Chaos, Solitons and Fractals 162 (2022) 112448

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

journal homepage: www.elsevier.com/locate/chaos

Implementing and morphing Boolean gates with adaptive synchronization: The case of spiking neurons

J. Yang [a,b,1], E. Primo [b,c,1], D. Aleja [b,c,d,*], R. Criado [b,c,d], S. Boccaletti [b,c,e,f], K. Alfaro-Bittner [b,c]

[a] Unmanned Systems Research Institute, Northwestern Polytechnical University, Xi'an 710072, Shaanxi, PR China
[b] Universidad Rey Juan Carlos, Calle Tulipán s/n, Móstoles 28933, Madrid, Spain
[c] Laboratory of Mathematical Computation on Complex Networks and their Applications, Universidad Rey Juan Carlos, Calle Tulipán s/n, Móstoles 28933, Madrid, Spain
[d] Data, Complex Networks, and Cybersecurity Research Institute, Universidad Rey Juan Carlos, 28028, Madrid, Spain
[e] Moscow Institute of Physics and Technology, Dolgoprudny 141701, Moscow, Russian Federation
[f] CNR - Institute of Complex Systems, Via Madonna del Piano 10, I-50019 Sesto Fiorentino, Italy

ARTICLE INFO

Keywords:
Boolean logical gates
Synchronization
Dynamical systems
Spiking neurons

ABSTRACT

Boolean logic is the paradigm through which modern computation is performed in silica. When nonlinear dynamical systems are interacting in a directed graph, we show that computation abilities emerge spontaneously from adaptive synchronization, which actually can emulate Boolean logic. Precisely, we demonstrate that a single dynamical unit, a spiking neuron modeled by the Hodgkin-Huxley model, can be used as the basic computational unit for realizing all the 16 Boolean logical gates with two inputs and one output, when it is coupled adaptively in a way that depends on the synchronization level between the two input signals. This is realized by means of a set of parameters, whose tuning offers even the possibility of constructing a *morphing* gate, i.e., a logical gate able to switch efficiently from one to another of such 16 Boolean gates. Extensive simulations demonstrate the efficiency and the accuracy of the proposed computational paradigm.

1. Introduction

Boolean logic is that branch of algebra that defines logical operations on variables which may assume only a *truth* or *false* value, denoted respectively as 1 and 0. Its fundamental concepts and main principles were set already in 1847 by George Boole, in his book entitled "The Mathematical Analysis of Logic" [1]. But it was only in the early 20th century that the American mathematician and electrical engineer Claude Shannon described (in his MIT master thesis) the equivalence of Boolean logic to the binary properties of electrical switches performing logic functions [2], which later became the foundation of digital circuit design. Thanks to the successive, continuous, and progressive technological advances in the miniaturization of electronic components (such as high-speed circuits, or capacitive or ferromagnetic storage devices), all Computer Processing Units (CPU's) which are today equipping our smart-phones, desktops and laptops perform their functions via Boolean logic.

In more recent years, the interest shifted from Boolean computability toward defining alternative paradigms of computation, in a trial to unveil some mechanisms through which information processing takes place, for instance, in human or animal brains, and to set new paradigms for logical operations in bio-informatics and quantum computing. When computation is investigated in connection with dynamical systems and neural networks, a fertile approach which has been introduced is that of *reservoir* computing. There, input signals are mapped into higher dimensional spaces through the (transient) dynamics of a non-linear system (the reservoir).The accuracy and efficiency of this technique in performing computation has been demonstrated in several different configurations and task resolving problems [3–7].

Another proposed method was that of showing that computation abilities may emerge spontaneously from adaptive synchronization [8–10], when nonlinear dynamical systems are interacting in a directed graph via a coupling that adapts itself to the synchronization level between two input signals. In this paper, we follow this latter approach, and show how a single dynamical unit, a spiking neuron modeled by the Hodgkin-Huxley model [11], can be used as the basic computational

* Corresponding author at: Universidad Rey Juan Carlos, Calle Tulipán s/n, Móstoles 28933, Madrid, Spain.
E-mail address: david.aleja@urjc.es (D. Aleja).
[1] J.Y and E.P. equally contributed to the Manuscript

https://doi.org/10.1016/j.chaos.2022.112448
Received 17 June 2022; Accepted 11 July 2022
Available online 19 August 2022

unit for realizing all the 16 Boolean logical gates with two inputs and one output, and how a suitable tuning of a set of parameters provides actually a *morphing* gate, i.e., a logical gate able to switch from one to another of such 16 logical functions.

The paper is organized as follows: in Section 2 we describe the basic model allowing to use the dynamics of a single spiking neuron as a computational unit. In Section 3 we show how the 16 logical gates can be implemented as a function of 8 morphing parameters, and give two illustrative examples: the OR and the universal NAND gates. Finally, Section 4 reports our discussions and conclusions.

2. The computational paradigm

Our basic computational unit is pictorially sketched in Fig. 1a). It consists of two input ports A and B, one output port $[O(t)]$, and a dynamical system, namely a neuron whose internal dynamics $[ID(t)]$ evolves in time following the Hodgkin-Huxley model

$$C_m\frac{dV}{dt} = I - G_n - W(V_A - V), \tag{2.1}$$

where V stands for the membrane potential of the excitable neuron, C_m is the membrane capacitance, and I is the ionic membrane current [11].

