

righto.com

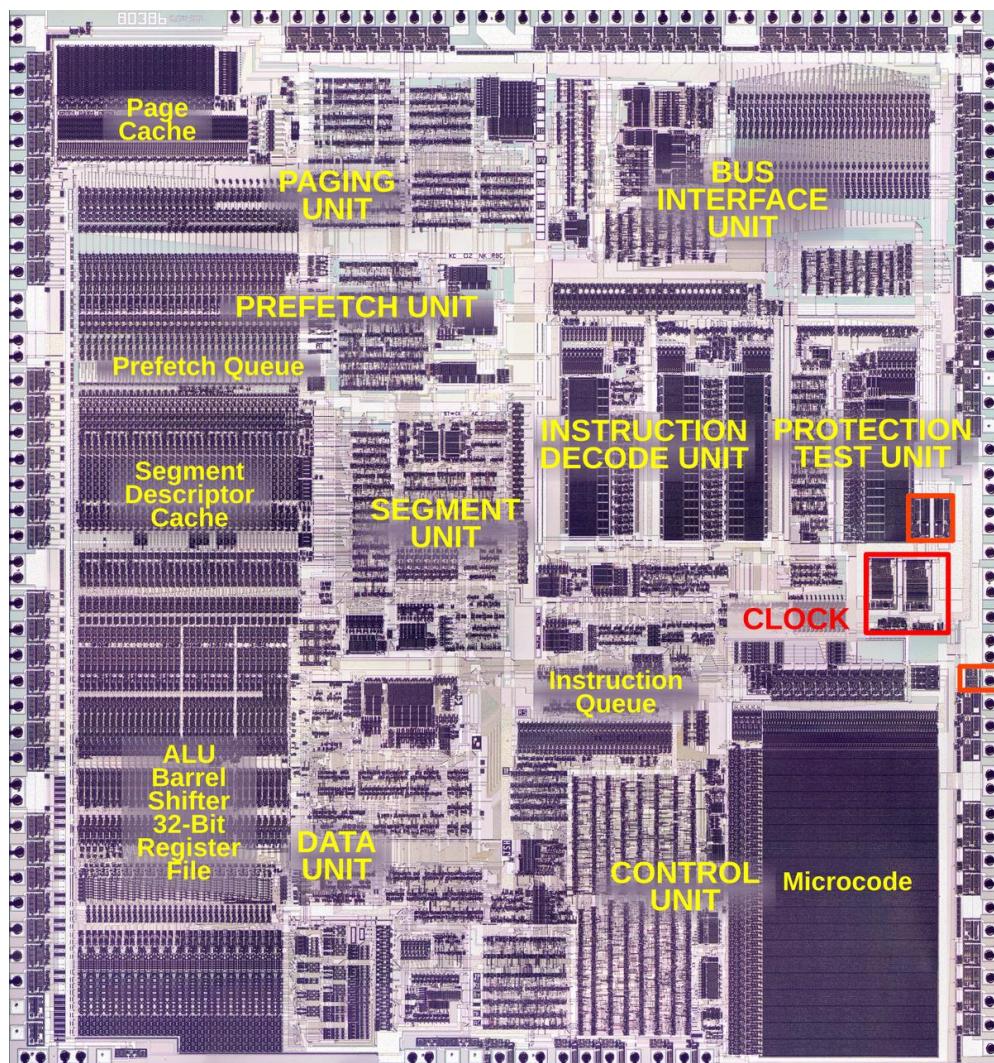
Inside the Intel 386 processor die: the clock circuit

19–24 minutes

Processors are driven by a clock, which controls the timing of each step inside the chip. In this blog post, I'll examine the clock-generation circuitry inside the Intel 386 processor. Earlier processors such as the 8086 (1978) were simpler, using two clock phases internally. The Intel 386 processor (1985) was a pivotal development for Intel as it moved x86 to CMOS (as well as being the first 32-bit x86 processor). The 386's CMOS circuitry required four clock signals. An external crystal oscillator provided the 386 with a single clock signal and the 386's internal circuitry generated four carefully-timed internal clock signals from the external clock.

The die photo below shows the Intel 386 processor with the clock generation circuitry and clock pad highlighted in red. The heart of a processor is the datapath, the components that hold and process data. In the 386,

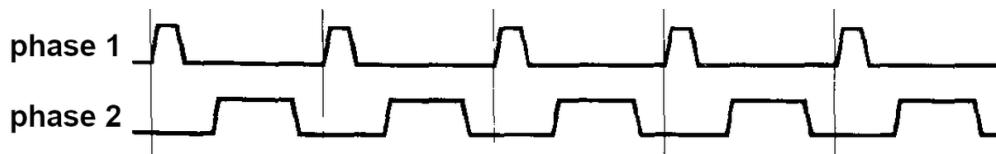
these components are in the lower left: the ALU (Arithmetic/Logic Unit), a barrel shifter to shift data, and the registers. These components form regular rectangular blocks, 32 bits wide. In the lower right is the microcode ROM, which breaks down machine instructions into micro-instructions, the low-level steps of the instruction. Other parts of the chip prefetch and decode instructions, and handle memory paging and segmentation. All these parts of the chip run under the control of the clock signals.



The 386 with the main functional blocks labeled. Click this image (or any other) for a larger version.

