MOSFET deaths is gate-oxide break-down. The major cause of the breakdown is too high gate-source voltage on a MOSFET. The datasheets always specify this value, and for power MOSFETs at least the value is usually +/-20V. Getting out of this region will very quickly destroy the FET.

This presents a problem for the high-side P-MOS drivers: if V_{bat} is higher than 20V, we can't allow the gate drive to go down to 0V any more for low levels. It can go only as low as – say – V_{bat}-15V to allow for some safety margin as well. This is usually accomplished by adding a Zener diode to the drive circuit:

If you set the Zener voltage to about 15V, it will limit your voltage difference between the output and V_{bat} to be within the safe limits.

Similar limitations are needed on the low-side as well, if your driver works from V_{bat} and not from a separate power supply.

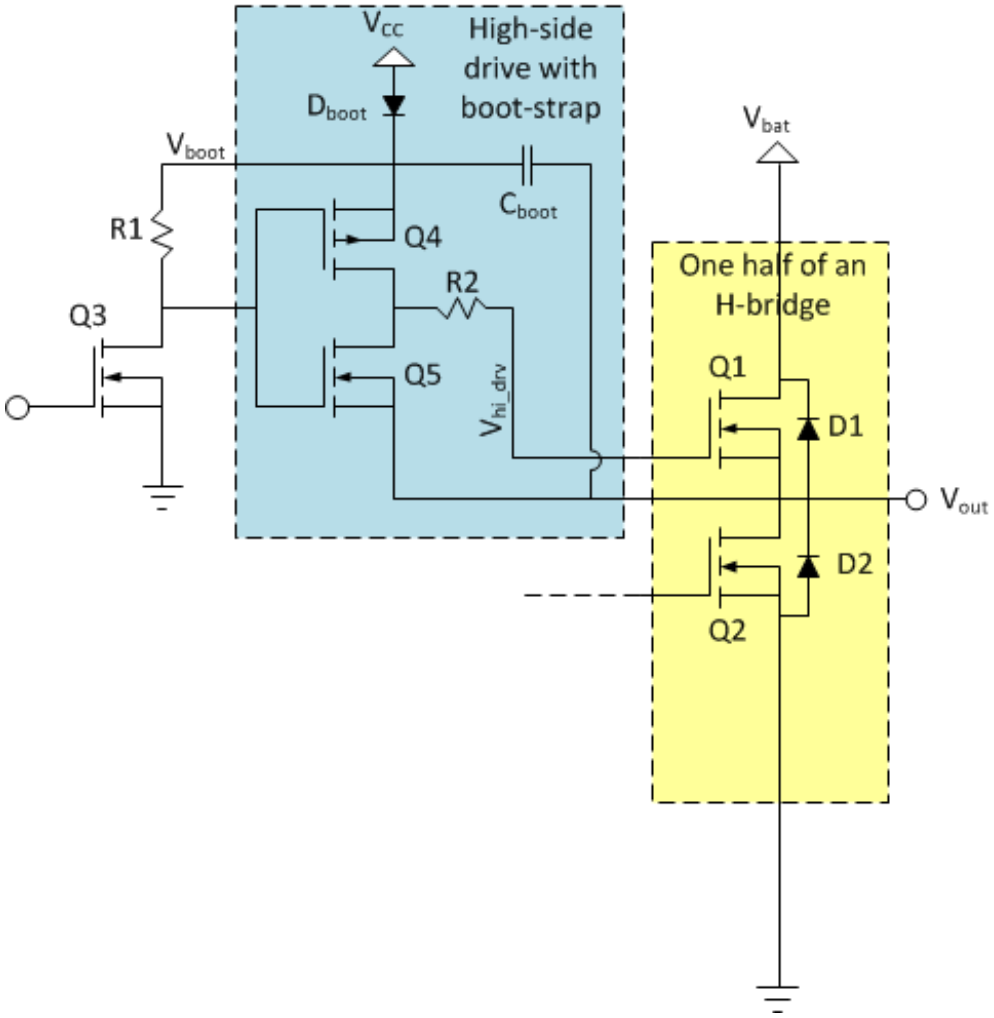


N-MOS high-side drive circuits

So far we've only discussed driving P-MOS devices on the high-side. With all the complexities of level-shifting and voltage-limiting, P-MOS drivers are still simpler than drivers for an N-channel device.

