How Calculators Remember

Tony Duell

Presented at HPCC Mini-Conference October 2023

One reason that I am interested in desktop calculators is that I get to see methods and technologies that were quickly superseded in larger machines and thus almost impossible to find examples of there. Calculators were normally smaller, cheaper, and simpler than full computers and thus old methods were kept in use rather later. Today I want to look at one aspect of this, memory.

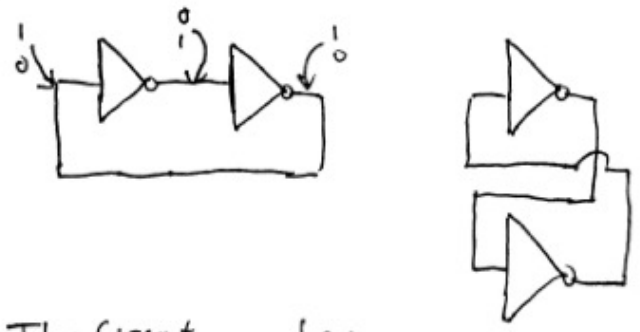
It is obvious that any electronic calculator, even a simple 4-function machines, has to have some storage for the numbers it is working with. So, let's see how that was done. I am concentrating on desktop machines as the handhelds have all the interesting bits hidden inside integrated circuits where I can't examine them.

Eccles-Jordan flip-flop

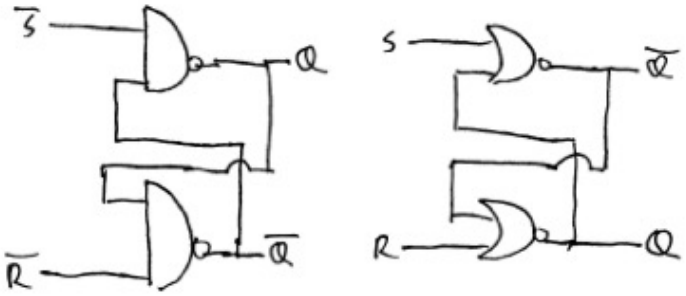
Let's begin with the obvious method which wasn't generally used in older calculators for a reason that I will come to in a moment.

I am sure everybody here recognises 'this statement is false' as a paradox. It can't be either true or false. But what about 'this statement is true'? That is indeterminate. It could be true (in which case what it says is correct, namely that it is true) or it could be false (in which case what it says is wrong, so it's saying it's not true). Now if we consider the equivalent statement 'this statement is not false' and take that one stage further to 'this statement is not not-true' we come to an interesting result

SLIDE 1



This circuit can be in either of 2 states, but is useless since it can't be forced into either of them.



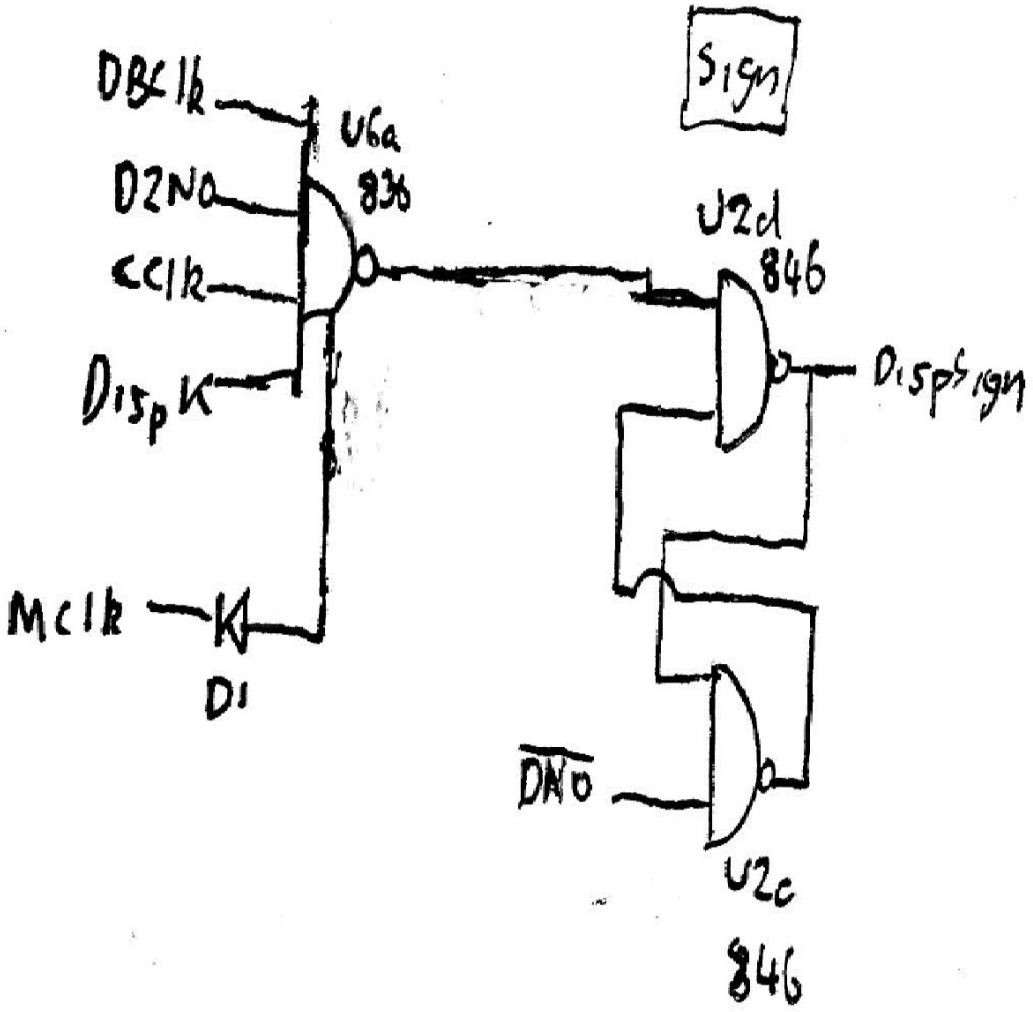
Useful SR Flip-flop circuits

A ring of 2 NOT gates -- the top diagram on this slide -- is stable. Either inverter could have a '1' at the output which would then cause a '0' at its input due to the other inverter. In this form it is USELESS. It's a circuit that indeed has 2 stable states but no way of forcing it to either one or the other.

But now consider a NAND gate with 2 inputs. Anything NANDed with '1' is simply inverted. Anything NANDed with '0' gives '1' . So a NAND gate can be used as a NOT gate where you can force the output to '1'. Put 2 of those in a ring -- the second diagram -- and you have a practical circuit to store a single bit. Normally both S/ and R/ are '1'. so the thing is a ring of 2 NOT gates. Bring either of them low and you can force it to one state or the other.

Of course, you can play the same sort of trick with NOR gates. Anything NORed with '0' is simply inverted, anything NORed with '1' is '0'. So again 2 of those in a ring make a single-bit store, the third diagram.

SLIDE 2

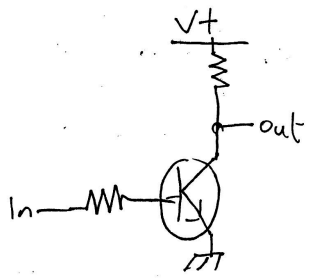


This circuit is of course very common in digital electronics. This slide shows an example of its use in the Olympia ICR412 desktop calculator. This flip-flop simply stores whether or not to turn the '-' sign indicator on on the display.

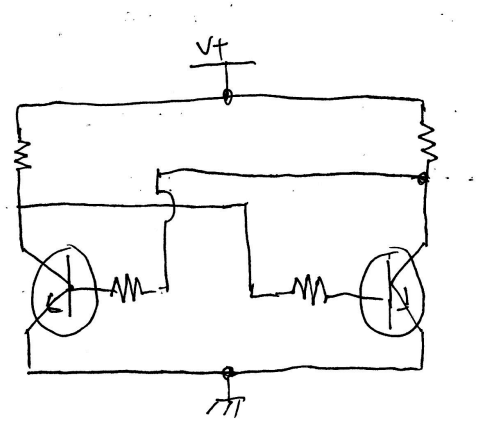
As an aside, if you make a circuit that accepts 3 different voltages -- let's call them '-', '0', '+' where the output is a cyclic permutation of the input -- say '-' gives '0', '0' gives '+' and '+' gives '-' then a ring of 3 such circuits forms a device with 3 stable states. This has been termed a 'flip flap flop'. It's practically useless as a pair of normal flip-flops is simpler, but it's one of those curiosities that's fun to think about.

