

# UC Berkeley Computer Science

**Subject:** A Six-Four GasP Tutorial  
**Date:** November 8, 2007  
**From:** Ivan Sutherland  
**UCIES** #2007-is49

## References:

UCIES #2007-is46 Infinity: A Proposed Test Chip, Ivan Sutherland, 18 April 2007  
UCIES #2007-is47: An Outline for the Review of Infinity, Ivan Sutherland, 13 May 2007  
UCIES #2007-is48: Results of a Design Review of Infinity, Ivan Sutherland, 17 May 2007  
web site: [research.cs.berkeley.edu/class/fleet/](http://research.cs.berkeley.edu/class/fleet/)

## PURPOSE

UCIES 2007-is48 presents the immediate results of the 15 May design review of Infinity. Over time, however, the review produced a collection of changes to the 6-4 GasP circuits that I plan to use in FLEET. This memo presents the revised ideas.

Three “tutorial” circuit diagrams appear with this memo, labeled A-C. There are also the circuits for some “real” GasP modules, PLAIN, BRANCH, and MERGE, and two simple combinations of them called RING5 and INFINITY5. Each such circuit tells something about our design style. This memo provides an English explanation of each of these figures.

## GENERAL

The circuit diagrams in this memo come from the Electric design system in use at Sun and elsewhere. In these circuit diagrams, purple logic gates and transistors are subject to automatic sizing, but red logic gates, red transistors, and yellow wire symbols are of fixed size. Although each yellow wire symbol includes a length and width in arbitrary “lambda” units, and a layer, I have used only 3 lambda wide wires in layer one.

The sizes shown are calculated by Electric in “X” values to represent the drive strength of each logic gate relative to that of a minimum-width inverter. In doing this calculation, Electric takes into account the load imposed on each gate including the self-loading of its own drains, the gate load of anything it drives, the capacitive load of the interconnect wire, and the Logical Effort of both the driving and the driven logic gate. A parameter of the calculation is the “step up” desired by the designer; that step up is sometimes called the “fan out.”

---

This document is a product of a collaboration between Sun Microsystems and the University of California at Berkeley. The ideas contained herein are freely available for any academic purpose.

In these figures I have chosen a step up of three and so Electric calculates the sizes required to make each and every purple logic gate do its task in the same time that an inverter would take to drive three copies of itself. At the upper left of the first three figures you will see strings of inverters driving various loads. Notice the top line, for example, where a purple inverter, whose size is calculated to be  $X = 50$ , drives the red inverter whose size is fixed at  $X = 150$ . I have included these strings of inverters to show how much drive is needed for some sample loads, including wire loads. The only property of wire that matters here is the capacitance it presents to its driver.

Electric's Logical Effort calculation provides a useful guide to the layout artist. In a corresponding layout the transistor widths should crudely approximate the calculated  $X$  values shown here. Because the delay in a string of gates is only slightly sensitive to changes in transistor widths, it is OK to treat the  $X$  values calculated by Electric as indications of relative width and not a requirement that must rigidly be met.

I have attached dummy loads to some circuits to indicate where I expect major loads. Driving many latches and the long wires required to reach them is a major task that requires wide transistors. In this memo I have used fixed-size inverters, colored red, to represent the load of groups of latches because using actual latches would introduce much unnecessary complexity. Other large loads are the long "state wires" that connect adjacent GasP modules.

Finally, as you will see in later figures, the Electric design tool represents single wires with narrow blue lines and collections of wires as wide green lines. Input and output terminal symbols are blunt red hollow arrows with appropriate names. Electric's naming convention permits indexing of names using a square bracket notation where `tom[1:15]` means fifteen wires with names `tom[1]` through `tom[15]`. Electric permits commas and letter indices, and so the expression `jim[A,B,4:6],mary[1:3]` means the eight wires `jim[A]`, `jim[B]`, `jim[4]`, `jim[5]`, `jim[6]`, `mary[1]`, `mary[2]`, `mary[3]`.

## **BASIC GasP – A-tutorial**

Figure A-tutorial shows a "typical" 6-4 GasP stage. The predecessor state wire, `pred[1]`, appears at the left of the figure; the successor state wire, `succ[1]`, appears at the right. Note that the LO condition of a state wire denotes EMPTY and the HI condition of a state wire denotes FULL. I only recently adopted the "HI is FULL" convention for FLEET; previous versions of this tutorial may have used the "LO is FULL" convention.

The FULL condition of the predecessor state wire and the EMPTY condition of the successor state wire combine to produce the "FIRE" signal, a "HI is ACTIVE" signal. FIRE does three things. First, it makes the latches transparent. Second, it renders the predecessor state wire EMPTY. Third, it renders the successor state wire FULL. These last two actions cause FIRE to go LO and returns the latches to their normally opaque condition.

Because all of the logic gates shown here are sized to have essentially the same delay, counting the number of logic gates approximates the timing of the circuit. If the predecessor is last to become FULL, FIRE occurs four gate delays later. If the successor is last to become EMPTY, FIRE occurs three gate delays later. From predecessor FULL to successor FULL takes six gate delays; from successor EMPTY to predecessor EMPTY takes four gate delays. Hence the name 6-4 GasP.

Because FIRE renders the predecessor EMPTY and the successor FULL, the FIRE signal destroys the very state that caused it. You can confirm that this destruction takes five gate delays by counting the number of gates in each of the two loops of the figure. Thus the FIRE signal will persist for 5 gate delays. In effect, 6-4 GasP circuits are ring oscillators of 5 gates each suitably coupled by AND functions.

