Mechanical Computing System Using Only One Physical Object - *qb) cube*

Albert Vučinović

January 14, 2018

Abstract

A new paradigm for mechanical computing is demonstrated that requires only one part. This basic part is combined to create locks and balances, which suffice to create all necessary combinatorial and sequential logic required for Turing-complete computational systems.

1 Introduction

There are many designs for mechanical computing. Most of them require many mechanical parts. In this paper we are following closely the proof from [1], which demonstrates that one can create mechanical computing systems using only links and joints. [1] proceeds by building locks and balances and proving that those are sufficient for Turing complete computing systems. We will prove that locks and balances such as used in [1] can be created using only one physical object, a *qb) cube*. The rest of the proof then follows from [1].

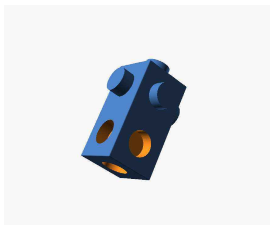

Figure 1: *qb) cube* in perspective view.

2 Computing With Only One Part

qb) cube is a depicted in Figure 1. *qb) cube* is a shape made from a *plus* and a *minus*, which are joined together into one firm object. The *plus* is a cube which has centered protrusions on every face of the cube. A *minus* is a cube which has cavities on every face of the cube. The only constraint on the protrusions on every face of the *plus*, and cavities on every face of *minus* is that their shape allows for rotation of joined *plus* and *minus* cubes. An example of such protrusions is a cylinder protrusion depicted in Figure 1. In this case the holes are of complementary shape. Of course, the *qb) cube* is a stiff object with no moving parts.

2.1 Outline of the proof

We will now proceed as follows:

- Construct a 4-Bar Linkage
- Construct a bellcrank
- Construct a lock from [1]
- Construct a balance connecting two locks on a fixed platform

When we have 4-Bar Linkages, bellcranks, locks and balances; construction of NAND gates, shift registers and thus Turing-complete computational system follows from [1].

2.2 4-Bar Linkage

Please see Figure 2, Figure 3.

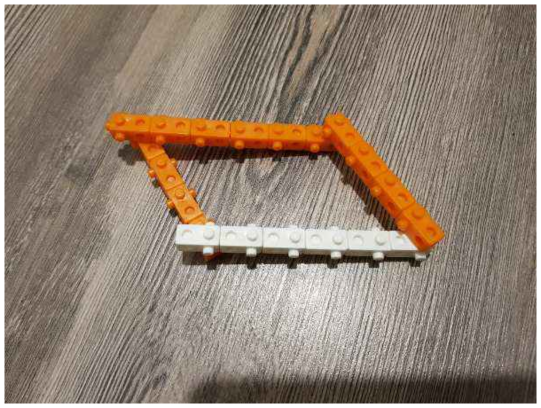

Figure 2: 4-Bar Linkage scewed to the left.

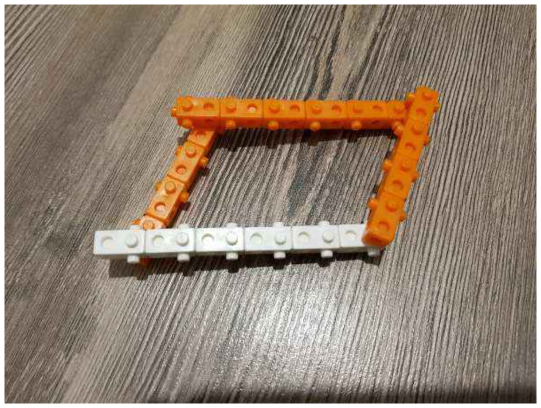

Figure 3: 4-Bar Linkage scewed to the right

2.3 Bellcrank

Please see Figure 4, Figure **??**.

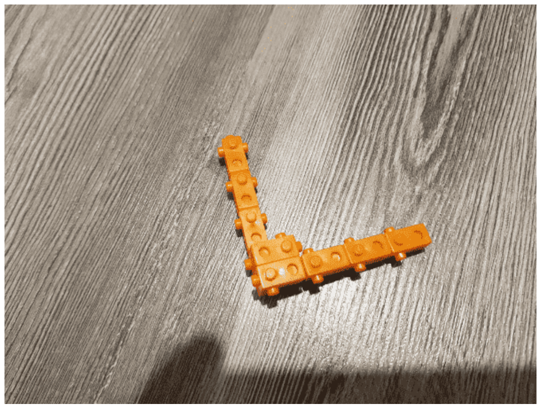
Figure 4: An example bellcrank implementation.