A brief discussion of clock phases

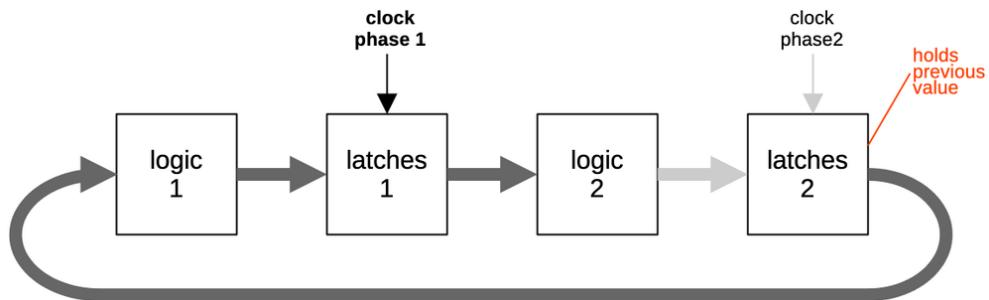
Many processors use a two-phase clock to control the timing of the internal processing steps. The idea is that the two clock phases alternate: first phase 1 is high, and then phase 2 is high, as shown below. During each clock phase, logic circuitry processes data. A circuit called a "transparent latch" is used to hold data between steps.² The concept of a latch is that when a latch's clock input is high, the input passes through the latch. But when the latch's clock input is low, the latch remembers its previous value. With two clock phases, alternating latches are active one at a time, so data passes through the circuit step by step, under the control of the clock.



The two-phase clock signal used by the Intel 8080 processor. The 8080 uses asymmetrical clock signals, with phase 2 longer than phase 1. From the [8080 datasheet](#).

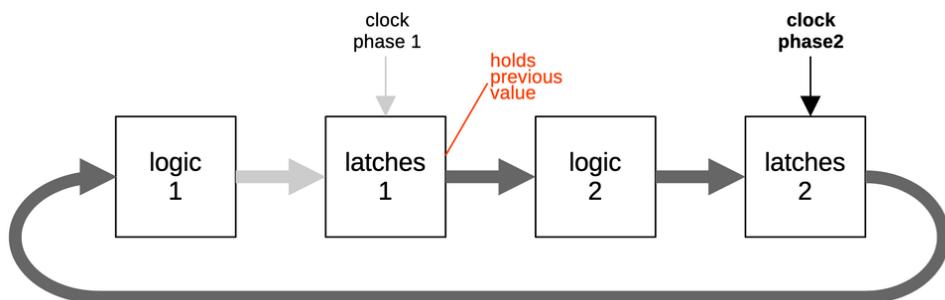
The diagram below shows an abstracted model of the processor circuitry. The combinational logic (i.e. the

gate logic) is divided into two blocks, with latches between each block. During clock phase 1, the first block of latches passes its input through to the output. Thus, values pass through the first logic block, the first block of latches, and the second logic block, and then wait.



Action during clock phase 1.

During clock phase 2 (below), the first block of latches stops passing data through and holds the previous values. Meanwhile, the second block of latches passes its data through. Thus, the first logic block receives new values and performs logic operations on them. When the clock switches to phase 1, processing continues as in the first diagram. The point of this is that processing takes place under the control of the clock, with values passed step-by-step between the two logic blocks.¹



Action during clock phase 2.

This circuitry puts some requirements on the clock timing. First, the clock phases must not overlap. If both clocks are active at the same time, data will flow out of control around the loop, messing up the results.³

Moreover, because the two clock phases probably don't arrive at the exact same time (due to differences in the wiring paths), a "dead zone" is needed between the two phases, an interval where both clocks are low, to ensure that the clocks don't overlap even if there are timing skews. Finally, the clock frequency must be slow enough that the logic has time to compute its result before the clock switches.

Many processors such as the 8080, 6502, and 8086 used this type of two-phase clocking. Early processors such as the 8008 (1972) and 8080 (1974) required complicated external circuitry to produce two asymmetrical clock phases.⁴ For the 8080, Intel produced a special clock generator chip (the [8224](#)) that produced the two clock signals according to the required timing. The Motorola 6800 (1974) required two non-overlapping (but at least symmetrical) clocks, produced by the [MC6875](#) clock generator chip. The MOS 6502 processor (1975) simplified clock generation by producing the two phases internally ([details](#)) from a

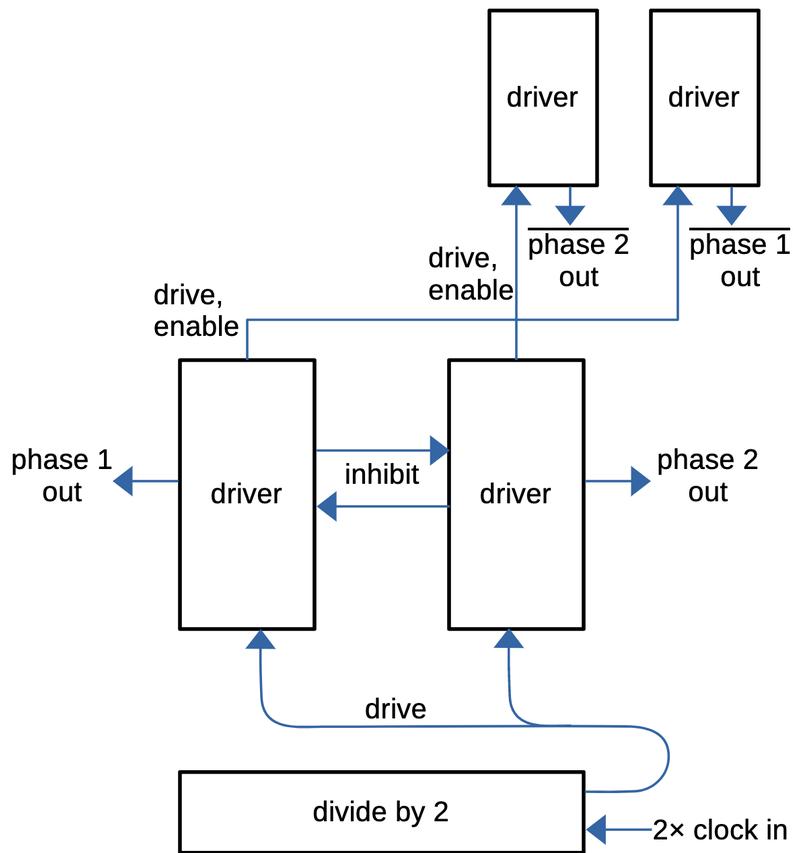
single clock input. This approach was used by most later processors.

