How to analyze the chip

Analyzing and reverse-engineering a chip such as the Z80 is a complex process.

Die photos
Analysis starts with detailed photos of the die. But before these photos can be obtained, the die must be exposed from its package. This process is called decapsulation.

A chip such as the Z80 is packaged in epoxy. Unfortunately, it is very difficult to dissolve epoxy. To remove the epoxy, typically strong acids such as concentrated nitric acid or concentrated sulfuric acid are required. To improve the dissolving process, these acids may need to be heated. Needless to say, the decapsulation process is dangerous and should only be done by experienced people.

<photo of decapsulated Z80>

Once the die is exposed, many photos are taken of the die through a microscope. A standard (biological) microscope shines light through the sample from below. This works fine for cells, but won't work for an opaque object such as chip. Instead, a metallurgical microscope is used. This microscope shines light from above, through the objective lens, allowing the surface of the die to be illuminated.

Different types of illumination such as darkfield or brighfield can be used to show different details of the chip.

A series of photographs is taken using the microscope, for instance a 10x10 grid. These photographs are then combined into one large image using mosaic software such as <name of software>. The final image is very large, 4700x5000 pixels in the case of the Z80 images.

The initial photograph shows the metal layer of the chip, which is on top. Next, the metal layer is dissolved in acid (usually using dangerous

hydrofluoric acid to remove the protective oxide layer). This exposes the polysilicon underneath, and photographs are taken of the polysilicon layer.

Finally, the polysilicon is removed to expose the bare silicon. Another set of photographs of this layer reveals the underlying transistor structures.

At the end of this process, a set of large images shows the different layers of the chip. With luck, these images are clear and show all the chip structures. However, bits of debris on the chip or scratches may obscure parts of the chip. In addition, it is difficult to entirely remove a layer without removing part of the layer below, so parts of the chip may be slightly obscured. Finally, depending on the characteristics of the chip and the microscope, the features of the chip may be more or less difficult to distinguish.

Circuits in the Z80 can be analyzed from these images. Picking out high-level features is straightforward - if you have experience with other chips, it is straightforward to see the register file, the instruction decode PLA, the ALU, and data buses. However, doing a gate-level analysis is very difficult and error-prone. You need to examine the images carefully to pick out individual transistors and their interconnections, figure out how the transistors are combined into gates, and slowly work your way through the circuitry.

Forming masks

The next step in the Z80 analysis was to convert the images into mask diagrams showing the structure of each layer and the connections between layers. For the Z80, this process was mostly manual, with members of the visual6502 team drawing polygons over each structure in the images. This is a very time-consuming process since there are thousands of structures.

It would be nice to automate this process but unfortunately that didn't work out well with the Z80. One problem is the image quality made it hard to automatically distinguish between layers.

In addition, many of the features are hard to see, such as the contacts between layers. This gets easier with practice, though, since the same patterns are

repeated in the chip, so you can tell where you'd expect to see a contact.

The result of this process is multiple mask diagrams showing silicon, polysilicon, metal, metal-polysilicon contacts, polysilicon-silicon buried contacts, and other features such as pins.

<insert diagram here>

At this point, Pavel Zima(?) wrote a simulator that used these masks as input. The simulator processed the masks to generate a list of transistors and interconnections. The simulator then used a physics-based approach to move charge around the transistors and wires. Because a small amount of charge is moved in each simulation step, the simulator requires many steps for each Z80 clock cycle and is very slow, performing only a few Z80 clocks per second.

With a bit of investigation, we could determine which transistors corresponded to various registers, allowing values to be read out of the simulation. By feeding appropriate values into the (simulated) data pins, memory is simulated and the desired instructions can be executed.

The simulator was very useful to understand the operation of the chip. The simulator also helped find errors in the mask drawings. For instance, <give example of a problem>. By carefully examining the circuitry and comparing with the expected behavior, problems could be tracked down and the masks corrected.

Another problem with the masks is the Z-80 contains multiple “traps” - transistors that looked like one type of transistor but were invisibly modified by ion injection to act as a different type of transistor. The motivation behind this was to make it more difficult for other companies to clone the Z80 - they couldn't just copy the visible circuitry.