My group at Sun has long had the notion of “kiting” data. Just as kiting a check means passing the check before the funds to cover it are actually in the bank, kiting data means announcing its imminent arrival before it actually arrives. Thus the FULL condition of the predecessor wire actually means “by the time you can act on this signal, there will be data available.”

Our 6-4 GasP circuits use two levels of “kiting” for “address” and “data” values. There are 15 “address” bits, fourteen numeric ones and a “token” bit, T. In a later figure you will see them named “a[1:14,T]”. The FIRE signal itself renders the latches for these bits transparent. In addition there are 37 “data” bits, named d[1:37]. The latches for these bits become transparent two gate delays after the latches for the address bits. The two extra gate delays give time for an AND function and the additional amplification required to drive the larger number of data latches. The amplifiers that deliver control signals to these latches appear in the upper right of the figure.

It is possible to kite the address bits less than the data bits because there are fewer of them. It is desirable to kite the address bits less because, as I explain later on, the address bits sometimes steer signals to the state wires. To provide proper control, the address bits must be available early enough to interact with the control signals. On the other hand, because the data bits control nothing, but simply “go along for the ride,” they can be tardy, giving additional time to amplify the signal required to drive so many latches.

To compute the sizes of the gates in this circuit requires some dummy load on the predecessor and successor state wires. The 7000 lambda long wire shown at the bottom of the figure is this load. To simulate the total load as it would appear in a long string of identical circuits, the figure uses a simple circular connection. Connecting the wire load to both the predecessor and the successor wire also simulates the load of the adjacent stages. Later figures will show the impact of reduced loads.

## **KEEPERS – A-tutorial**

A six-four GasP module drives its predecessor state wire LO to indicate that the data presented by a predecessor have been accepted. It drives its successor state wire HI to indicate that it has, or very soon will have, data available for its successor. After each drive event, the module stops driving the state wire.

The GasP modules include “keepers” that retain the voltage on each state wire. These keepers appear in the figure as pairs of red series transistors driven by an inverter from the state wire. Each module is responsible for keeping its successor state wire LO and its predecessor state wire HI. Thus each state wire is kept in both states, one state from one end and the other state from the other end. I chose to split the keeper to permit each GasP module to shut off the keeper when the module drives the state wire to change its state.

## **DATA CLOCK GATING, THE “T” BIT – A-tutorial**

The switch fabric for FLEET can carry two kinds of messages, “tokens” and “data”. The distinction between tokens and data involves a form of clock gating illustrated in the figure. To save energy, one address bit, called T, controls the latch pulse that drives the 37 data latches. The NAND gate that appears near the top of this figure omits the latch pulse for the data bits if the T bit is ZERO.

Layouts for the data path locate the T bit near the data bits, but nevertheless, the T bit shares the timing of the address bits. Because the data path for GasP must be faster than the GasP control modules, data waits at the input of the latches for them to become transparent. The FIRE signal itself makes the address latches transparent to capture the address value, including the T bit, from the previous stage. That same T bit from the previous stage conditions the NAND gate to avoid unnecessarily making the data latches transparent. The T bit is used twice in this stage at the same time, once as input to the T latch and once to condition the NAND gate. If it is present in time for one use, it will likewise be present for the other use.

## **A BRANCH UNIT – B-tutorial**

Driving each of the two successor wires in this figure is a P-transistor called a “fill” driver. A NAND gate just prior to the fill driver can make the fill function conditional on an address bit. B-tutorial shows a module with two successor state wires, succA[1] and succB[1]. When the module fires, one or the other of the two fill drivers will render its state wire FULL, i.e. HI. The upper state wire, succA[1], will fill only if the input called “direction” is LO. The lower state wire, succB[1] will fill only if the direction input is HI.

Note that both successor state wires must be empty, i.e. LO, before the module can fire. Thus both state wires enter into the AND function at the center of the figure.

The branch module sends the data and address bits it captured to both of its successor modules. Although the same data and address bits go to the latches of both successors, different state wires serve each successor. Only one successor state wire is rendered FULL, and so only one successor module takes the values.

The most critical timing issue in 6-4 GasP is the relative timing of the direction signal from a predecessor stage and the FIRE signal of the branch stage. The direction signal must be available at the NAND gates soon enough to select properly which successor state wire to fill. Moreover, the direction signal and its complement must remain stable long enough to fill the successor state wire completely.

The address bits provide the direction signal because of their lesser kiting. Note that it is the input address bits that provide direction control rather than the bits captured in this stage by the fire signal. The lesser kiting of the address bits brings the direction signal in soon enough. It will be interesting to see if this works out in practice.

### **A MERGE UNIT – C-tutorial**

This figure shows an arbitrated merge that will accept inputs from two places on demand. An arbiter in the merge unit renders the two FIRE signals at the top and bottom of the figure mutually exclusive. The arbiter appears as the cross-connected NAND gates near the center of the figure. Only one of these inverters at a time can produce a LO output. Thus the two signals called “fire[A]” and “fire[B]” are likewise mutually exclusive. In the 6-4 GasP merge module illustrated here each fire signal renders its predecessor EMPTY, and either fire signal renders the successor FULL. As noted, I have omitted the anti-metastability circuit from the arbiter for simplicity.

There is a peculiar timing issue in the Merge unit. Let us suppose that two inputs arrive on the A and B predecessor wires in rapid sequence, A first. The arbiter grants the first request, and so the logic gates act in the sequence A B C D (E&F) (G&A) H, where letters in parenthesis act at about the same time. Notice that gate J gets the disable signal from G barely one gate delay before the enable signal from H. This may be an unfortunate race condition. Only careful analysis of timing will reveal whether this may cause undesirable actions.