The reason is the following: The source of an N-channel device on the high-side has to be connected to the motor terminal and its drain to the power supply, otherwise the body-diode would be forward-biased and would always conduct. To turn off an N-MOS device in that configuration, you can connect the gate to ground or to the source: gate-source voltage is going to be below or equal to 0. But where to connect the gate to turn on the device? The power supply is not enough, since, if the device is already conducting, it's source and drain are roughly at the same potential. As the drain is connected to power, the source will be at that level as well, but than gate should be **higher** than that to keep the device on. We've seen that the gate should be at minimum 4.5V higher for our example above, and for some other devices maybe as much as 10-15V higher.

As V_{bat} is usually the highest voltage directly available in a system, this voltage needs to be generated. In most cases some kind of a charge-pump is used for that generation, mostly in a boot-strapped configuration:



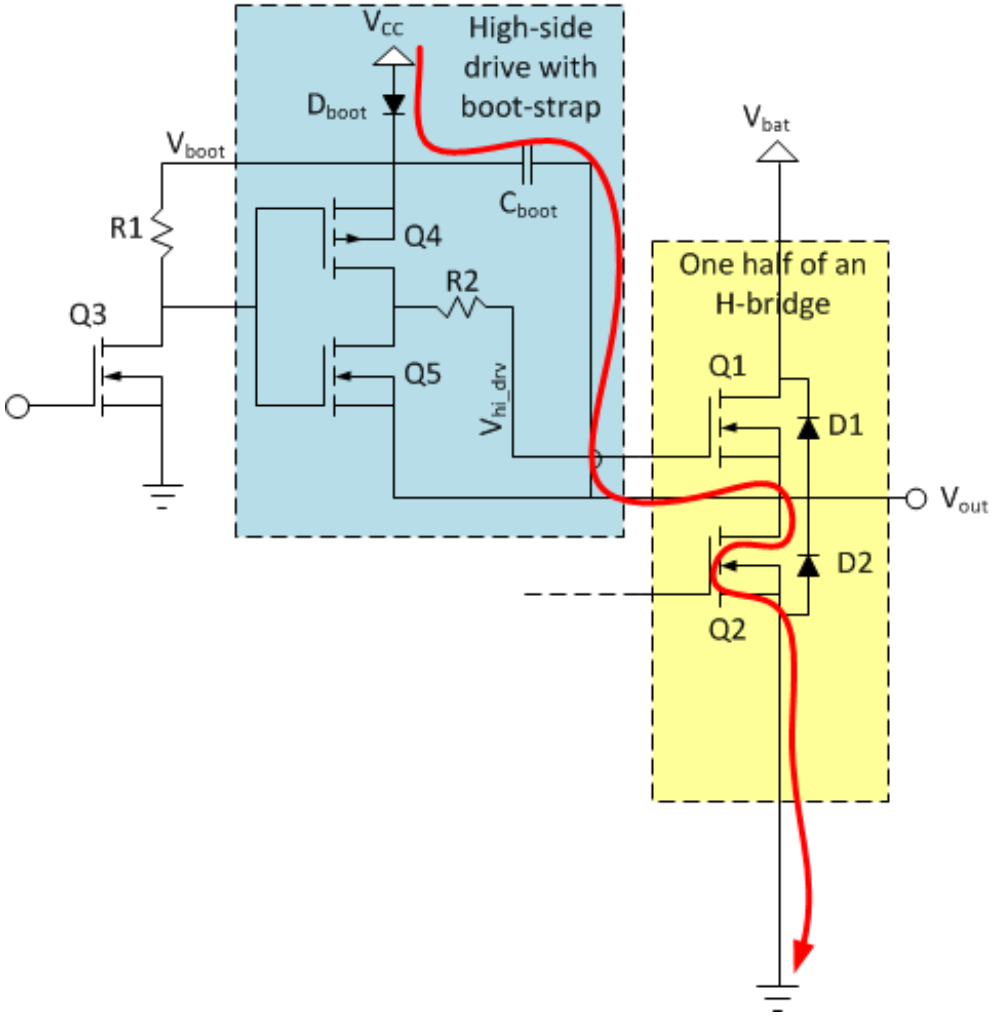
While actual implementations could be quite a bit more complex, I will use this simplified variant to explain how things work. These circuits are usually operated form a voltage supply (V_{cc}) somewhere between 8 and 15V, for our discussion let's say it's working from 12V.