These traps caused problems for us too since the masks were generated based on the visible transistors. In several cases the simulator generated the wrong

results. By analyzing the circuitry and by comparing with other Z80 variants, we were able to find and correct some of the traps. This will be discussed more later.

Circuit analysis

Once the transistor “net” has been obtained, the circuitry can be analyzed by hand. This is tedious and only small parts of the chip can be analyzed at a time. This process consists of drawing a schematic of the transistors in a particular circuit and staring at it to figure out what is going on. With a bit of familiarity with transistor logic, it is straightforward to pick out gates and build them into larger circuits. Since this process is very slow and tedious, it's hard to analyze more than a few gates this way.

The next technique to analyze the Z80 was to convert the “net” of transistors and interconnections into higher-level structures such as gates. Many of the gates fall into simple patterns such as NAND, NOR, and inverters, which can be extracted from the transistor list by looking for particular patterns. For instance, a transistor with one connection to ground and another connection to a depletion pull-up transistor forms an inverter. Two of these transistors in parallel form a NOR gate. By looking for simple patterns such as these, many of the gates can be extracted from the net.

The Z80 has many complex gates, such as OR-AND-NOR gates, especially in the instruction decoding circuitry. These gates pose a problem for analysis, since they come in a wide variety and don't fit any fixed pattern. They consist of a bunch of transistors eventually connected to ground or to a depletion pull-up, but the interconnections in between can be complex.

My first approach to analyzing these gates was to use a bipartite graph algorithm to find points where the transistors could be split into groups in series (corresponding to AND terms). Each of these groups was split into groups in parallel (corresponding to OR terms). The splitting took part recursively until individual transistors were obtained. This approach required a complex algorithm, but could handle complex gates. The main flaw with this algorithm was figuring out which transistors were part of a gate and which transistors were pass transistors on the output.

I eventually moved to a simpler, but more accurate approach. Instead of starting with a group of transistors that form a gate and splitting it apart, I started with individual transistors and joined them together. Any transistors in series were joined together as AND terms. Any transistors in parallel were joined together as OR terms. This process was repeated as long as possible. At the end, a gate consisted of a single transistor connected to ground and a pull-up. Any pass transistors on the output could not be merged, so they didn't end up confused with the gate.

One special case that needed to be handled with both algorithms was XOR gates. These gates are expensive to implement with Boolean logic because of the structure of their Karnaugh maps. Instead, the Z80 uses a combination of pass transistors and logic to implement XOR gates. My gate analysis handled XOR gates separately.

After the transistors have been combined into gates, analysis of the circuitry is much easier. The gates can be drawn out on paper into larger circuits. Some of the circuitry is easy to understand, for instance pairs of gates that form latches. Other circuitry, such as the ALU, takes a lot more thought to understand. Some pieces are much more mysterious and can't be understood without understanding other pieces.

At this point, the analysis is kind of like a puzzle which becomes easier to understand as more pieces fit into place. For example, you see that a circuit takes a bunch of mysterious inputs and creates outputs. But once it is determined that the inputs are M1, M2, T1, etc. it becomes clearer that the circuit is a timing circuit. And then looking where the output goes may clarify the meaning of the output.

Some tools for analysis

The simulator is very helpful in understanding the circuitry. By looking at the activity of a transistor or wire in the simulator, it becomes easier to understand what it does.

I wrote a tool to help analyze the activity of different wires for different

instructions. I started by running each instruction through the simulator and collecting the state of every wire at every clock. (This took a day of CPU and generated a very large output file.) Next I wrote a program that lets you select a wire and it graphically shows the activity of that wire for every instruction. If, for instance, the wire is active for T3 of arithmetic operations, this is a good indication that it is controlling a particular ALU operation. If a wire is active for everything involving the B register, it's probably a B register control wire.

Another tool I wrote shows the die and lets you click on a particular wire or transistor. It lists all the connected transistors and displays any known information, such as the gate this transistor is part of, and other connected gates. In this way, the physical chip can be mapped to a gate-level schematic.

In any case, analysis of the Z80 is a slow process, with a lot of puzzling out different parts of the chip. Subsystems of the chip can be analyzed in sequence, for example the decode PLA, the ALU, and the register file. The control signals for these components then become clear, allowing the logic that generates these signals to be understood. There are surprises and unexpected things along the way (such as the 4-bit ALU), but eventually most of the chip can be understood.