Let us assume to have a networked ensemble of such computational units. We also assume that the input arriving to one of these units be described by $W = W(\Delta V_{AB}, \mu)$, i.e., be a function of a coupling parameter and depends on the difference of the membrane potentials entering ports A and B. Furthermore, each neuron is under the influence of *i*) an external source of Gaussian white noise $G_n(t)$ that equally affects all units of the ensemble and *ii*) the existence of a reference signal $R(t)$. The 1-state is postulated to be that spiking dynamical state which is *synchronous* with R.

Notice that, in the absence of network interactions, each neuron would evolve according to its own internal dynamics. However, due to the very well-known phenomenon of noise induced synchronization in spiking-like dynamics [12–14], the term $G_n(t)$ will induce the internal dynamics ID of all units to synchronize, after a suitable transient, to a unique dynamical state $S(t)$, which, from now on, will be associated to the 0-state.

As for V_A and V_B (i.e., the input voltages entering from ports A and B),

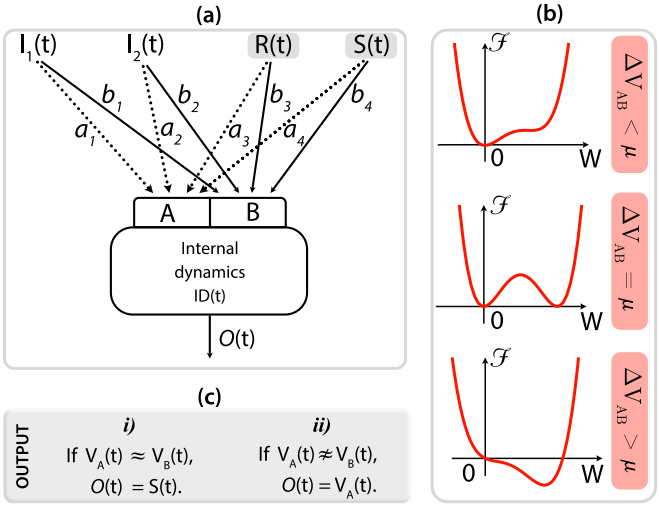

Fig. 1. The computational unit. Schematic representation of the computational unit. (a) The unit is constituted by 1) a neuron whose internal dynamics (ID) follows Eq. (2.1), 2) two input ports A and B, and 3) the output port $O(t)$. The input voltage entering from port $A(B)$ is given by a linear combination of signals $I_1(t)$, $I_2(t)$, $R(t)$, and $S(t)$, as described in Eq. 2.2. (b) The stability of the equilibria points of Eq. (2.3), for $\mu = 0.25$, $k = 0.3$, $w_1 = 0.5$, and $w_2 = 1$. (c) The outputs $O(t)$ when *i*) $V_A \approx V_B$ and *ii*) $V_A \not\approx V_B$.

they can be defined as linear combinations of $I_1(t)$, $I_2(t)$ (the two input signals which will be actually processed by the computational unit), $R(t)$, and $S(t)$, i.e.,

$$\begin{aligned} V_A &= a_1 I_1(t) + a_2 I_2(t) + a_3 R(t) + a_4 S(t), \\ V_B &= b_1 I_1(t) + b_2 I_2(t) + b_3 R(t) + b_4 S(t). \end{aligned} \tag{2.2}$$

On its turn, this leads to the introduction of eight *morphing parameters* $(a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4)$ which, as we will see momentarily, define the logical operation that the unit is performing on the input signals I_1 and I_2.

Finally, in the Hodgkin-Huxley model [11], the ionic membrane current I comes from the contributions of sodium (Na), potassium (K) and other (l) ions' currencies, such that $I = -(I_l + I_K + I_{Na})$ where $I_l = g_l(V - V_l)$, and I_{Na} is defined as

$$\begin{aligned} I_{Na} &= g_{Na} m^3 h (V - V_{Na}), \\ \frac{dm}{dt} &= \alpha_m(1-m) - \beta_m m, \\ \frac{dh}{dt} &= \alpha_h(1-h) - \beta_h h, \\ \alpha_m &= \frac{0.1(V+25)}{e^{\frac{V+25}{10}} - 1}, \\ \beta_m &= 4e^{\frac{V}{18}}, \\ \alpha_h &= 0.07 e^{\frac{V}{20}}, \\ \beta_h &= \frac{1}{e^{\frac{V+30}{10}} + 1}, \end{aligned}$$

and I_K is given by

$$\begin{aligned} I_K &= g_K n^4 h (V - V_K), \\ \frac{dn}{dt} &= \alpha_n(1-n) - \beta_n n, \\ \alpha_n &= \frac{0.01(V+10)}{e^{\frac{V-10}{10}} - 1}, \\ \beta_n &= 0.125 e^{\frac{V}{80}}. \end{aligned}$$

Additionally, the strength W of the coupling to the input signal that enters from port A evolves as

$$\dot{W} = -W(W - w_1)(W - w_2) + k[\Delta V_{AB} - \mu], \tag{2.3}$$

where k is an adaptation speed, ΔV_{AB} is a positive function that quantifies the synchronization error between the voltages or signals entering from port A and B, and μ is a threshold used to filter small synchronization errors coming from random sources of noise. Unless otherwise specified, the parameters used in our study are $\mu = 0.25$ and $k = 0.3$.