An important factor is that the Intel 386 processor was implemented with CMOS circuitry, rather than the NMOS transistors of many earlier processors. A CMOS chip uses both NMOS transistors (which turn on when the gate is high) and PMOS transistors (which turn on when the gate is low).⁷ Thus, the 386 requires an active-high clock signal and an active-low clock signal for each phase,⁵ four clock signals in total.⁶ In the rest of this article, I'll explain how the 386 generates these four clock signals.

The clock circuitry

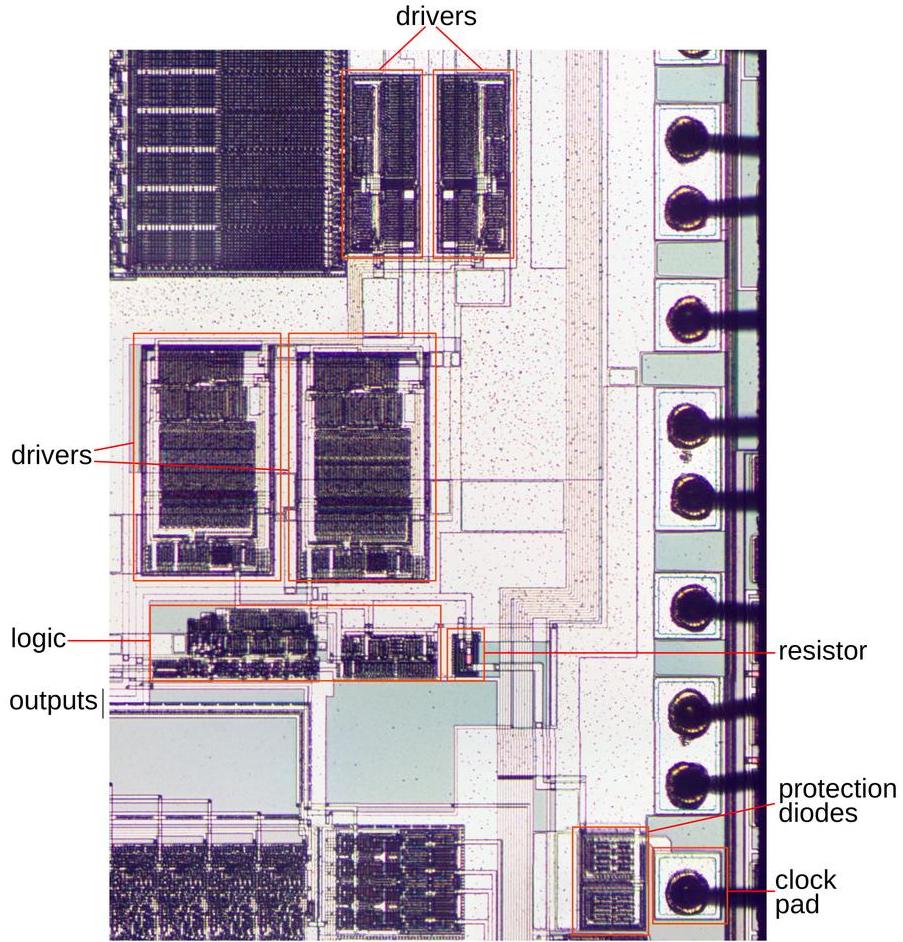
The block diagram below shows the components of the clock generation circuitry. Starting at the bottom, the input clock signal (*CLK2*, at twice the desired frequency) is divided by two to generate two drive signals with opposite phases. These signals go to the large driver circuits in the middle, which generate the two main clock signals (phase 1 and phase 2). Each driver sends an "inhibit" signal to the other when active, ensuring that the phases don't overlap. Each driver also sends signals to a smaller driver that generates the inverted clock signal. The "enable" signal shapes the output to prevent overlap. The four clock output signals

are then distributed to all parts of the processor.



Block diagram of the clock circuitry. The layout of the blocks matches their approximate physical arrangement.

The diagram below shows a closeup of the clock circuitry on the die. The external clock signal enters the die at the clock pad in the lower right. The signal is clamped by protection diodes and a resistor before passing to the divide-by-two logic, which generates the two clock phases. The four driver blocks generate the high-current clock pulses that are transmitted to the rest of the chip by the four output lines at the left.

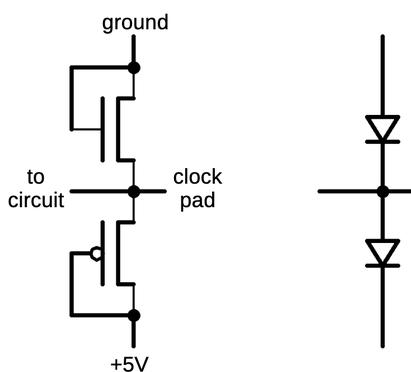


Details of the clock circuitry. This image shows the two metal layers. At the right, bond wires are connected to the pads on the die.

