Paper Computers

by Louis H. Kauffman (1981)

I. Introduction

There is a well-known and fundamental connection between boolean algebra and switching circuits. First formalized by Claude Shannon in [ ], this connection has formed the basis for the development of computer circuitry, and it underlies many subsequent developments in programming, machine design and mathematical logic (see). The purpose of this paper is to put forth an elementary version of this connection that goes one step farther than the usual approach. I wish to show, using very simple mathematical models, how when boolean algebra "turns on itself", that is when it becomes self-referential, then memory and counting are born!

Last this last statement seem excessively esoteric, let's look at once at the case of memory: Imagine building a machine from basic elements that <u>invert a signal</u>. There are two signal types labelled 0 and 1 and satisfying:

$$00 = 0$$
$$01 = 10 = 0$$
$$11 = 1$$

Thus the 0 signal is dominant; $01 = 0$ denotes the fact that the 1 is drowned out by the prescence of the zero (0). We denote our inverting element by a diagram of form

$$\longrightarrow$$

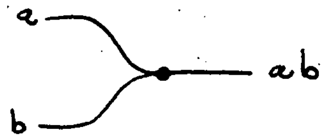
where the signals travel along the line in the direction of the arrowhead. Thus

$$0 \longrightarrow 1$$
$$1 \longrightarrow 0$$

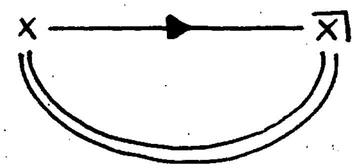
are indicators of the inversive process.

Let $\overline{a|}$ denote the result of inverting a. Thus $\overline{0|} = 1$ and $\overline{1|} = 0$. Then we can write $a \longrightarrow \overline{a|}$ to indicate the process of inverting a.

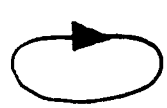
Since we also have a notion of signal combination ab, this also deserves a diagram



Now comes the self-reference! Consider the equation $(*)\ x = \overline{x|}$. *This is* the boolean analog of the *liar* paradox: "*This* statement is false." For if we assume that $x = 0$ then $(*) \Rightarrow x = 1$ and $x = 1 \Rightarrow x = 0$. *So it is neither*, or both! *Nonsense?* Well maybe, but let's diagram $(*)$:



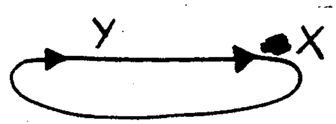
You see that $x = \overline{x|}$ says that the output of the inverter has been turned back and plugged into the input:



Here the self-reference has become a circularity and we suddenly get a new view of the paradox. For such a circuit *will simply oscillate!* ...0101010101...

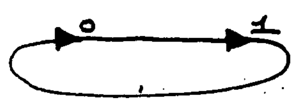
That is, if you actually make such a device then there will be a little time delay as the signal passes through the inverter. During this time interval the output remains 0, then flips to 1, then flips to 0, ... Incidentally there has been a bit of mathematical-linguistic sleight-of-hand here! The concept of time-delay is associated with physical inverting elements. By <u>interpreting</u> $X = \overline{X}\rceil$ in this frame-work the sense of paradox has shifted into the understanding of the possibility of temporal oscillation. This is quite legitimate, but you should watch very carefully people (like mathemagicians) who keep shifting their language base.

On to memory: Look at this circuit.

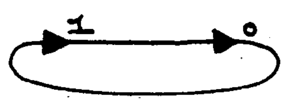


Here Y is the output of the left inverter, and X is the output of the right inverter. Thus $X = \overline{Y}\rceil$ and $Y = \overline{X}\rceil$. Circularity again.

