Computing with Planar Toppling Domino Arrangements

William M Stevens

Oxford University Department of Psychiatry, Warneford Hospital, Oxford, OX3 7JX
william@stevens93.fsnet.co.uk

Abstract. A method for implementing Boolean logic functions using arrangements of toppling dominoes is described. Any desired combinational function can be implemented. A circuit constructed using this method has no timing or order constraints on its inputs and requires no out-of-plane bridges for passing one line of dominoes over another. Since it is built using toppling dominoes, a circuit can be used only once.

Keywords: Logic circuit, dual-rail logic, one-shot logic, non-electronic logic, domino

1 Introduction

There are several different reasons for studying domino computation. There are didactic reasons that have nothing directly to do with science: for people interested in the physical basis of computing it is interesting and enjoyable to build computing primitives from simple materials that are easy to manipulate and which do not require elaborate tools to construct. Domino computers have been technologically possible for many thousands of years, but only since the advent modern mathematics and digital computing theory has anybody had the insight and motivation to both build one and tell other interested people about it.

Just as pure mathematicians strive to explore and understand all areas of mathematics, regardless of their practical utility, so some scientists apply the same philosophy to physical phenomena. Even those who have a different philosophy and who believe that all scientific endeavour should be directed towards practical outcomes must admit that there are well known examples in mathematics of ideas that seemed entirely useless at the time that they were first conceived of and developed, but which later turned out to be fundamental to theoretical science, which is in turn fundamental to several modern technologies. Domino systems are one-shot, collision-based systems, so studying their information processing potential could lead to insights into other systems that fall into one or both of these categories.

A one-shot computing system is a system that can be configured to carry out a computation, but it can only perform the computation once. A collision-based system is one in which mobile objects or patterns interact with one another to carry out information processing operations. Examples of collision-based systems include the systems based on the Belousov-Zhabotinsky (BZ) reaction that have been devised by Gorecki et al. [1], and Fredkin and Toffoli's billiard ball model [2], which is also a one-shot system. A comprehensive collection of work related to collision-based computing systems can be found in [3].

1.1 Previous Work

The idea of using interacting dominoes to carry out Boolean operations has occurred to many people, as a web search around the subject area will reveal.

O'Keefe [4] seems to have been the first to explore the area in some detail and publish his findings. O'Keefe described a system of logic gates made from dominoes in which a travelling disturbance on a line of dominoes represents a Boolean value of 1, and the absence of a disturbance represents a value of 0. Figure 1 shows a NOR gate in O'Keefe's system. The relative timing of the inputs to this gate is critical to its correct functioning. In the NOR gate, the A and B inputs to the gate must be applied before the auxiliary 1 input has been applied so that they have time to prevent the auxiliary 1 input from causing a value of 1 to be output at C.

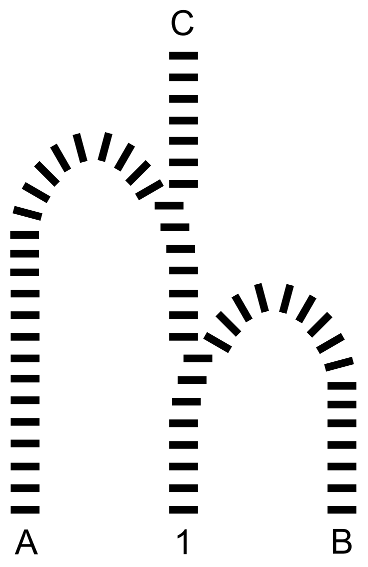

Fig. 1. A NOR gate in O'Keefe's system.

O'Keefe did not restrict his dominoes to a two-dimensional surface, he allowed bridges so that one line of dominoes could cross another without interference. O'Keefe points out that constraining a system to planar geometry requires that the flow of information in a domino system must be carefully managed so as to avoid signals being blocked by barriers of de-energised dominoes.

1.2 Challenges

This paper addresses two of the challenges that O'Keefe's work raises.

- Can a toppling domino logic system be devised that has no timing constraints on its inputs?
- Can arbitrary combinational functions be implemented in a planar system of dominoes, without requiring bridges to allow one domino line to cross another?

2 Domino Interactions and Boolean Algebra

There are a large number of ways that dominoes can be arranged. This paper makes use of conventional arrangements in which all dominoes initially stand upright, spaced apart from one another, and topple over in the course of propagating a disturbance. There are two different transitions that an individual upright domino can undergo: it can topple in one direction, or it can topple in the opposite direction. This paper does not make user of the finer details of the physics of toppling waves of dominoes, but interested readers can find out about some of these in the paper by Wagon et al. [5].