Now let's go down a level.

SLIDE 3



Not gate



Eccles-Jordan Flip-flop

'When I was a lad in 1919

Eccles and Jordan came upon the scene

With a pair of triodes and some resistor gates

They built a circuit with 2 stable states'

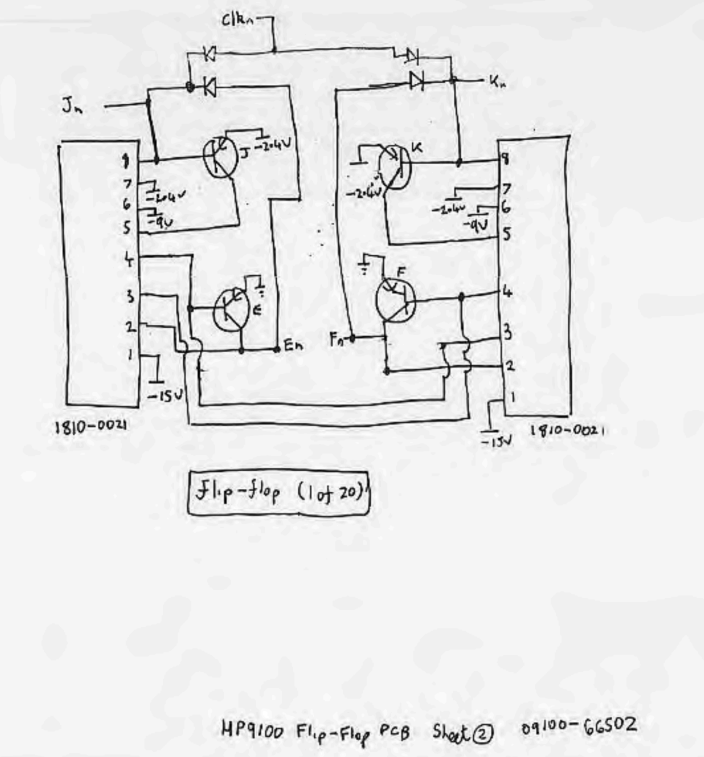
as the filkbook puts it. Now I don't intend to talk about triodes today, but the interesting thing is that the circuit works with just about any component that can be used as a switch. Triode valves, transistors, FETs, relays, etc.

At the top of this slide is a simple circuit that works as a NOT gate. 2 resistors and a transistor. If the input is '0' then the transistor is cut off, so the output is a '1', there is little voltage drop across the collector resistor. If the input is '1' then current flows through the base resistor into the base of the transistor which then saturates. The collector voltage is thus low, and the output is a '0'.

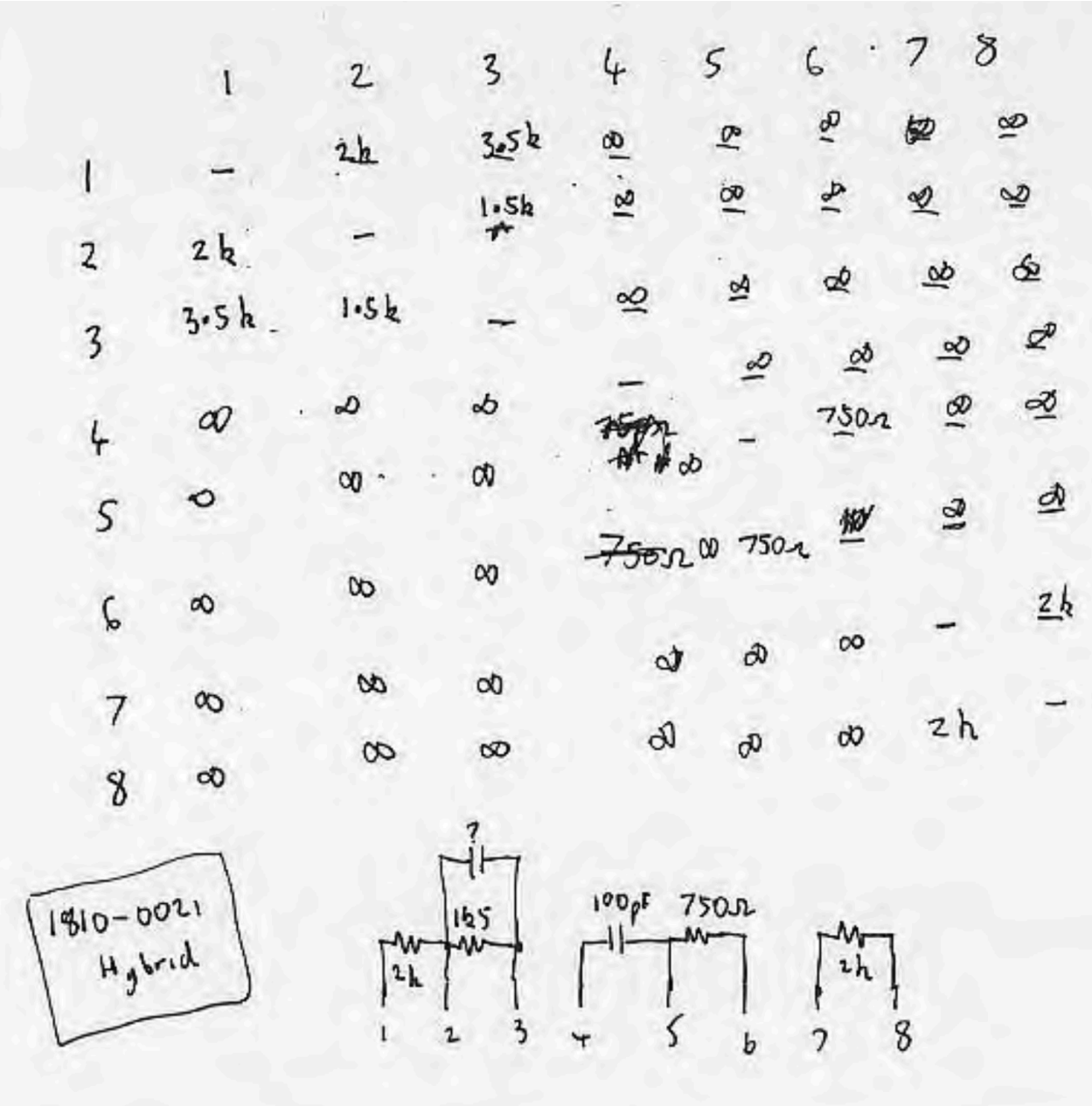
The next circuit shows how 2 of these can be interconnected to make the classic Eccles-Jordan bistable circuit. It's just the ring of 2 NOT gates. You can force it into a particular state either by injecting current into one or other of the transistors, making it saturate or by grounding the collector of one or other transistor forcing the input of the other NOT gate to '0'.

A practical example of this circuit is found in the HP9100 calculator.