Input protection

The 386 has a pin "*CLK2*" that receives the external clock signal. It is called *CLK2* because this signal has twice the frequency of the 386's clock. The chip package connects the *CLK2* pin through a tiny bond wire (visible above) to the *CLK2* pad on the silicon die. The *CLK2* input has two protection diodes, created from

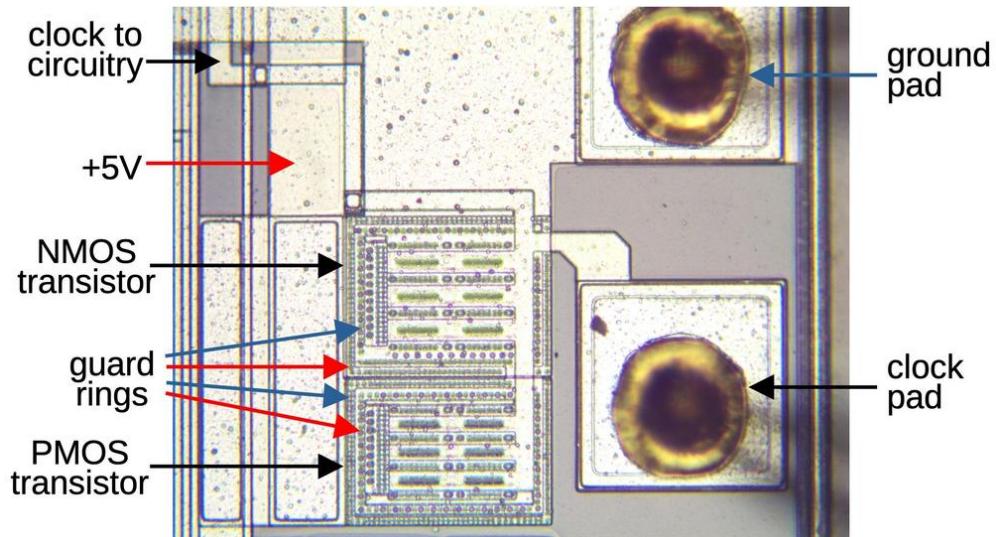
MOSFETs, as shown in the schematic below. If the input goes below ground or above +5 volts, the corresponding diode will turn on and clamp the excess voltage, protecting the chip. The schematic below shows how the diodes are constructed from an NMOS transistor and a PMOS transistor. The schematic corresponds to the physical layout of the circuit, so power is at the bottom and the ground is at the top.



The input protection circuit. The left shows the physical circuit built from an NMOS transistor and a PMOS transistor, while the right shows the equivalent diode circuit.

The diagram below shows the implementation of these protection diodes (i.e. transistors) on the die. Each transistor is much larger than the typical transistors inside the 386, because these transistors must be able to handle high currents. Physically, each transistor consists of 12 smaller (but still relatively large) transistors in parallel, creating the stripes visible in the image. Each transistor block is surrounded by two

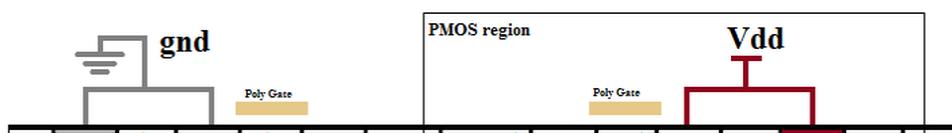
guard rings, which I will explain in the next section.

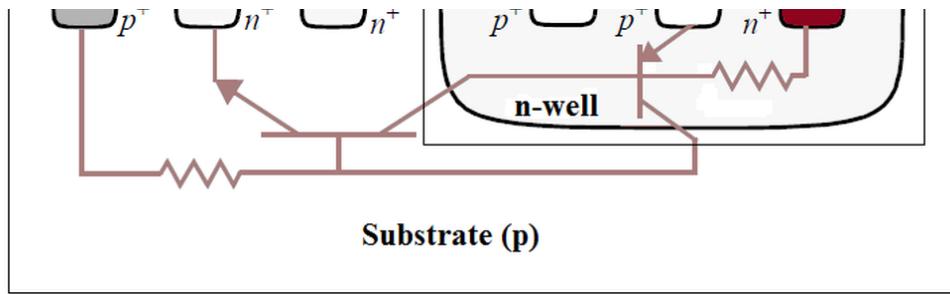


This diagram shows the circuitry next to the clock pad.

Latch-up and the guard rings

The phenomenon of "latch-up" is the hobgoblin of CMOS circuitry, able to destroy a chip. Regions of the silicon die are doped with impurities to form N-type and P-type silicon. The problem is that the N- and P-doped regions in a CMOS chip can act as parasitic NPN and PNP transistors. In some circumstances, these transistors can turn on, shorting power and ground. Inconveniently, the transistors latch into this state until the power is removed or the chip burns up. The diagram below shows how the substrate, well, and source/drain regions can combine to act as unwanted transistors.⁸





This diagram illustrates how the parasitic NPN and PNP transistors are formed in a CMOS chip. Note that the 386's construction is opposite from this diagram, with an N substrate and P well. Image by [Deepon, CC BY-SA 3.0.](#)

Normally, P-doped substrate or wells are connected to ground and the N-doped substrate or wells are connected to +5 volts. As a result, the regions act as reverse-biased diodes and no current flows through the substrate. However, a voltage fluctuation or large current can disturb the reverse biasing and the resulting current flow will turn on these parasitic transistors.

Unfortunately, these parasitic transistors drive each other in a feedback loop, so once they get started, they will conduct more and more strongly and won't stop until the chip is powered down. The risk of latch-up is highest with circuits connected to the unpredictable voltages of the outside world, or high-current circuits that can cause power fluctuations. The clock circuitry has both of these risks.