But now there is no oscillation. If $Y = 0$ then $X = 1$ (and $X = 1 \Rightarrow Y = 0$). On the other hand, if $Y = 1$ then $X = 0$. The circuit has two <u>stable</u> <u>states</u>:

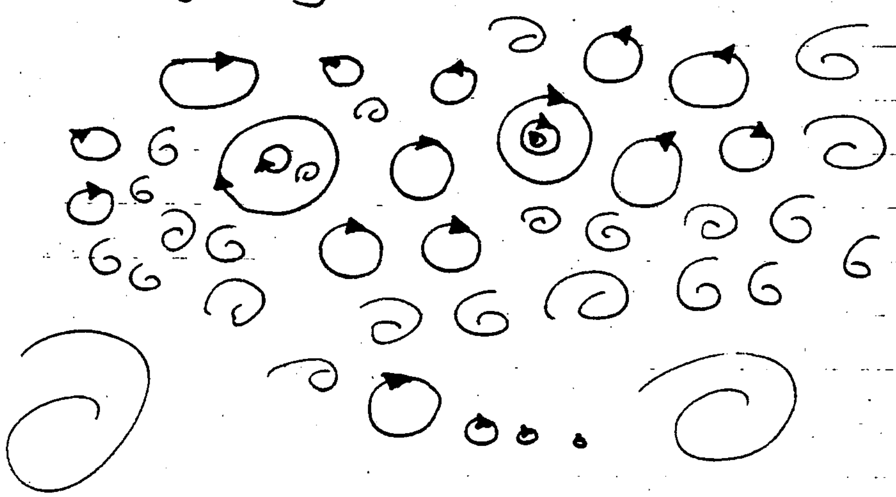


and

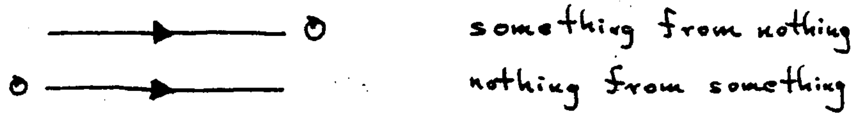


This is an entirely new and unprecedented phenomenon. By allowing apparently paradoxical circularity in our descriptions, we have produced the design of simple machines that can maintain themselves in more than one state of existence. For these stable states have nothing to do with time delays. The left inverter steadily gives its 0 signal to the right inverter who, in turn, steadily provides a 1 to the left. They hold each other up in the mutual embrace that is the stable state. Engineers call this the steady state behaviour. It is adequately modelled, as we have done, by asking that the graph of inverters and lines be labelled so that each inverter is balanced $0 \to 1$ or $1 \to 0$.

But how did the stable state arise? Well first of all the two inverters had to get together. This is a matter of conjecture and mythology, but I sometimes imagine the following scenario: Long ago and far away there was a primeval soup of curled up single self-referential inverters, each oscillating away, ignoring all the others. (Perhaps they were a by-product of the big-bang.)

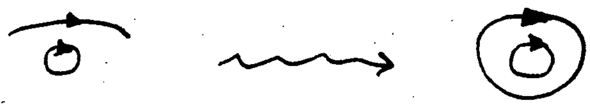


These were very primitive times indeed and in fact the only "things" around that could be used as signals were the little curls themselves! And there weren't any 1's. But that was allright, since an inverter would just merrily produce a zero (0 (↻)) if nothing came to its input and it would <u>stop</u> if something came along:



If it weren't for that sense of direction given to each inverter, this might seem a bit confusing. But that was the legacy of the big bang.

Oh! I didn't tell you how this works! It went this way. In order for → to act on ↻ and cancel it. The inverter would curl around its victem and engulf it right properly:



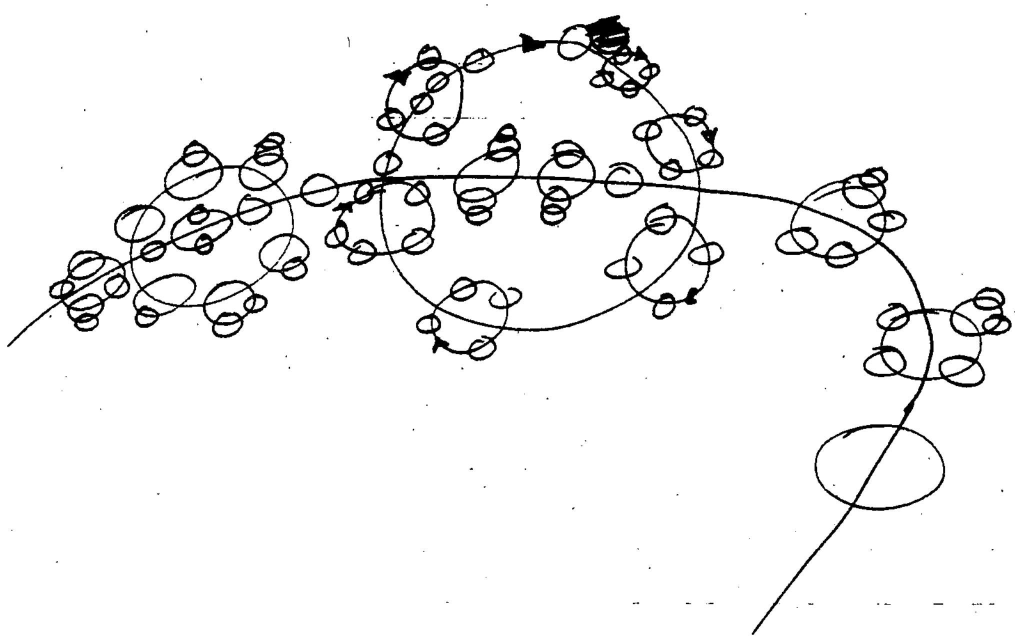
Then it would move right down and superimpose itself on the unsuspecting curl

and due to a very ancient universal law; (To the effect that you can't have two different things in the same place (cf W. Pauli   )) the two curls would just vanish!

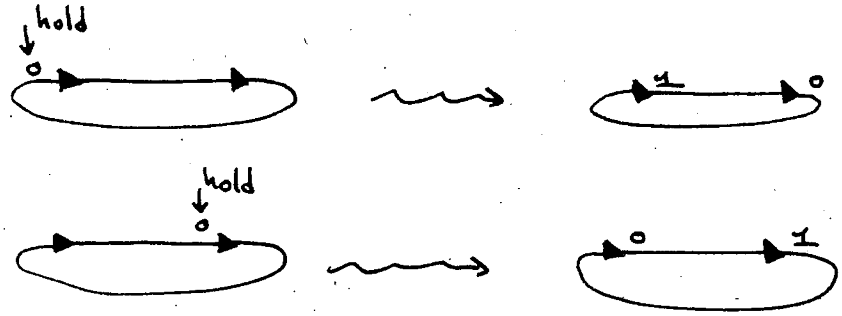
Getting something from nothing is a bit harder, but I'm told that it was entirely due to the inverters' tendency to curl on themselves.

Ah but I digress. Back to memory. Consider the following scene. Two curls find themselves in close proximity (the initial high temperature of the big bang has cooled considerably). They meet, touch along part of their self-referential boundaries. These cancel Pauli-wise and lo! an interlocked pair is born.

[This is only the very beginning of a long and complex history. Unfortunately, due to enormous cancellations, there are very few further references. I have it on good authority however that the big-bang itself was the result of a <u>single</u> self-reference. In those ancient epicyclic days of wheels on wheels on wheels, there was much to be grateful for in the <u>idemposition law</u> (as the Pauli Principle was later called [ ]) that allowed even a <u>little</u> cancellation.



Returning to the mathematics of memory, we can imagine the stable state being induced by <u>holding</u> one input at 0 until the memory flips to the appropriate state.



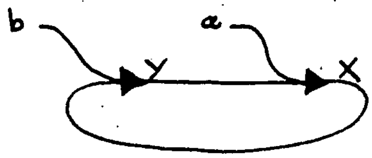
If the zero is held long enough then it gets through both inverters, completing the cycle by itself and the need for outside assistance is gone. The state is remembered. Since the circuit can be influenced into either of its two states, it becomes a repository for a single bit of information.

In section 2 we shall carry this story further and examine the design of circuits that can count and do arithmetic operations.

II. <u>Memory and Flip-Flops</u>

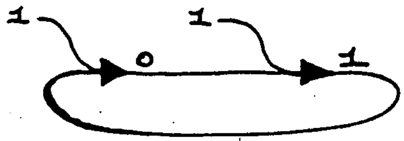
Lets examine the memory circuit $\mathcal{M}$ further. We allow inputs a and b so that

$\mathcal{M}$:

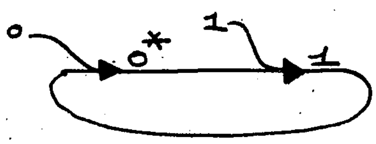


$X = \overline{aY}$, $Y = \overline{bX}$.