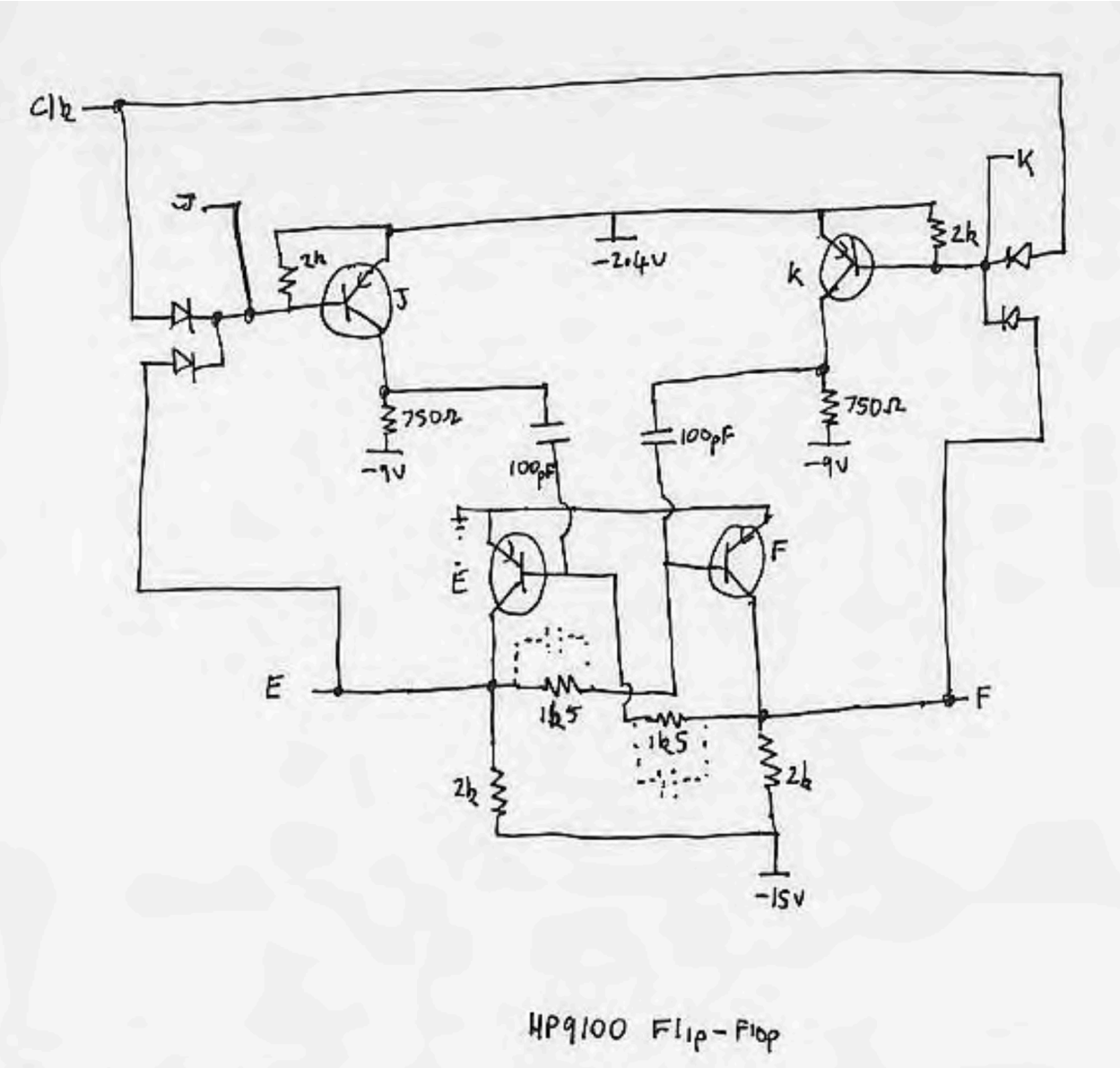
SLIDE 4



The first figure shows a single flip-flop. It is composed of 4 transistors, 4 diodes and 2 custom thick flim hybrid circuits.



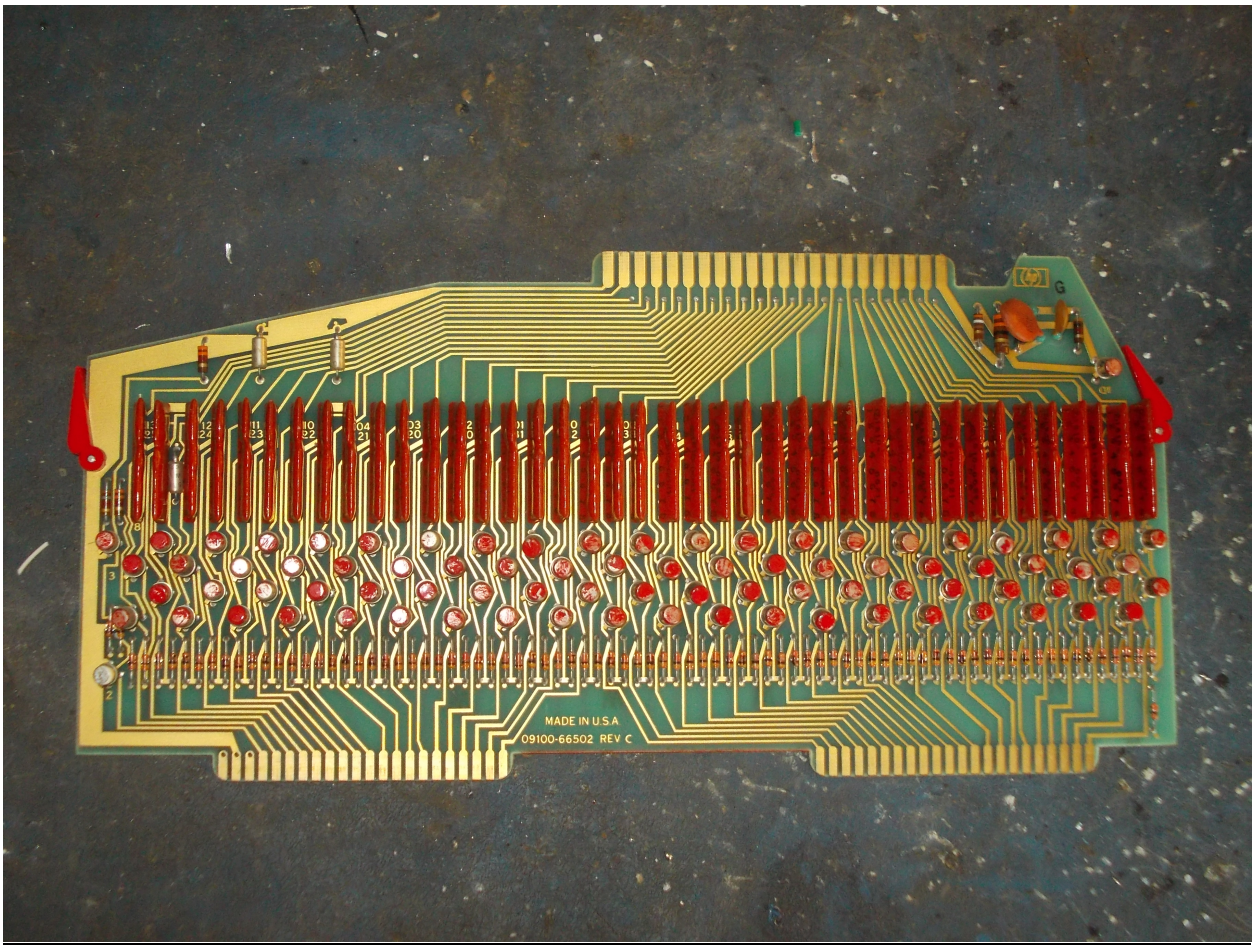
The next figure shows what's inside the hybrid, leading to the last figure showing the full circuit diagram.



It uses PNP transistors so the collector voltage is negative and everything appears upside down, but you can recognise the classic flip-flop circuit round transistors E and F. The other components are there to cause the flip-flop to be set or cleared with the correct combination of input signals.

And here is the problem. To store 1 bit takes all those components. In the HP9100, there are PCBs with 20 of those flip-flops and not much else on them.

SLIDE 5



But even an 8-digit calcuator takes 32 bits to store each of its numbers. So, the number of components needed gets excessive fast. As the filkbook goes on to say

> *'But for main store it cost too much*
>
> *So we tried CRTs, delay lines and such'*

I don't know of any calculator which uses a Williams tube -- a CRT with a metal plate over the screen -- as storage. But other methods were certainly used, and that's what I want to talk about.