In this paper two domino configurations are defined that will be used exclusively as the basis for constructing other arrangements. These configurations are shown in Figure 2, along with symbols that will be used in schematic diagrams of more complex arrangements later on.

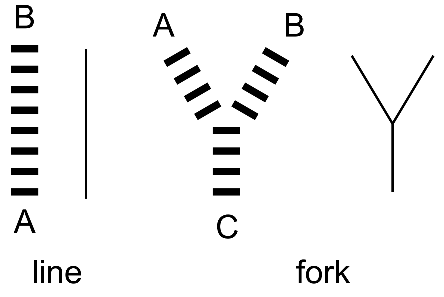

Fig. 2. Elementary domino configurations.

The first is the *line* configuration. When A topples towards B then A will cause B will topple in the same direction as A. When B topples towards A then A will topple in the same direction as B. If A and B both topple towards one another at the same time then the two toppling wavefronts will collide and nothing further will happen. Once a domino is toppled it can no longer take part in any further interaction.

The second is the *fork* configuration. This configuration has three lines of dominoes meeting at a junction, dominoes at the external ends of these three lines are labelled A,B and C. When C topples toward the junction, then it will cause both A and B to topple away from the junction. When A topples toward the junction it will cause C to topple away from the junction, but B will be unaffected. When B topples toward the junction it will cause C to topple away from the junction, but A will be unaffected. It is possible that two or three lines leading into the fork configuration topple simultaneously. Clearly if all three of A, B and C topple towards the junction simultaneously there can be no resulting output. If both A and B topple toward the junction at the same time this will cause C to topple away from the junction. If A and C (or equivalently, B and C) topple toward each other at approximately the same time then whether or not B will topple is dependent on the precise timing and location of the resulting collision. Clearly if the collision happens nearer to C than to A then B will not topple. If the collision happens nearer to A than to C then B will topple. If the collision happens in the vicinity of the fork junction then whether or not B will topple is unpredictable.

The word ‘signal’ refers to a toppling domino wavefront. So the phrase ‘a signal is applied to an input of a mechanism’, means that the domino located at that input is toppled in the direction that will cause a toppling wavefront to travel into the mechanism. The phrase ‘a signal emerges from an output of a mechanism’ means that the domino located at the output is toppled by a wavefront emerging from the mechanism.

2.1 One Way Line

A one way line can be constructed using two forks as shown in Figure 3. A signal entering the configuration from A will split at fork F into two signals that will collide with each other, preventing any signal from emerging at B. In the other direction, a signal entering the configuration from B will pass through fork G, through F and then emerge from A. Correct operation of the one way line requires that the rate of signal propagation along any two domino lines is similar, otherwise the collision that is required after a signal from A splits at fork F may happen in the wrong location. Figure 3 also shows the schematic symbol that will be used for a one way signal line.

2.2 Single Line Crossover

A single line crossover is a mechanism with two inputs A and B and two outputs A’ and B’ which are crossed over topologically. A signal may arrive either at input A or at input B. A signal arriving at A will be passed to output A’, a signal arriving at B will be passed to B’. This definition of a single line crossover does not specify what will happen if a signal arrives at both inputs: in this paper the single line crossover will only be used in situations where it is guaranteed that only one of the inputs will ever receive a signal.

A pair of forks can be used as the basis of a single line crossover, with one way lines on each input to prevent signals from one input propagating back to the other input. Figure 4 shows the single line crossover and the schematic symbol

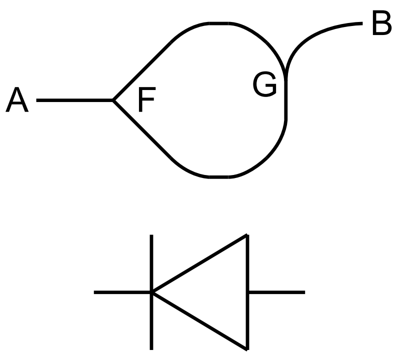

Fig. 3. A one way line.

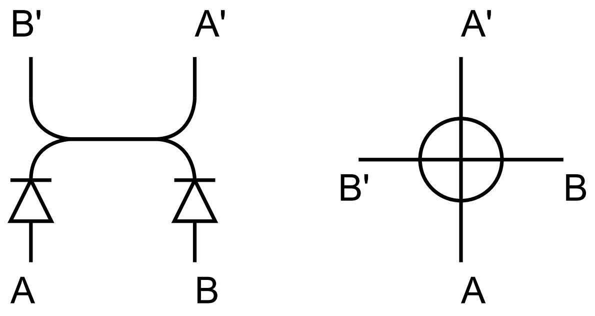

Fig. 4. A single line crossover.

that will be used to represent it. A total of six forks are used in a single line crossover.