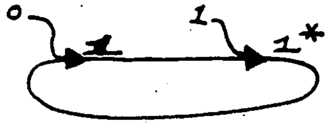
Suppose that $a = 1$, $b = 1$, $Y = 0$, $X = 1$.



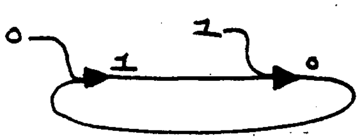
This is a stable state for $\mathcal{M}$. Now let b change to 0 (We write $a \rightsquigarrow 0$.)



The * indicates that the circuit is unbalanced at Y. Since b is being held at 0, $Y \rightsquigarrow 1$:



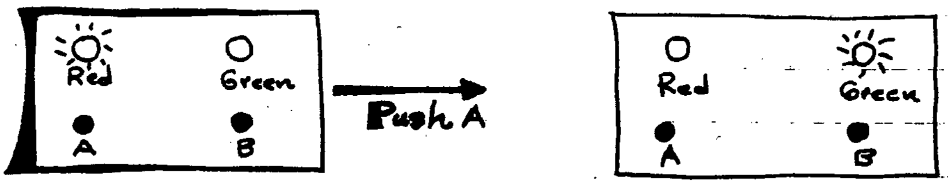
Now $X \rightsquigarrow 0$:



This is the new stable state.
Now <u>note</u> : <u>In this state the circuit is opaque to changes in the input b</u> . We must change a to cause another transition.

This is an important characteristic of this memory circuit. It has two inputs and two states. For each state only one input is sensitized. Flipping states also flips input sensitivity.

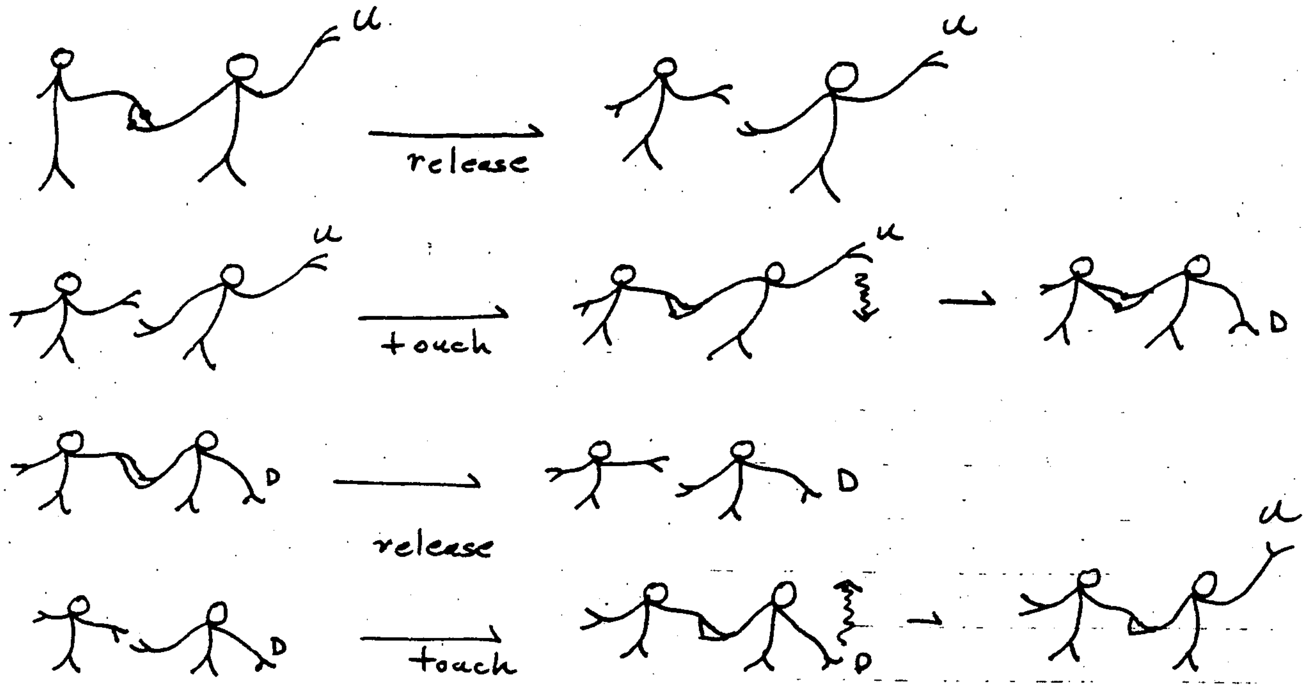
Lets put the situation in another frame. Our memory circuit is analogous to a box with two buttons (A and B) and two lights (Red and Green). At any given time