Jo Ebergen and I found a way to make the circuit more robust. The LO-active AND function of gate J could easily include another input. Taking this third input from the opposite side’s fire signal, fire[A] for gate J and fire[B] for gate B, disables the undesirable actions. This extra input costs logical effort. Is the extra input necessary? Are there other changes that would be equally effective and less costly in logical effort?

### **LAYOUT CONSIDERATIONS and WIRE – A,B & C-tutorial**

In a 90 nanometer process, with latches in two rows, the GasP stages for Infinity are about 0.5 mm wide and less than 30 microns high. That means that the longest horizontal wires in them are “short” from the point of view of resistance, but

“long” from the point of view of capacitance. Such wires are long enough that driving their capacitance in addition to the latch load requires wide transistors, but the resistive delay in the wire is small enough to ignore.

There are three long wires of interest. These appear as the yellow lozenges in the circuit diagram with values 5000, 6000, and 7000. Although the address latches driven by the FIRE signal are located near the GasP logic part of the stage, the T bit is far away. This leads to the 5000 lambda estimate of wire length.

The data drive amplifier, which follows the NAND gate, must drive a total of 37 latches. Although the driver is in the center of the latches, and the control wire is shared as much as possible, the estimate of 6000 lambda remains appropriate. The transistor width of the latch driver is heavily influenced by the lengths of wire that it drive. Indeed, nearly half of the drive current from the latch driver,  $X = 98.889$ , goes to charging and discharging wire.

The third long wire of interest is the state wire itself. Some stages may be close to their successor stages, but some may be far separated. This third long wire appears as a load of 7000 or more lambda in the figures. It will be important to know the impact of putting stages capable of driving such long wires closer together than expected.

### **LATCHES – D-register and E-register**

In order for later figures to show how these modules actually fit together, I need icons to represent the data latches. These two figures show what’s inside my register icons. Building up registers from a simple latch circuit involves more complexity than is appropriate for this tutorial. Instead, I show two dummy register circuits, one with a single data input and one with twin data inputs. You will recognize these circuits from the upper right corner of the figure called A-tutorial. You must provide your own notion of appropriate latch circuits; here I emphasize only the load they impose on their driver. Later on this memo seeks timing specifications for the latches.

### **REAL GasP CIRCUITS – F-gaspPlain, G-gaspBranch and H-gaspMerge**

Making assemblies of GasP modules requires icons to represent them. Moreover, the actual circuits omit parts unnecessary to function that appear in the tutorial diagrams. The result appears in the three GasP module circuits. The reader should be able to understand these circuits from the description of the tutorial examples above.

Please notice that the module circuits augment the FIRE signal by an export called fire, fire[A] or sometimes fire[B]. This export will connect the icon for the GasP circuit to the icon for the latches it controls. In the merge circuit, the two fire signals called fire[A] and fire[B] share a single export called fire[A,B].

In each of these circuits there is an isolated purple transistor that drives the predecessor or successor wire. Examination of the calculated “right” sizes of these transistors will reveal that they vary; some are larger than others. This is particularly evident in the figures F-gaspPlain BEFORE and F-gaspPlain AFTER. These differences are the result of calculating sizes for equal time delay in the face of different external loads on the modules. Later on I’ll say more about where these loads differ.

## **RING-5**

The ring5 circuit is merely five GasP modules and five registers connected in a ring. There is a ring like this, but with more stages, on the test chip. The ring on the test chip also includes stages capable of loading and reading data values, a stage with an arbiter that can stop the ring cleanly, and a counter to measure throughput.

To see how the separation of stages affects the calculated transistor widths for the GasP modules, the ring5 figure has only 3000 lambda of wire between two of its stages. The figures called “F-gaspPlain BEFORE” and “F-gaspPlain AFTER” show transistor widths calculated for these two GasP stages. Notice that the purple successor driver for the “BEFORE” stage and the purple predecessor driver for the “AFTER” stage are narrower than those shown in the earlier F-gaspPlain diagram. Notice that the X values of other logic gates also differ because they drive reduced load.

Although not shown here, the impact of the reduced load of 3000 lambda appears in Electric’s calculations for the “proper” X values in all of the stages in the ring. Even for stages far away from the reduced load the calculations permit slightly narrower transistors, although the effect diminishes with increasing logical distance from the anomaly. In practice this is merely an interesting numeric curiosity for the calculations are based on assumptions not nearly good enough to support meaning in such small differences.

I want a set of identical GasP stages that will work for a variety of loads. Naturally where loads are lighter, the modules may operate slightly faster. Nevertheless, because the control delays considered here and the data path delays of the latches both increase with greater separation of GasP modules, it should be possible to use a single design in a variety of situations. How tolerant will such modules be to variations in spacing?

## **INFINITY-5**

The final figure of this memo, called “infinity5” shows a pair of coupled rings. Each ring has 10 stages, but they share 5 stages in common. At the beginning of the shared section a “merge” element takes input from either branch on demand. At the end of the shared section, a “branch” element distributes data to the two branches. The test chip includes a similar structure, but with two rings of 100 stages that share 50 stages in common. The test chip also includes stages that can load and read data from the rings and stop the rings cleanly. It also provides counters to measure throughput.

Notice that one of the state wires in the upper ring has less load than the others. In fact, the size calculations for the merge stage shown earlier exhibit the impact of this reduced load. Notice that the two purple predecessor driver transistors in the merge stage differ slightly in width. This difference occurs because of the anomalous load in the upper ring. I believe the branch stage fails to exhibit a similar effect because it is too far removed from the anomaly.