The stability properties of the equilibria of Eq. (2.3) depend on the parameters w_1, w_2, k, μ, and ΔV_{AB}. Panel (b) in Fig. 1 shows the stability properties of such equilibria, by considering that Eq. (2.3) can be written as $\dot{W} = -\delta F/\delta W$ with $F = W^4/4 - (w_1 + w_2)W^3/3 + w_1 w_2 W^2/2 - k(\Delta V_{AB} - \mu)W$.

The adaptive dynamics of $W(\Delta V_{AB}, \mu)$ induces alternation of synchronization and desynchronization processes, in that it drives the coupling strength toward zero (or close to zero) or to a positive value. Precisely, if $V_A \approx V_B$, then $W \approx 0$ and the dynamics of the unit synchronizes to the state $S(t)$ induced by influence of the Gaussian white noise, i.e.,

$$O(t) = S(t), \quad V_A \approx V_B. \tag{2.4}$$

Instead, if $V_A \approx V_B$, W will converge toward a positive value, and therefore the dynamics of the unit will synchronize to that exhibited by the voltage entering from port A, i.e.,

$$O(t) = V_A(t), V_A \approx V_B. \tag{2.5}$$

It is worth mentioning that the latter condition can be rigorously proven only when V_A exhibits a dynamics which is compatible with the Hodgkin-Huxley model, i.e. when either $V_A = R$ or $V_A = S$. In the more general case of Eqs. 2.2 i.e., when V_A is a generic linear combination of $I_1(t)$, $I_2(t)$, $R(t)$, and $S(t)$ the condition is not automatically guaranteed and has to be checked numerically.

3. Implementation of the Boolean logical gates

3.1. The general procedure

The goal is to use the computational unit described in the previous section for implementing all the 16 possible logic Boolean operations (corresponding to two inputs and one output) whose truth table is reported in Table 1. Any of such logic operations returns 0 or 1 depending on the values (also 0 or 1) that the two inputs (p, q) are featuring. In our framework, the latter sentence means that the output of our neuron dynamics will be $S(t)$ (the 0-state) or $R(t)$ (the 1-state), depending on the signals $I_1(t)$ and $I_2(t)$ that, in this case, are playing the role of p and q. As it can be seen from Eq. (2.2), there are eight unknown morphing parameters, whose values actually define the specific gate that is being implemented. The method for their determination can be described as follows.

The first step is to consider the inputs I_1 and I_2 of the "true table". Both inputs are necessarily synchronized to either S (and therefore take the value 0) or R (and therefore take the value 1). In addition, each pairs (I_1, I_2) defines V_A and V_B: if $(I_1, I_2) = (R, R)$ then $V_A = (a_1 + a_2 + a_3)R + a_4 S$ and $V_B = (b_1 + b_2 + b_3)R + b_4 S$, see Table 2, which is valid regardless on the specific logic gate to be implemented.

Once the expressions for the input voltages entering ports A and B (V_A and V_B) are calculated, one immediately obtains a set of equations by applying the condition (2.4) (when $O(t) = S$) or (2.5) (when $O(t) = V_A$) to each row of Table 2. To illustrate such a latter step, let us suppose

Table 1
The truth table for the 16 Boolean logic gates: contradiction $\perp$, logical conjunction AND, material no-implication $\not\rightarrow$, converse no-implication $\not\leftarrow$, logical NOR, projection functions p and q, logical bi-conditional XNOR, tautology T, logical NAND, material implication $p \rightarrow q$, converse implication $p \leftarrow q$, logical disjunction OR, negations $\neg p$ and $\neg q$, and exclusive disjunction XOR.

(p, q)	$\perp$	AND	$\not\rightarrow$	$\not\leftarrow$
(1,1)	0	1	0	0
(1,0)	0	0	1	0
(0,1)	0	0	0	1
(0,0)	0	0	0	0
(p, q)	NOR	p	q	XNOR
(1, 1)	0	1	1	1
(1, 0)	0	1	0	0
(0, 1)	0	0	1	0
(0, 0)	1	0	0	1
(p, q)	⊺	NAND	p→q	p←q
(1, 1)	1	0	1	1
(1, 0)	1	1	0	1
(0, 1)	1	1	1	0
(0, 0)	1	1	1	1
(p, q)	OR	¬p	¬ q	XOR
(1, 1)	1	0	0	0
(1, 0)	1	0	1	1
(0, 1)	1	1	0	1
(0, 0)	0	1	1	0

Table 2
General expressions for the voltages V_A and V_B when the controllable signals I_1 and I_2 are R (1-state) or S (0-state).

(I_1, I_2)	V_A	V_B
(R, R)	$(a_1 + a_2 + a_3)R + a_4 S$	$(b_1 + b_2 + b_3)R + b_4 S$
(R, S)	$(a_1 + a_3)R + (a_2 + a_4)S$	$(b_1 + b_3)R + (b_2 + b_4)S$
(S, R)	$(a_2 + a_3)R + (a_1 + a_4)S$	$(b_2 + b_3)R + (b_1 + b_4)S$
(S, S)	$a_3 R + (a_1 + a_2 + a_4)S$	$b_3 R + (b_1 + b_2 + b_4)S$

that the output signal for $(I_1, I_2) = (R, R)$ is R. Then, condition (2.5) must be satisfied, yielding

$$\left.\begin{array}{rcl} O(t) = V_A = R & \rightarrow & a_1 + a_2 + a_3 = 1 \wedge a_4 = 0 \\ V_A \approx V_B & \rightarrow & b_1 + b_2 + b_3 \neq 1 \vee b_4 \neq 0. \end{array}\right\} \tag{3.1}$$

Eventually, the procedure leads to a set of equations whose solution is, in principle, not unique. In other words, all the conditions on the morphing coefficients defining the 16 possible Boolean logic cases are satisfied for *a family of solutions*. Table 3 reports one of such possible solutions for each of the 16 gates. For the sake of clarity and exemplification, in the next subsections we illustrate the family of solutions corresponding to some specific case, starting from the logical disjunction (OR) gate.

