

Title: **Static latches not using NOR and NAND functions**

A novel approach for analyzing logic feedback circuits.

Author: **Peter Lablans, Ternarylogic LLC.**

Date: **June 5, 2005**

Memory latches

Data storage is probably the most unambiguous of memory solutions. In data storage (such as magnetic or optical media) a data element is written as a particular physical state (for instance reflective or non-reflective). Until that state is physically changed it remains unchanged, thus a true memory.

Other memory elements have to be refreshed, because over time they lose their capability to “remember” or hold data.

Yet another way of realizing memory is the destructive reading element, which can retain data, but loses that data when it is read.

The above examples are physical memories. Another form of memory is the logical memory, which is created by executing logical functions in feedback configuration. Its basic realization is known as a latch. Its performance depends on the execution of logical functions, not how it is physically realized.

There may be confusion about the terminology, as latches in VLSI realization are often inverters with feedback and clock-controlled gates. Latches in this article are assumed to be “logic function based” memory elements. Its well known structure is provided in the following figure 1.

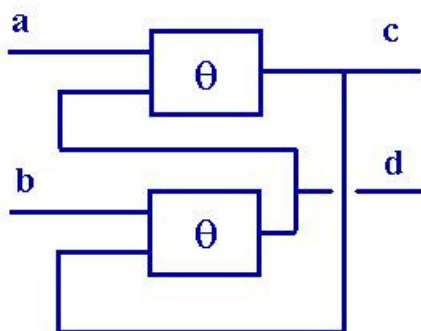


Figure 1.

The author has developed a model to create novel non-binary or multi-valued logic based memory elements. Because of the generic nature of the model it also applies to binary logic based latches. Consequently the model should correctly predict the performance of the known and perhaps novel binary latches.

The known binary latches have the structure as shown in Figure 1 and apply either the binary logic function NAND for θ or the logic function NOR.

The novel model (which is part of methods and apparatus claimed in a United States Patent Application and will be published in the USPTO Pre-Grant database in December 2005) successfully describes the behavior of binary and non-binary latches.

The model also explains what the memory effect of a latch is. It shows that a NAND or NOR latch actually does not 'store' information. What this configuration as shown in Figure 1 does is best described in the following steps:

1. for certain inputs (a,b) the device of Figure 1, executing an n-valued (with n an integer 2 or greater) logic function θ will generate an output (c,d);
2. for certain functions θ outputs (c,d) will be stable;
3. for certain functions θ certain outputs (c,d) will also be uniquely related to inputs (a,b);
4. an n-valued function θ exists in a device described by Figure 1, wherein:
 - n different combinations of (a,b) generate n uniquely related and stable outputs of (c,d);
 - at least one 'neutral' combination of inputs (a,b) exists, not equal to one of the 'n' input combinations, such that when an input combination being one of the 'n' input combinations is changed to this 'one' (neutral) combination of (a,b) the output combination (c,d) remains unchanged;

When a device complies with those conditions, we have a latch. The model will also explain what a 'forbidden state' (x,y) of a latch is. Because a stable output (c,d) is the initial state for a new input (a,b), a forbidden state (x,y) is an input state that will generate a stable output (c0,d0). When the input (x,y) is changed to the neutral combination with (c0,d0) as its initial state the outputs will become non-stable.

The memory effect of a latch is in essence the existence of a neutral input that leaves the outputs generated by the previous input states unchanged in the binary case.

Binary latches not using NAND or NOR functions

Applying all 10 non-trivial binary functions of all possible 16 binary logic functions in the novel model will generate all possible binary latch solutions. Some of those solutions will have non-commutative logic functions. The following truth table is of a non-commutative function that will realize a static latch.

| | | |
|----|---|---|
| NC | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |

Though non-commutative functions sometimes appear to be exotic, a closer look will show that they have attractive properties in for instance CMOS, as they require a minimal number of components.

The model can also be executed for the top device and the bottom device in the diagram of Figure 1 having dissimilar binary functions. For instance the combination of an OR function with an AND function will create a latch.

This and other resulting binary latch configurations can be simulated in the Java Applet “Simcir Digital Circuit Simulator Version 1.2.1” which was developed by [Kazuhiko Arase](#) who owns all related rights. A copy of the applet is widely available on-line.