## PROBLEMS

The X values for purple logic gates show the results of theoretical calculations based on logical effort and estimates of the capacitance of the wires they drive. In fact, the logical effort of real gates varies depending on how much their layout shares adjacent drains. Moreover, it's very hard to calculate exactly the impact of wire loads, because wire load depends on the details of what other structures are nearby. Thus the calculated widths shown are merely estimates.

In fact, I want a set of modules with fixed transistor widths that I can assemble into systems that will work. Thus I want to know what variations in transistor widths or wire length will cause failure and which ones don't matter. For example, all of the data drivers will be the same, at least to the precision of manufacture. Does their particular size matter? How sensitive is the design to variations in these drivers?

Moreover, in each of these modules, the predecessor and successor loops share the central three gates.<sup>1</sup> This leaves only, at most, two gates where differences in delay might matter. Is that enough sharing to ensure adequate timing margin?

The largest single source of variation is the separation between modules. The state wires, here estimated at about 7000 lambda, may vary from as little as one thousand to as much as ten thousand lambda. Is it necessary to adjust the sizes of the predecessor and successor drive transistors to match the actual lengths of the state wires? We had to do that for four-two GasP. I hope the more conservative six-four GasP will prove more robust and obviate the need to so carefully adjust transistor widths.

Finally, the latches must meet some requirements. What are those requirements? Can they be expressed in terms of "thru time", the delay from receipt of the transparency signal to change in data output? I expect that the latches must meet both a minimum and a maximum thru time.

In this memo I have no doubt missed some problems that I should have been able to identify. What are they, please?

---

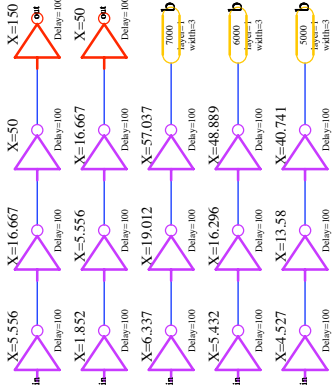
<sup>1</sup> The real circuits use a three-input NOR rather than inverters and a NAND, but a suitable symbol was not available for this tutorial.



# B-tutorial

## Branching GasP cell

ies 8 November 2007



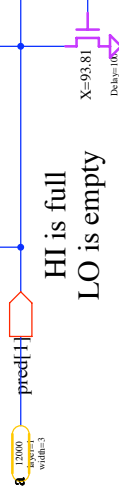
```

sw=3.7
max_in=0.22
max_out=0.100
max_jk=0.30
gas_cop=0.167
diffp=0.7
LESETTINVERSE=0.1
    
```

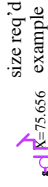
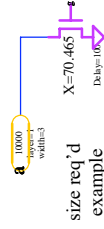


half keeper

bigger dummy load  
because not connected  
to successor

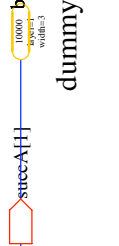


HI is full  
LO is empty



HI is full  
LO is empty

dummy load



half keeper

direction

dummy load



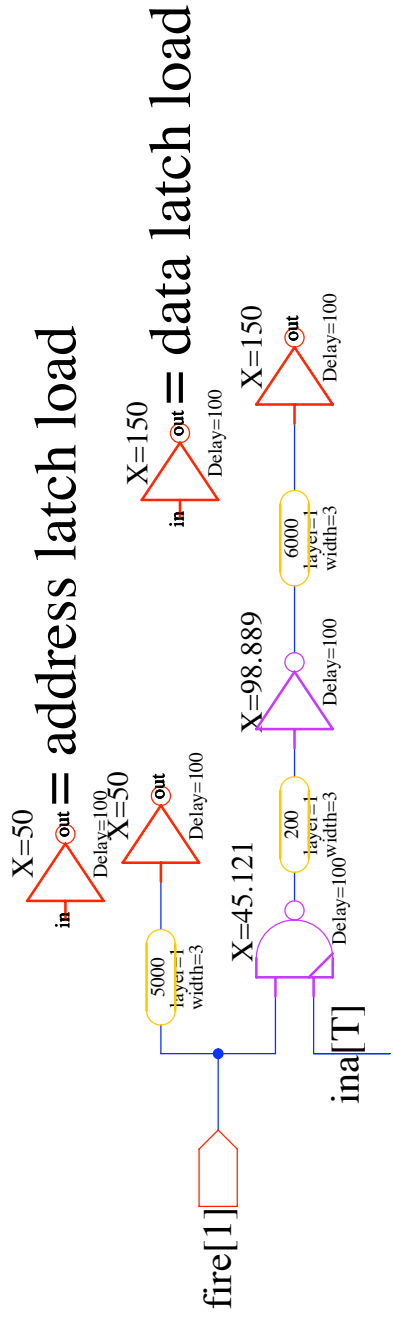
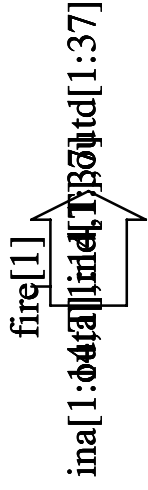
half keeper



# D-registerOne

this is a dummy one-input register and driver

ies 3 November 2007



ina[1:14,T],ind[1:37]