3.2. The OR Gate

The logical disjunction gate is that gate whose output $O(t)$ is S (0-state) when $I_1 = I_2 = S (p = q = 0)$ and R otherwise. Application of the method described in the previous subsection yields the following set of equations

- $(I_1, I_2) = (R, R) \rightarrow O(t) = R$,

$$\left.\begin{array}{ll} a) & O(t) = V_A \rightarrow a_1 + a_2 + a_3 = 1 \wedge a_4 = 0 \\ b) & V_A \approx V_B \rightarrow b_1 + b_2 + b_3 \neq 1 \vee b_4 \neq 0. \end{array}\right\} \tag{3.2}$$

- $(I_1, I_2) = (R, S) \rightarrow O(t) = R$,

$$\left.\begin{array}{ll} a) & O(t) = V_A \rightarrow a_1 + a_3 = 1 \wedge a_2 + a_4 = 0 \\ b) & V_A \approx V_B \rightarrow b_1 + b_3 \neq 1 \vee b_2 + b_4 \neq 0. \end{array}\right\} \tag{3.3}$$

- $(I_1, I_2) = (S, R) \rightarrow O(t) = R$,

$$\left.\begin{array}{ll} a) & O(t) = V_A \rightarrow a_2 + a_3 = 1 \wedge a_1 + a_4 = 0 \\ b) & V_A \approx V_B \rightarrow b_2 + b_3 \neq 1 \vee b_1 + b_4 \neq 0. \end{array}\right\} \tag{3.4}$$

Notice that Eqs. (3.2–3.4) are directly determined from Eq. (2.5). The remaining logical operation ($(I_1, I_2) = (S, S) \rightarrow O(t) = S$) leads to two

Table 3
A possible choice of the morphing parameters realizing the different 16 Boolean logic gates.

Gate	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4
$\perp$	0	0	0	1	0	0	0	1
AND	0	1	0	0	−1	1	0	1
$\not\rightarrow$	1	0	0	0	0	1	0	0
$\not\leftarrow$	0	1	0	0	1	0	0	0
NOR	0	−1	1	1	1	0	0	0
p	1	0	0	0	0	0	0	1
q	0	1	0	0	0	0	0	1
XNOR	−1	1	1	0	0	0	2	−1
⊺	0	0	1	0	1	0	0	1
NAND	0	0	1	0	2	−1	0	0
p → q	0	0	1	0	1	−1	0	1
p ← q	0	0	1	0	−1	1	0	1
OR	0	0	1	0	−1	−1	1	2
¬p	0	0	1	0	1	0	0	0
¬q	0	0	1	0	0	1	0	0
XOR	−1	−1	2	1	0	0	2	−1

separate conditions

- Condition 2.4, $V_A \approx V_B$

$$\rightarrow a_1 + a_2 + a_4 = b_1 + b_2 + b_4 \wedge a_3 = b_3. \tag{3.5}$$

- Condition 2.5

$$\left.\begin{array}{ll} a) & O(t) = V_A \rightarrow a_1 + a_2 + a_4 = 1 \wedge a_3 = 0. \\ b) & V_A \approx V_B \rightarrow b_1 + b_2 + b_4 \neq 1 \vee b_3 \neq 0. \end{array}\right\} \tag{3.6}$$

Now, the morphing parameters a_i and b_i, $i = 1, \ldots, 4$ must satisfy Eqs. (3.2–3.4) and either Eq. (3.5) or Eq. (3.6). In particular, one immediately sees that the values of a_i are completely determined by Eqs. (3.2a, 3.3a, 3.4a). After straightforward calculations, a possible choice is $a_1 = 0$, $a_2 = 0$, $a_3 = 1$, and $a_4 = 0$. Notice furthermore that, for the chosen values for a_i, Eq. (3.6) has no solutions in that $a_3 \neq 0$. This implies that the remaining parameters (b_1, b_2, b_3, and b_4), must satisfy Eqs. (3.2b, 3.3b, 3.4b) and Eq. (3.5), i.e., $(b_3 = 1 \wedge b_1 + b_2 + b_4 = 0) \wedge (b_4 \neq 0 \vee b_1 + b_2 \neq 0) \wedge (b_1 \neq 0 \vee b_2 + b_4 \neq 0) \wedge (b_2 \neq 0 \vee b_1 + b_4 \neq 0)$.

Simple calculations lead one to deduce that the unknown parameters have to lay on the plane π defined by $b_1 + b_2 + b_4 = 0$ with $b_1 \neq 0$, $b_2 \neq 0$, and $b_4 \neq 0$, as it is shown in Fig. 2(a), where the forbidden values are marked by the blue, red, and black straight lines $l_1 - l_3$.

