Chip structure

The Z80 was released in 1976, so it is a relatively old chip. The original chip was built with NMOS technology, although newer variants use CMOS. The NMOS chip is what will be described here.

The chip consists of a few layers. The base is the silicon die. On top of this, a layer of polysilicon provides interconnections as well as implementing transistors. The top layer is a metal interconnect layer. Oxide layers (silicon dioxide) provide insulation between the conductive layers. Connection points provide connections between the different layers as needed.

This chip technology is more advanced than earlier chips (such as some TI calculator chips), which did not have a polysilicon layer. But it is much simpler than modern chips, which may have a dozen or more metal layers.

The feature size of the Z80 is xxx micrometers. This compares to xxx for current chips.

The photograph below shows the basic structure of the Z80 chip. Square pads around the outside of the die are used to connect the chip to the package with very thin metal wires. These wires connect the chip pads to thicker metal wires in the package that lead to the pins.

The Z80 is said to have approximately 8500 transistors. The exact transistor count depends on how transistors are counted. For instance, input pins have transistors that act as clamp diodes. In many cases, two transistors operate in parallel, so they act as one functional transistor. Transistor counts often count potential transistors in a PLA - sites where there could be a transistor, whether or not one is actually there. (This avoids changing the transistor count of a chip if the PLA is reprogrammed.) For these reasons, the transistor count is not as solid a number as you might expect.

The Z80 is powered through a +5V pin and a ground pin. This voltage made the Z80 easy to power and compatible with standard TTL logic. This is simpler than earlier chips such as the 8080 which required +5V, -5V and

+12V. In the Z80, a +5V signal indicates high, or 1, and a 0V (ground) signal indicates low, or 0.

From looking at the die photo, you can see that power distribution is a major factor in the layout of the Z80. The metal layer is used for almost all the power distribution because it has much lower resistance than silicon or polysilicon. There are a few exceptions, though. In the register file, the power is provided through the silicon layer.

Because the Z80 has a single metal layer, routing both power and ground in the metal puts a lot of constraints on on the routing and layout. (In contrast, modern chips with multiple metal layers can devote one or more separate layers to power and ground, making the routing much easier.)

The ground pin is connected to the middle left of the die, and the power pin is connected to the middle right of the die. One ground path flows from the pad around the outside of the die, mainly providing ground for the output pins. (This separate path probably avoids power spikes from the high-current pins from affecting power to the rest of the chip.) A second ground path flows to the right, branching and narrowing in a tree-like structure. As the paths split up and get further away from the ground pad, they get thinner as they don't need to carry as much current.

The +5 supply has a similar pattern, branching off from the pad on the right. One trace circles the chip just inside the pins provides power to the output pins. The other trace goes to the left, branching off and getting thinner. The power and ground traces intermesh like fingers, but don't cross, of course.

The metal layer isn't used only for power and ground; it is also used for long-distance interconnections and bus lines, as well as short-range connection. These metal traces typically run parallel to the power and ground traces, since they can't cross.

The polysilicon layer is very important because polysilicon forms the gates of the NMOS transistors. Whenever polysilicon crosses doped silicon, a transistor is formed.

The polysilicon is also important for chip wiring. Although it has higher resistance than the metal layer, it has lower resistance than silicon. Polysilicon traces can cross metal traces since they are separate layers. Thus, polysilicon traces are especially useful when they run perpendicular to the metal layer.

At the bottom layer is the silicon itself. The silicon is doped with different elements to form transistors. The undoped silicon can be considered an insulator for our purposes. The resistance of doped silicon is fairly high, so it is generally not used for long-distance connections. It is used, however, to connect neighboring transistors. In particular, the transistors in a gate are often formed from one region of doped silicon, separated by gates.

The Z80 uses two types of NMOS transistors: enhancement and depletion. The type of transistor depends on how the silicon is doped. For the purposes of this Z80 analysis, the enhancement transistors can be considered simple switches that turn on (close) when a high voltage is applied to the gate, and turn off (open) when the gate is grounded. The depletion transistors can be considered resistors, which are used as pull-ups to pull an output high.

Building gates with NMOS transistors

Several gates are easy to construct with NMOS. An inverter (also known as a NOT gate) consists of a single enhancement transistors and a single depletion pull-up as shown below. If the input is 1, the transistor closes, connecting ground to the output, which pulls the output low. If the input is 0, the transistor opens and the depletion transistor pulls the output high.

A NOR gate is similar with an extra transistor. A high input on either transistor (or both) will connect the output to ground. Otherwise the pull-up pulls the output high. This implements a NOR gate.

<insert image>

For a NAND gate, the transistors are connected in parallel. If both transistors have a high input, the output will be connected to ground. Otherwise the output is pulled high. Thus, the output is 0 if input A AND input B are 1, implementing a NAND gate.

<insert image>

NAND and NOR gates with more than two inputs are easy to implement, simply by adding more transistors. However, an AND or OR gate cannot be implemented directly in NMOS. Such a gate must be implemented by combining a NAND/NOR gate and an inverter.