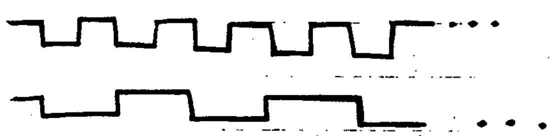
either Red or Green is on. If Red is on, then a push on A will switch the box to Green. Further pushing on A won't have any effect. To change back to Red, you must push the B button. A second push on B has no effect.

The remarkable fact is that we can use this very insensitivity of the memory to counting beyond two in the design of circuits that can actually count! The basic idea is a device called a <u>flip-flop</u> and the best description I know involves people. Here are the rules:

1) Your right arm is either <u>Up</u> or <u>Down</u>.
2) When your left palm is <u>touched</u> <u>change</u> the position of your right arm.
3) When your left palm is <u>released</u> <u>do not</u> <u>change</u> the position of your right arm.



release	u	R T R T R T R T...
touch	D	u D D u u D D u...
release	D	
touch	u	
release	u	
⋮		

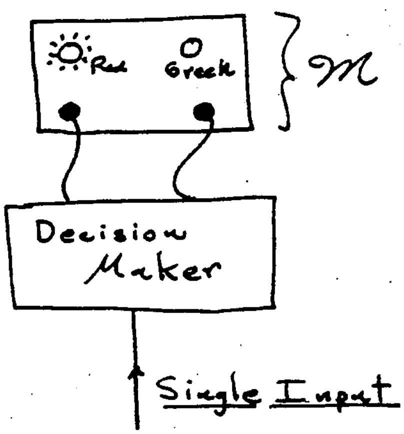


Note that under repeated release and touch (R and T) the up-down (u and D) behaviour occurs with half the frequency. This flip-flop <u>divides by two</u>.

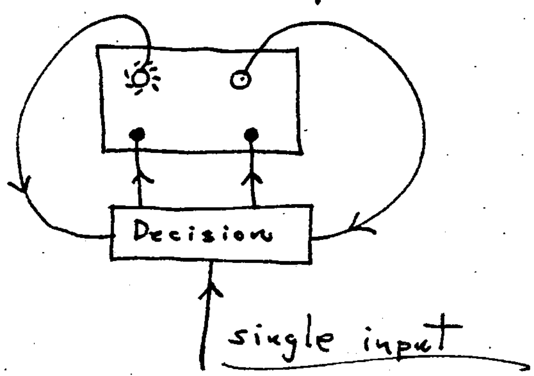
Now it is possible to use our memory box to produce this behaviour. Just <u>alternate</u> buttons!

Touch A	Green
Release A	Green
Touch B	Red
Release B	Red
Touch A	Green
Release A	Green
Touch B	Red
Release B	Red

Now this suggests that **if** we **could somehow** surround the memory with a decision apparatus that would alternately direct a signal to one side and then the other, we would be able to create a circuit with flip-flop behaviour.