Panel (b) of Fig. 2 reports the numerical simulations of Eq. (2.1) with the input voltages V_A and V_B obtained through the morphing parameters $a_1 = a_2 = a_4 = 0, a_3 = b_3 = 1, b_1 = b_2 = -1, b_4 = 2$ (i.e., the possible solution discussed above). All our simulations have been performed using the Euler integration method, with integration time step $h = 0.01$ unit time. The full simulation ($2*10^6$ integration time step's units) is actually divided in four equally long time intervals (each one made of $0.5*10^6$ integration steps), which are separated with vertical dashed lines. Each of such intervals corresponds to one of the four different settings of the input signals ($(I_1, I_2) = (R, R)$, $(I_1, I_2) = (R, S)$, $(I_1, I_2) = (S, R)$, and $(I_1, I_2) = (S, S)$), as specified on top of the curve of $O(t)$ (last row of panel b). Furthermore, red and blue colors are used to refer to the R (1-state) and S (0-state) signals, respectively. It is seen that the computational unit correctly processes all logical operations, with a rather little transient needed to pass from one state to the other (visible in the two gray regions of the time evolution of $O(t)$ located at the very beginning of the simulation and after $1.5*10^6$ integration steps), which is the time needed by the system from desynchronizing from a dynamical state and to re-synchronize to the other.

In order to quantify the accuracy, or precision, with which the computation task is performed, one can adopt the following procedure. One first fixes an observation time window $\Delta_1 t$ (in our case $\Delta_1 t = 600$ integration time steps), on which the accuracy measure $0 \leq \Delta(x, y, t) \leq 1$ is defined in the interval $[t, t + \Delta_1 t]$. On its turn, $\Delta(x, y, t)$ is calculated as follows. Initially, the x signal is taken as a reference, and $num(x(t), y(t))$ is calculated as the number of spikes (in the interval $[t, t + \Delta_1 t]$) featured by the signal x (the local maxima in $x(t)$ which exceed a given threshold) that correspond also to spikes featured by the signal y around the same spiking time (i.e., for each spike in x at time $t_j \in [t, t + \Delta_1 t]$ one searches for the existence of a spike in y in the interval $[t_j - \Delta_2 t, t_j + \Delta_2 t]$, with $\Delta_2 t = 100$ integration time steps in our case). The same process is repeated, taking y as reference signal, for the calculation of $num(y(t), x(t))$. Let us furthermore denote with $n_s(x(t))$ and $n_s(y(t))$ the total number of spikes featured by the signals x and y, respectively, within the interval $[t, t + \Delta_1 t]$. Then, one has $\Delta(x, y, t) = \frac{num(x(t), y(t)) + num(y(t), x(t))}{n_s(x(t)) + n_s(y(t))}$.

In our case, we consider

$$Sync(t) \equiv \Delta(O(t), R, t), \tag{3.7}$$

which implies $Sync(t) \sim 1$ when $O(t) = R$ and $Sync(t) \sim 0$ when $O(t) = S$. Fig. 2(c) reports $Sync(t)$ for our simulations of the OR gate, from which and one can see that the computation is indeed quite accurate.