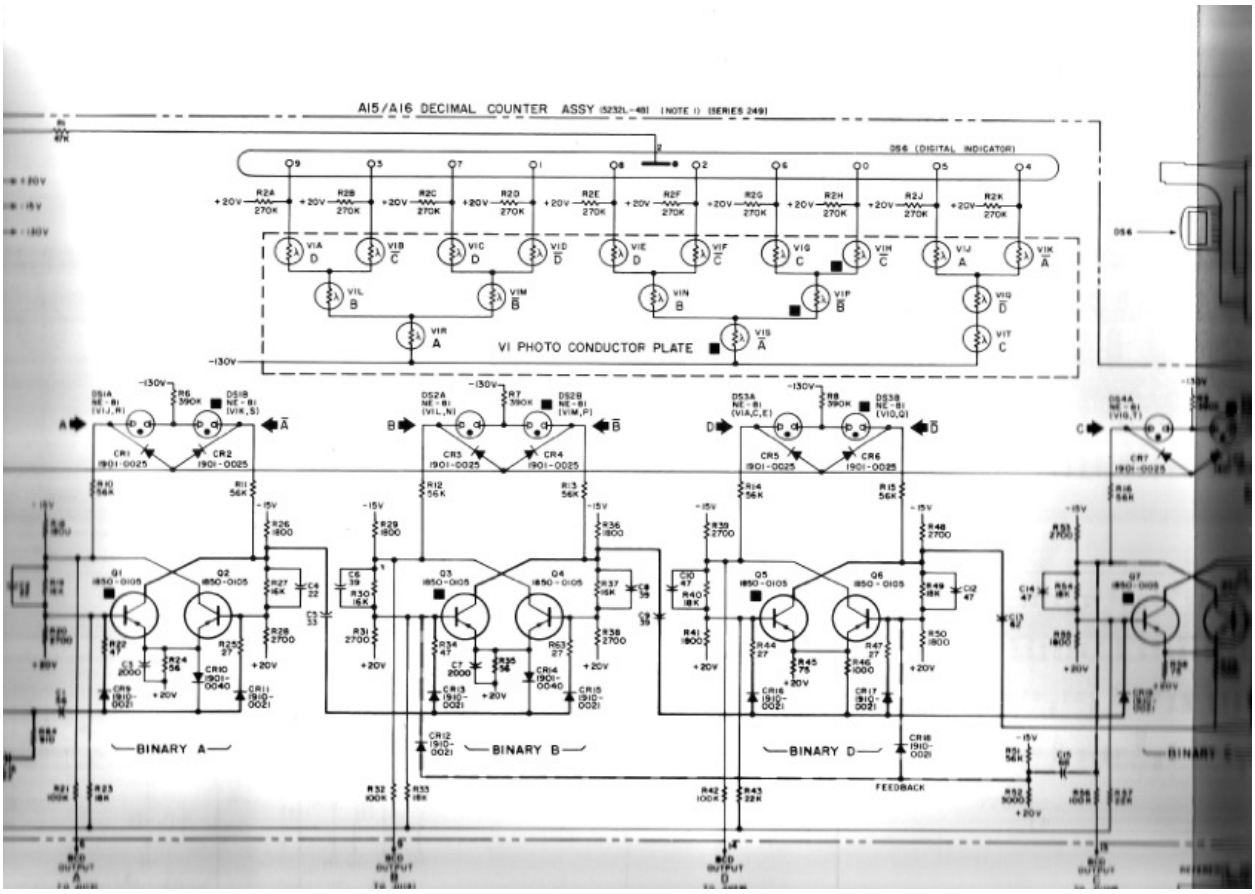
Trigger tubes

The next method is based upon a property of the gas discharge glow tube -- like a common neon bulb. It takes a higher voltage to strike (start) the glow discharge than to maintain it. Therefore, if you supply such a tube from a voltage between the maintaining and striking voltage, it can act as a 1 bit storage device. If the discharge is extinguished, then it will not start. But if the discharge is taking place, it will be maintained and carry on.

The discharge can be extinguished by reducing the applied voltage below that maintaining voltage limit. It can be started either by raising the voltage above the striking level or by using an auxiliary trigger electrode in the tube. A discharge started between the trigger electrode and the cathode will spread and form a discharge between the main anode and cathode.

I don't have an example of a calculator which uses this method of storage, although I believe one early Anita model used trigger tubes -- fancy glow tubes. However, I must show you an interesting storage example from an HP measuring instrument, the 5245L counter.

SLIDE 6



This shows part of one of the decade counter/display boards in the instrument. There are some flip-flops which form the counter itself, but the interesting part is what is connected to the collector of the flip-flop transistors, that pair of neon bulbs and the associated components. As HP explain here, the transfer pulse causes the state of the counter to be transferred to the neon bulbs and latched there. Thus the state of the counter is held while the counter counts the input signal again. Being HP, the way that the neon bulbs are used is ingenious. They are not just the storage elements, as the next page shows, they are part of the display driver. The neons illuminate a thick-film circuit of cadmium sulphide LDRs. Depending on which neons are lit, a particular path through the decoder tree conducts and energises the appropriate cathode of the display tube.

Dynamic memory (capacitors)

Now let's consider yet another method of storing bits. If you ask a relative beginner in electronics how to store electricity, they will hopefully suggest a capacitor. Can you store data that way, having a capacitor charged for a '1' and discharged for a '0'?

Of course you can. This is commonly called dynamic memory, it has the disadvantage that the charge leaks away with time and thus the memory is not permanent, not even while the power is kept on. The way round that is to read out all the locations periodically and to restore -- or refresh -- the charge on those which read as '1's.

SLIDE 7

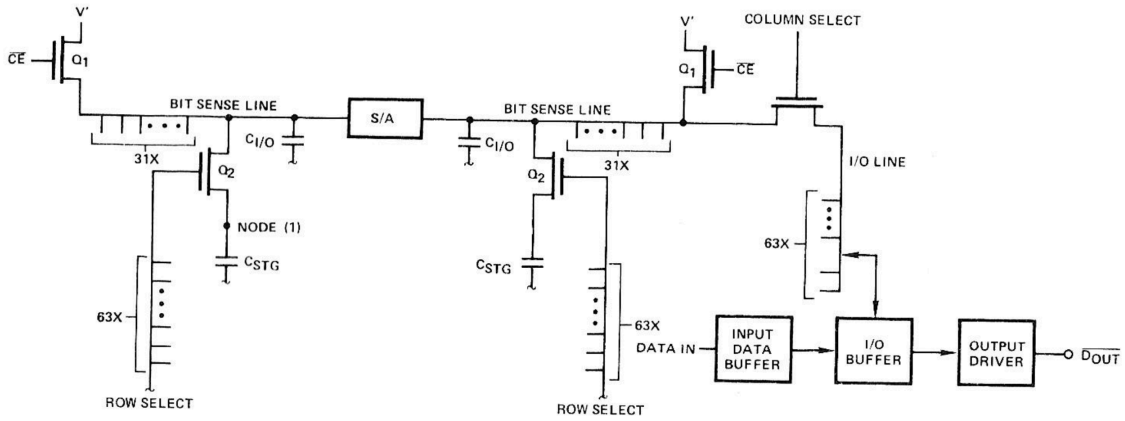
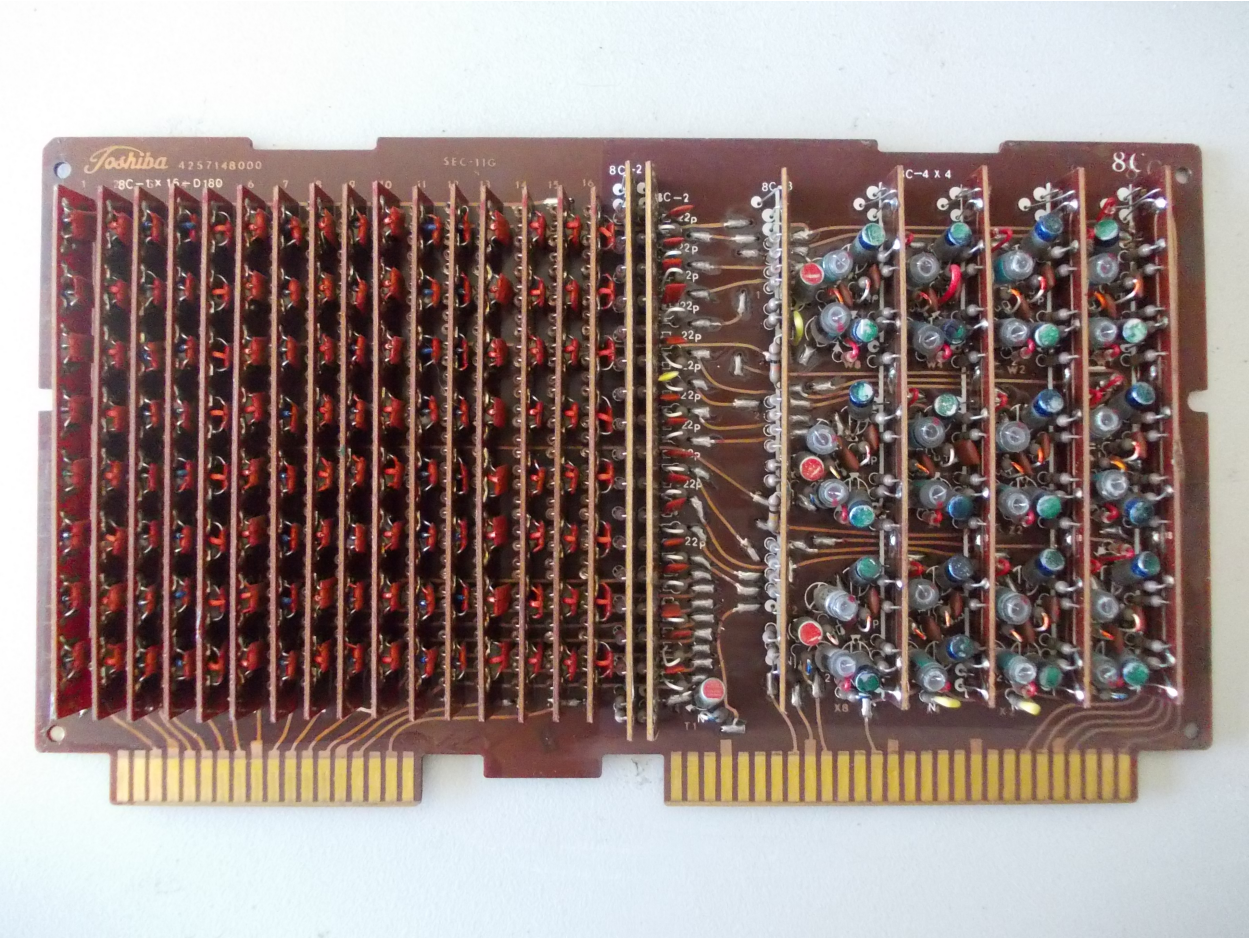