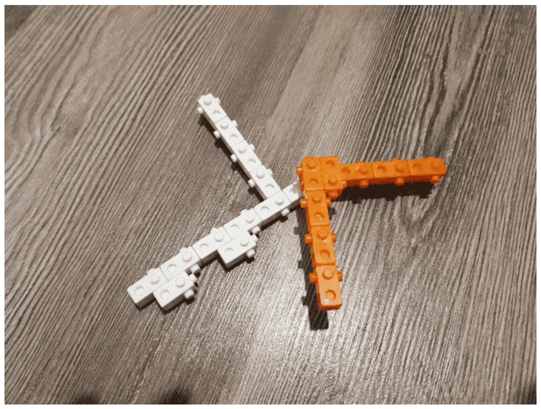
Figure 5: Bellcrank on a fixture.

2.4 Lock

Please see Figure 6, Figure 7.

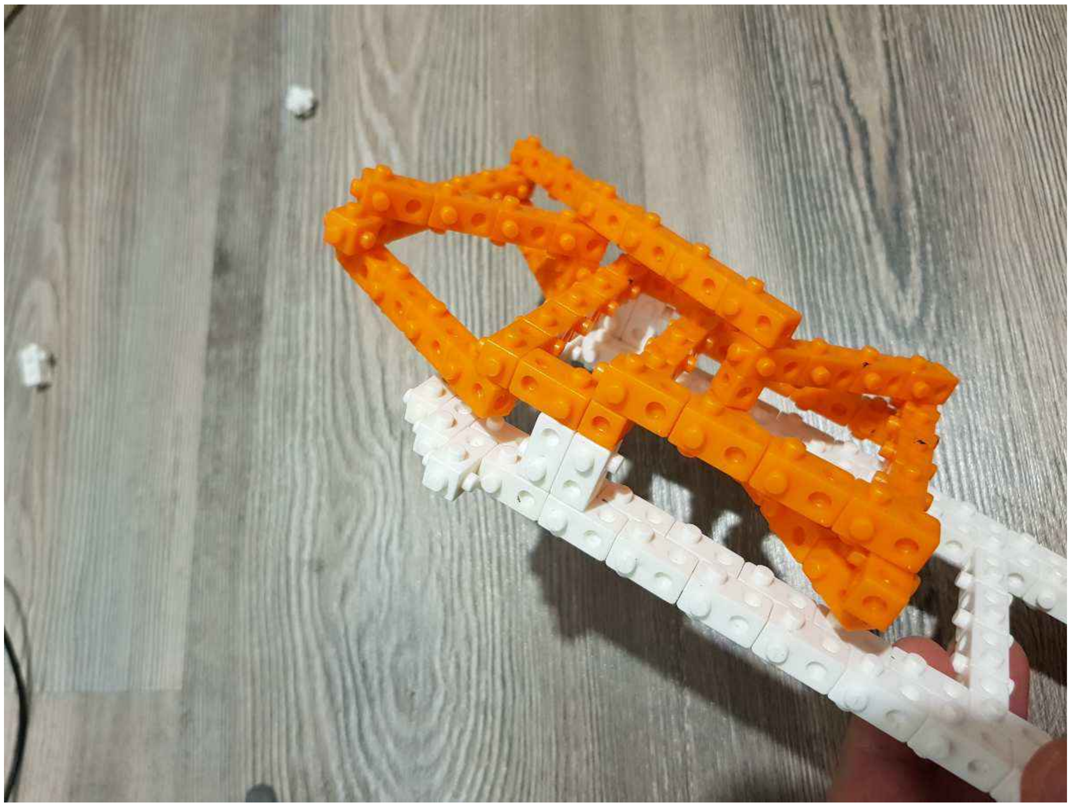

Figure 6: Lock with upper 4-bar scewed, which locks the lower in place, perspective view.