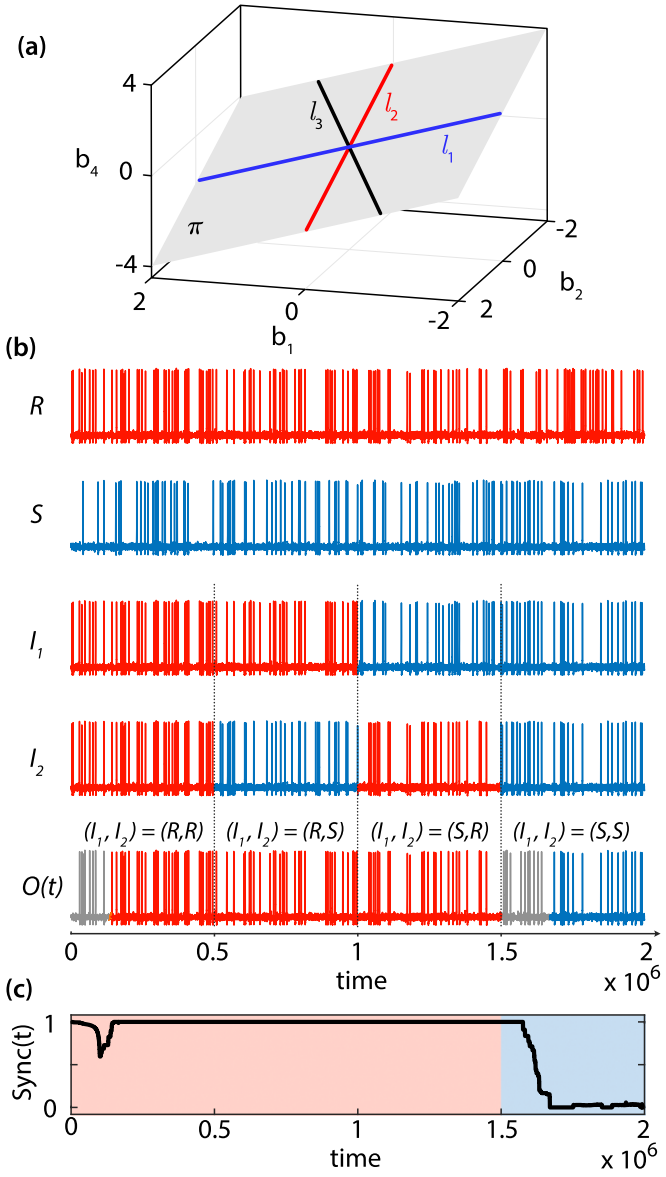

Fig. 2. (Colour online) **The OR gate.** (a) Plot of the family of solutions available for the coefficients b_1, b_2, and b_4. Red, blue, and black straight lines represents the forbidden values for $\{b_1, b_2, b_4\}$, respectively, in particular $b_1 \neq 0$, $b_2 \neq 0$, and $b_4 \neq 0$. (b) Numerical simulations of Eq. (2.1) for $a_1 = a_2 = a_4 = 0, a_3 = b_3 = 1, b_1 = b_2 = -1, b_4 = 2$. The panel reports the time evolution of the $R(t)$ (first row), $S(t)$ (second row), $I_1(t)$ (third row), $I_2(t)$ (fourth row), and $O(t)$ (fifth row) signals. Red and blue colors are used to plot the signals which are synchronized to $R(t)$ (1-state), and $S(t)$ (0-state), respectively. The gray color is used to plot the signal during the transition between the two states. The vertical lines separate time intervals where different inputs are used, i.e., $(I_1, I_2) = (R, R)$, $(I_1, I_2) = (R, S)$, $(I_1, I_2) = (S, R)$, and $(I_1, I_2) = (S, S)$. (c) The computation accuracy measure [see Eq. (3.7)]. The red and blue background colors stays for the R and S state featured by the signal $O(t)$, respectively. Time is reported in units of the integration step.

3.3. The universal NAND Gate

The logical gate NAND is a gate of particular importance, since it has the property (together with the NOR gate) of functional completeness, and for this it is called *universal*. It is possible, indeed, to demonstrate that any Boolean function can be implemented using only NAND gates [15], and therefore implementing efficiently the NAND operation

corresponds, in practice, to being able of performing any computational task, i.e., of constructing a universal Turing machine [16]. The NAND gate produces an output signal $O(t)$ which is S (0-state) for $I_1 = I_2 = R$ ($p = q = 1$) and R otherwise, as it can be seen in Table 1.

With the same procedure adopted in the previous subsection, the following set of equations is obtained for the morphing coefficients:

- $(I_1, I_2) = (S,S) \rightarrow O(t) = R$,

$$\left.\begin{array}{ll} a) & O(t) = V_A \rightarrow a_3 = 1 \wedge a_1 + a_2 + a_4 = 0 \\ b) & V_A \not\approx V_B \rightarrow b_3 \neq 1 \vee b_1 + b_2 + b_4 \neq 0. \end{array}\right\} \quad (3.8)$$

- $(I_1, I_2) = (R,S) \rightarrow O(t) = R$,

$$\left.\begin{array}{ll} a) & O(t) = V_A \rightarrow a_1 + a_3 = 1 \wedge a_2 + a_4 = 0 \\ b) & V_A \not\approx V_B \rightarrow b_1 + b_3 \neq 1 \vee b_2 + b_4 \neq 0. \end{array}\right\} \quad (3.9)$$

- $(I_1, I_2) = (S,R) \rightarrow O(t) = R$,

$$\left.\begin{array}{ll} a) & O(t) = V_A \rightarrow a_2 + a_3 = 1 \wedge a_1 + a_4 = 0 \\ b) & V_A \not\approx V_B \rightarrow b_2 + b_3 \neq 1 \vee b_1 + b_4 \neq 0. \end{array}\right\} \quad (3.10)$$

The remaining logical operation $(I_1, I_2) = (R,R) \rightarrow O(t) = S$, is guaranteed for either one of the two following conditions:

- Condition 2.4, $V_A \approx V_B$

$$\rightarrow a_1 + a_2 + a_3 = b_1 + b_2 + b_3 \wedge a_4 = b_4. \quad (3.11)$$

- Condition 2.5,

$$\left.\begin{array}{ll} a) & O(t) = V_A \rightarrow a_1 + a_2 + a_3 = 0 \wedge a_4 = 1. \\ b) & V_A \not\approx V_B \rightarrow b_1 + b_2 + b_3 \neq 0 \vee b_4 \neq 1. \end{array}\right\} \quad (3.12)$$

Therefore, the parameters a_i and b_i, $i = 1, \ldots, 4$, must satisfy Eqs. (3.8–3.10) and no matter which one of Eqs. (3.11,3.12). Once again, the coefficients of V_A are completely determined by Eqs. (3.8a, 3.9a, 3.10a). A possible choice is $a_1 = 0$, $a_2 = 0$, $a_3 = 1$, and $a_4 = 0$, which make Eq. (3.12) impossible to be satisfied, since $a_4 \neq 1$. The remaining parameters (b_1, b_2, b_3, and b_4) must therefore satisfy Eqs. (3.8b, 3.9b, 3.10b) and Eq. (3.11), i.e., $(\, b_4 = 0 \wedge b_1 + b_2 + b_3 = 1) \wedge (b_3 \neq 1 \vee b_1 + b_2 \neq 0\,) \wedge (b_1 + b_3 \neq 1 \vee b_2 \neq 0) \wedge (b_2 + b_3 \neq 1 \vee b_1 \neq 0)$.

Straightforward calculations allow one to deduce that the parameters b_i $(i = 1, 2, 3)$ must lay on the plane $b_1 + b_2 + b_3 = 1$ with $b_1 \neq 0$, $b_2 \neq 0$, and $b_3 \neq 1$. The results of our simulations are reported in Fig. 3 (a). The first two rows show the temporal evolution of the uncontrolled signals R and S, respectively, while the third, fourth and fifth row report the controlled inputs I_1 and I_2 and the output $O(t)$, for $a_1 = a_2 = a_4 = b_3 = b_4 = 0$, $a_3 = 1$, $b_1 = 2$, and $b_2 = -1$. The same color stipulations have been used as in Fig. 2. The synchronization level is shown in Fig. 3 (b).