Figure 4. 2107B Memory Cell and Associated I/O Circuitry

This shows part of the circuitry of an Intel 4K-bit dynamic memory IC. The actual memory cell -- the circuit to store 1 bit -- is just a single mosfet switching transistor and a capacitor. The rest of the circuitry is common to many cells. As a result of the simplicity of the memory cell, dynamic memory has historically had a higher data storage capacity per chip than static memory (which uses the flip-flop circuits I talked about earlier). In 1970 Intel produced a 1kilobit dynamic RAM chip, the 1103, when static RAM was typically limited at 256bits on a single IC. The first major use of this device was the HP9810 calculator.

Can you make dynamic memory using discrete capacitors rather than ones formed on a silicon integrated circuit? Of course you can.

SLIDE 8



This shows the memory board from a Toshiba Toscal calculator. I have such a machine which will restore sometime, as I've not done so yet, I don't have the circuit diagrams. The left-hand half of that board consists of 16 identical modules, each stores 16 bits corresponding to the 4 bits for a particular digit in each of the 4 registers of the calculator

SLIDE 9

This is a close-up of the storage modules. Each of the browny-orange parts with 3 leads is a double capacitor which stores 2 bits. Refreshing the store is quite simple : In order to simplify the display circuitry, the display is scanned, the digits are turned on one after the other. As each display digit is selected, the appropriate location in the display register has to be read out to send its value to the display. At the same time that location is refreshed. And a little extra circuitry reads and refreshes the other registers too.

Delay line

The next storage method goes back to one of the first computers in England, the EDSAC. That stood for 'Electronic Delay Storage Electronic Calculator' and as the filkbook goes on to say :

> *'The main memory of EDSAC was, you see*
>
> *Acoustical pulses in mercury'*

The idea is to use a delay line, a device where a pulse fed into it comes out the other end a certain time later. What makes it useful for storage is that several pulses can be fed in one after another and they come out in the same sequence after the delay time. You capture the pulses coming out of the delay line, re-synchronise them to the processor's master clock, and send them back into the delay line. Of course you can ignore the output data to store new data in the delay line. So by putting in, say, a pulse for a '1' and no pulse for a '0' you can store binary data, which will then circulate through the delay line and associated circuitry until it is re-written with new data.

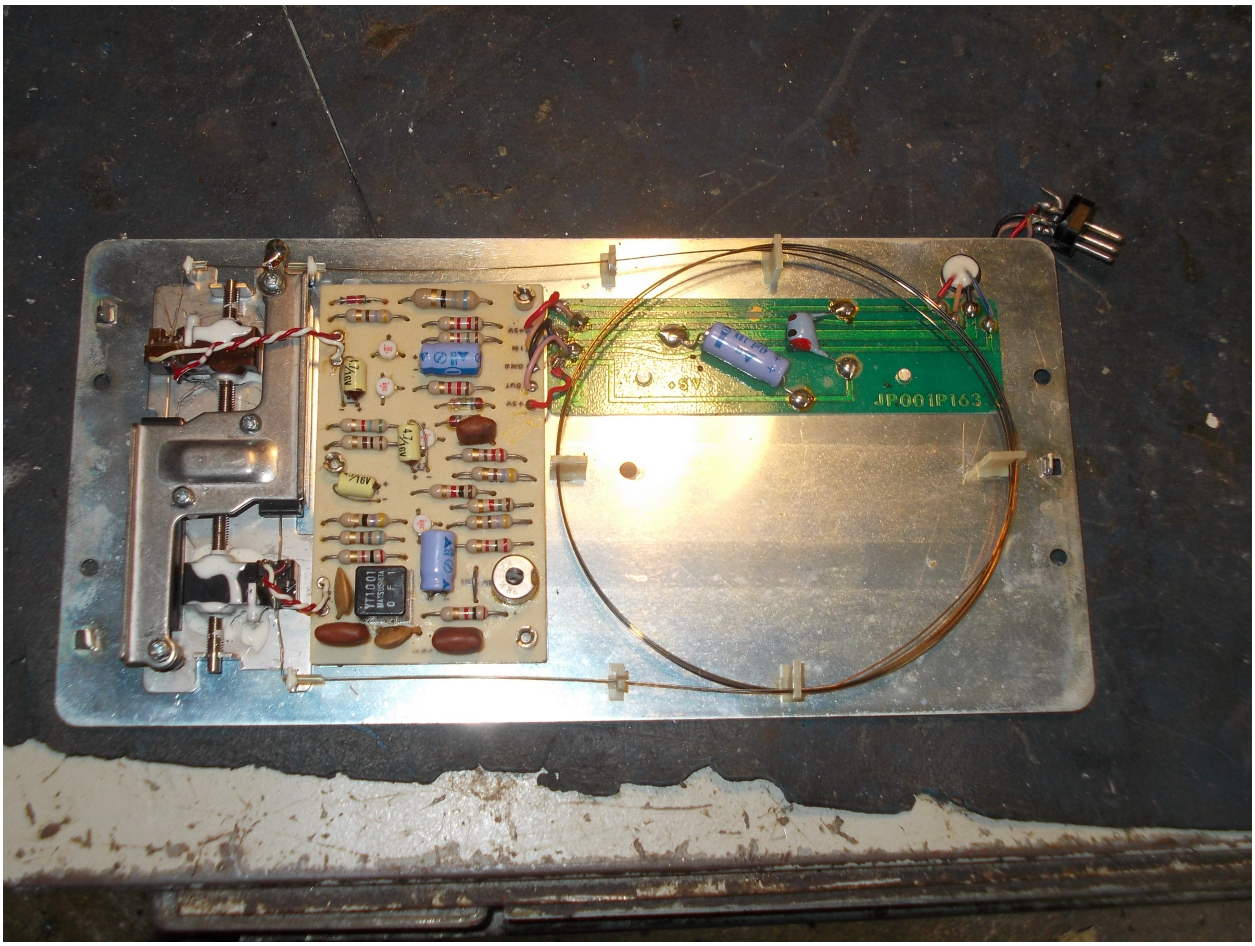
Obviously the storage depends on the time of the shortest pulse that it can reliably handle and

the total delay time. A device that can handle 1MHz pulses and delays for 500us (and I will show a picture of such a device in a few minute's time) will store 500 bits.