2.3 Both Mechanism

A *both* mechanism has two inputs and one output. Only when signals have arrived at both inputs will it generate an output. It is not immediately obvious that this mechanism can be implemented using only the fork and line configurations. Figure 5 shows the both mechanism and the schematic symbol that will be used to represent it.

If a signal is applied to A well before a signal is applied to B, then the one of the two paths that emanate from the fork near input B will be interrupted, permitting a signal derived from B to emerge at C. If a signal is applied to B well before a signal is applied to A then the same process happens, with A and B exchanged. A situation that might occur if both A and B are applied close together in time is that in which A prevents B from propagating to C, but B also prevents A from propagating to C. To prevent this situation from occurring, the length of the path T must be long enough so that if a signal from B reaches

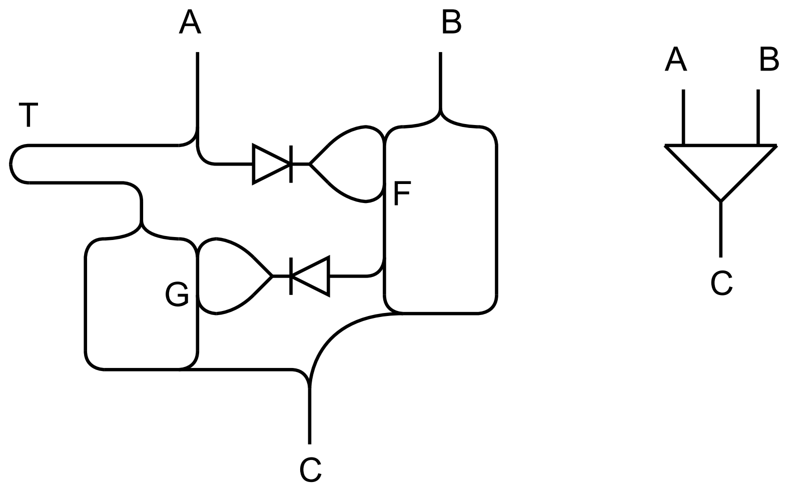

Fig. 5. A both mechanism.

fork F before a signal from A, then a signal from B will reach fork G before a signal from A. A total of 17 forks are used in the both mechanism.

In part because the both mechanism uses one way lines, and in part because of the timing constraint in the previous paragraph, correct operation of the both mechanism requires that the rate of signal propagation along any two domino lines is similar.

2.4 Dual-Rail Boolean Algebra

The mechanisms described in the previous sections can be mapped onto some of the constituents of a Boolean algebra. The way in which we choose to map signals onto Boolean values means that it is impossible to construct a mechanism to correspond to the negation operation within this Boolean algebra. Using von Neumann's double-line trick, introduced in [6], this incomplete Boolean algebra can be used to construct a complete *dual-rail* Boolean algebra that does have a negation operation. Because we will be dealing simultaneously with two different Boolean algebras, one constructed from the other, we will need to be clear at every point which one we are referring to. Let us call the incomplete Boolean algebra (without negation) B, and the complete Boolean algebra that we construct B'. All of the constituents of B' will be marked with an apostrophe.

The familiar definition of a two-element Boolean algebra is used, consisting of a set containing two elements $\{0, 1\}$ and three operations AND, OR and NOT defined in the conventional way.

We use the passage of a single signal along a propagation path at any time to represent the element 1 in algebra B, and the non-occurrence of a signal on a path at any time to represent the element 0. The both mechanism corresponds to the AND operation in algebra B. The fork mechanism corresponds to the

OR operation in algebra B. Given the way that we have chosen to represent the elements 0 and 1, we cannot construct a mechanism that corresponds to the NOT operation in algebra B because detecting 'the non-occurrence of a signal on a path at any time' cannot be done in a finite time. We would need to wait forever to be sure that an event was never going to happen.

Algebra B' is constructed from algebra B as follows. The two elements $\{0', 1'\}$ of algebra B' are defined as $0' = (1, 0)$ and $1' = (0, 1)$. That is, each element in algebra B' is made using an ordered pair of elements from algebra B. When we use a letter, such as x, to refer to an element of B' we use subscripts x_0 to refer to the first member of the ordered pair and x_1 to refer to the second member of the ordered pair.