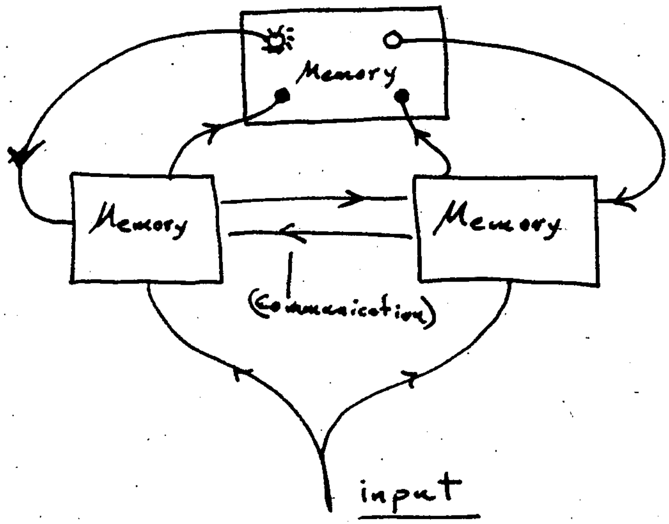


However, if this decision-maker does not make use of the information in M itself, then it is as good as a flip-flop by itself. Thus we should expect a more intimate connection between M and the signal switching apparatus. The picture would be more like

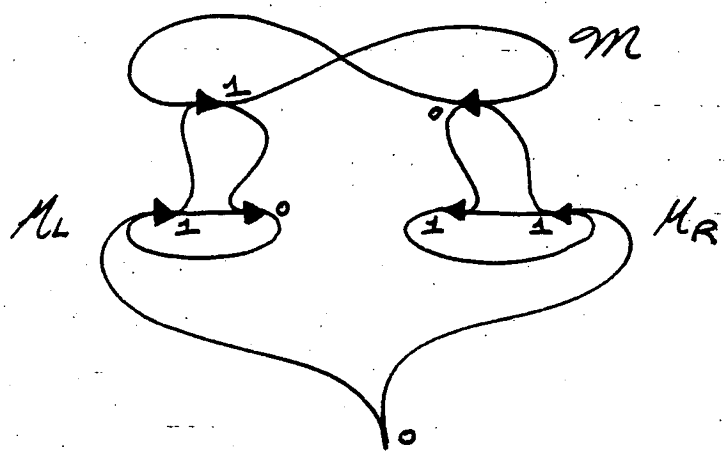


Now however we do design the decision maker it will be symmetrical with respect to each side of the memory.

For a given side there must be a part of the decision-maker that has <u>two</u> states, one that lets the signal pass, one that inhibits it. This suggests using another memory! And if we need one extra memory, then by the above reasoning we shall need <u>two</u> of them!

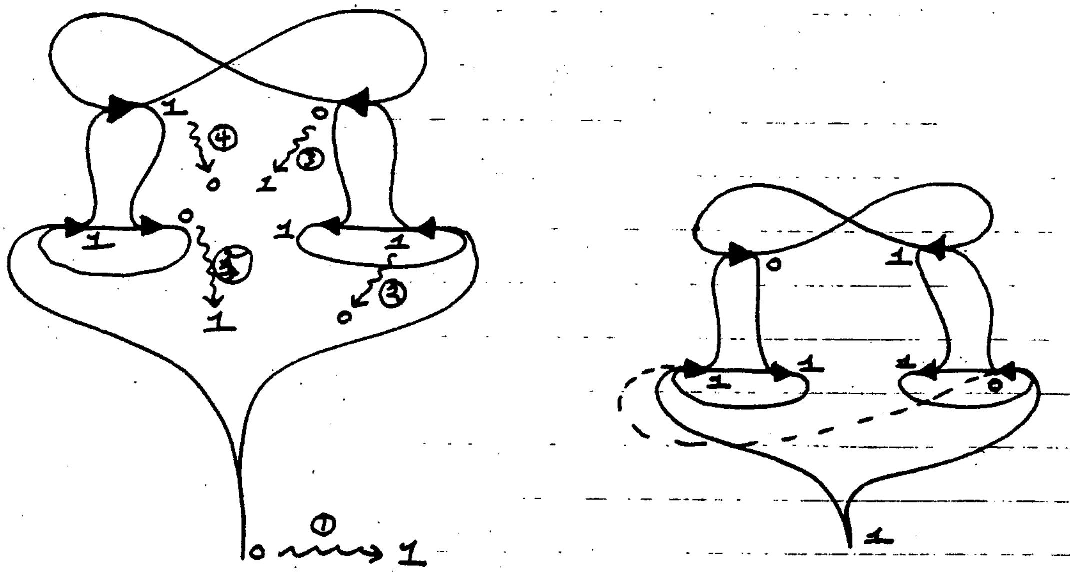


Thus now our projected design must appear as above. Presumably there may be need of communication lines between the two memories.