3.4. The other logical gates

The procedure described in Section 3.1 can be straightforwardly applied to implement all other Boolean logical gates. It should be remarked that, in all cases, the resulting set of equations defines a family of solutions for the morphing parameters. In Table 3 we have indicated one of the possible solutions for each of the 16 gates. Furthermore, Fig. 4 reports the quality of computation [Eq. (3.7)] obtained in our simulations when the morphing parameters are set to the values reported in Table 3, and one can see that, in all cases, the computation is performed in a rather accurate way.

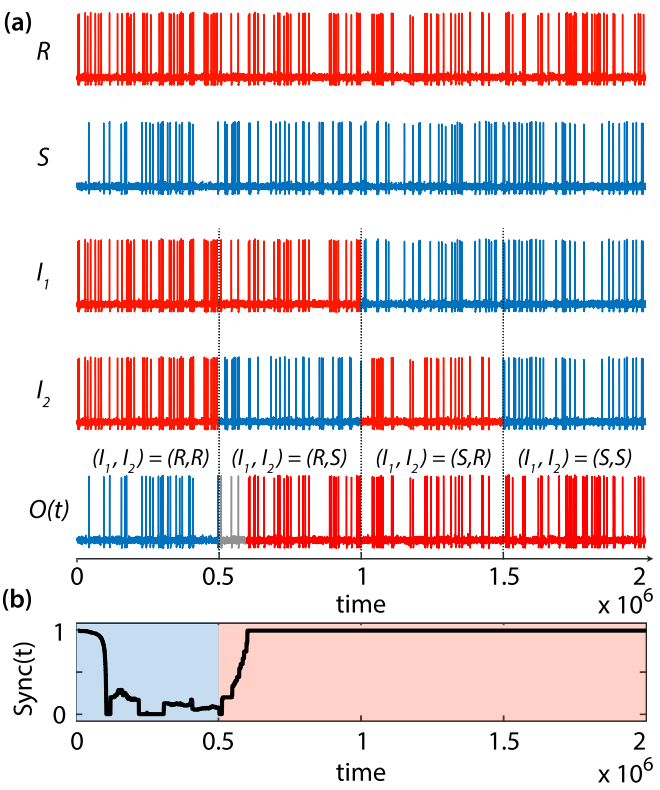

Fig. 3. (Colour online) **The NAND gate.** (a) Numerical simulations of Eq. (2.1) with $a_1 = a_2 = a_4 = b_3 = b_4 = 0$, $a_3 = 1$, $b_1 = 2$, and $b_2 = -1$. From the first to the fifth row: time evolution of the R, S, I_1, I_2, and $O(t)$ signals. Red and blue colors indicate the R (1-state), S (0-state), and the gray color is used to plot the output signal during the transition between R and S. The vertical lines separate the different inputs, i.e., $(I_1, I_2) = (R,R)$, $(I_1, I_2) = (R,S)$, $(I_1, I_2) = (S,R)$, and $(I_1, I_2) = (S,S)$. (b) The computation accuracy measure [see Eq. (3.7)]. Red and blue background colors indicate R (1-state) and S (0-state), respectively. Time is reported in units of the integration step.

4. Conclusions

In summary, we have given evidence that computation abilities emerge spontaneously from adaptive synchronization of dynamical systems. Namely, we have considered an ensemble of spiking neurons (each one obeying the Hodgkin-Huxley model) subjected to a common source of noise and interacting in a directed graph via a coupling that adapts itself to the synchronization level between two input signals.

We have demonstrated that such neurons can be used as the basic computational units for realizing all the 16 Boolean logical gates with two inputs and one output. This is realized by a suitable tuning of a set of parameters which provide therefore a *morphing* gate, i.e., a logical gate able to switch from one to another of such 16 logical functions.

Specifically, we have explicitly extracted a possible solution for the NAND gate, which has the property of functional completeness. This implies that our framework for computation is able to implement any Boolean function and/or operation and to perform, in principle, any computational task as a universal Turing machine.

Our results are of value, in that they potentially enlighten mechanisms at the basis of bio-computation processes. Moreover, it is important to remark that computation is, in our framework, an emergent feature and as so it is not limited to only binary Boolean logic, but it can be extended to a larger number of states (by having several reference signals R, as it was demonstrated in Ref. [10]) in order to perform multiple-input Boolean and even non-Boolean operations.