This high-side driver shows strong similarities to the P-MOS high-side drivers we've discussed before, there are significant differences. While it also consists of a level-shifter (Q3, R1) followed by a C-MOS driver stage (Q4, Q5), this stage is neither grounded nor is connected to power. The lower leg of it is connected to the middle terminal of the bridge, or more importantly to the source of the power FET it drives, Q1. This means that the low-level output voltage of this stage (V_{hi_drv}) will be the same potential that the source of Q1 is, whatever that happens to be.

Let's now see, what the voltage (V_{boot}) of the higher leg of the driver – that determines the high-level output voltage – is! To understand the operation of the circuit you'll have to imagine that both the high- and the low-sides are driven by a PWM signal. We close Q2 for some portion of every cycle, and Q1 for the rest (not counting shoot-through protection for a minute).

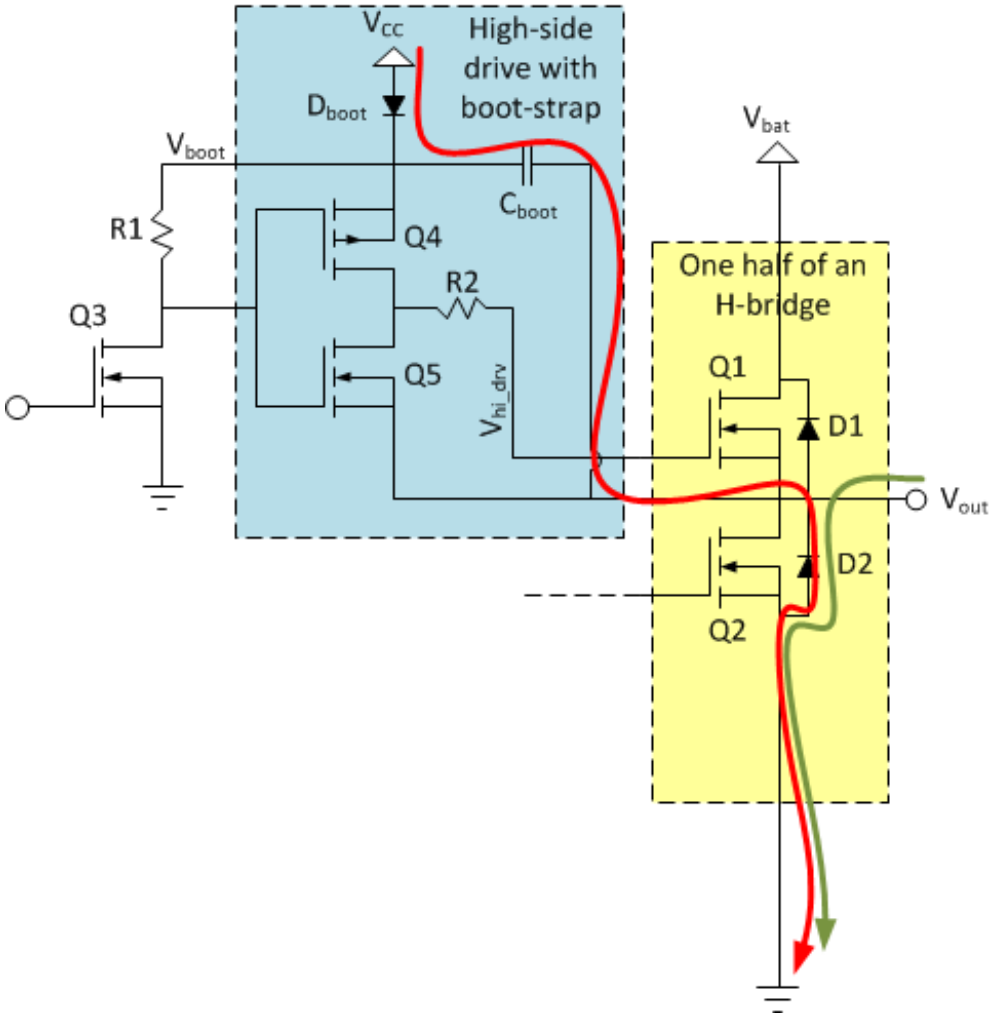
For the part of the cycle, when Q2 conducts, the output terminal voltage (V_{out}) is 0V, or very close to it. Since one side of C_{boot} is connected to this node, it is also grounded. D_{boot}, which is connected between V_{cc} and the other side of C_{boot} will make sure that

C_{boot} is charged up to V_{cc}:



