Homotopy Level Two Computing

by Sven Nilsen, 2022

In this paper I extend the functional complete NAND gate to homotopy level 2.

The NAND[1] gate has the property that it is functional complete. It means that any Boolean expression can be re-expressed by an equivalent expression utilizing only NAND operations.

In the paper “Path Semantical Qubit”[2], I introduced the \`qubit/~\` operator for use in Path Semantics[3]. The \`qubit/~\` operator has the following invariant:

¬~a == ~¬a

With this operator, one can express path semantical quality, aquality, contravariant quality and homotopy equivalence of higher homotopy levels[4]. In classical logic, the \`qubit\` operator generates a new random proposition that depends on the argument for a single case (synchronized per case).

It turns out that there is a 4-valued logic[5] which has the following property:

~~a == a

This 4-valued logic can be thought of as truncating or folding homotopy levels to level 2.

The truth table for the extended NAND gate to this 4-valued logic is the following:

A	**B**	**NAND(A, B)**
00	00	01
00	01	01
00	10	01
00	11	01
01	00	01
01	01	00
01	10	11
01	11	10
10	00	01
10	01	11
10	10	11
10	11	01
11	00	01
11	01	10
11	10	01
11	11	10

Using NAND gates only, the extended NAND gate can be constructed as following:

> nand_2 := \(a : [bool; 2], b : [bool; 2]) = [
>
> $a_0 \barwedge b_0$,
>
> $(\neg a_1 \wedge b_1 \wedge a_0) \vee (a_1 \wedge \neg b_1 \wedge b_0) \vee (a_1 \wedge b_1 \wedge (a_0 == b_0))$
>
>]

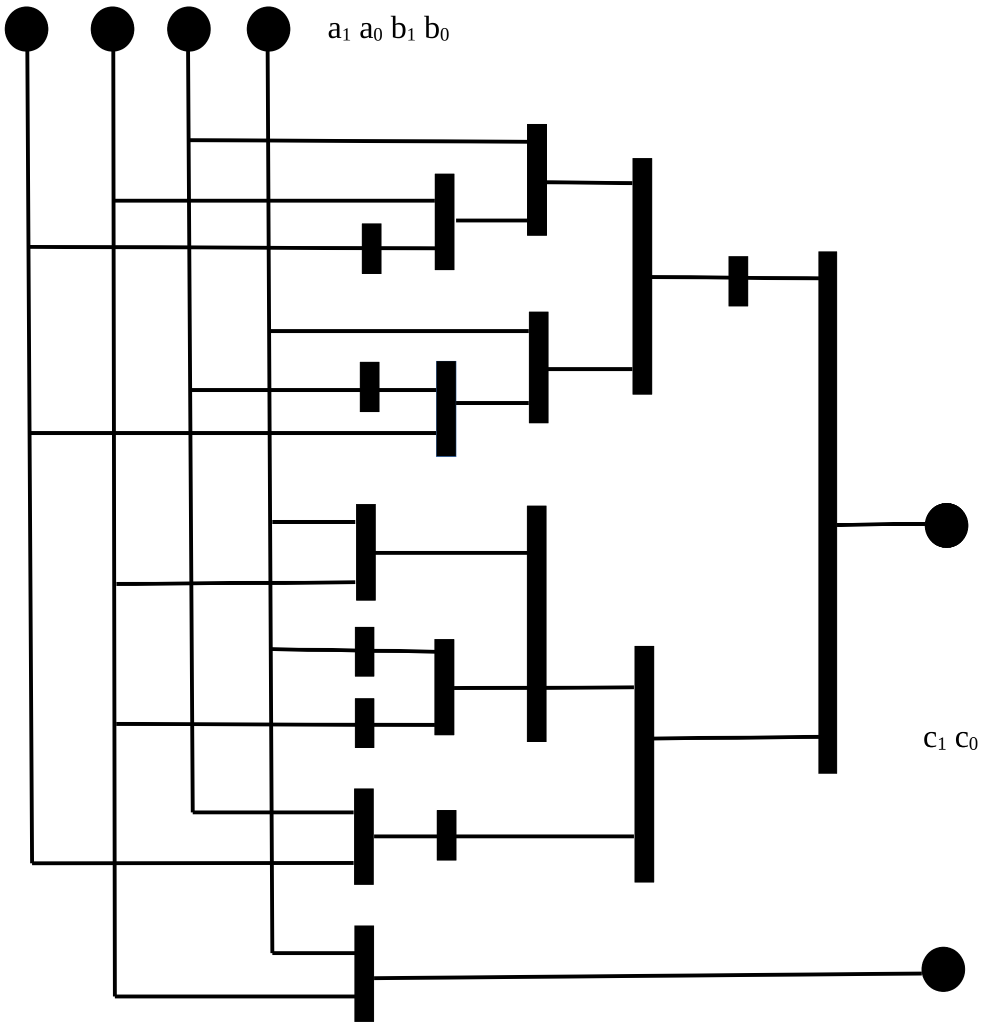
Where the operations `¬, ∧, ∨, ==` are defined using `⊼` (NAND):

> $\neg a := (a \barwedge a)$
>
> $(a \wedge b) := \neg(a \barwedge b)$
>
> $(a \vee b) := (\neg a \barwedge \neg b)$
>
> $(a == b) := (a \wedge b) \vee (\neg a \wedge \neg b)$

Here is a definition that describes the full wiring using only NAND (plus NOT for brevity):

> nand_2 := \(a : [bool; 2], b : [bool; 2]) = [
>
> $a_0 \barwedge b_0$,
>
> $\neg((\neg(\neg a_1 \barwedge a_0) \barwedge b_1) \barwedge (\neg(\neg b_1 \barwedge a_1) \barwedge b_0)) \barwedge (((a_0 \barwedge b_0) \barwedge (\neg a_0 \barwedge \neg b_0)) \barwedge \neg(a_1 \barwedge b_1))$,
>
>]

Here is a visual representation (`$a_1$, $a_0$, $b_1$, $b_0$` are inputs and `$c_1$, $c_0$` are outputs):



Using the extended NAND gate, one can bootstrap into homotopy level two computing by overloading the notation for logic using this 4-value logic instead. The construction of gates works in the same way as before, except that the extended NAND gate is used instead of the ordinary NAND gate.

The `qubit` gate can be defined using a pseudo-random synchronized `rbit` operation:

```
qubit := \(a : [bool; 2]) =
        ((a == 00) => [1, ¬rbit()]) ∧
        ((a == 01) => [1, rbit()]) ∧
        ((a == [1, ¬rbit()]) => 00) ∧
        ((a == [1, rbit()]) => 01)
```

Notice that these operations are based on the extended NAND gate.

The `rbit` operation returns `0` or `1` randomly, but stays fixed for each computational cycle.

References:

[1] "NAND logic"
Wikipedia
https://en.wikipedia.org/wiki/NAND_logic

[2] "Path Semantical Qubit"
Sven Nilsen, 2022
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/path-semantical-qubit.pdf

[3] "Path Semantics"
AdvancedResearch
https://github.com/advancedresearch/path_semantics

[4] "PSQ – Path Semantical Quantum Propositional Logic"
AdvancedResearch – Summary page on Path Semantical Quality
https://advancedresearch.github.io/quality/summary.html#psq---path-semantical-quantum-propositional-logic

[5] "Four-valued logic"
Wikipedia
https://en.wikipedia.org/wiki/Four-valued_logic