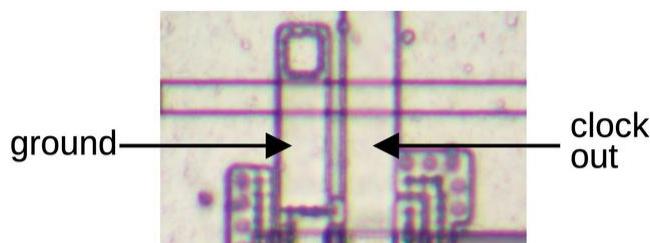
One way of protecting against latch-up is to put a guard

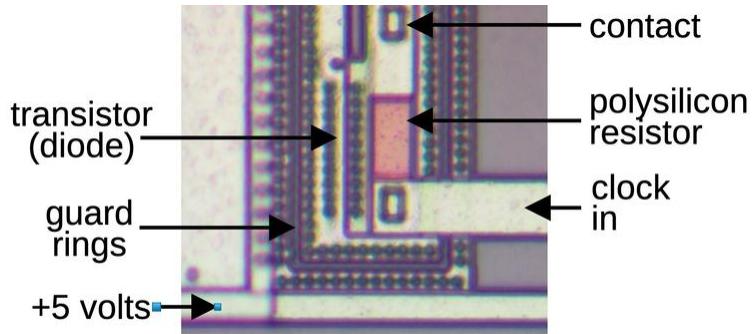
ring around a potentially risky circuit. This guard ring will conduct away the undesired substrate current before it can cause latch-up. In the case of the 386, two concentric guard rings are used for additional protection.⁹ In the earlier die photo, these guard rings can be seen surrounding the transistors. Guard rings will also play a part in the circuitry discussed below.

Polysilicon resistor

After the protection diodes, the clock signal passes through a polysilicon resistor, followed by another protection diode. Polysilicon is a special form of silicon that is used for wiring and also forms the transistor gates. The polysilicon layer sits on top of the base silicon; polysilicon has a moderate amount of resistance, considerably more than metal, so it can be used as a resistor.

The image below shows the polysilicon resistor along with a protection diode. This circuit provides additional protection against transients in the clock signal.¹⁰ This circuit is surrounded by two concentric guard rings for more latch-up protection.

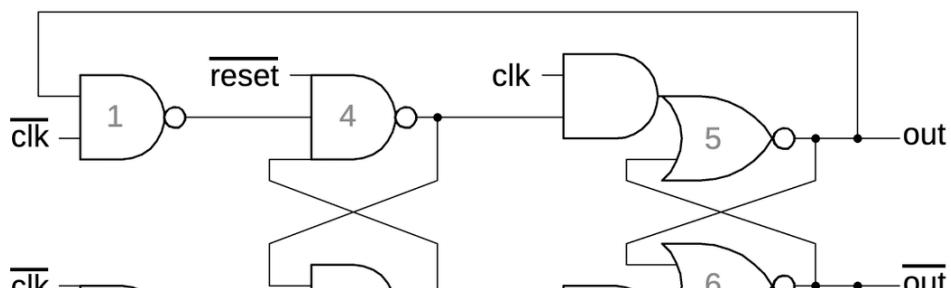


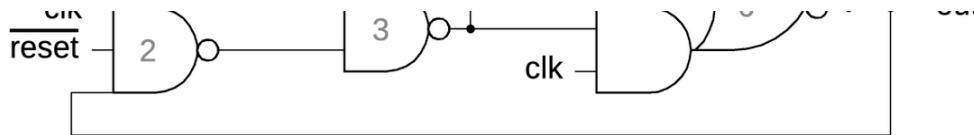


The polysilicon resistor and associated diode.

The divide-by-two logic

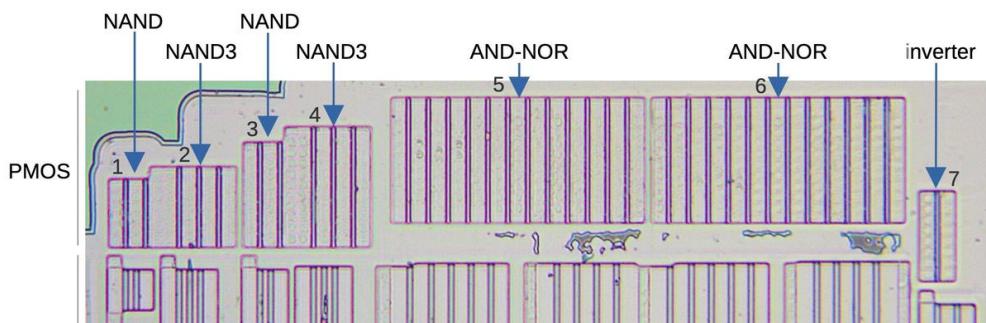
The input clock to the 386 runs at twice the frequency of the internal clock. The circuit below divides the input clock by 2, producing complemented outputs. This circuit consists of two set-reset latch stages, one driven by the input clock inverted and the second driven by the input clock, so the circuit will update once per input clock cycle. Since there are three inversions in the loop, the output will be inverted for each update, so it will cycle at half the rate of the input clock. The reset input is asymmetrical: when it is low, it will force the output low and the complemented output high. Presumably, this ensures that the processor starts with the correct clock phase when exiting the reset state.

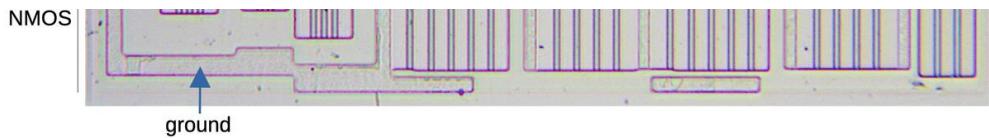




The divide-by-two circuit.