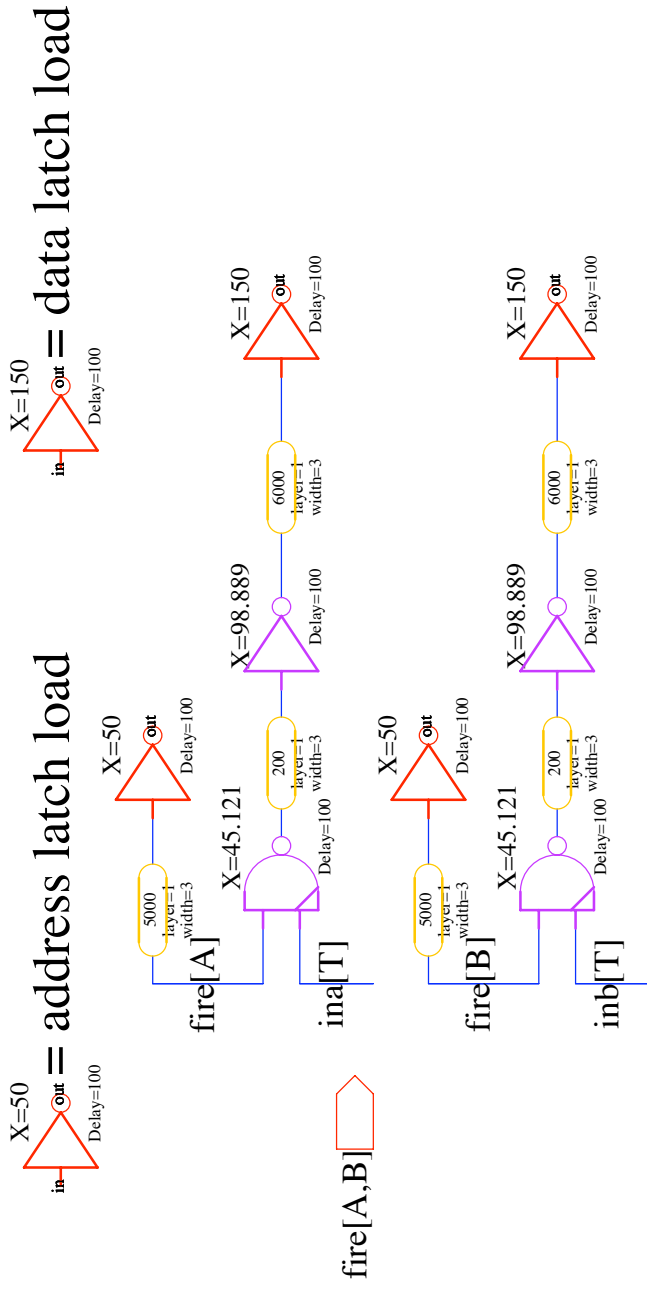
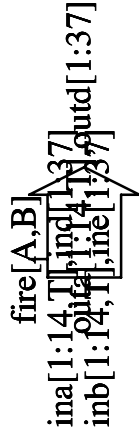
15+37=52 latches here

outa[1:14,T],outd[1:37]