Obviously this is a sequential access store, you have to wait for the bits you want to come out, and for the correct time to write new data into the bit sequence. But for a calculator this is not a major disadvantage, many of the operations are sequential anyway. Addition starts by adding up the 2 least significant digits of the numbers, then the next 2 digits, and so on. So if the numbers are stored interleaved in the delay line, the machine can grab the 2 lowest digits (into 4-bit registers of flip-flops), add them and send the data back into the delay line. And then do the same with the next pair of digits.

EDSAC used tubes of liquid mercury as the delay line, with ultrasonic transducers at the ends. Fortunately, as far as I know, no calculator ever used mercury. But solid metal delay lines, typically using a nickel alloy wire were not uncommon.

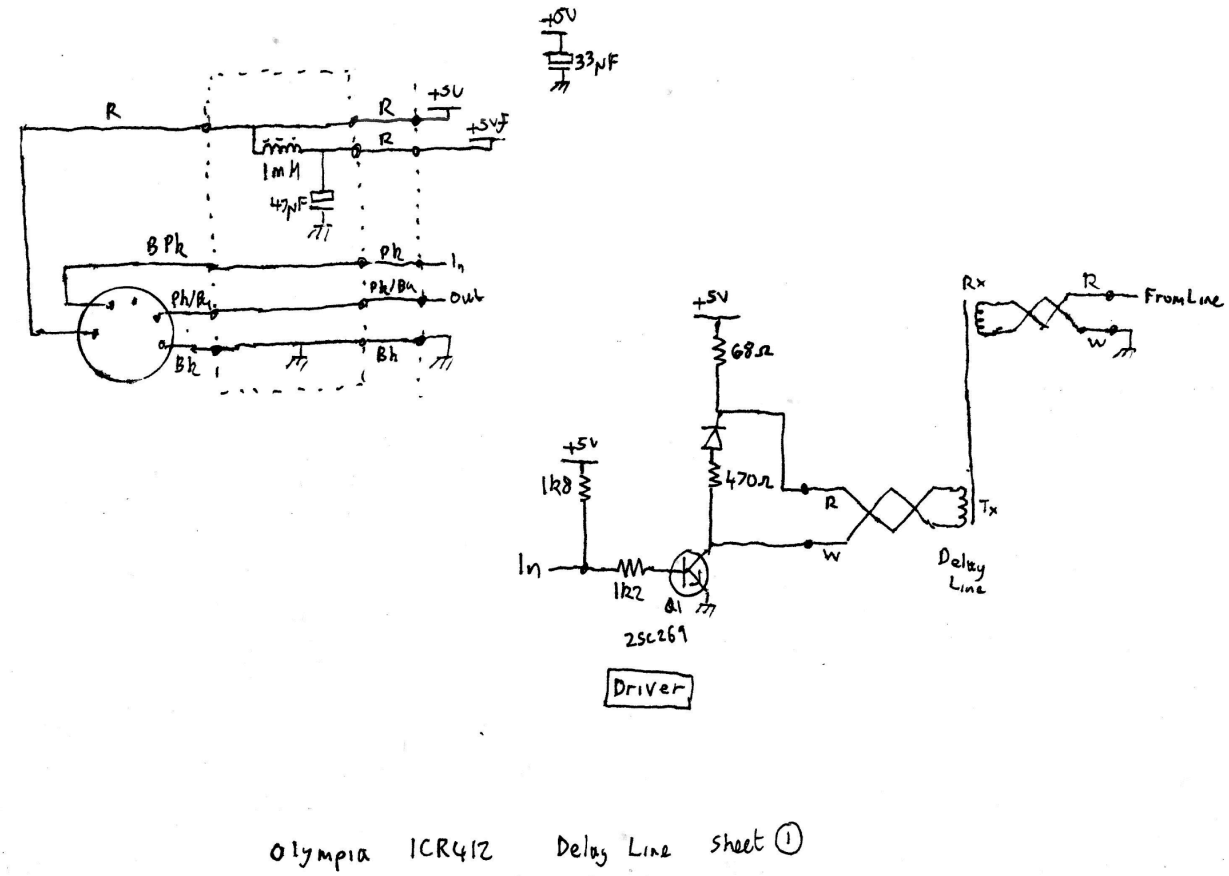
SLIDE 10



This is the delay line store from an Olympia ICR412 desktop calculator, a simple 4-function machine. It works at 1MHz and has a delay time of a little over 500us. The actual delay line is the coil of wire on the right. at the far left are the transducers to send an acoustic pulse into the

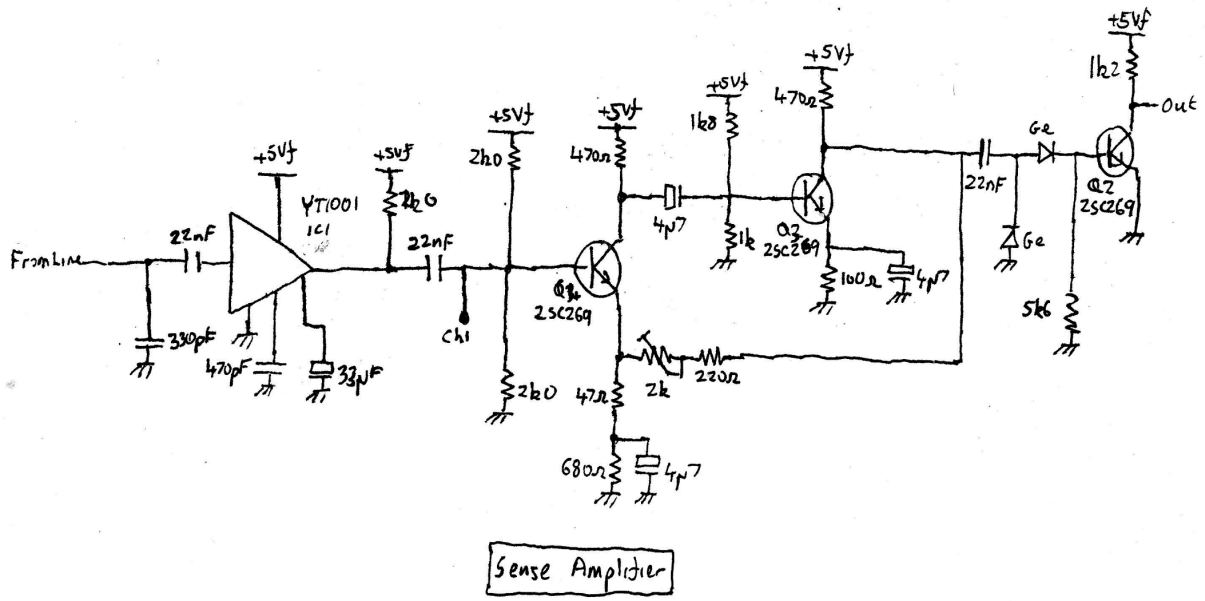
wire, it then propagates along the wire at the speed of sound and is detected by the other transducer.

SLIDE 11



Olympia ICR412 Delay Line Sheet ①

This shows the simple circuitry on the PCB in the delay line unit. Sheet 1 shows how an incoming pulse is fed to the driver transistor and then to the transmitting transducer.



Olympia ICR412 Delay Line sheet (2)

Sheet 2 shows the signal from the receiving transducer being amplified by a custom device and some more transistors to be fed back to the logic.

Magnetic Core

Finally, I am going to mention the storage method that almost everyone thinks of when discussing old computers and calculators -- magnetic core memory.