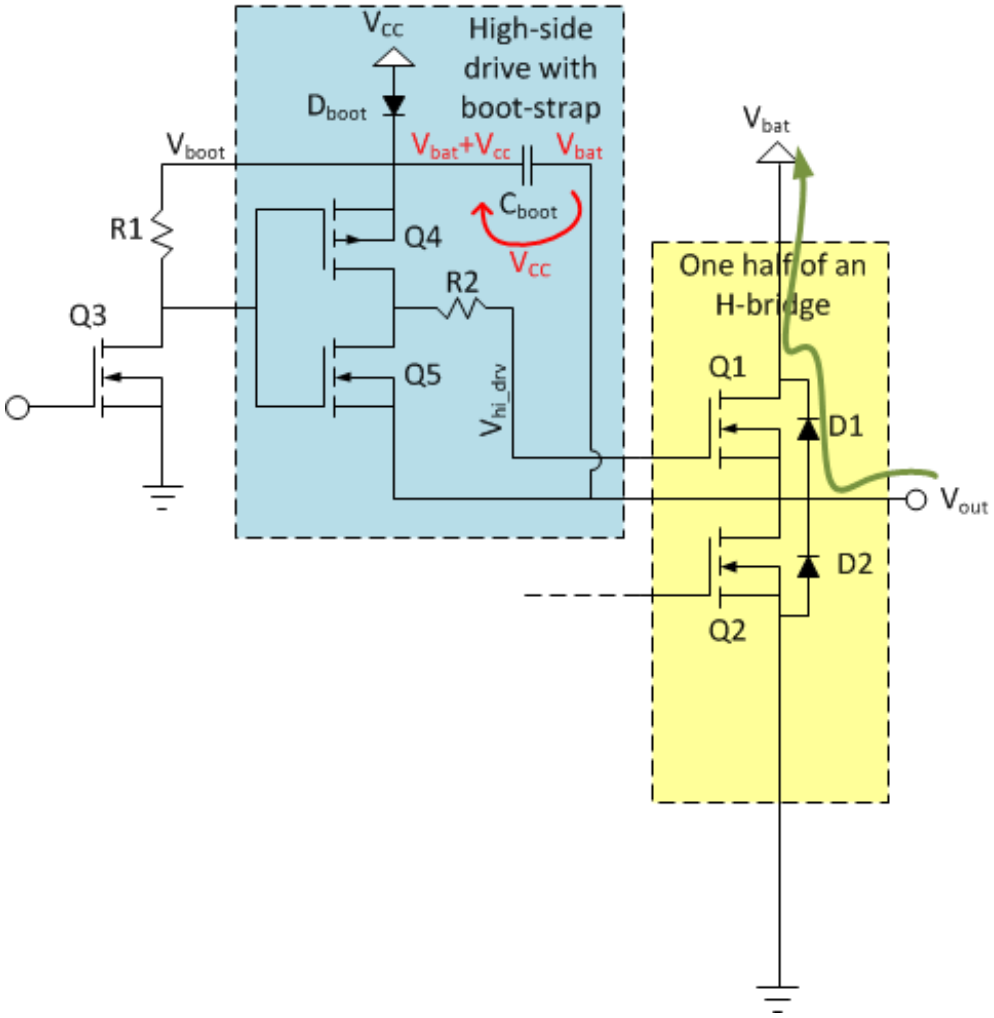
This of course also means that V_{boot} is equal to V_{cc}, 12V in our example. When it comes to turning Q2 off, V_{out} starts floating. Depending on what the motor, and the other side of the bridge does, either D1 or D2 will open and continue conducting the motor current.

If D2 openes, V_{out} would stay at 0V (actually it would go slightly negative for D2 to conduct, but let's ignore that for now), so C_{boot} and V_{boot} stays as they were: charged up to V_{cc}, or 12V higher than V_{out}.



If we wanted to turn Q1 on at this point, Q4 can easily drive it's gate 12V higher than it's source terminal voltage (V_{out}), so we can clearly turn it on.

If however D1 starts conducting after we turned Q2 on, V_{out} rises to V_{bat} (again, slightly higher, to forward-bias the diode, but I'll ignore that here). When that happens – since the voltage across C_{boot} can't change abruptly – V_{boot} has to rise and reach $V_{bat}+V_{cc}$. Normally, C_{boot} would discharge quickly towards V_{cc}, but in our case D_{boot} closes and lets V_{boot} rise as high as it wishes:



At that point V_{out} is at V_{bat} and V_{boot} is higher than that by V_{cc} (12V). If we wanted to turn Q1 on at this point, we can still do it: the high-level output voltage of the driver (V_{boot}) is 12V higher than the source voltage of Q1, as it is connected to V_{out}.