I have numbered the gates above to match their physical locations below. In this image, I have etched the chip down to the silicon so you can see the active silicon regions. Each logic gate consists of PMOS transistors in the upper half and NMOS transistors in the lower half. The thin stripes are the transistor gates; the two-input NAND gates have two PMOS transistors and two NMOS transistors, while the three-input NAND gates have three of each transistor. The AND-NOR gates need to drive other circuits, so they use paralleled transistors and are much larger. Each AND-NOR gate contains 12 PMOS transistors, four for each input, but uses only 9 NMOS transistors. Finally, the inverter (7) inverts the input clock signal for this circuit. The transistors in each gate are sized to maximize performance and minimize power consumption. The two outputs from the divider then go through large inverters (not shown) that feed the driver circuits.[11](#)





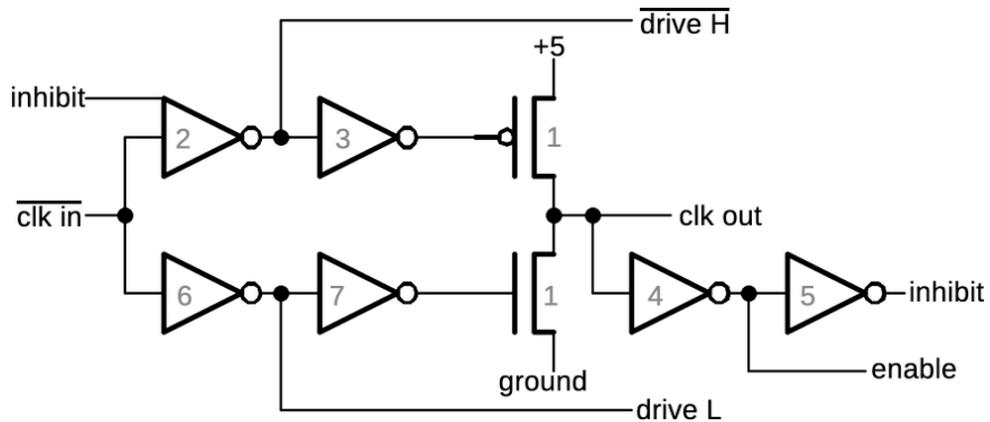
The silicon for the divide-by-two circuit as it appears on the die.

The drivers

Because the clock signals must be transmitted to all parts of the die, large transistors are required to generate the high-current pulses. These large transistors, in turn, are driven by medium-sized transistors. Additional driver circuitry ensures that the clock signals do not overlap. There are four driver circuits in total. The two larger, lower driver circuits generate the positive clock pulses. These drivers control the two smaller, upper driver circuits that generate the inverted clock pulses.

First, I'll discuss the larger, positive driver circuit. The core of the driver consists of the large PMOS transistor (1) to pull the output high, and the large NMOS transistor (1) to pull the output low. Each transistor is driven by two inverters (2/3 and 6/7 respectively). The circuit also produces two signals to shape the outputs from the other drivers. When the clock output is high, the "inhibit" signal goes to the other lower driver and inhibits that driver from pulling its output high.[12](#) This

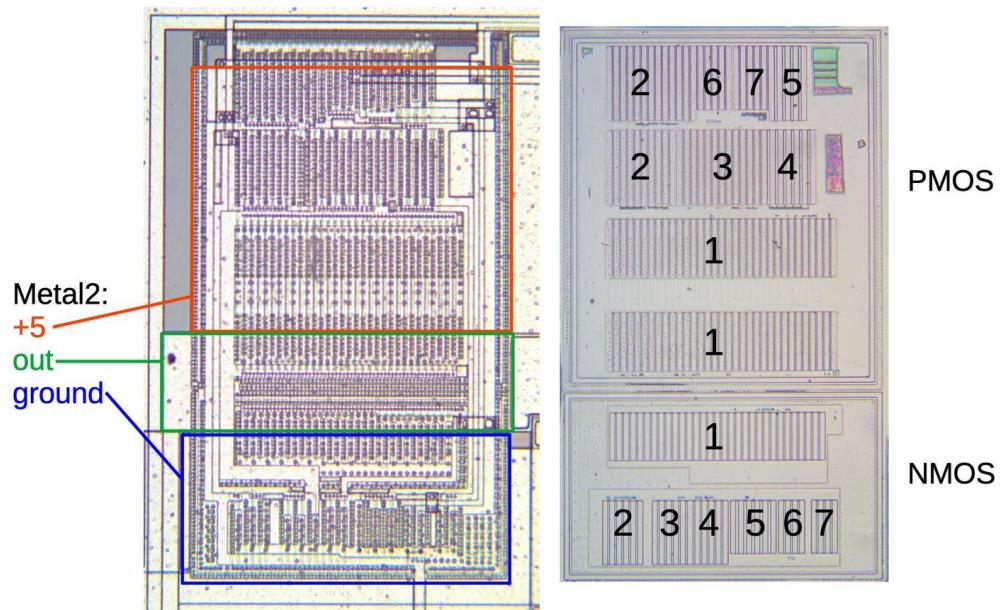
prevents overlap in the output between the two drivers. When the clock output is low, an "enable" output goes to the inverted driver (discussed below) to enable its output. The transistor sizes and propagation delays in this circuit are carefully designed to shape the internal clock pulses as needed.



Schematic of the lower driver.

The diagram below shows how this driver is implemented on the die. The left image shows the two metal layers. The right image shows the transistors on the underlying silicon. The upper section holds PMOS transistors, while the lower section holds NMOS transistors. Because PMOS transistors have poorer performance than NMOS transistors, they need to be larger, so the PMOS section is larger. The transistors are numbered, corresponding to the schematic above. Each transistor is physically constructed from multiple transistors in parallel. The two guard rings are visible in the silicon, surrounding and separating the PMOS and

NMOS regions.