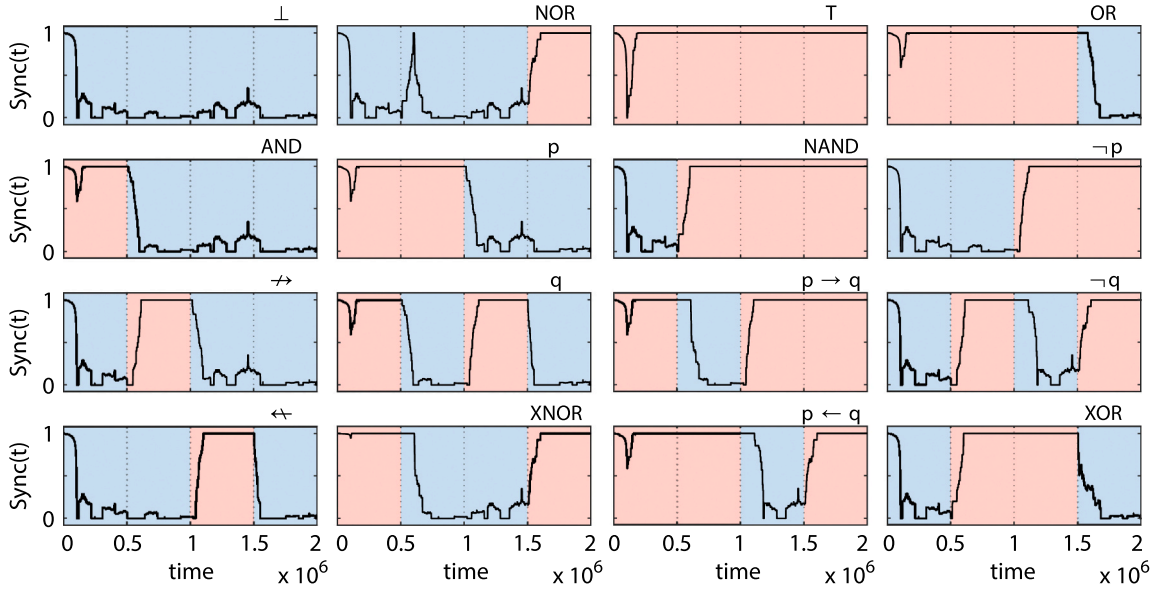

Fig. 4. (Colour online) **The 16 Boolean logic gates with two inputs and one output.** The computation accuracy [Eq. (3.7)] for all the 16 Boolean logical gates. The morphing parameters used in the simulations are reported in Table 3. Red and blue background colors indicate the R (1-state) and S (0-state) regions, respectively. In all cases, vertical lines separate the different inputs, i.e., $(I_1, I_2) = (R, R)$, $(I_1, I_2) = (R, S)$, $(I_1, I_2) = (S, R)$, and $(I_1, I_2) = (S, S)$. If compared with the truth Table 1, one sees that the computation is performed accurately in all cases. In all plots, time is reported in units of the integration step.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

Authors would like to thank Massimiliano Zanin, David Papo, Irene Sendiña-Nadal, and Zhang Shenggui for many inspiring discussions. J. Y. acknowledges support from China Scholarship Council (No. 202106290085). Work partially supported by projects PGC2018-101625-B-I00 (Spanish Ministry, AEI/FEDER, UE) and M1993 (URJC Grant). The usage of the resources, technical expertise, and assistance provided by the supercomputer facility CRESCO of ENEA in Portici (Italy) is also acknowledged.

References

[1] Boole G. The mathematical analysis of logic. Philosophical Library; 1847.
[2] Shannon CE. A symbolic analysis of relay and switching circuits. Electr Eng 1938; 57(12):713–23.
[3] Vandoorne K, Mechet P, Van Vaerenbergh T, Fiers M, Morthier G, Verstraeten D, Schrauwen B, Dambre J, Bienstman P. Experimental demonstration of reservoir computing on a silicon photonics chip. Nat Commun 2014;5(1):1–6.
[4] Haynes ND, Soriano MC, Rosin DP, Fischer I, Gauthier DJ. Reservoir computing with a single time-delay autonomous boolean node. Phys Rev E 2015;91(2): 020801.
[5] Du C, Cai F, Zidan MA, Ma W, Lee SH, Lu WD. Reservoir computing using dynamic memristors for temporal information processing. Nat Commun 2017;8(1):1–10.
[6] Pathak J, Hunt B, Girvan M, Lu Z, Ott E. Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. Phys Rev Lett 2018;120(2):024102.
[7] Gauthier DJ, Bollt E, Griffith A, Barbosa WAS. Next generation reservoir computing. Nat Commun 2021;12(1):1–8.
[8] Zanin M, Del Pozo F, Boccaletti S. Computation emerges from adaptive synchronization of networking neurons. PLoS One 2011;6(11):e26467.
[9] Zanin M, Papo D, Sendiña-Nadal I, Boccaletti S. Computation as an emergent feature of adaptive synchronization. Phys Rev E 2011;84(6):060102.
[10] Zanin M, Papo D, Boccaletti S. Computing with complex valued networks of phase oscillators. EPL (Europhys Lett) 2013;102(4):40007.
[11] Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol 1952;117(4):500.
[12] Zhou C, Kurths J. Noise-induced synchronization and coherence resonance of a hodgkin–huxley model of thermally sensitive neurons. Chaos: an interdisciplinary Journal of Nonlinear Science 2003;13(1):401–9.
[13] Zhou CS, Kurths J, Allaria E, Boccaletti S, Meucci R, Arecchi FT. Constructive effects of noise in homoclinic chaotic systems. Phys Rev E 2003;67(6):066220.
[14] Lai YM, Porter MA. Noise-induced synchronization, desynchronization, and clustering in globally coupled nonidentical oscillators. Phys Rev E 2013;88(1): 012905.
[15] Sheffer HM. A set of five independent postulates for boolean algebras, with application to logical constants. Trans Am Math Soc 1913;14(4):481–8.
[16] Turing AM. On computable numbers, with an application to the entscheidungsproblem. Proc Lond Math Soc 1937;s2-42(1):230–65.