All in all, C_{boot} and D_{boot} will make sure that V_{boot} is always at a higher voltage than V_{out} by V_{cc}. Or at least most of the time. The problem is that whenever V_{boot} is higher than V_{cc}, the only thing that keeps it at that level is the charge kept in C_{boot}. Any current that's flowing out of that node will discharge the capacitor and eventually bring back V_{boot} to only V_{cc}. In our simple circuit, most of that current will flow through R1, but even if we solved that, other leakage currents through the various components will eventually do that. It takes a long time, probably seconds, but inevitably it will happen. What it means is that, while this circuit can certainly turn Q1 on, it can't keep it turned on indefinitely.

The most important consequence is that bridges driven by this type of driver can't operate at 100% duty-cycle: you'll have to give some in every cycle for C_{boot} to re-charge.

As far as drive modes go, the circuit doesn't put a significant limitation on the number of options available. For lock anti-phase drive, all four FETs are switching, so there is not problem at all, but for the two sign-magnitude drives, you'll have to make sure that the FET that's continuously on is on the low-side.

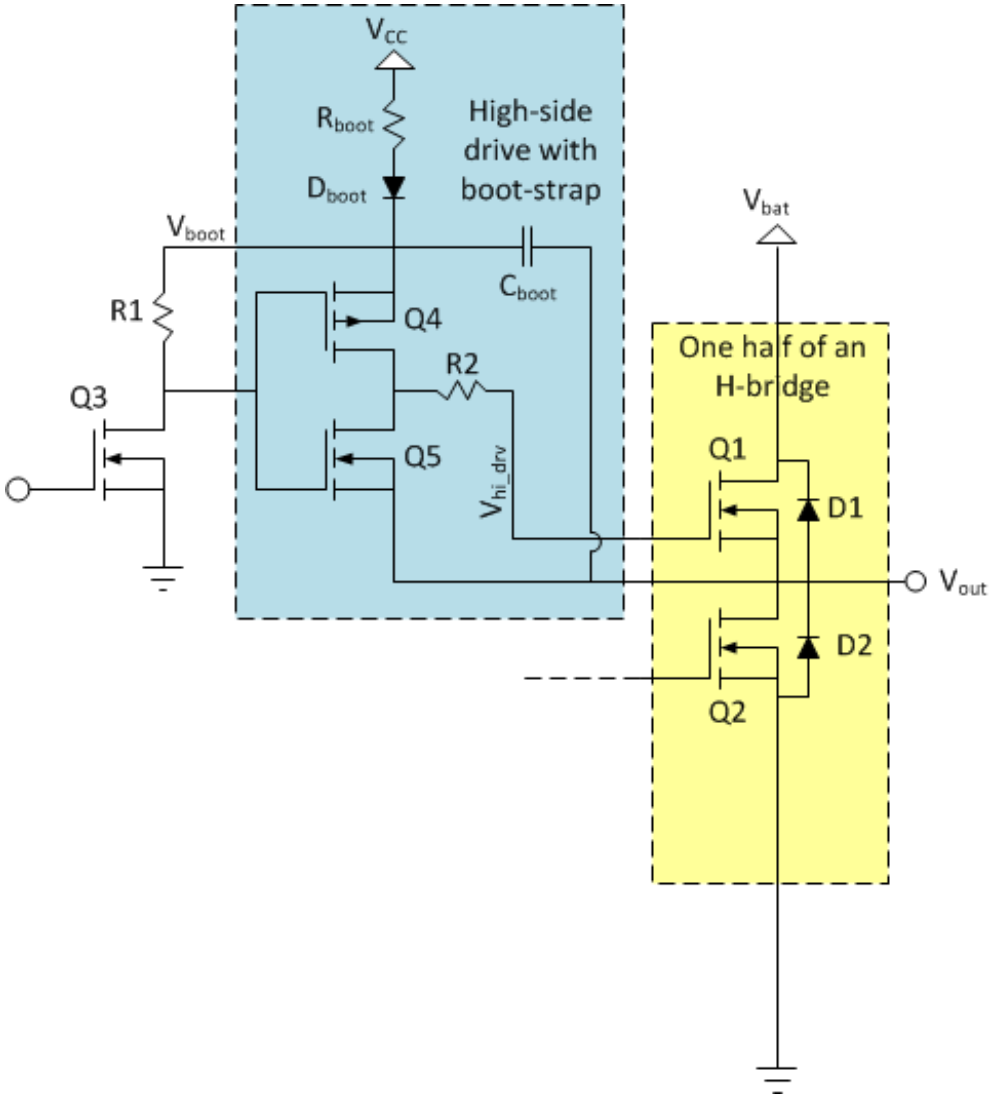
Another problem worth mentioning is this: when you try to turn Q1 on, that is you turn