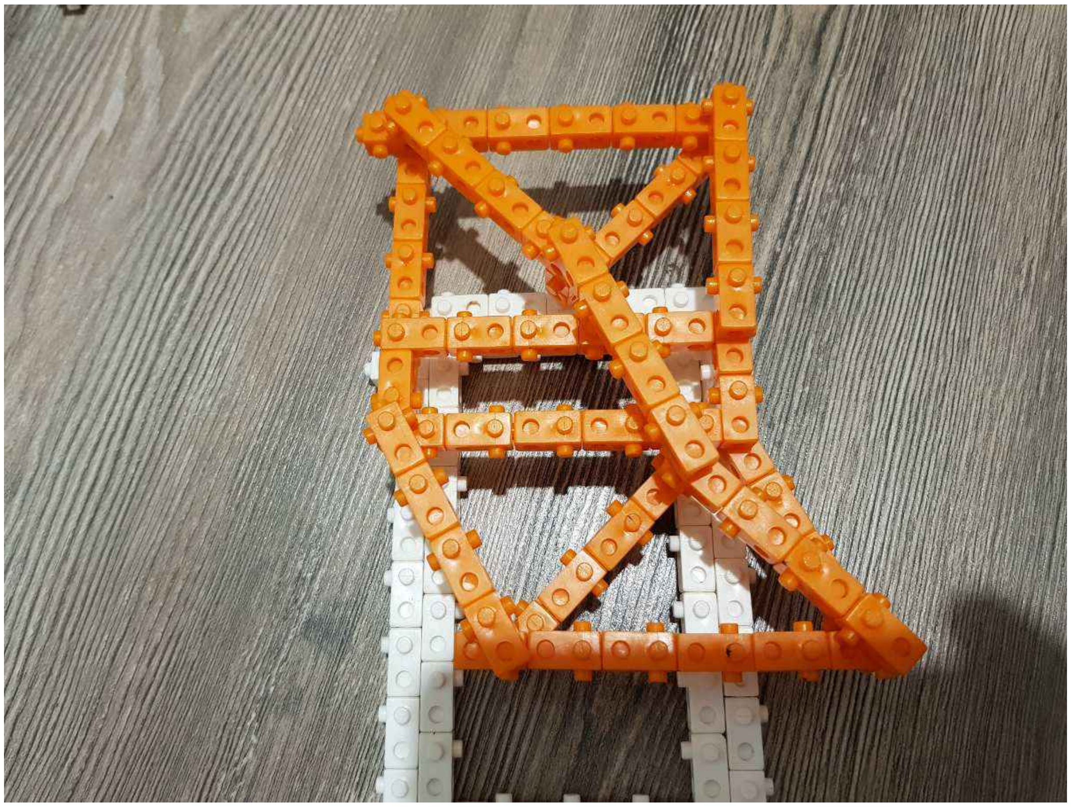

Figure 7: Lock with lower 4-bar scewed, which locks the upper in place.

2.5 A balance connecting two locks on a fixed platform

Please see Figure 8,Figure 9.

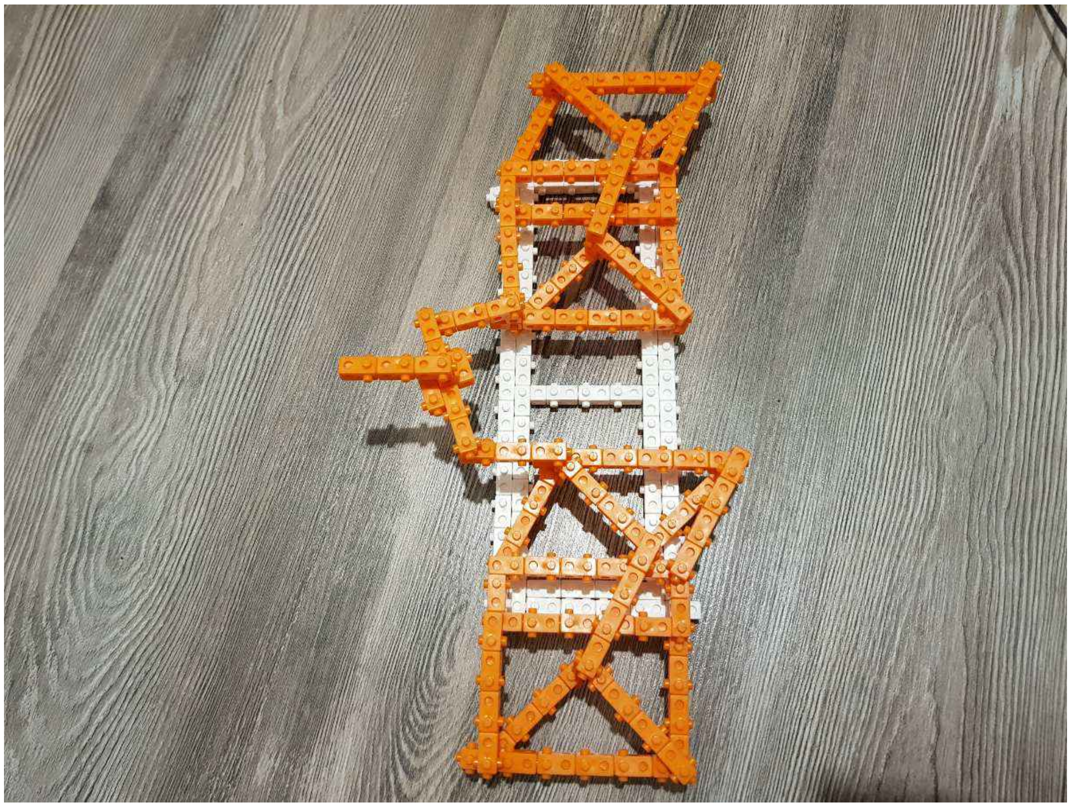

Figure 8: A balance connected to two locks.