The AND', OR' and NOT' operations of algebra B' are defined in terms of the constituents of B as follows:

$$\begin{aligned} x \text{ AND' } y &= (x_0 \text{ OR } y_0, x_1 \text{ AND } y_1) \\ x \text{ OR' } y &= (x_0 \text{ AND } y_0, x_1 \text{ OR } y_1) \\ \text{NOT' } x &= (x_1, x_0) \end{aligned}$$

It is straightforward to show that these definitions satisfy the conventional definitions for conjunction, disjunction and negation.

With this definition of the NOT' operation, negation of a value x in algebra B' can be realised simply by crossing over the signal paths along which the signals representing x_0 and x_1 travel. This can be accomplished using the single line crossover of Figure 4.

In addition to allowing a negation operation to be realised topologically by crossing over propagation paths, the other advantage that the dual-rail system has is that it removes any need to be concerned with the relative timing of signals, and so allows asynchronous circuits to be constructed. When the occurrence or non-occurrence of a signal is used to represent a Boolean value, without any other signal for timing reference, then it is not possible to know how long to wait before deciding that the signal had not occurred. In the dual-rail system used here we can expect either a signal corresponding to 0', or a signal corresponding to 1' to arrive at each dual-rail output once the circuit has finished operating. We will not be left in any state of ambiguity over whether the circuit has finished or not.

3 Logic Functions and Planar Crossover

We are now in a position to be able to construct a planar, dual-rail, one-shot, asynchronous Boolean logic system using the mechanisms introduced above. A dual-rail NAND gate is shown in Figure 6. This gate uses a total of 30 forks.

The NAND gate is a universal gate, and can be used as the basis for realising any Boolean function. To complete the proof that any Boolean function can be realised it must be shown that dual-rail signals can be split into two, so that a single output can feed into several gates. It must also be shown that dual-rail

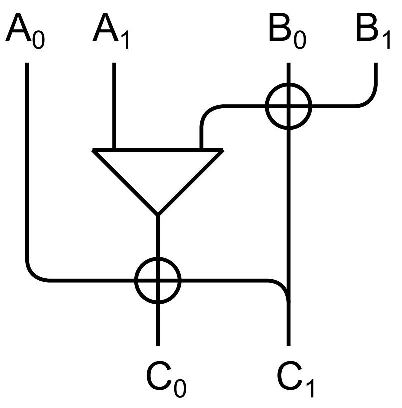

Fig. 6. A dual-rail NAND gate.

signals can be routed from one gate to another without any restriction, perhaps crossing each other.

Dual-rail signal paths can be split using the configuration shown in Figure 7, which makes use of 8 forks.

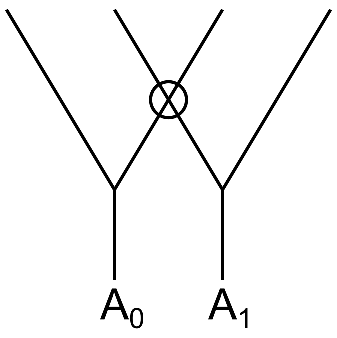

Fig. 7. A dual-rail fanout mechanism.

When two dual-rail signal paths need to cross each other we could use the planar crossover network made from AND gates and NOT gates described by Dewdney in [7]. This method would use a total of 266 forks. An alternative planar arrangement, which uses only 140 forks, is shown in Figure 8. It is not known whether this arrangement is minimal.

Since the dual-rail system does not have any timing restrictions, the precise length or route of any path between gates is unimportant, so long as the path starts and ends in the right locations. Gates can be spaced as far apart as is necessary to fit as many signals as required between them.