Q4 on, you basically connect C_{boot} and the gate-capacitance of Q1 in parallel. If D_{boot} is closed, so the only place to get charge to the gate is from C_{boot}, you are essentially just re-distributing the charge between the two capacitors. As that happens, the voltage on them will get lower, and the gate-voltage you can achieve depends on the ratio of the two capacitances:

$V_{gate} = V_{cc} * C_{boot} / (C_{boot} + C_{gate})$

What this means is that if you want to make sure that V_{gate} is – let's say – within 10% of V_{cc}, you have to make sure that C_{boot} is about an order of magnitude higher than the gate-capacitance of Q1. We've seen that the gate-capacitance of large FETs can be several nF, so C_{boot} should be in the 47-100nF range for those devices.

The last thing to talk about is that as the bridge is switching, D_{boot} keeps opening to top-up C_{boot}. But how much current will flow through it? As I've drawn the circuit above, the only thing limiting this current is the internal resistance of D_{boot} and the wires connecting them. All in all, there could be a significant current-spike on V_{cc} through D_{boot} due to the operation of the charge-pump. The power supply generating Vcc has a finite internal resistance, so this current-spike will translate into a periodic voltage drop on V_{cc}. To prevent that, usually a resistor is connected in series with D_{boot} to control the current flowing in the capacitor. It's value is determined to make sure that you can completely re-charge C_{boot} even under worst-case duty-cycle conditions:



Let's say you allow for a maximum of 99% duty cycle, your gate capacitance is 5nF, C_{boot} is 100nF, Vcc is 12V and the operating frequency of the bridge is 20kHz. This means that you will charge the gate of Q1 up to 12V (disregarding now the voltage drop on D_{boot} and due to the charge-redistribution phenomena we've discussed above) 20000 times a second. Each time you charge the capacitor up, you need 5nF * 12V = 60nQ of charge. Since you do that 20000 times a second, the total charge transferred to the capacitor over a second is 1.2mQ, or in other words, your average gate-current is **1.2mA**.

Let's say (due to charge-redistribution and other leakage paths) under worst-case conditions C_{boot} looses 10% of its charge during the on-time. This results in a 1.2V drop on V_{boot}, that needs to be replenished during the off-time, which is (under worst-case conditions) 1% of the total cycle-time, or 500ns. To charge a capacitor of 100nF up by 1.2V in 500ns, you need **0.24A** of (peak) charge current to do it.

From this quick calculation you see that the peak current (flowing through R_{boot} and D_{boot}) can be quite large compared to the modest average current flowing to the gate of Q1. This is important as this high current will stress your power delivery network, and can be a strong noise source for other parts of your design.

If you are curious, there's plenty of more detail about these boot-strap circuits in device datasheets and application notes, like this one: http://www.fairchildsemi.com/an/AN/AN-6076.pdf

Integrated drivers

We haven't talked much about that so far, but the fact is that H-bridges and step-down DC/DC converters share a lot in common. So much so, that the driver circuits we've talked about here are the same that people use for high-current, synchronous DC/DC converters. Such converters are in use on PC motherboards these days, and that brought about an abundance of cheap and high-performance half-bridge drivers. These are almost exclusively of the boot-strap-based dual-N-channel MOSFET driver kind, but are equipped with additional goodies, like shoot-through protection, various input configurations, several voltage options, enable pins, built-in boost diode etc. They usually come in SO-8 packages, but of course other options are also available. You would need to use two of them to make a full bridge driver. Some examples include the L6743Q or L6387A from ST, the ADP3120A from OnSemi. You can also find full H-bridge drivers like the venerable – and expensive – HIP4081A from Intersil.

Conclusion

This was a loooong article I have to admit, but hopefully it gives you some background into the design challenges of the drive circuitry of H-bridges. For all but the most simple applications, specialized drive circuitry is needed as while low-side drive is quite often possible from simple logic signals, high-side drive is usually more involved. Today, the availability of cheap, boot-strapped half-bridge driver ICs makes all N-channel bridges a very attractive design option.

Now, that we've covered the basic construction of the bridges and their associated drive circuits, in the next installment of the series, we will look into the control of the bridge and some interesting variations on the previously established sign-magnitude and asynchronous sign-magnitude drive options.