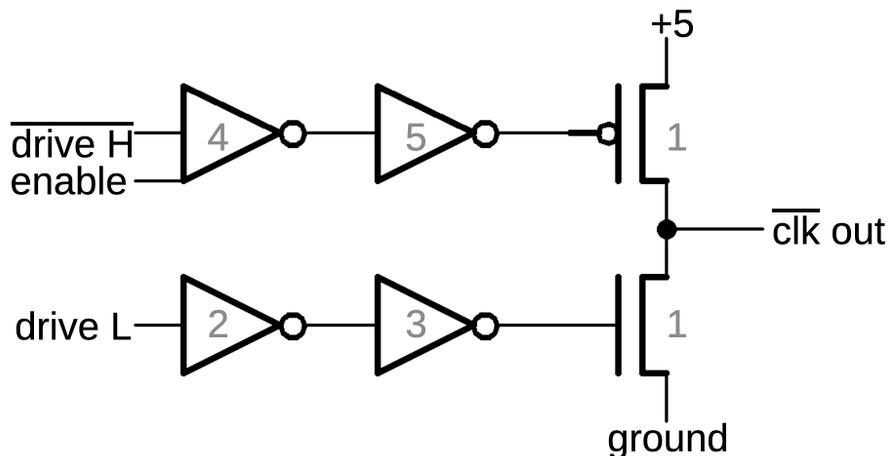


One of the lower drivers. The left image shows metal while the right image shows silicon.

The 386 has two layers of metal wiring. In this circuit, the top metal layer (M2) provides +5 for the PMOS transistors, ground for the NMOS transistors, and receives the output, all through large rectangular regions. The lower metal layer (M1) provides the physical source and drain connections to the transistors as well as the wiring between the transistors. The pattern of the lower metal layer is visible in the left photo. The dark circles are connections between the lower metal layer and the transistors or the upper metal layer. The connections to the two guard rings are visible around the edges.

Next, I'll discuss the two upper drivers that provided the

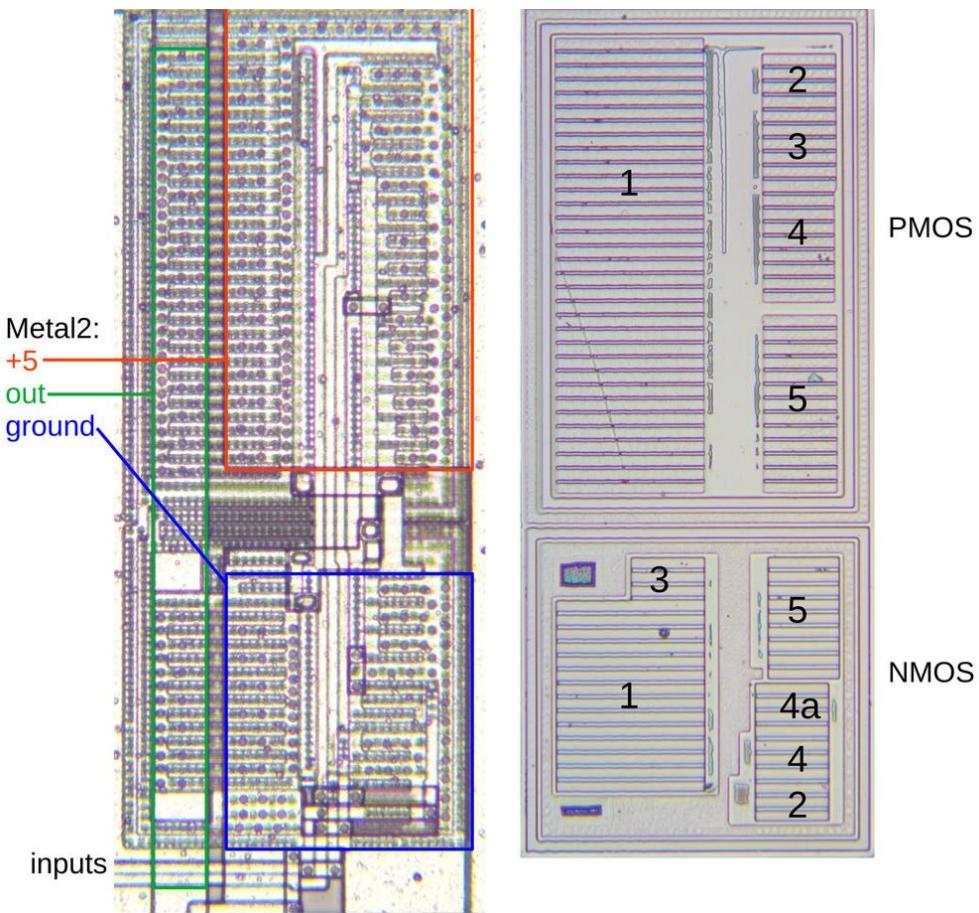
inverted clock signals. These drivers are smaller, presumably because less circuitry needs the inverted clocks. Each upper driver is controlled by enable and drive from the corresponding lower driver. As before, two large transistors pull the output high or low, and are driven by inverters. The enable input must be high for inverter 4 to go low. Curiously, the enable input is wired to the output of inverter 4. Presumably, this provides a bit of shaping to the signal.



Schematic of the upper driver.

The layout (below) is roughly similar to the previous driver, but smaller. The driver transistors (1) are arranged vertically rather than horizontally, so the metal 2 rectangle to get the output is on the left side rather than in the middle. The transistor wiring is visible in the lower (metal 1) layer, running vertically through the circuit. As before, two guard rings surround the PMOS and NMOS regions.