The following figures show screen shots from Simcir simulating an AND-OR latch.

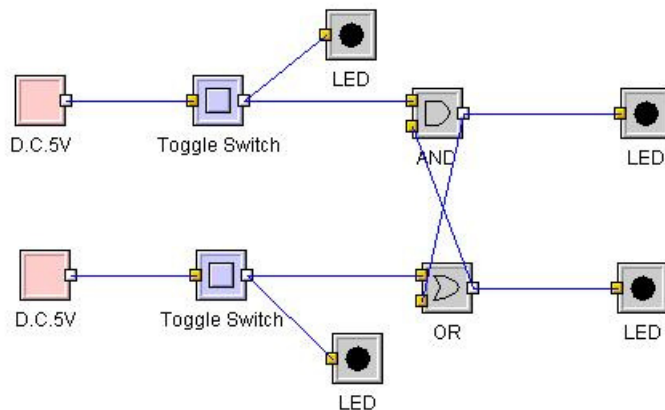


Figure 2.

Figure 2 shows a latch similar to the one of Figure 1 with an AND and an OR function. The input is (0,0) and the output is (0,0). As it turns out: input (0,0) no matter what the previous output was, will always generate (0,0).

The following Figure 3 shows that when the input is changed to (1,0) the output remains (0,0). In fact it will turn out that (1,0) is the neutral input.

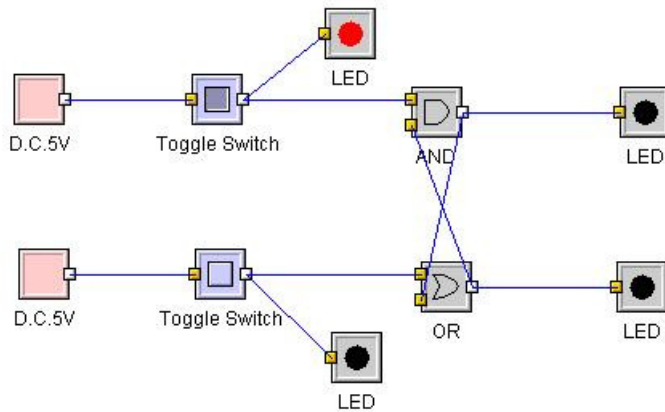


Figure 3.

The following Figure 4 shows the AND/OR latch with input (1,1) and having output (1,1).

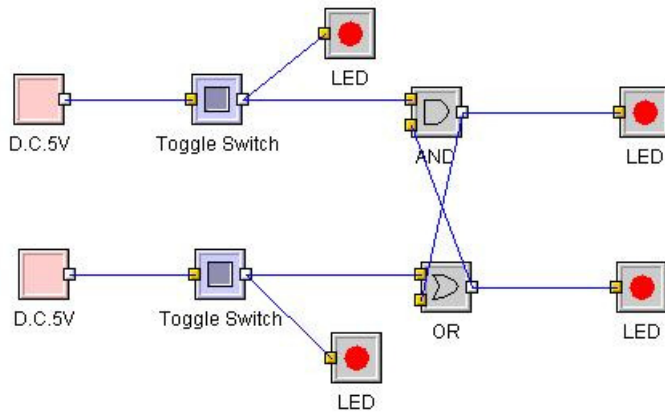
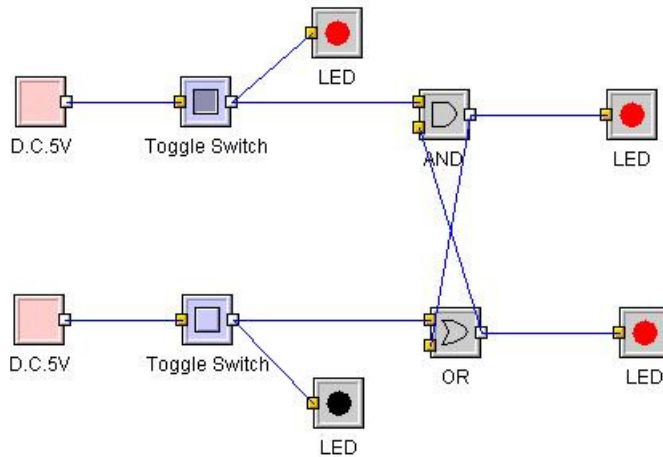


Figure 4.