SLIDE 12

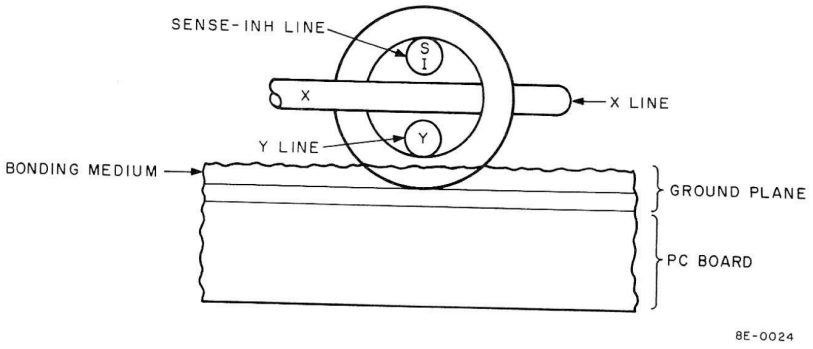


Figure 3-34 Magnetic Core

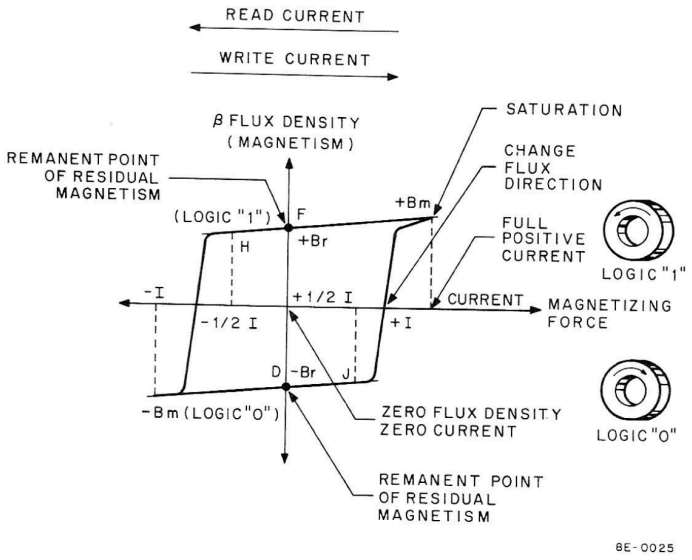


Figure 3-35 Magnetic Core Hysteresis Loop

The principle is shown by these diagrams from the DEC PDP8/e manual, back when computers came with useful manuals. Each bit is stored on a toroid made from a material with suitable magnetic properties. Threaded through the toroid are 3 (as shown here) or 4 wires. I'll explain the difference in a moment.

Two of the wires are the address lines and carry enough current that when the resulting magnetic fields coincide in a single core, the field is sufficient to 'flip' the magnetisation of the core from one state to the other, as shown in the hysteresis loop. However, the field from one wire has no effect. This allows for an X-Y addressing scheme of a matrix of course. And of course, the currents are sent in one direction

to magnetise the core one way -- say to write a '0' and in the other direction to magnetise the core the other way to write a '1'.

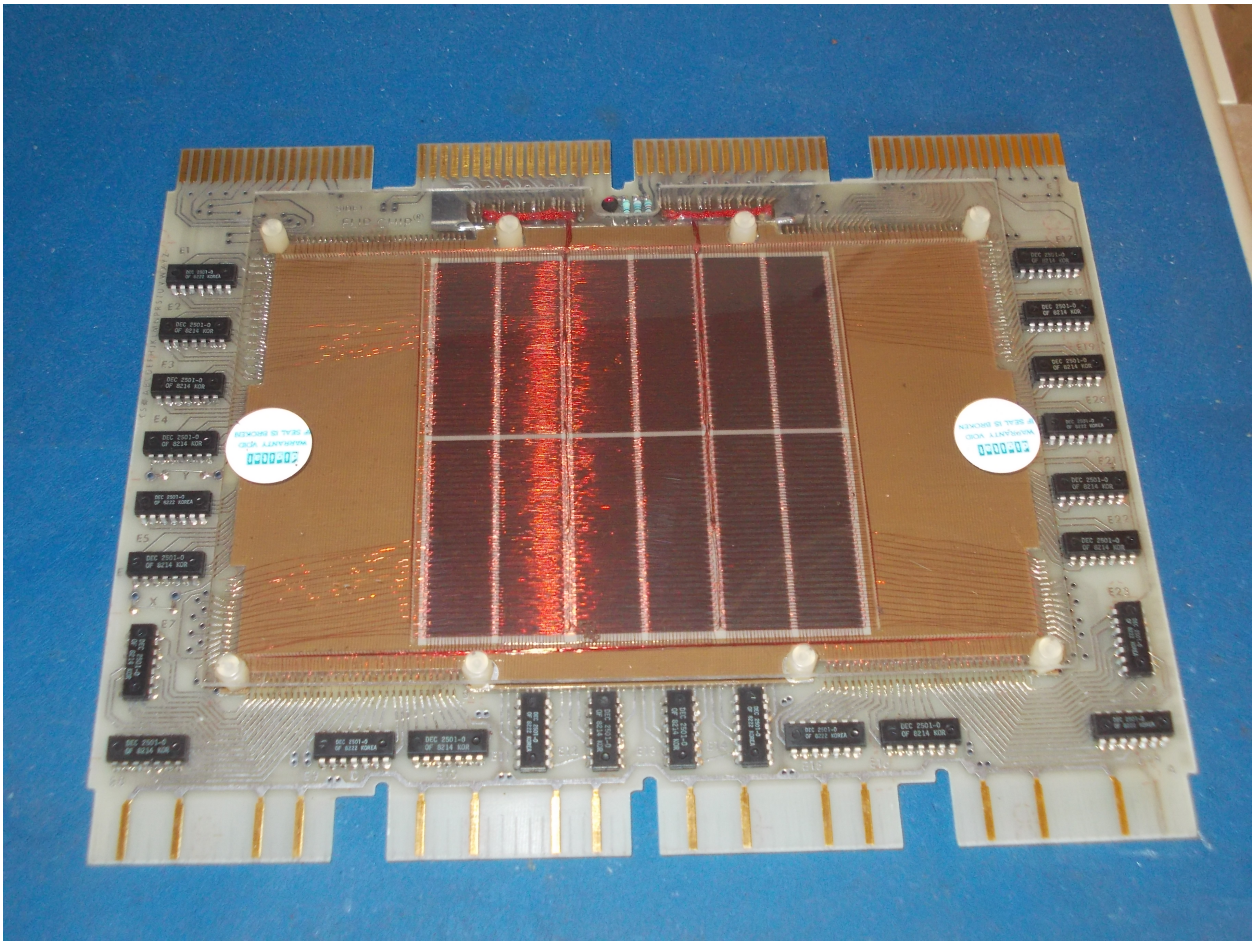
Now we come to the third wire. The only way to determine the state of the core is to write a '0' to it and see if the magnetic flux in it changes. If it was magnetised as a '1' then it will, whereas if it was magnetised as a '0' it won't. The third wire detects this change in flux. If the flux changes, a voltage is induced in that wire -- termed the 'sense wire' and this is detected by the sense amplifier. The address wires are common to all the bits of a particular word, whereas the sense wire loops through all the cores corresponding to the same bit in every word of the memory.

This third wire has another function. The PDP8/e memory cycle, which is typical of the core memory cycle in many machines, is in fact a read-modify-write cycle. The cycle starts by reading the addressed location, which of course clears it to 0. Then the address wire currents are reversed, writing '1's into each bit of the selected location. However, an inhibit current can be passed through the third wire, now termed the inhibit wire, to cancel out part of the magnetic field from the address wires, preventing the core's magnetisation from changing. Thus the bits of the word where the inhibit line is energised remain as '0's, those where the inhibit line is turned off are set to '1'.

And this of course explains why some core memory units have 4 wires. They have separate sense and inhibit wires rather than using the same wire for both functions. The electronic circuitry to operate a 4-wire core memory unit is simpler than that for a 3-wire one, at the cost of a more complex core plane.

The reason I have left core memory to the end is that, unlike the other memory systems I've described, it is not that difficult to find a full computer system using core memory. You would not manage to find a computer with delay line storage or discrete capacitor dynamic memory.