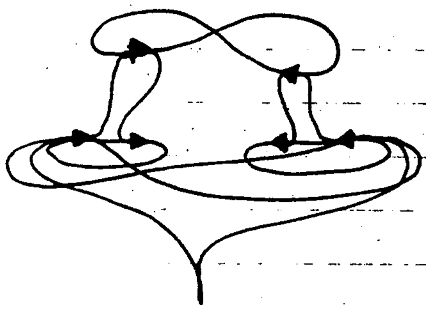
Here is a first-stage attempt to implement our ideas. The main memory is represented by the large ∞ form. The two auxiliary memories by ⬭ forms. In the state illustrated the left memory is opaque to a change 0 ⇝ 1, but the right memory will change (and note that this is due to the fact that right now both its sides are held at 1: one side by the input + one side by 𝔐.

Lets change the input 0 ⇝ 1 and watch what happens :

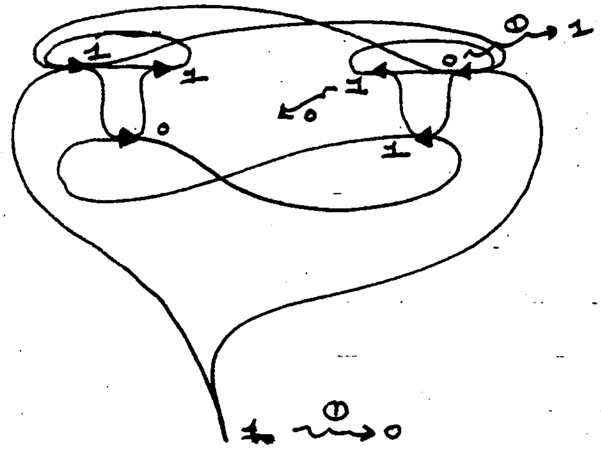


After these five changes the main memory has flipped over as desired, but the situation is still unstable. We would like a nice zero to hold M_L inplace! Well, a zero <u>is</u> available from one side of M_R!! See the dotted line in the accompanying diagrams.

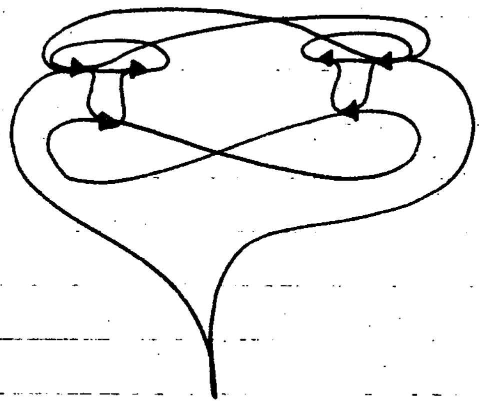
Thus the situation suggests that we add this communication line and (of course) its mirror image to find as next guess for the flip-flop :

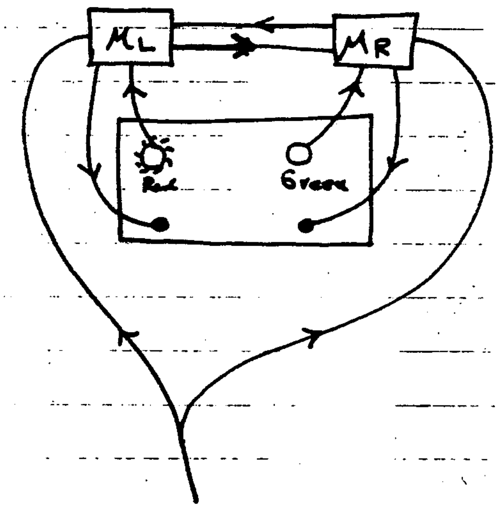


Here's a better drawing:



Here we have labeled the drawing with our last stable state. If the input 1 now changes 1 ⇝ 0, then only M_R changes as the transitions indicate. Hence by symmetry (this last state is the mirror image of the state in which we began) this circuit *is* a flip-flop.





It conforms precisely to our general design notion, and is remarkably simple in its operation.

Transitions