# E-registerTwo

this is a dummy two-input register and driver

ies 3 November 2007

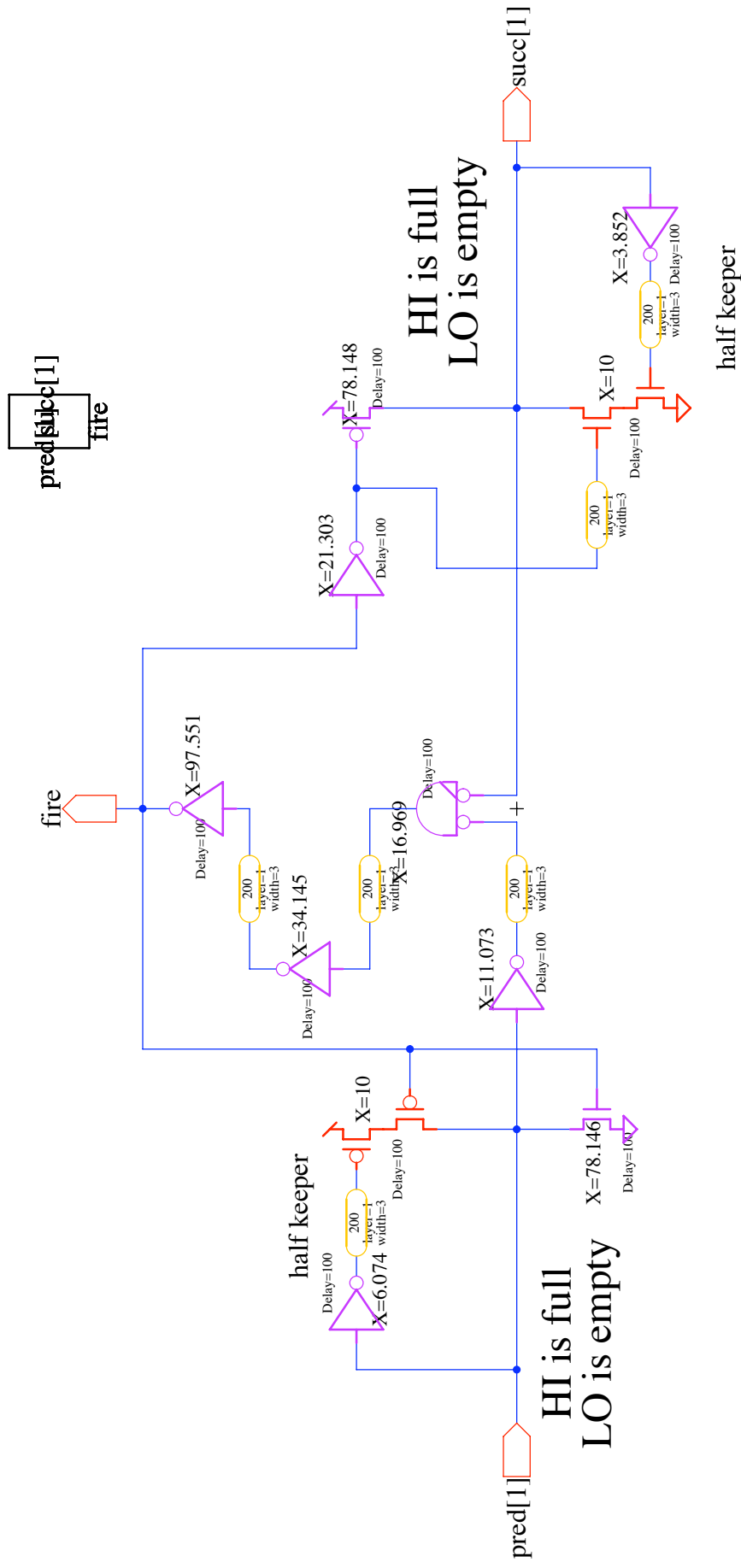


$$\text{ina}[1:14, T], \text{ind}[1:37] \text{ (latch)} + \text{inb}[1:14, T], \text{ine}[1:37] \text{ (latch)} = \text{outa}[1:14, T], \text{outd}[1:37] \text{ (latch)}$$