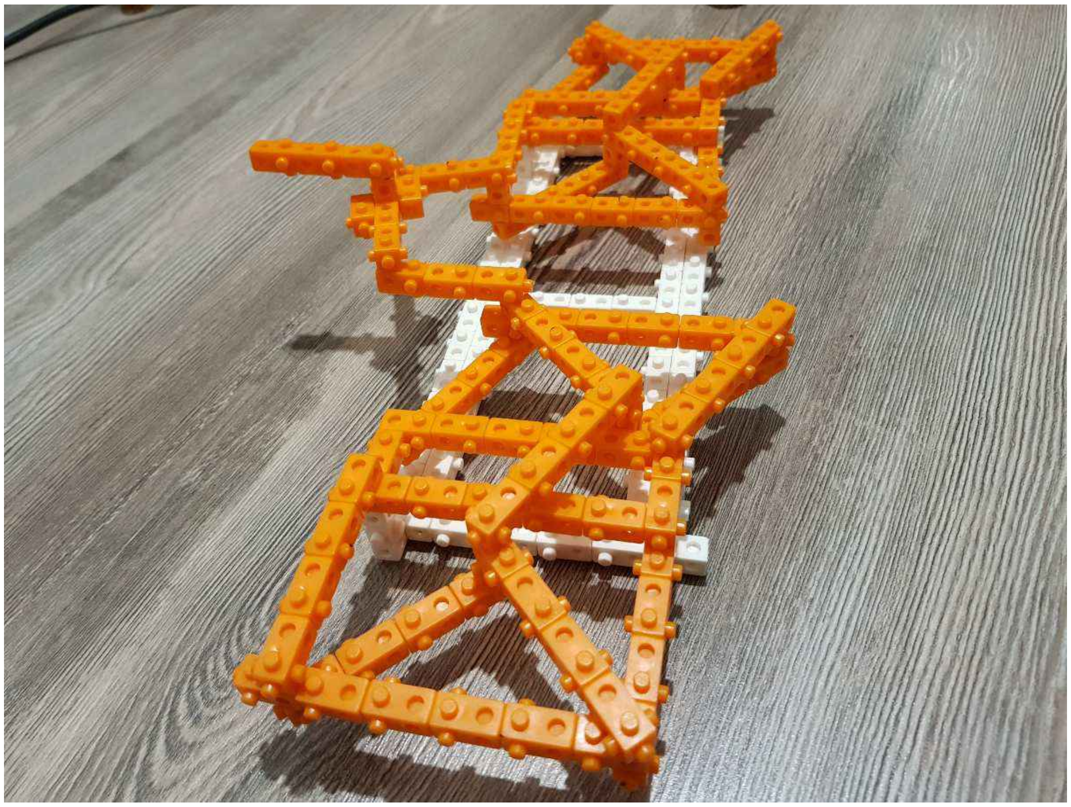

Figure 9: A balance connected to two locks in perspective view.

3 Caveats and remarks

There are a few caveats when implementing such a computational system on any scale.

- Physical characteristics of the physical *qb) cube* which is used, such as rigidity and friction in various directions. The plastic implementation of *qb) cube* would for example be unable to handle complex computations.
- Mathematical precision. For example when creating the lock, we are using some flexibility inherent in the plastic implementation of *qb) cube* in a reasonable manner. We assume that similar flexibility is available in any other system that would try to implement such a system.

In the original paper ([1]) the fixture on which the whole system would be created is not counted as an item, only links and joints. With *qb) cube* we can create that fixture as well which we think is an advantage.

4 Conclusions

We have in principle demonstrated that using only *qb) cube* a Turing-complete computational system can be created.

References

[1] Ralph C. Merkle, Robert A. Freitas Jr., Tad Hogg, Thomas E. Moore, Matthew S. Moses, James Ryley Mechanical Computing Systems Using Only Links and Rotary Joints *https://arxiv.org/abs/1801.03534*, 2018.