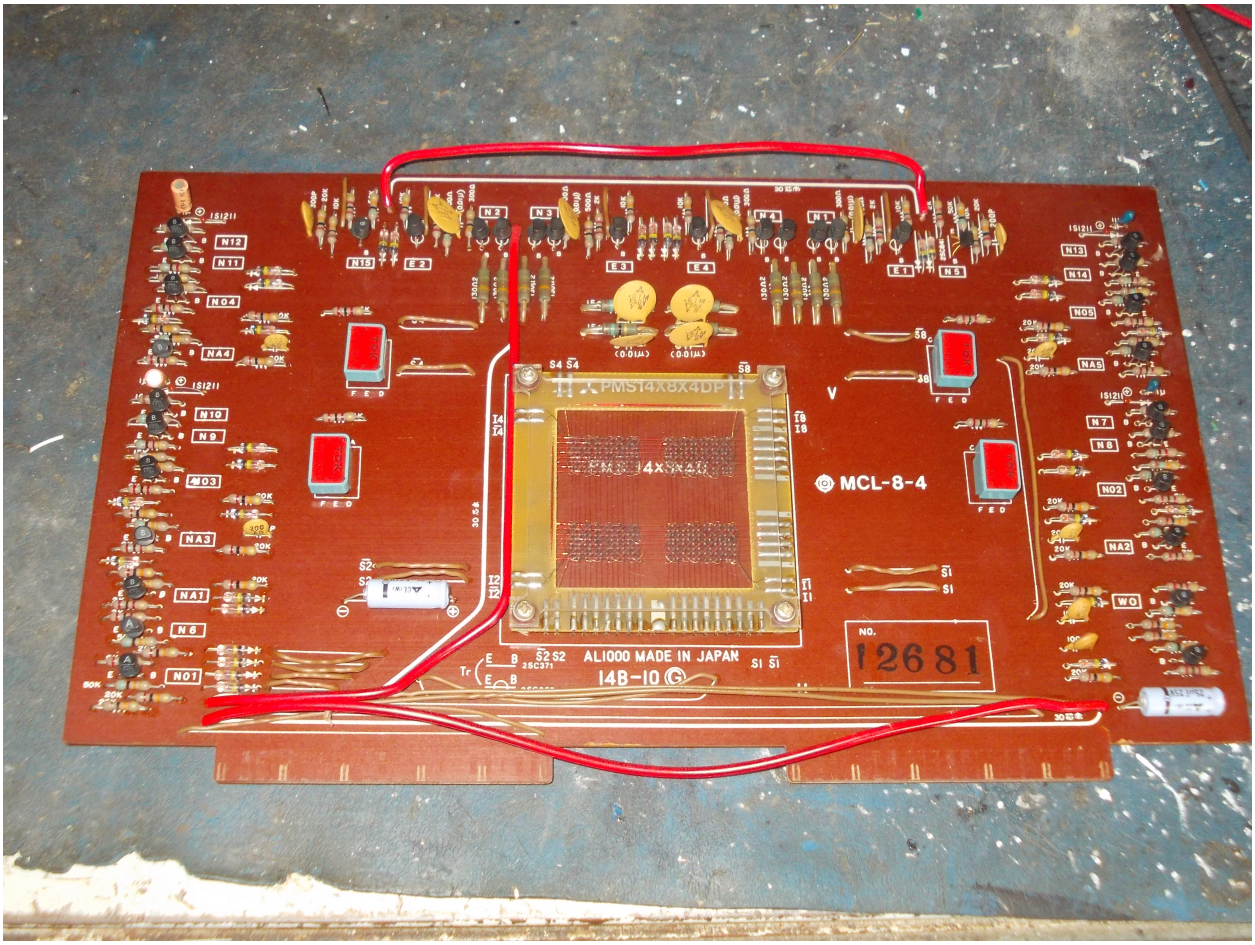
SLIDE 13



I own a PDP8/e computer with 32K words of magnetic core memory. This shows one of the 8K word core memory boards from that machine. Each of the black rectangles is an array of 8192 tiny magnetic cores threaded on the wires.

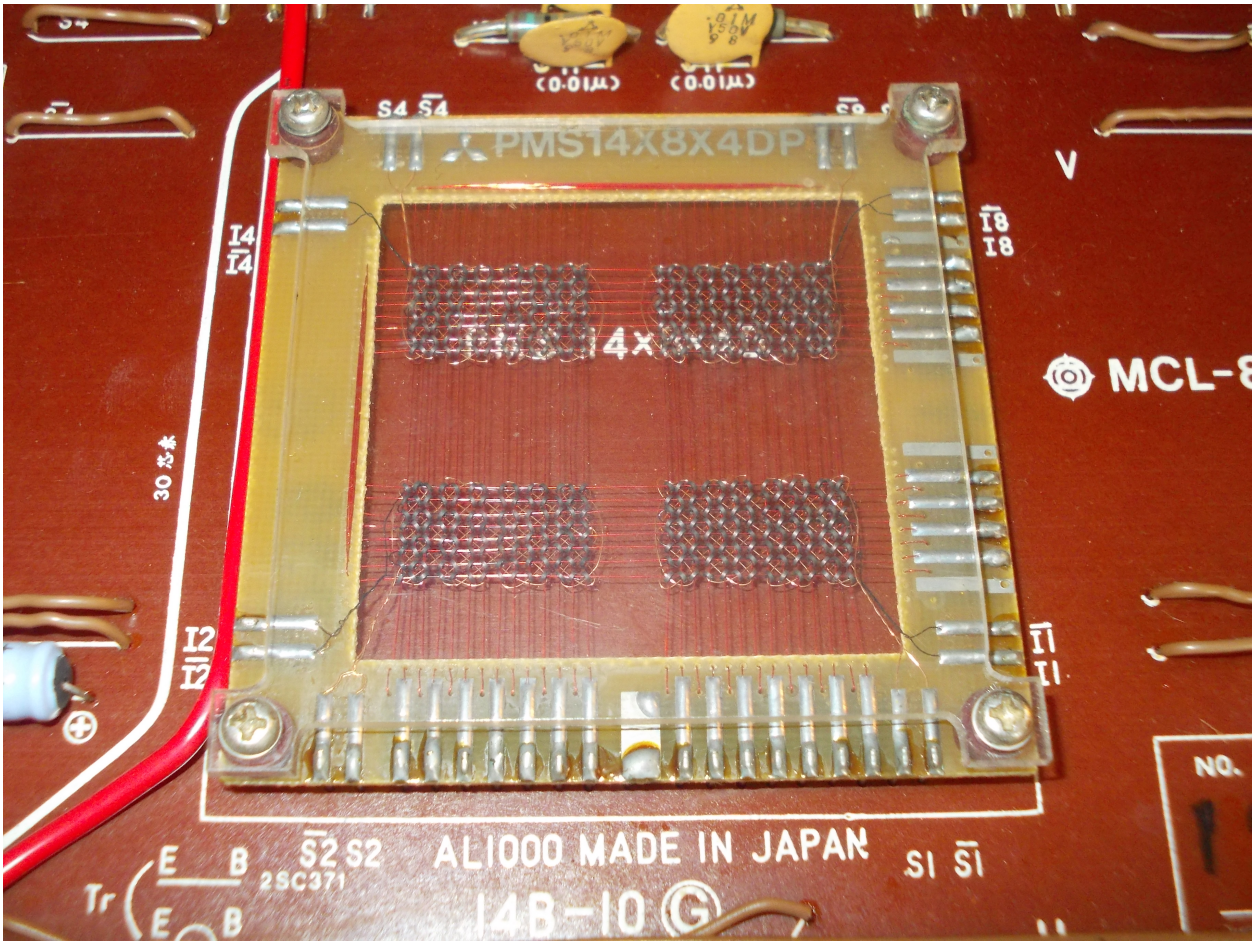
However, core memory was also used in calculators. The obvious HP example is the HP9100 family, but for a change, I'll go to a different manufacturer.

SLIDE 14



This is the memory PCB from the Casio AL1000 calculator, a simple programmable machine sold under the Commodore name in the States. The core memory itself is the square unit in the middle of the board.

SLIDE 15



And my last silde is a closeup of that core memory. As it's a much lower-density unit than the board from the PDP8/e, the individual cores are visible.

So, there you have it. Many methods of storing numbers were used in calculators before static RAM (flip-flops) and dynamic RAM (capacitors) ICs replaced them.