# F-gaspPlain

## Basic GasP cell

ies 8 November 2007

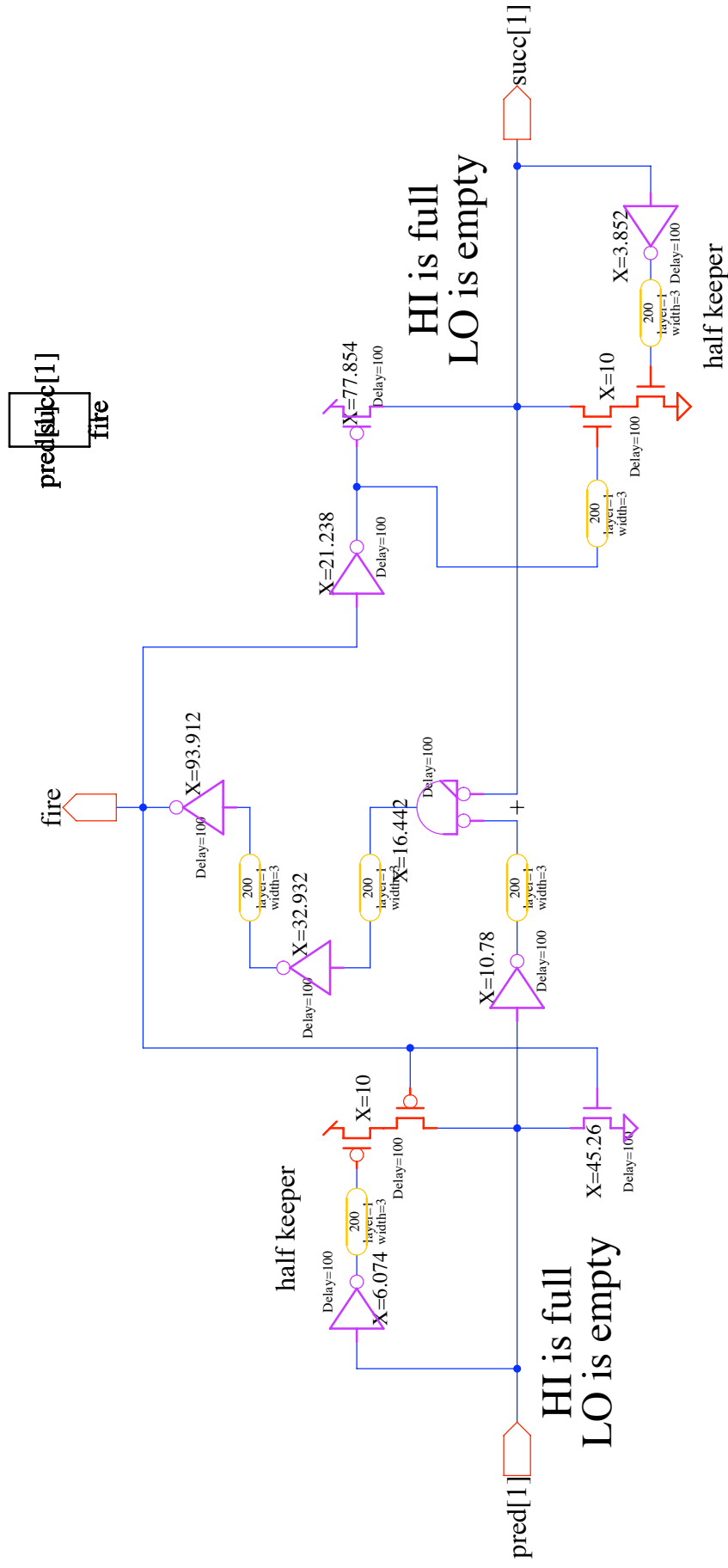


# F-gaspPlain

## Basic GasP cell

ies 8 November 2007

# AFTER

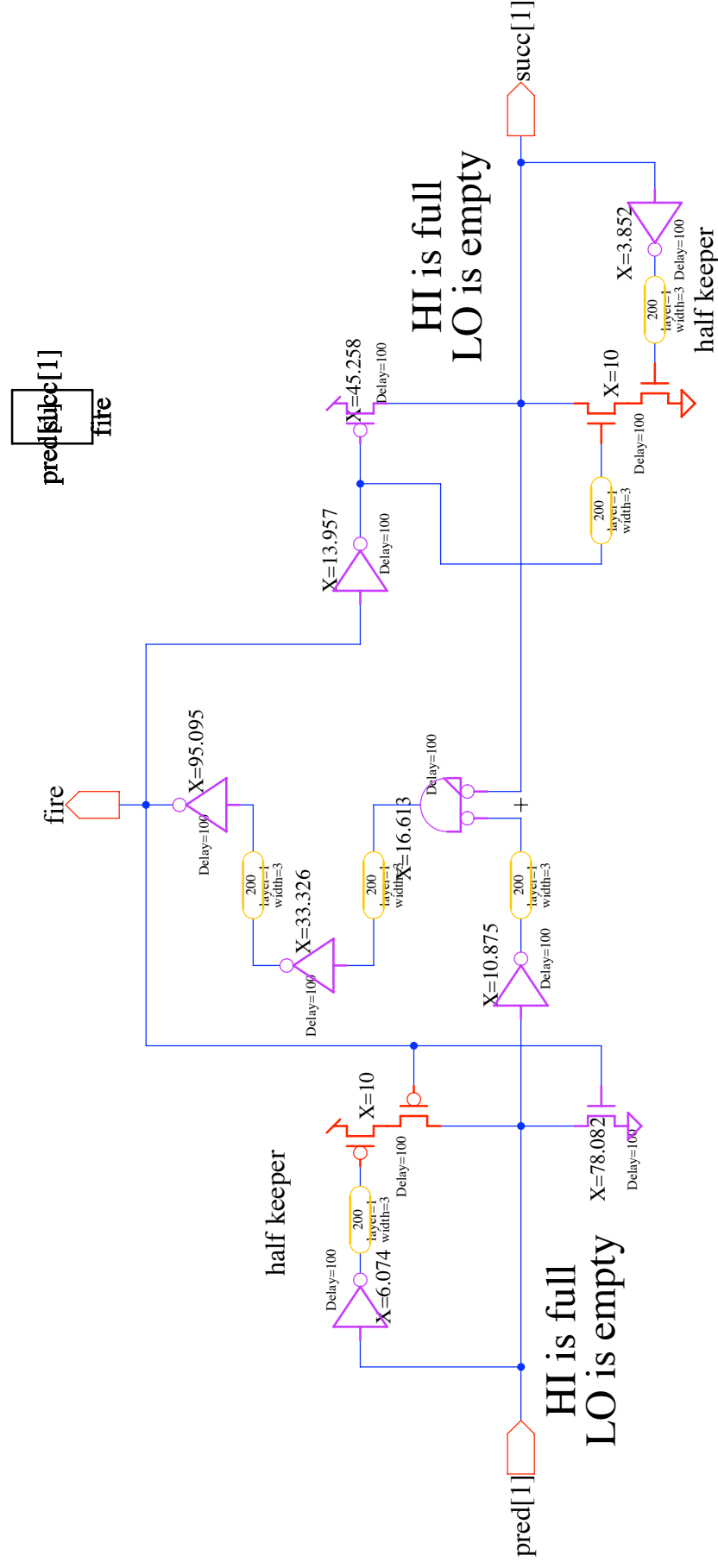


# F-gaspPlain

Basic GasP cell

ies 8 November 2007

# BEFORE

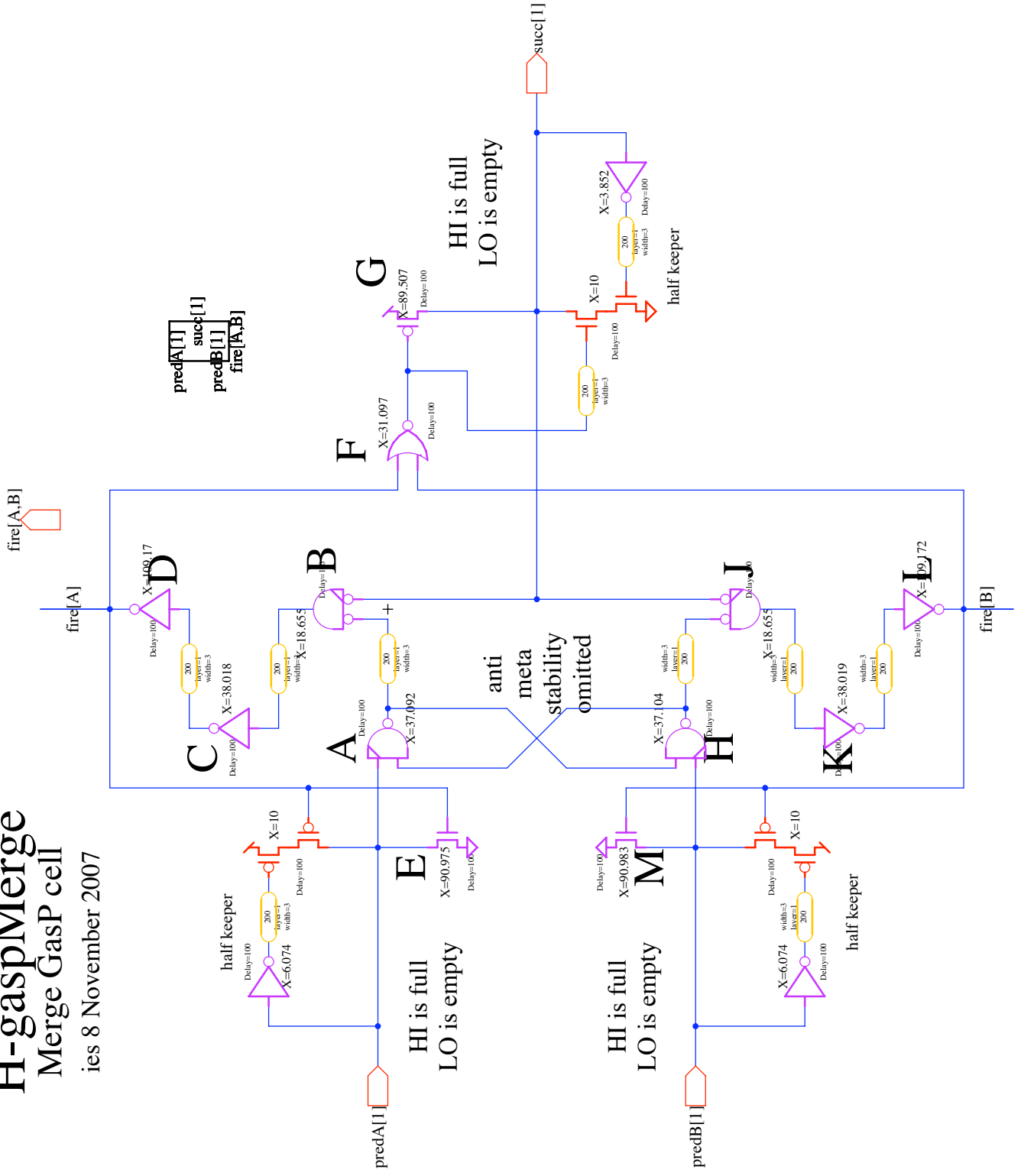




# H-gaspMerge

## Merge GasP cell

ies 8 November 2007



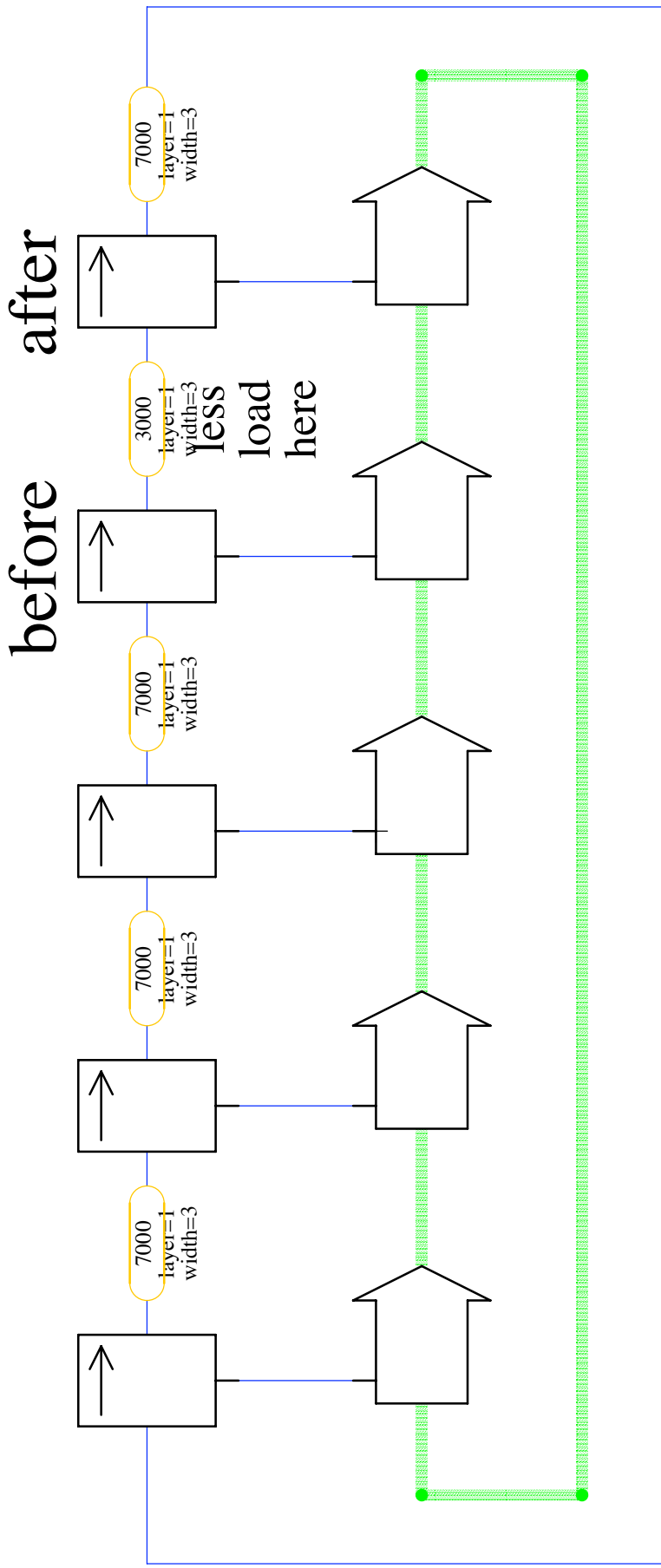
# J-rings5

a test ring of five cells

ies 29 October 2007

```
su=3.7  
wire_ratio=0.22  
epsilon=0.001  
max_iter=30  
gate_cap=0.167  
diffn=0.7  
diffp=0.7  
net_ratio=0.1
```

LESETT



```

su=3.7
wire_ratio=0.22
epsilon=0.01
k=30
gate_cap=0.167
diffp=0.7
diffn=0.7
RESET=RESET_ratio=0.1

```

# K-infinity<sup>5</sup>

twin rings of five cells each

ies 29 October 2007