One surprising feature of NMOS gates is that complex gates such as OR-AND-NOR are straightforward to implement, just by adding another transistor for each input. For example, the circuit below implements OR-AND-NOR. The logic can be verified by looking at which input combinations pull the output low. The Z80 makes a lot of use of complex gates such as this, especially in the instruction decode logic.

<insert image>

Because of the construction of NMOS gates, with a pull-up resistor, the gates are somewhat asymmetric. In particular, they can produce a “strong” 0 value, sinking significant current due to the connection to ground. But the 1 value is “weak”, since the current through the pull-up resistor (transistor) is limited. One consequence is a gate can pull a wire low faster than it can pull a wire high, especially if the wire is long and thus has higher capacitance.

One solution to this problem is to use a “superbuffer”, which can provide a high current both for a 0 and for a 1 output. In a superbuffer, instead of using a pull-up resistor (depletion transistor), an active transistor connected to +5 volts is used to pull the output high. This transistor is fed with the opposite signal as the pull-down transistors so the output is either actively pulled to ground or pulled to +5 volts. If both transistors were active at the same time, it would be a short circuit and high current would flow through the transistors, which would be bad.

One variant is a superbuffer with a tri-state output. In this case, the output is either +5, ground, or no output. The no output case is implemented by turning off both the transistor to +5 and the transistor to ground. This is useful for a connection to a bus, where the buffer is providing a signal onto the bus only some of the time, while other circuits use the bus other times. A similar tri-state output is used for most of the Z80's output pins, which can be effectively disconnected in order to allow peripherals to control the external buses.

More can be said about the output pins. The output pins provide signals outside the chip at levels of XXX mA, compared to XXX microamps for signals inside the chip. Thus, the transistors must be much larger to source or sink these currents. The image below shows the transistors used for one of the output pins.

Pass transistor logic

The Z80 makes a lot of use of pass transistor logic, which may seem strange if you're only familiar with regular gates.

The circuit below shows a simple example of a circuit with a pass transistor. If the clock is high, the pass transistor is closed, so the input is fed into the inverter and the complement comes out. However, if the clock is low, the pass transistor is opened, and the circuit will keep its previous output, like a latch.

This behavior may seem surprising if you're used to TTL logic, where an unconnected input has undefined behavior. But the characteristics of NMOS transistors make pass transistors very useful.

An NMOS transistor has very high input impedance, because of the insulating oxide layer between the gate and the transistor. As a result, if there is charge on the gate of an NMOS transistor, it will just stay there when the input is disconnected. That is, an NMOS transistor can stay in an on or off state even if nothing is connected to the gate input.

The charge is held on the gate due to the gate's capacitance. The impedance isn't quite infinite, so eventually the charge will leak away. It may take hundreds of microseconds for the charge to leak away, which is a long time by CPU standards. This is one reason why CPUs with pass transistors have a minimum clock speed - the CPU has to move along before the charge leaks away.

Pass transistors are widely used in the Z80. The most common use is a pass transistor controlled by the clock. This synchronizes the input signal with the clock, so it can only change while the clock is high and will be latched while the clock is low.

Pass transistors are also used for multiplexing multiple signals, as shown below. If control A' is high, input A flows to the output. If control B' is high, input B flows to the output, and so on. This provides a more efficient way of combining signals than using logic gates.

Another use of pass transistors is connecting signals to buses, or connecting buses together. As will be explained in a later chapter, the data bus is segmented into multiple pieces by pass transistors. Another example is the register storage. A particular register is connected to the bus by pass transistors as needed. This will be explained in chapter X.

Finally, pass transistors are used for the XOR gate. An XOR gate is inconvenient to implement in this sort of logic, and would require multiple NAND or NOR gates. Instead, the Z80 uses a somewhat unusual circuit, shown below to implement XNOR. If input A is 1, input B is connected to the output. If Input B is 1, input A is connected to the output. Otherwise, the output is pulled high. Working through the cases verifies that the result is XNOR (i.e. inverted XOR). Note that this gate doesn't provide the buffering/amplification of other gates, since the inputs provide the current sink for the output.

<image>

Latches

To store data, the Z80 uses a variety of latches.

As described above, a pass transistor can be used to hold a signal for a clock cycle.

Many latches (such as the flags) use two inverters and a pass transistor as shown below. When the pass transistor is closed, the latch holds its value. When the pass transistor is opened, a new value can be stored in the latch.

The register file uses a pair of inverters as a latch. Instead of using a pass transistor to control writing, a stronger signal is used to overpower the inverters. This will be discussed in more detail in chapter X.

A common type of latch in the Z80 is a flip flop formed from a pair of OR-NAND gates. A typical example is below. When the clock is high, the circuit is stable with each gate outputting an opposite value (either 0, 1 or 1, 0). When the clock is low, a 0 input on a data line will force the corresponding output high. This value will be latched when the clock goes low again. Typically one data line is the inverted value of the other. This latch forms a D flip flop.

The same type of latch can be formed from a pair of AND-NOR gates. The latch functions the same way, except the operation of the clock is reversed. This latch holds it value while the clock is low, and reads the input value while the clock is high.