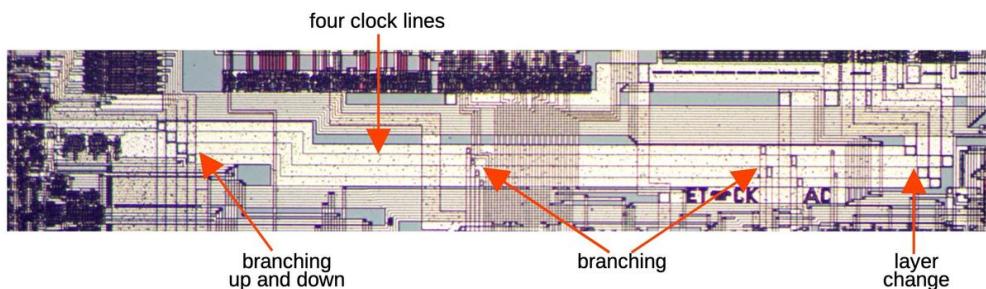
One of the upper drivers. The left image shows metal while the right image shows silicon.

Distribution

Once the four clock signals have been generated, they are distributed to all parts of the chip. The 386 has two metal layers. The top metal layer (M2) is thicker, so it has lower resistance and is used for clock (and power) distribution where possible. The clock signal will use the lower M1 metal layer when necessary to cross other M2 signals, as well as for branch lines off the main clock

lines.

The diagram below shows part of the clock distribution network; the four parallel clock lines are visible similarly throughout the chip. The clock signal arrives at the upper right and travels to the datapath circuitry on the left. As you can see, the four clock lines are much wider than the thin signal lines; this width reduces the resistance of the wiring, which reduces the RC (resistive-capacitive) delay of the signals. The outlined squares at each branch are the vias, connections between the two metal layers. At the right, the incoming clock signals are in layer M1 and zig-zag to cross under other signals in M2. The clock distribution scheme in the 386 is much simpler than in modern processors.



Part of the wiring for clock distribution. This image spans about 1/5 of the chip's width.

Clocks in modern processors

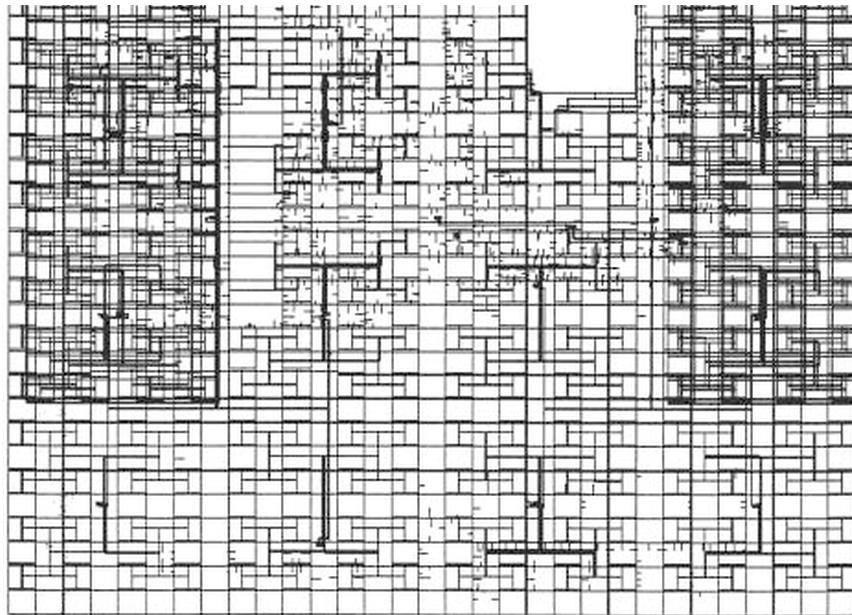
The 386's internal clock speed was simply the external clock divided by 2. However, modern processors allow the clock speed to be adjusted to optimize performance

or to overclock the chip. This is implemented by an on-chip PLL (Phase-Locked Loop) that generates the internal clock from a fixed external clock, multiplying the clock speed by a selectable multiplier. Intel introduced a PLL to the 80486 processor, but the multiplier was fixed until the Pentium.

The Intel 386's clock can go up to 40 megahertz. Although this was fast for the time, [modern processors](#) are over two orders of magnitude faster, so keeping the clock synchronized in a modern processor requires complex techniques.¹³ With fast clocks, even the speed of light becomes a constraint; at 6 GHz, light can travel just 5 centimeters during a clock cycle.

The problem is to ensure that the clock arrives at all circuits at the same time, minimizing "clock skew". Modern processors can reduce the clock skew to a few picoseconds. The clock is typically distributed by a "clock tree", where the clock is split into branches with each branch buffered and the same length, so the delays nearly match. One approach is an "H-tree", which distributes the clock through an H-shaped path. Each leg of the H branches into a smaller H recursively, forming a space-filling fractal, as shown below.





Clock distribution in a PowerPC chip. The recursive H pattern is only approximate since other layout factors constrain the clock tree. From [ISSCC 2000](#).

Delay circuitry can actively compensate for differences in path time. A Delay-Locked Loop (DLL) circuit adds variable delays to counteract variations along different clock paths. The Itanium used a clock distribution hierarchy with global, regional, and local distribution of the clock. The main clock was distributed to eight regions that each deskewed the clock (in 8.5 ps steps) and drove a regional clock grid, keeping the clock skew under 28 ps. The Pentium 4's complex distribution tree and skew compensation circuitry got clock skew below ± 8 ps.

Conclusions

The 386's clock circuitry turned out to be more complicated than I expected, with a lot of subtlety and complications. However, examining the circuit illustrates several features of CMOS design, from latch circuits and high-current drivers to guard rings and multi-phase clocks. Hopefully you have found this interesting.

I plan to write more about the 386, so follow me on Twitter [@kenshirriff](#) or [RSS](#) for updates. I'm also on Mastodon occasionally as [@kenshirriff@oldbytes.space](https://oldbytes.space/@kenshirriff).

Thanks to William Jones for discussing a couple of errors.

Notes and references