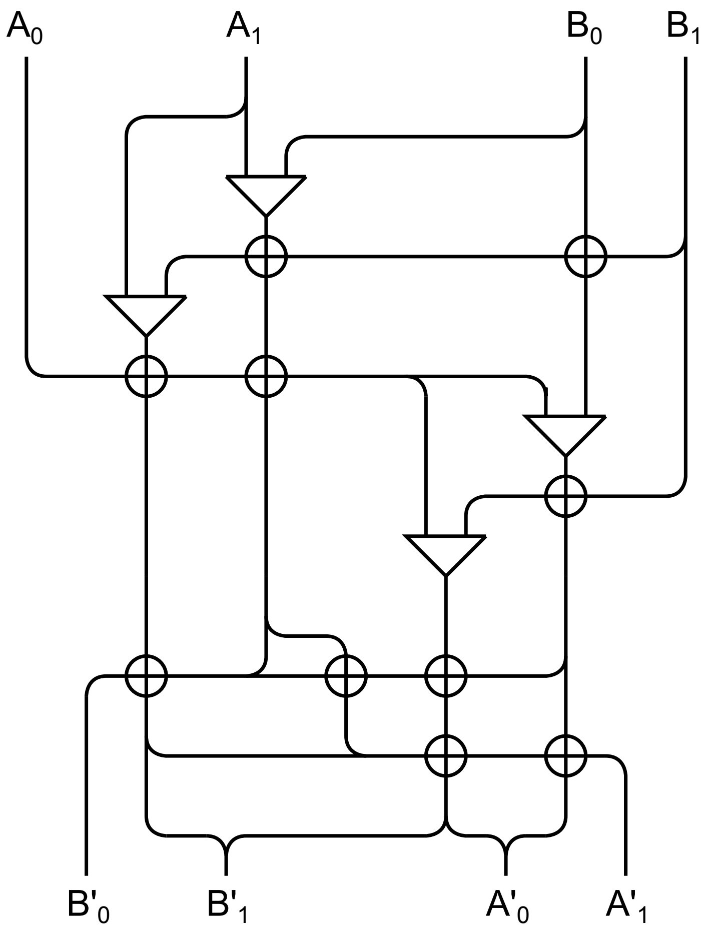

Fig. 8. A dual-rail crossover mechanism.

4 Conclusion

This paper shows that a pair of domino interactions, the fork and the line, are logically universal in the sense that they can be used to implement any combinational Boolean logic function. The method used to show this involved taking a Boolean algebra that was incomplete, but which could be implemented using dominoes in a straightforward way, and using it to construct a complete Boolean algebra.

Since the domino fork and line interactions are logically universal, subject to the assumption that the rate of signal propagation along any two domino lines is approximately uniform, any other system supporting identical or similar interactions is also likely to be a good candidate for logical universality. The direct construction of Boolean gates in a particular medium, based on these interactions, may not in itself be significant, but knowing whether or not a medium is logically universal can influence the decision about whether to investigate the ability of that medium to support other, perhaps undiscovered, information processing structures.

It is noteworthy that in the system described here a large number of elementary fork configurations seems to be required for implementing even simple Boolean logic functions, and that proving how arbitrary Boolean logic functions can be implemented is not as straightforward as it is, for example, in relay circuits. It seems that within planar domino systems, asynchronous Boolean functions are relatively complex phenomena.

A possible route for further research in this area is to formalise and to generalise the concepts used in this paper. One way that this could begin is illustrated in Figure 9. Here a one way line is split into four regions, each of which contains either a fork or a line, and has timing parameters related to line lengths. The response of a single region to all possible combinations of input events could be described by formalising the descriptions given in section 2. Composition rules could then be formulated that would permit the behaviour of a mechanism to be deduced from the behaviour of its constituent mechanisms. Working out the details of a formal system of this type is likely to be simpler in a domino system, where each event can only occur once and where each region can undergo only a finite number of state transitions, than in the more general case where events can occur any number of times and where regions may return to a previous state. Similar schemes have been used in the domain of asynchronous circuit design and also in process calculi, but (to my knowledge) none of these have been extensively used for describing, modelling or understanding information processing in event-propagating media such as domino systems, collision-based systems and spiking neural networks.

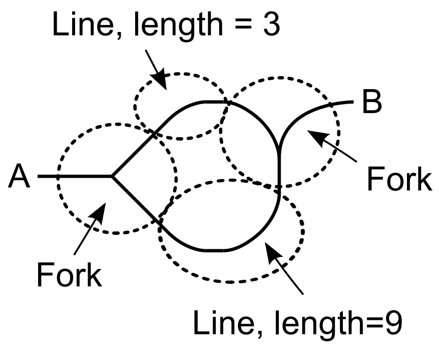

Fig. 9. A one way line split into four regions.

References

1. Gorecki, J., Gorecka, J.N.: Multi-argument logical operations performed with excitable chemical medium. J. Chem. Phys. 124(084101) (2006)
2. Fredkin, E., Toffoli, T.: Conservative Logic. J. Theo. Phys. 21(3,4), 219–253 (1982)
3. Adamatzky, A. (ed): Collision Based Computing. Springer, London (2002)
4. O'Keefe, S.: Implementation of Logical Operations on a Domino Substrate. Int. J. Unconventional Computing 5(2), 115–128 (2009)
5. Wagon, S., Briggs, W., Becker, S.: The Dynamics of Falling Dominoes. The UMAP Journal 26(1), 35–47 (2005)
6. Von Neumann, J.: Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components. In: Shannon C. and McCarthy J. (eds.) Automata Studies, pp. 43–98, Princeton University Press, Princeton, New Jersey (1956)
7. Dewdney, A. K.: Logic circuits in the plane. ACM SIGACT News 10(3), 38–48 (1979)