The next Figure 5 shows the latch with neutral input (1,0). The output remains (1,1).



The circuit works like a true latch: it has output (0,0) with input (0,0), no matter what the initial conditions are. When the input is changed to (1,0) the output remains (0,0).

When the input is (1,1), no matter the initial conditions, the output is (1,1). When the input is changed to (1,0) the output remains (1,1).

The modeled output

It is possible to switch in the Simcir model from input (1,1) with output (1,1) to input (0,1). The same applies to switching from input (1,1) with output (0,0) to input (0,1). In both situations the output will become (0,1). However by the nature of the program it is impossible to change input (1,0) directly to (0,1) without an intermediate step or without inserting inverters.

Running the latch model based on previously defined conditions will result in the following table:

| a | b | c | d | c0 | d0 |
|---|---|--------------|--------------|----|----|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | undetermined | undetermined | 0 | 1 |
| 1 | 0 | undetermined | undetermined | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

The table shows that input $(a,b) = (0,0)$ will generate output $(c,d) = (0,0)$ no matter what initial condition $(c0,d0)$ is. When $(a,b) = (0,1)$ the output (c,d) will be $(0,1)$; and when $(a,b) = (1,1)$ then (c,d) will be $(1,1)$. When $(a,b) = (1,0)$ the output (c,d) will depend on the state of $(c0,d0)$. When $(c0,d0) = (0,0)$ with $(a,b) = (1,0)$ the output (c,d) will be $(0,0)$. When $(c0,d0) = (1,1)$ with $(a,b) = (1,0)$ then $(c,d) = (1,1)$.

However with input $(a,b) = (1,0)$ and $(c0,d0)$ was either $(0,1)$ or $(1,0)$ then the output is not determined, unstable or will reach some device favored state. So input $(0,1)$ is a forbidden state.

It should be clear that the circuit does not ‘store’ any information. It merely maintains the previous state by not changing its output values when it changes from a certain set of input signals to another set of input signals.

Relation with n-valued logic

The model, which will be published in December 2005 in the USPTO Pre-Grant Patent Application database, was developed to successfully create a series of non-binary, strictly logic based information retaining, sequential n-valued circuits, including logic based n-valued memory elements and latches. The model has been applied to generically solve the issue of the true n-valued static logic latch. The model has been run in binary mode to validate its correctness.

Other inventions and articles related to strictly logic based n-valued sequential and memory circuits exist.

In 1958 Brousentsov created the ternary (or trinary) Setun computer. In an article from 2001 Alexey Stakhov describes the 3-valued arithmetic that is applied and the 3-valued memory element called a “flip-flap-flop”. This article can be found on-line at: http://heim.ifi.uio.no/~olavky/artikler/Brousentsovs_Ternary_Principle_2002.pdf

In 1972 Maley, et al of IBM obtained patents on what they called Ternary Latches (for instance US Patent 3,671,763). These latches are multi-component.

In 2002 the USPTO published Patent Application 20020158663 by Eliahu Shemesh entitled: Non-binary digital logic element and circuit design therefor. The specification provides only one ternary function for only a 3-valued latch. The truth table for a ternary latch in that Application is validated by the model used in this paper.

A more recent publication is a thesis from February 2004: Design of Many-valued Logical Circuits by Milan Petrik, focused on 2-function latches and flip-flops. It can be found on-line at: <http://cmp.felk.cvut.cz/~petrikm/mvdesign/index.php> .

These publications all deserve close study by people interested in binary and n-valued latches.

United States Patents and Published Pre-Grant Patent Applications can be viewed at www.uspto.gov .

About Ternarylogic LLC: Ternarylogic LLC is a New Jersey based R&D company. Its mission is to create novel MVL technology. The company owns a portfolio of inventions related to scramblers/descramblers, sequence generators and sequence detectors, sequence correlators, gates and inverters based circuitry, non-binary multipliers, latches and other non-volatile memory elements, optical disk applications and MC-DSSS technology.

© 2005, Ternarylogic LLC. All rights reserved.